

VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

Nikolaj GORANIN

GENETIC ALGORITHM APPLICATION IN INFORMATION SECURITY SYSTEMS

DOCTORAL DISSERTATION

TECHNOLOGICAL SCIENCES,
INFORMATICS ENGINEERING (07T)



Vilnius LEIDYKLA TECHNICA 2010

Doctoral dissertation was prepared at Vilnius Gediminas Technical University in 2006–2010.

Scientific Supervisor

Prof Dr Habil Antanas ČENYS (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07T).

VGTU leidyklos TECHNIKA 1756-M mokslo literatūros knyga
<http://leidykla.vgtu.lt>

ISBN 978-9955-28-588-5

© VGTU leidykla TECHNIKA, 2010
© Nikolaj Goranin, 2010
ngrnn@fmf.vgtu.lt

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Nikolaj GORANIN

GENETINIŲ ALGORITMŲ TAIKYMAS INFORMACIJOS SAUGOS SISTEMOSE

DAKTARO DISERTACIJA

TECHNOLOGIJOS MOKSLAI,
INFORMATIKOS INŽINERIJA (07T)



LEIDYKLA
Vilnius TECHNIKA 2010

Disertacija rengta 2006–2010 metais Vilniaus Gedimino technikos universitete.

Mokslinis vadovas

prof. habil. dr. Antanas ČENYS (Vilniaus Gedimino technikos universitetas,
technologijos mokslai, informatikos inžinerija – 07T).

Abstract

Information infrastructure protection against malicious software, such as viruses, Internet worms and botnets is a crucial task that requires development and implementation of both reactive, such as patches, removal tools, antivirus software updates and proactive, such as countermeasure planning in advance, measures.

Success of these measures application highly depends on reaction time and understanding of malware evolution trends respectively. Reaction time should be directly proportional to the risk level posed by malware. Currently risk evaluation task is based on expert knowledge and no automatic evaluation systems exist. Existing malware models mainly concentrate on malware epidemic consequences modelling, i.e. forecasting the number of infected computers, simulating malware behaviour or economic propagation aspects and are based only on current malware propagation strategies.

Present study presents an innovative genetic algorithm based malware risk level evaluation and malware evolution forecasting model. Genetic algorithm approach was selected taking into consideration its efficiency while solving tasks with large solution space and ability to model the evolution process which is the case for malware, often considered as a form of artificial life, evolution forecasting. The model presented covers the malware feature representation description in the genetic algorithm suitable format, evolution evaluation fitness functions for propagation and survivability strategy evolution forecasting of several malware types in friendly and hostile environments, algorithm operating conditions and a genetic algorithm based method for decision tree generation, used for malware risk evaluation.

Correctness of the proposed fitness evaluation criteria is tested on historical data; malware evolution modelling and risk evaluation experiments are performed; conclusions regarding the malware evolution trends are provided.

Study also reviews genetic algorithm application in information security systems, provides technical analysis of modern malware types, that are used for evolution modelling, and analyses existing malware models.

It is concluded that the proposed model is easily extensible for other malware types and parameters. It can be successfully used for evolution forecasting and risk evaluation of newly appearing malware samples.

Reziumė

Apsauga nuo kenksmingo programinio kodo, tokio kaip kompiuteriniai virusai, interneto kirminai ir kenksmingi botnet tinklai, tampa gyvybiškai svarbiu uždaviniu užtikrinant informacinės infrastruktūros saugą ir stabilumą. Taikomos apsaugos priemonės gali būti suskirstytos į reaktyvias, tokias kaip programinės įrangos ir antivirusinių programų duomenų bazių atnaujinimai, šalinimo įrankių kūrimas, ir proaktyvias, tai yra planuojamas iki grėsmės atsiradimo.

Apsaugos priemonių grupių taikymo efektyvumas priklauso nuo reakcijos laiko pirmu, ir kenksmingo programinio kodo evoliucijos tendencijų supratimo, antru atveju. Reakcijos laikas turėtų būti tiesiogiai proporcingas naujai atsirandančio kenksmingo programinio kodo pavojingumo lygiui, kurio įvertinimas šiuo metu pagrįstas ekspertiniais vertinimais. Šiuolaikiniai kenksmingo programinio kodo modeliavimo įrankiai yra skirti epidemiologinių arba ekonominių pasekmių prognozavimui ir kenksmingo programinio kodo elgesio simulavimui bei nevertina perspektyvių kenksmingo programinio kodo strategijų.

Šioje disertacijoje siūlomas genetiniais algoritmais paremtas kenksmingo programinio kodo rizikos vertinimo ir evoliucijos prognozavimo modelis. Genetiniai algoritmai pasirinkti atsižvelgiant į metodo efektyvumą sprendžiant daugelio sričių, tame tarpe ir informacinės saugos, uždavinius, bei galimybę modeliuoti evoliucijos procesą, kuris gali būti taikomas ir kenksmingo programinio kodo, dažnai traktuojamo kaip dirbtinės gyvybės atmaina, evoliucijos modeliavimui. Modelio aprašymą sudaro kenksmingo programinio kodo atvaizdavimo aprašymas, tikslo funkcijos skirtingų parametrų evoliucijos prognozavimui, algoritmo veikimo parametrai ir genetiniais algoritmais paremtas sprendimų medžių, naudojamų rizikos vertinimui, generavimo metodas.

Pasiūlytų tinklo funkcijų korektiškumas tikrinamas pritaikant istorinius duomenis, atliekami evoliucijos modeliavimo ir rizikos vertinimo eksperimentai, padaromos išvados dėl kenksmingo programinio kodo evoliucijos tendencijų.

Disertacijoje taip pat analizuojami genetinių algoritmų taikymai informacijos saugos sistemose, pateikiama kelių šiuolaikinių kenksmingo programinio kodo rūšių, kurių evoliucija yra modeliuojama, techninė analizė, bei nagrinėjamos egzistuojančios kenksmingo programinio kodo modeliavimo priemonės.

Darbe teigiama, kad pasiūlytas modelis yra lengvai plečiamas ir gali būti taikomas kenksmingo programinio kodo evoliucijos modeliavimui bei naujai atsirandančių egzempliorių rizikos vertinimui.

Notations

Abbreviations

2D – two dimensional (geometrics);
3D – three dimensional (geometrics);
AES – advanced encryption standard;
AI – artificial intelligence;
AL – artificial life;
ANN – artificial neural networks;
C&C – command & control (center);
CPU – central processing unit;
CS – classifier systems;
CVE – common vulnerabilities and exposures;
DARPA – The Defense Advanced Research Projects Agency
DDoS – distributed denial of service (attack);
DES – data encryption standard;
DNA – deoxyribonucleic acid;
DNS – domain name system;
DOS – denial of service;
EA – evolutionary algorithm;
EEG – electroencephalogram;
EER – error rate;

ENISA – The European Network and Information Security Agency;
ES – evolutionary strategies;
FDR – Fischer discriminant ratio;
FTP – file transfer protocol;
FVC – Fingerprint Verification Contest;
GA – genetic algorithm;
GP – genetic programming;
HDD – hard disk drive;
HIDS – host-based IDS;
HTTP – hypertext transfer protocol;
ICIGA – improved cryptography inspired by genetic algorithms;
ICMP – Internet control message protocol;
ID – intrusion detection;
IDC – International Data Corporation;
IDS – intrusion detection system;
IIS – Internet Information Services;
IP – Internet protocol;
IPS – intrusion prevention system;
IRC – Internet relay chat;
IT – information technology;
LDA – linear discriminant analysis;
LPR – line printer remote;
MMS – multimedia messaging service;
NIDS – network-based IDS;
NIST – National Institute of Standards and Technology;
NLFFSR – non-linear function feedback shift register;
OS – operating system;
P2P – peer-to-peer;
R2L – Remote2Local;
RCS – The Random Constant Spread;
RSA – Rivest, Shamir and Adleman algorithm;
SIR – susceptible-infected-recovered;
SIS – susceptible-infected-susceptible;
SMS – short message service;
SMTP – simple mail transfer protocol;
SNR – signal-to-noise ratio;
TC – termination condition;
TCP – transmission control protocol;
U2R – User2Root;
UDP – user datagram protocol;
UTC – coordinated universal time;
VEP – visual evoked potential.

Symbols

A – public key;

$a(t)$ – proportion of vulnerable machines which have been compromised at the instance t ;

A' – private key;

a_{ij} – adjustable parameter (biometry);

c_i – greyscale intensity;

CPU_LOAD_i – average gene's method load on the infected computer's CPU during time t_j ;

DDF – measured frequency for a diagram;

DF – measured character's frequency in the decoded ciphertext;

F – fitness function, that evaluates the malware strategy's efficiency under in terms of survivability;

f_i – fitness of the i^{th} individual in population of solutions;

F_{SC} – fitness function, that evaluates the malware strategy's efficiency under pressure of countermeasures;

k – activity level of malware strategy being evaluated, either in terms of number of action performed (propagation forecasting) or computer resource usage level (survivability forecasting);

K – value based on $K(S)$ function calculations, that evaluated the S strategy's fitness in sense of propagation speed efficiency;

m – relatively primary number (cryptography) or number of points considered (biometry);

Matched – revellance of gene in historical data;

N – population size;

N_{CC} – number of Command and Control servers, needed for botnet to remain stable in time interval T ;

p_i – probability assigned to the specific gene i , describing its effectiveness probability or probability of the i^{th} individual in population to be selected;

ranking – difficulty of intrusion detection;

S – malware strategy representing variable;

SDF – standard frequency for a diagram;

SF – standard character's frequency in English plaintext;

T – template image (biometry) or time interval (Random Constant Spread model);

t_{CC_block} – the average time needed for botnet fighters to block the Command and Control center of a specific hierarchy;

t_j, t_{ij} – time consumptions for the specific gene functionality or gene's activity period;

T_S – time interval necessary for botnet to remain stable;

t_x, t_y – adjustable parameter (biometry);

w – relatively primary number;

w^{-1} – inverse to the relatively primary number w ;

$Weight_i$ – conditional gene's weight;

x, y – Cartesian spacial dimension.

Contents

INTRODUCTION	1
The Investigated Problem.....	1
Importance of the Thesis	2
The Object of Research	3
The Goal of the Thesis	3
The Tasks of the Thesis.....	3
Research Methodology	4
Importance of Scientific Novelty	4
Practical Significance of Achieved Results.....	5
The Defended Statements.....	5
Approval of the Results.....	6
Structure of the Dissertation.....	6
Acknowledgements	7
 1. GENETIC ALGORITHMS AND THEIR APPLICATION IN INFORMATION SECURITY SYSTEMS	 9
1.1. Genetic Algorithm Principles	10
1.2. Genetic Algorithm Application in Cryptology	14
1.2.1. Cryptographic Genetic Algorithm Implementations.....	14
1.2.2. Cryptological Genetic Algorithm Inspired Attacks	17
1.3. Genetic Algorithms Approach in Intrusion Detection Systems.....	20
1.3.1. Review of the IDS Technology.....	20

1.3.2. Genetic Algorithm Application Research Trends and Neuro-Genetic Approach.....	22
1.3.3. GA Application for Automatic IDS Rule Generation.....	23
1.4. Genetic Algorithms Application Analysis in Biometric Systems	25
1.4.1. Genetic Algorithm Application for Fingerprint Biometric Information Processing	26
1.4.2. Genetic Algorithm Application for Face Feature Processing	33
1.4.3. Other Biometric Information Processing by Genetic Algorithms	37
1.5. Research on Information Security in Lithuania.....	39
1.6. Conclusions of Chapter 1 and Formulating Tasks for the Dissertation.....	40
 2. MALWARE TECHNICAL ANALYSIS AND MALWARE MODELS.....	43
2.1. Malware Strategy Definition	44
2.2. Internet Worms.....	44
2.2.1. Internet Worm – CodeRed	45
2.2.2. Internet Worm – Ramen.....	46
2.3. Mobile Malware	46
2.4. Botnets.....	47
2.4.1. IRC Botnet – Agobot	51
2.4.2. P2P Botnet – STORM.....	53
2.5. Malware Models.....	54
2.5.1. General or Malware Specific Models	55
2.5.2. Genetic Algorithm Based Models.....	58
2.6. Conclusions of Chapter 2	61
 3. AUTOMATIC MALWARE RISK EVALUATION MODEL	63
3.1. Risk Definition and General Model Requirements	63
3.2. Decision Trees Optimization by Genetic Algorithms	64
3.3. GAtree Modelling Tool.....	65
3.4. Data Representation for Decision Tree Generation.....	67
3.5. Experiment Conditions and Results	68
3.6. Conclusions of Chapter 3	70
 4. MALWARE EVOLUTION FORECASTING MODEL.....	71
4.1. General Model Assumptions	72
4.2. Model Correctness Evaluation	73
4.3. Internet Worm Evolution Modelling	74
4.3.1. Internet Worm Features Representing Chromosome	74
4.3.2. Genetic Algorithm Operation Parameters	79
4.3.3. Fitness function.....	79
4.3.4. Model Limitations.....	81
4.3.5. Experiment Results	81
4.3.6. Fitness Function and Modelling in Hostile Environment	82
4.4. Mobile Malware Evolution Modelling	84
4.4.1. Mobile Malware Features Representing Chromosome	84

4.4.2. Genetic Algorithm Operation Parameters	85
4.4.3. Fitness Function	86
4.4.4. Experiment Results	86
4.5. Botnet Evolution Modelling	88
4.5.1. Botnet Features Representing Chromosome	89
4.5.2. Genetic Algorithm Operation Parameters	93
4.5.3. Fitness Function – Propagation Technique Evolution Forecasting	93
4.5.4. Fitness Function – Survivability Technique Evolution Forecasting	95
4.5.5. Model Limitations	97
4.6. Concluding Remarks of Chapter 4	97
GENERAL CONCLUSIONS	99
REFERENCES	101
LIST OF PUBLICATIONS BY THE AUTHOR ON THE TOPIC OF THE DISSERTATION	113

Turinys

IVADAS	1
Tiriamoji problema.....	1
Darbo aktualumas.....	2
Tyrimų objektas	3
Darbo tikslas.....	3
Darbo uždaviniai	3
Tyrimų metodika.....	4
Darbo mokslinis naujumas ir jo reikšmė.....	4
Darbo rezultatų praktinė reikšmė	5
Ginamieji teiginiai.....	5
Darbo rezultatų aprobavimas.....	6
Disertacijos struktūra.....	6
Padėkos	7
 1. GENETINIAI ALGORITMAI IR JŲ TAIKYMAS INFORMACIJOS SAUGOS SISTEMOSE	 9
1.1. Genetinių algoritmų principai	10
1.2. Genetinių algoritmų taikymas kriptologijoje.....	14
1.2.1. Kriptografiniai genetinių algoritmų taikymai	14
1.2.2. Kriptologinės genetiniais algoritmais paremtos atakos.....	17
1.3. Genetinių algoritmų metodų taikymas įsiskverbimų detektavimo sistemose.....	20
1.3.1. IDS technologijos apžvalga	20

1.3.2. Genetinių algoritmų taikymų ir tyrimų tendencijos bei neuro-genetinis metodas	22
1.3.3. GA taikymas automatiniam IDS taisyklių generavimui.....	23
1.4. Genetinių algoritmų taikymo analizė biometrinėse sistemose	25
1.4.1. Genetinių algoritmų taikymas pirštų atspaudų biometrinės informacijos apdorojimui	26
1.4.2. Genetinių algoritmų taikymas veido atvaizdų apdorojimui	33
1.4.3. Kitos biometrinės informacijos apdorojimas genetiniais algoritmais	37
1.5. Informacinės saugos tyrimai Lietuvoje	39
1.6. Pirmojo skyriaus išvados ir disertacijos uždavinių formulavimas.....	40
 2. KENKSMINGO PROGRAMINIO KODO TECHNINĖ ANALIZĖ IR KENKSMINGO PROGRAMINIO KODO MODELIAI	43
2.1. Kenksmingo programinio kodo strategijos apibrėžimas	44
2.2. Interneto kirminai	44
2.2.1. Interneto kirminas – CodeRed	45
2.2.2. Interneto kirminas – Ramen	46
2.3. Mobilus kenksmingas programinis kodas	46
2.4. Botnet tinklai	47
2.4.1. IRC botnet tinklas – Agobot	51
2.4.2. P2P botnet tinklas – STORM.....	53
2.5. Kenksmingo programinio kodo modeliai	54
2.5.1. Bendri arba pagal kensmingo programinio kodo tipą specializuoti modeliai.....	55
2.5.2. Genetiniais algoritmais paremti modeliai	58
2.6. Antrojo skyriaus išvados	61
 3. AUTOMATINIS KENKSMINGO PROGRAMINIO KODO RIZIKOS VERTINIMO MODELIS.....	63
3.1. Rizikos apibrėžimas ir bendri modelio reikalavimai	63
3.2. Sprendimų medžių optimizavimas genetiniais algoritmais	64
3.3. GAtree modeliavimo įrankis	65
3.4. Duomenų atvaizdavimas sprendimų medžių generavimui	67
3.5. Eksperimento sąlygos ir rezultatai.....	68
3.6. Trečiojo skyriaus išvados	70
 4. KENKSMINGO PROGRAMINIO KODO EVOLIUCIJOS PROGNOZAVIMO MODELIS	71
4.1. Bendros modelio prielaidos.....	72
4.2. Modelio korektiškumo vertinimas.....	73
4.3. Interneto kirminų evoliucijos modeliavimas	74
4.3.1. Interneto kirminų savybes atvaizduojanti chromosoma.....	74
4.3.2. Genetinio algoritmo parametrai	79
4.3.3. Tikslų funkcija	79
4.3.4. Modelio apribojimai.....	81

4.3.5. Eksperimento rezultatai	81
4.3.6. Tikslų funkcija ir modeliavimas priešiškoje aplinkoje.....	82
4.4. Mobilus kenksmingo programinio kodo evoliucijos modeliavimas	84
4.4.1. Mobilus kenksmingo programinio kodo savybės atvaizduojanti chromosoma	84
4.4.2. Genetinio algoritmo parametrai	85
4.4.3. Tikslų funkcija	86
4.4.4. Eksperimento rezultatai	86
4.5. Botnet tinklų evoliucijos modeliavimas	88
4.5.1. Botnet tinklų savybės atvaizduojanti chromosoma.....	89
4.5.2. Genetinio algoritmo parametrai	93
4.5.3. Tikslų funkcija – plitimo metodų evoliucijos prognozavimas	93
4.5.4. Tikslų funkcija – išgyvenamumo savybių evoliucijos prognozavimas ...	95
4.5.5. Modelio apribojimai.....	97
4.6. Ketvirtąjo skyriaus išvados	97
BENDROSIOS IŠVADOS	99
LITERATŪRA IR ŠALTINIAI.....	101
AUTORIAUS PUBLIKACIJOS DISERTACIJOS TEMA	113

List of Figures

Fig. 1.1. Genetic algorithm block diagram	10
Fig. 1.2. Binary solution representation in GA	11
Fig. 1.3. Single point crossover.....	12
Fig. 1.4. Mutation in GA.....	13
Fig. 1.5. A General model of 4 bit NLFFSR (Almarimi <i>et al.</i> 2008)	15
Fig. 1.6. Encryption / decryption example (Almarimi <i>et al.</i> 2008)	16
Fig. 1.7. GA-based hardware AES encryption scheme (Wang, L. <i>et al.</i> 2006).....	16
Fig. 1.8. Sample IDS rule (Sinclair <i>et al.</i> 1999).....	23
Fig. 1.9. Chromosome representing IDS rule (Li 2004)	24
Fig. 1.10. Order of variable <i>Weight</i> change order (Li 2004).....	24
Fig. 1.11. Elementary geometric transforms for planar surface elements (El-Emary and Abd El-Kareem 2008)	27
Fig. 1.12. a) Fingerprint sample with minutiae marked (Fingerprint 2004) b) minutiae types (Fingerprint 2005)	29
Fig. 1.13. GA-based fingerprint image generation method (Cho <i>et al.</i> 2007).....	31
Fig. 1.14. Generated fingerprint images (Cho <i>et al.</i> 2007)	32
Fig. 1.15. Block diagram of 2D face feature extraction method (Yen and Nithianandan 2002).....	34
Fig. 1.16. Chromosome for 2D face image segmentation (Yen and Nithianandan 2002).....	34
Fig. 1.17. Chromosome for 2D face image feature extraction (Yen and Nithianandan 2002).....	35

Fig. 1.18. 3D facial feature extraction (Sun and Yin 2007)	36
Fig. 1.19. Learning and recognition model for the Neuro-Genetic hybrid system (Islam and Rahman 2009).....	38
Fig. 2.1. Botnet with centralized architecture	51
Fig. 2.2. Decentralized P2P botnet.....	53
Fig. 2.3. Malware propagation graph.....	55
Fig. 2.4. Architecture of the malware evolution model (Noreen <i>et al.</i> 2009)	59
Fig. 3.1. Mutation on tree data structure (Papagelis and Kalles 2001)	66
Fig. 3.2. Crossover on tree data structure (Papagelis and Kalles 2001).....	66
Fig. 3.3. Sample Internet worm strategy representation for risk evaluation	68
Fig. 3.4. Risk evaluation decision tree fragment.....	69
Fig. 4.1. Sample Internet worm strategy randomly generated	78
Fig. 4.2. Change in fitness of the whole population and the best individual (Internet worms).....	82
Fig. 4.3. Best strategy fitness change graph (Mobile malware).....	87
Fig. 4.4. Average population fitness change graph (Mobile malware)	87

Paveikslų sąrašas

1.1 pav. Genetinio algoritmo blokinė diagrama.....	10
1.2 pav. Dvejatainis sprendimo atvaizdavimas GA	11
1.3 pav. Vieno taško kryžminimas	12
1.4 pav. Mutacija GA.....	13
1.5 pav. Bendras 4 bitų NLFFSR modelis (Almarimi <i>et al.</i> 2008)	15
1.6 pav. Šifravimo / dešifravimo pavyzdys (Almarimi <i>et al.</i> 2008).....	16
1.7 pav. GA-paremta aparatinė AES šifravimo schema (Wang, L. <i>et al.</i> 2006).....	16
1.8 pav. IDS taisyklės pavyzdys (Sinclair <i>et al.</i> 1999).....	23
1.9 pav. IDS taisyklę atvaizduojanti chromosoma (Li 2004).....	24
1.10 pav. Kintamojo <i>Weight</i> keitimosi tvarka (Li 2004).....	24
1.11 pav. Pagrindinės geometrinės transformacijos plokšties elementams (El-Emary and Abd El-Kareem 2008).....	27
1.12 pav. a) Piršto antspaudas su pažymėtomis detalėmis (Fingerprint 2004) b) detalių tipai (Fingerprint 2004)	29
1.13 pav. GA-paremtas piršto antspaudų atvaizdų generavimo metodas (Cho <i>et al.</i> 2007).....	31
1.14 pav. Sugeneruoti pirštų antspaudų atvaizdai (Cho <i>et al.</i> 2007).....	32
1.15 pav. 2D veido bruožų išrinkimo metodo blokinė diagrama (Yen and Nithianandan 2002).....	34
1.16 pav. 2D veido atvaizdų segmentavimo chromosoma (Yen and Nithianandan 2002).....	34

1.17 pav. Chromosoma naudojama bruožų išrinkimui iš 2D veido atvaizdų (Yen and Nithianandan 2002)	35
1.18 pav. 3D veido bruožų išrinkimas (Sun and Yin 2007)	36
1.19 pav. Apsimokantis atpažinimo modelis hibridinėje Neuro-Genetinėje sistemoje (Islam and Rahman 2009).....	38
2.1 pav. Centralizuotos architektūros botnet tinklas	51
2.2 pav. Decentralizuotas P2P botnet tinklas	53
2.3 pav. Kenksmingo programinio kodo plitimo grafikas.....	55
2.4 pav. Kenksmingo programinio kodo evoliucijos modelio architektūra (Noreen <i>et al.</i> 2009).....	59
3.1 pav. Mutacija medžio duomenų struktūroje (Papagelis and Kalles 2001)	66
3.2 pav. Kryžminimas medžio duomenų struktūroje (Papagelis and Kalles 2001).....	66
3.3 pav. Pavyzdinio Interneto kirmino strategijos atvaizdavimas rizikos vertinimui.....	68
3.4 pav. Rizikos vertinimo sprendimo medžio fragmentas	69
4.1 pav. Pavyzdinė atsitiktinai sugeneruota Interneto kirmino strategija.....	78
4.2 pav. Populiacijos ir geriausio individo tinkamumo keitimasis (Interneto kirminai) .	82
4.3 pav. Geriausios strategijos tinkamumo keitimosi grafikas (Mobilus kenksmingas programinis kodas)	87
4.4 pav. Vidutinio populiacijos tinkamumo keitimosi grafikas (Mobilus kenksmingas programinis kodas)	87

List of Tables

Table 2.1. Abstract Representation of Bagle (Noreen <i>et al.</i> 2009)	60
Table 3.1. Representation of the Internet Worm strategies for risk evaluation.....	67
Table 3.2. Internet Worm population decrease categories	68
Table 4.1. Internet Worm feature representing chromosome.....	75
Table 4.2. Sample S_i Internet worm strategy fitness calculation parameters	80
Table 4.3. Mobile malware feature representing chromosome	84
Table 4.4. Botnet feature representing chromosome.....	90

Lentelių sąrašas

2.1 lentelė. Abstraktus Bagle viruso atvaizdavimas (Noreen <i>et al.</i> 2009).....	60
3.1 lentelė. Interneto kirminų strategijų atvaizdavimas rizikos vertinimui	67
3.2 lentelė. Interneto kirmino populiacijos kritimo kategorijos	68
4.1 lentelė. Interneto kirminų savybes atvaizduojanti chromosoma	75
4.2 lentelė. Pavyzdinės S_i Interneto kirmino strategijos tinkamumo įvertinimo skaičiavimo parametrai	80
4.3 lentelė. Mobilų telefonų kenksmingo programinio kodo savybes atvaizduojanti chromosoma	84
4.4 lentelė. Botnet savybes atvaizduojanti chromosoma.....	90

Introduction

The Investigated Problem

The information security system term covers a wide range of information systems the main aim of which is insurance of information security, i.e. protection of information confidentiality, integrity, availability, non-repudiation or performing the related applied or scientific tasks, such as attack modelling, user authentication verification, malware spread modelling, electronic signatures, information security system reliability and security level verification, etc.

Malware, i.e. software created with specific aim to harm legal information system user is considered to be among the biggest threats to modern information technology infrastructure and harm done by it is constantly increasing. Protection against current malware species is implemented by the means of signature-based or other antivirus systems, intrusion detection systems, spam-filters and other countermeasures, but the need of evaluating the potential risk of a newly appearing malware species as well as forecasting malware evolution tendencies remains uncovered or is based on an empirical evaluation.

Malware risk level evaluation and evolution forecasting tasks are especially important for quick response implementation and countermeasure planning in advance respectively. The first task requires the high solution velocity and the second has a large solution space.

In this research modern malware mechanisms are analysed and application of genetic algorithms for newly appearing malware risk evaluation and malware evolution forecasting is proposed.

Importance of the Thesis

Business processes of companies and organizations nowadays are highly dependant on the information technology infrastructure. To remain competitive and effective so should be their information systems. Malware is considered to be among the biggest threats to information resources availability, confidentiality and integrity, together with insider and hacker attacks. The negative malware economic effect is extremely high, due to both direct harm done by malware and expenses the companies have to spend on countermeasures.

Significant changes in malware creators' motivation from recognition in hacker community to financial gain make every company a potential victim, and the protection against malware a crucial task. Still in many cases countermeasures implemented remain inefficient or insufficient. The main reasons for this are dependence of current antivirus software on signature databases, gaps in the software update process, incorrect evaluation of the threat posed by a specific malware and inadequacy of countermeasures implemented to a threat caused by a new malware. While the solution of the first two problems lays in technical-organizational area (usage of heuristic or anomaly based malware detection methods, optimisation of signature database and software patch management process, implementation of strict information security standards) and an intensive research is done to improve technical characteristics of antivirus software, the malware risk level evaluation and evolution forecasting tasks remain uncovered by scientific models and rely on empirical expert knowledge and evaluation, although many malware researchers recognize the fact that creation of risk level evaluation and evolution forecasting model would be important both from practical and scientific point of view. Existing malware models mainly concentrate on malware epidemic consequences modelling, i.e. forecasting the number of infected computers, simulating malware behaviour or economic propagation aspects and are based only on current malware propagation strategies.

It is widely accepted that malware can be considered as a form of artificial life. Due to that reason it should follow the general Darwinian evolution rules, since methods implemented by malware creators in malware are forced by external factors: available technology, tasks assigned and countermeasure pressure. In this research we propose using the genetic algorithm modelling approach for malware risk level evaluation and evolution forecasting. The main attention is dedicated to three malware types: the most aggressive one – Internet worms,

rapidly evolving botnets and relatively new mobile malware. The genetic algorithm is selected as a modelling tool taking into consideration its ability to simulate natural evolution processes, efficiency while solving optimisation, modelling problems with large solution space and successful application for other information security tasks. The model includes the genetic algorithm description, operating conditions, chromosome that describes malware characteristics, the fitness function for evolution forecasting or evaluation criteria for risk level evaluation and the modelling platform.

The Object of Research

The object of present study is application of genetic algorithms in information security systems.

The Goal of the Thesis

The main objective is creation of malware risk level evaluation and malware evolution forecasting model.

The Tasks of the Thesis

In order to achieve the objective, the following problems had to be solved:

1. To review genetic algorithms and their application in different information security areas such as cryptology, intrusion detection systems, biometry and others.
2. To analyse existing malware models.
3. To analyse and describe features of selected modern malware types in the way suitable for representation in genetic algorithm model, dedicated to risk level evaluation and evolution forecasting.
4. To derive fitness evaluation criteria for the genetic algorithm based malware risk level evaluation model.
5. To derive a fitness functions 2–3 parameters evolution tendencies forecasting of several malware types in a friendly (no countermeasures applied) and, where applicable, hostile (countermeasures applied) environment.

6. To evaluate correctness of the fitness functions and fitness evaluation criteria by applying them to historical data.
7. To define genetic algorithm operation conditions such as mutation, crossover operators, population size, etc.
8. To develop the prototype, perform experiments on several malware types, collect and evaluate modelling results.

Research Methodology

To achieve the *objective*, the following *research methods* were used:

- *Comparative research* and literature research methods were used while analysing the genetic algorithm principles, their applications in information security systems, existing malware models and modern malware techniques.
- The analysis results were summarized and the approach was expounded using the research *generalization* and *logical induction methods*.
- The *experiment research* method was applied while performing the model tests.

Importance of Scientific Novelty

The aspects of scientific novelty on theoretical and experimental investigation of genetic algorithm application for malware risk level evaluation and evolution forecasting are as follows:

1. The model proposed is the first known malware model dedicated to malware evolution forecasting task.
2. The automated model of malware risk evaluation, based on decision tree generation by genetic algorithm, was proposed.
3. The first known application of genetic algorithms for malware evolution forecasting task and automatic classifying tree generation, dedicated to new malware risk level evaluation.

Practical Significance of Achieved Results

Malware risk level evaluation and forecasting evolution tendencies is important for scientific substantiation of expert prediction evaluation, understanding of malware development tendencies, malware evolution forecasting by separate parameters or their complexes, development of countermeasure techniques and prevention of malware epidemic outbreaks by implementing quick response mechanisms.

The proposed model was applied for Internet worms propagation strategy evolution forecasting (prior to satiation, both friendly and hostile environments), mobile malware (prior to satiation, friendly environment) and botnets (all propagation phases, friendly environment), for botnet survivability evolution forecasting (after satiation, hostile environment) and for evaluation of internet worm population stability threat evaluation and certain results were achieved, that allowed us to make evaluation about evolution tendencies of these malware types.

The applicability of the proposed model is highly dependent on the collection of the reliable statistical data. The proposed model is easily extensible for other malware types and parameters so only minor modifications should be made to apply the model for other malware types and parameters evolution forecasting.

The Defended Statements

The following statements based on the results of present investigation may serve as the official hypotheses to be defended:

1. Application of genetic algorithms allows forecasting malware evolution tendencies.
2. Genetic algorithm approach for decision tree, dedicated to malware risk level evaluation, generation, allows automatic malware risk level evaluation.
3. The proposed genetic algorithm based malware risk level evaluation and evolution forecasting model is expandable for different malware types and malware parameters.

Approval of the Results

The author has made 6 presentations at 6 scientific conferences (3 international):

- NATO RTO Information Systems Technology Panel Symposium, Information Assurance and Cyber Defense (IST-091 / RSY-021), Antalya, Turkey.
- The 16th International Conference on Information and Software Technologies, Kaunas, Lithuania.
- The 12th Conference for Lithuania Junior Researchers SCIENCE FOR FUTURE, Vilnius, Lithuania.
- The 11th Conference for Lithuania Junior Researchers SCIENCE FOR FUTURE, Vilnius, Lithuania.
- The 14th International Conference of ELECTRONICS, Vilnius, Lithuania.
- The 10th Conference for Lithuania Junior Researchers SCIENCE FOR FUTURE, Vilnius, Lithuania.

Structure of the Dissertation

The dissertation is composed of Introduction, four main chapters and general conclusions. The total dissertation scope is 115 pages, 40 equations, 30 pictures and 7 tables.

In Introduction topicality of the problem is analysed, work main objective and tasks are formulated, scientific novelty is proved, research methods used are listed with corresponding application areas and defended statements are presented.

Chapter 1 provides the analysis of genetic algorithms general principles and their application in information security systems such as biometric authentication systems, cryptology, intrusion detection systems and others. Special attention is dedicated to genetic algorithm representation methods and operating conditions dependability on the task solved: chromosome formatting methods, fitness functions or evaluation criteria, crossover and mutation parameters, etc. Neuro-genetic approach is also analysed on set of examples. Expediency of genetic algorithm application approach for each of the methods reviewed is evaluated and the most perspective research areas of genetic algorithm application in sphere of information security are established. The main aim of the analysis is to establish if there are any ready genetic algorithm based solutions suitable for the fulfilment of the main objective.

Chapter 2 describes features of three selected modern malware types: Internet worms, which can be classified as the most aggressive malware type, relatively new mobile malware and rapidly evolving botnets, which may be considered as the biggest threat to modern information infrastructure. Understanding of malware propagation, survivability and other mechanisms is important for representation in a genetic algorithm suitable format. Technical analysis covers an overview of the malware type and more in-deep description of the most typical malware type representatives. Existing malware models are analysed in order to compare their features and efficiency with those proposed in the genetic algorithm based malware risk evaluation and evolution forecasting model.

Chapter 3 presents malware risk level evaluation model and experimental investigation of its application to evaluation of Internet worm population stability threat after the satiation phase. Risk evaluation model is based on decision trees, composed out of the classified historical data and generated with the help of genetic algorithms (GAtree software). The model tests on Internet worms with known population stability after the satiation phase are described.

Chapter 4 provides the description of malware evolution forecasting model, based on genetic algorithms, that simulates the process of natural evolution to malware, which can be considered as a form of artificial life. Model is applied for three malware types: Internet worms, mobile malware and botnets, which were described previously. Evolution of two malware parameters: propagation techniques (for Internet worms, mobile malware, botnets) and survivability (for botnets) is being modelled. Model for Internet worm propagation technique evolution forecasting includes both friendly and hostile environment cases.

General conclusions as well as recommendations for further research summarises the present study. It is followed by an extensive list of 169 references and a list of 7 publications by the author on the topic of the dissertation.

Acknowledgements

I am grateful to my scientific supervisor Prof Dr Habil Antanas Čenys for guiding and helping me in my studies and preparing the thesis. I would also like to thank Prof Habil Dr Gintautas Dzemyda and reviewers Prof Habil Dr Romualdas Baušys and Prof Dr Dalius Navakas for valuable comments and remarks on this thesis. I am indebted to Vilnius Gediminas Technical University and Institute of Mathematics and Informatics academic staff and many of my colleagues for help and ideas provided. This thesis would not have been possible unless support from my parents and family. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the preparation of this thesis.

Genetic Algorithms and Their Application in Information Security Systems

This Chapter provides the analysis of genetic algorithms general principles and their application in information security systems such as biometric authentication systems, cryptology, intrusion detection systems and others. Special attention is dedicated to genetic algorithm representation methods and operating conditions dependability on the task solved: chromosome formatting methods, fitness functions or evaluation criteria, crossover and mutation parameters, etc. Neuro-genetic approach is also analysed on set of examples. Expediency of genetic algorithm application approach for each of the methods reviewed is evaluated and the most perspective research areas of genetic algorithm application in sphere of information security are established. The main aim of the analysis is to establish if there are any ready genetic algorithm based solutions suitable for the fulfilment of the main objective.

Analysis presented in this Chapter was published in (Goranin and Cenys 2007^{*}).

^{*} The references are given in the list of publications by the author on the topic of the dissertation

1.1. Genetic Algorithm Principles

Genetic algorithms (GA) is a family of computational algorithms which are based on evolutionary processes (Whitley 1993) and considered to be a part of Evolutionary algorithms (EA) which nowadays also cover such methods and techniques as evolutionary strategies (ES), genetic programming (GP), artificial life (AL), classifier systems (CS), concept of evolving mechanisms (Reeves and Rowe 2003) and many others. EA are inspired by Darwinian evolution theory of natural selection. GA could mimic nature to computationally emulate the same “survival of the fittest” paradigm for difficult problems.

Although several attempts were made to adopt some evolutionary concepts in algorithmic environment as early as in 50's (Barricelli 1957; Fraser 1957), 60's (Barricelli 1963) and almost all essential GA components were described in the beginning of 70's (Fraser and Donald 1970; Crosby 1973), John Holland is considered to be the formal GA author. In 1975 he proposed the optimisation method, based on evolutionary and genetic processes and introduced a formalized framework for predicting the quality of the next generation, known as Holland's Schema Theorem (Holland 1975).

GA method is based on population of individuals. Each individual is represented as a chromosome and stands for one of the possible problem solutions (Holland 1975). The general algorithm flow is presented in Fig. 1.1.

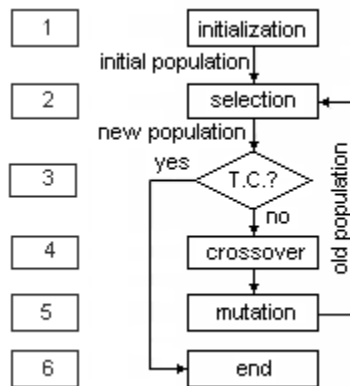


Fig. 1.1. Genetic algorithm block diagram

During the initialisation stage (Fig. 1.1 (Step 1)) initial population of solutions is generated. At the selection stage (Fig. 1.1 (Step 2)) solutions are selected through a fitness-based process and in case termination condition (TC) is not met (Fig. 1.1 (Step 3)) evolutionary mechanisms are started (Fig. 1.1 (Steps

4 and 5)). In case TC is reached, algorithm execution is ended (Fig. 1.1 (Step 6)).

Initial population of solutions of a fixed size n is usually generated on a random basis. Traditionally, individuals (or chromosomes) are represented in binary as strings of 0s and 1s, of a length m (Fig. 1.2), but other encodings are also possible.

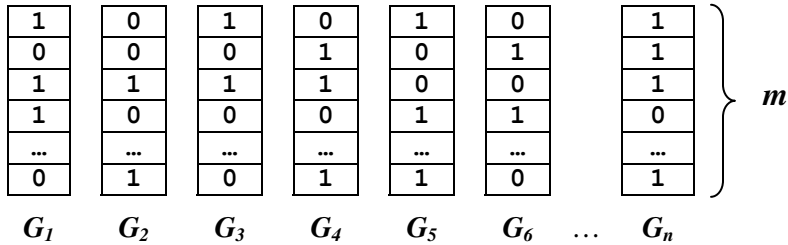


Fig. 1.2. Binary solution representation in GA

Individual genomes are chosen from a population at the selection stage for later breeding. Selection is performed through a fitness-based process, which accomplishes the “survival of the fittest” paradigm, i.e. fitter solutions evaluated by a fitness function or any other kind of evaluation criteria have better chances to be selected as parents for a new generation of solutions. By allocating more reproductive occurrences to above average individual solutions, the overall effect is to increase the population’s average fitness (Huang and Wechsler 1999). The fitness function is a particular type of objective function that quantifies the optimality of a solution (i.e. an individual in a population) so that the particular individual may be ranked against all the other individuals. Evaluation against the specified training set or even human evaluation (Kosorukoff 2001) can be used.

Several selection strategies exist. Most of them are based on fitness evaluation for each individuals and sorting the population by descending fitness values and differ in methods of parent selection from the sorted list. The most simple is the truncation or simple parent selection (Mühlenbein and Schlierkamp-Voosen 1993) when a proportion of the best-fit individuals is taken for reproduction. The more sophisticated one and one of the most popular is the proportionate or roulette-wheel selection (Bäck 1996) when each individual has a chance to be selected as parent depending on its fitness (fitness normalization of all individuals should be performed before that). Stochastic universal sampling (Baker 1987) developed of fitness proportionate selection and differs in a way that where fitness proportionate selection chooses several solutions from the population by

repeated random sampling, it uses a single random value to sample all of the solutions by choosing them at evenly spaced intervals. Tournament selection (Miller and Goldberg 1995) does not require sorting of all individuals. It involves running several tournaments (fitness comparisons) among a few individuals chosen randomly from the population and the winner of each tournament (individual that possesses the best fitness) is selected as a parent for the next population. Sometimes only a part of population with higher fitness is used, in other cases a concept of elitism is implemented, i.e. the best individuals are transferred to a new generation unchanged. In practical GA implementation some proportion of population with relatively low fitness is allowed to produce the offspring, since this keeps the diversity of population and prevents from convergence to local minimum.

GA is terminated in case one of the following TC is reached: necessary fitness is achieved; the maximum number of generations is reached; the budget is exhausted (e.g. computational resources); evolution stagnation (i.e. the fittest individual remains the same in a number of generations); combination of previous cases. If not, generation of a new population is started.

The new population generation stage employs two genetic operators: crossover and mutation. For each new individual to be produced, a pair of parents is selected from the array of possible parents. Offspring is generated by applying crossover and mutation operators and inherits some portion of features from both parents. The process of parent selection and offspring generation is continued till a new population of size n is generated. During the crossover two parents exchange by parts of their chromosomes. Many crossover techniques exist for individuals which use different data structures to store themselves. The most common one is the single-point crossover (Fig. 1.3).

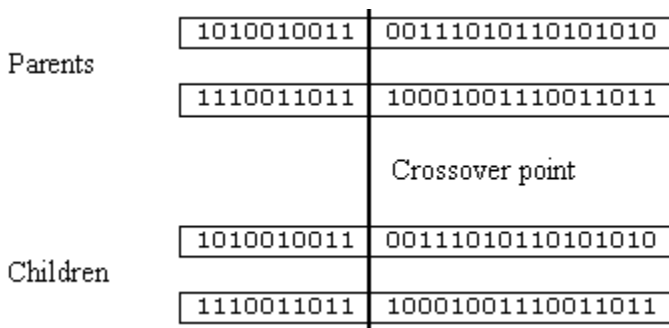


Fig. 1.3. Single point crossover

Other crossover methods also exist (two-point – employs two crossover points; cut and splice – single point, but its position in parent chromosomes is not the same and the offspring generated differ in length). Usually the crossover point position is selected randomly, unless order of genes in chromosome is important (e.g., travelling salesman problem). In that case additional ordering or continuity insurance methods are implemented.

The mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence (Fig. 1.4) and prevents convergence to a local minimum. Mutation rate is rarely high (usually $< 5\%$) since in that case GA would show low convergence rate or produce „noisy“ results.

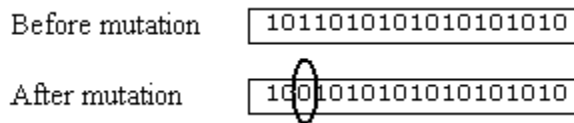


Fig. 1.4. Mutation in GA

While creating the GA based model there are three main tasks a researcher has to solve: firstly, it is the correct selection of chromosomes and genes representing solution; secondly – creation of the fitness equation (or fitness evaluation criteria, such as statistical evaluation, etc.), which evaluates the fitness of the solution (single chromosome), generated during evolution and selects the most appropriate solutions according to the selection strategy; and thirdly – defining the strategy for deriving the offspring and the composition of the next generation. It is obvious, that solution representation and offspring deriving strategy (also called operation conditions) is highly dependent on the analysed problem and its complexity, and the fitness function – on the evaluated criteria.

The power of a GA lies in its ability to exploit, in a highly efficient manner, information about a large number of individuals (Huang and Wechsler 1999). Other advantages GA gives are (Holland 1975; Tang *et al.* 1996):

- no prior assumptions about the solution are needed;
- works with a coding of parameter sets and not with the parameters;
- searches from a population of candidate solutions, not a single point;
- uses objective function information, not derivative or other auxiliary;
- uses probabilistic transition rules, not deterministic rules;
- results in a population of solutions instead of an individual solution.

GA approach has been proved to be effective in many areas such as business decision-making, bioinformatics and other (Stender *et al.* 1994).

Nowadays computer systems are facing increased number of security threats. Intruders can be divided into two groups, external and internal (Selvakani and Rajesh 2007). The unauthorized access is intended to steal or change information or to disrupt the valid use of the resource by an authorized user (Sindhu *et al.* 2009). The goal of a security system is to protect the most valuable assets of an organization: data and information. Different organizations will have very different security policies and requirements depending on their missions (Diaz-Gomez and Hougen 2005a). GAs are widely used in information security systems. When analysing GA application in information security systems it is necessary to note that there are three main problem areas they are used in: cryptology, optimisation of intrusion detection systems (IDS) and biometric authentication systems. Only a few authors use GAs for process modelling tasks, such as threat identification (Ahmad *et al.* 2010) or modelling product selection for information security technology system (Qi-Rong *et al.* 2008)

1.2. Genetic Algorithm Application in Cryptology

The application of GA to the field of cryptology is unique. Few works exist on this topic (Kumar and Ghose 2009). There are several main approaches of GA application in cryptology – improving strength of existing ciphers, their implementations or applications, creation of new GA inspired ciphers and cipher cryptanalysis or attacks against cryptographic systems in other words.

1.2.1. Cryptographic Genetic Algorithm Implementations

One of the first known attempts to employ GA's in cryptography was proposed by Tragha *et al.* in 2005 and later improved in 2006 (Tragha *et al.* 2006) and called ICIGA (Improved Cryptography Inspired by Genetic Algorithms). It is a symmetrical blocks ciphering system. System insures $O(n)$ complexity because all operations used are simple and they are applied randomly in ciphering. Furthermore, in the ICIGA system the user can choose the size blocks and the length key.

In (Almarimi *et al.* 2008) a new approach for encrypting real time data transmission is presented. Firstly, the pseudorandom binary sequence using Non-Linear Function Feedback Shift Register (NLFFSR) is generated. After that the generated pseudorandom sequence with crossover operator is used for encrypting the data. NLFFSR is a non-linear forward feedback shift register with a feedback function f and non-linear function g (Fig. 1.5).

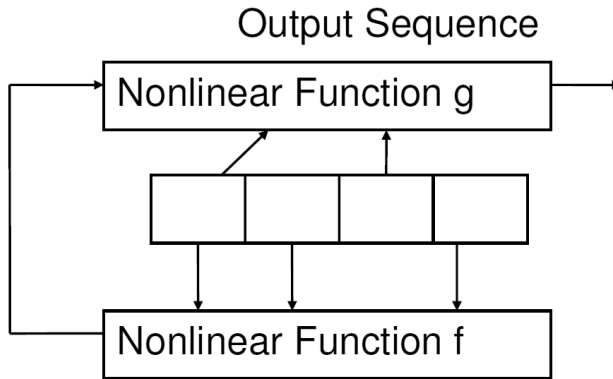


Fig. 1.5. A General model of 4 bit NLFFSR (Almarimi *et al.* 2008)

When the register is loaded with any given initial value (except 0 which will generate a pseudorandom binary sequence of all 0's), then pseudorandom binary sequence is generated with very good randomness and statistical properties. The only signal necessary for the generation of the binary sequence is a clock pulse. With each clock pulse a bit of the binary sequence is produced. The generated period of sequence by NLFFSR is the maximum, if the primitive polynomial is used. To design any stream cipher system, one needs to consider the NLFFSR with primitive feedback polynomials as the basic building blocks. The usefulness of these sequences depends in large part on there having nearly randomness properties. Therefore such sequences are termed as pseudorandom binary sequences. The balance, run and correlation properties of these sequences make them more useful in the selection of secret keys. The encrypting process emulates the working of the crossover operator using pseudorandom sequence and comprises the following steps:

1. Generation of the pseudorandom binary sequence using the NLFFSR as Z_n .
2. Taking mod 8 of the pseudorandom sequence to get the decimal value ranging from 0 to 7: $Z_n = \text{mod}(Z_n, 8)$;
3. Initialising $i=0$;
4. Taking two consecutive bytes of the data stream as A_1 and A_2 ;
5. Applying crossover to the two consecutive bytes of the data stream as B_1 and B_2 using the number Z_i .
6. Encrypting data as C_1 and C_2 , where

$$X_i = Z_i \oplus Z_i \ll 4$$

$$X_{i+1} = Z_{i+1} \oplus Z_{i+1} \ll 4$$

$$C_1 = B_1 \oplus X_i$$

$$C_2 = B_2 \oplus X_{i+1}$$

and $i=i+2$.

7. Repeating steps 4 to 6 until end of data.

The steps for decryption are just reversal of the encryption. The encrypted result of the representative image by this method are shown on Figure 1.6.

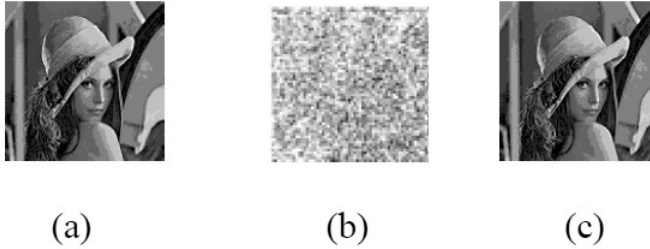


Fig. 1.6. Encryption / decryption example (Almarimi *et al.* 2008)

Authors state, that combining the concept of GA with the randomness properties of NLFFSR provides highly safe and reliable method of transferring secret information. The simulation results have indicated that the encryption results are completely chaotic by the sense of sight and very sensitive to the parameter fluctuation.

In (Wang, L. *et al.* 2006) the Advanced Encryption Standard (AES) encryption hardware implementation, using GA is proposed (Fig. 1.7).

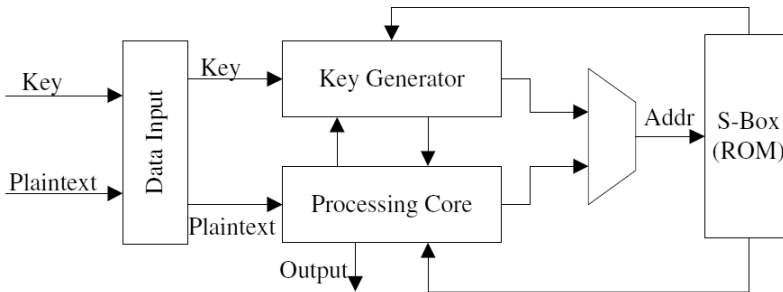


Fig. 1.7. GA-based hardware AES encryption scheme (Wang, L. *et al.* 2006)

AES – is a symmetric cipher, using data blocks of 128 bits length and 128, 196, or 256 bit encryption keys. Key Generator module generates the encryption keys by the means of GA. Bit combinations define the crossover point and mutation point and rate. Technological advances of the proposed method are the increased randomness of the encryption keys, usage of a new encryption key in each cycle, better resistance to cryptological attacks and effectiveness of CPU usage. The major minus of the solution proposed is its inadequacy to the AES standard.

1.2.2. Cryptological Genetic Algorithm Inspired Attacks

Application of GAs in cryptology is closely related with attack automation and effectiveness increase tasks (Delman 2004). There are over 10 GA-inspired attacks described against monoalphabetical, polialphabetical substitution, transposition and Knapsack ciphers. Some of the most typical examples are reviewed in this chapter. No effective GA-based attacks against modern symmetric (e.g. AES, DES) or asymmetric (e.g. RSA) are known or they are not more effective than the “brute-force” attack.

Attacks against Classic Cryptographic Ciphers

The first attack employing GA mechanisms was proposed in (Spillman *et al.* 1993). The monoalphabetic substitution cipher is attacked. Population of solutions represents the possible encryption keys and the fitness function (Eq. 1.1) is used to evaluate the individuals from the population. The function represents a normalized error value. Larger fitness values represent smaller errors. Sensitivity to small differences is increased by raising the result to the 8th power, while sensitivity to large differences is decreased by the division of the summation terms by 4.

$$Fit = \left(1 - \sum_{i=1}^{26} \left\{ |SF[i] - DF[i]| + \sum_{j=1}^{26} |SDF[i, j] - DDF[i, j]| \right\} / 4 \right)^8. \quad (1.1)$$

$SF[i]$ is the standard frequency of character i in English plaintext, $DF[i]$ is the measured frequency of the decoded character i in the ciphertext, SDF is the standard frequency for a diagram and DDF is the measured frequency. In case of 10–50 population size the exact key is found after 100 generations (approximately 1000 of possible key are evaluated), when population is larger than 100 individuals, 20 generations are needed on average. The defined optimal mutation rate is equal to 0.20. The main disadvantage of the method described is its inaccuracy (the exact key is not always found) and human interruption is needed for result correction.

Another attack proposed in 1993 is an attack against transposition cipher by Matthews (Matthews 1993). Its effectiveness rate seeks only 13.33 %. Results achieved by (Dimovski and Gligoroski 2003) against transposition ciphers are comparative and seek 13.25 %. (Clark 1994) describes the GA-based attack against the transposition/substitution cipher, but no efficiency evaluations are provided. Efficiency of the GA attack against Vingere cipher (Clark and Dawson 1997) was up to 90 %.

It can be noticed that GA-based attacks against classic cryptographic ciphers except one proposed by (Spillman *et al.* 1993) has lower efficiency rates compared to traditional ones, such as statistical analysis.

Attacks against Knapsack Ciphers

The most successful is GA method application against knapsack ciphers: “light”, Merkle-Hellman knapsack (Spillman 1993; Garg and Shastri 2007) and Chor-Rivest knapsack (Yaseen and Sahasrabuddhe 1999). “Light” and Merkle-Hellman knapsacks can be cracked by other analytical methods, but their efficiency is by rank lower than the GA-based methods. Chor-Rivest knapsack, based on discretionary logarithms, was considered to be secure if parameters are selected correctly. That means that the proposed GA-based attack is the only one known and successful except the “brute-force” method (Delman 2004).

The GA-based attack against Merkle-Hellman knapsack cipher is described in (Garg and Shastri 2007). Merkle-Hellman knapsack algorithm acts in the following way:

1. the “light” knapsack is selected, where each of the following elements is greater than the sum of all previous elements;
2. an integer m is selected, which is greater than the sum of whole knapsack;
3. integer w is selected, such that $\gcd(m, w) = 1$, i.e. m and w are relatively prime;
4. the inverse is calculated $w^{-1} = (w) \bmod(m)$;
5. sequence $A = w((A') \bmod(m))$ is constructed, i.e. $a_i = w((a'_i) \bmod(m))$. A is a Public key, A' and m, w, w^{-1} values – Private key.

GA-based attack against the Merkle-Hellman knapsack algorithm contains the following steps:

1. generation of initial population M of size 10–100;
2. individual solution fitness in population evaluation by Eq. 1.2.

$$Fitness = \begin{cases} 1 - \left(\frac{|T - Sum|}{Target} \right)^{\frac{1}{2}}, & \text{if } Sum \leq T; \\ 1 - \left(\frac{|T - Sum|}{MaxDiff} \right)^{\frac{1}{6}}, & \text{if } Sum > T, \end{cases} \quad (1.2)$$

where

$$Sum = \sum_{j=1}^n a_j m_j, \quad (1.3)$$

$$T = \sum_j a_j', \quad (1.4)$$

$$FSum = \sum_{j=1}^n a_j MaxDiff = \max\{T, FSum - T\}. \quad (1.5)$$

3. application of crossover and mutation operators;
4. repeating steps 2–3.

Operating conditions: text length (ASCII) < 100 symbols, „light“ knapsack – 8 elements, combined parent selection (25 % – elitism, 75 % – simple), population size – 75 individuals, crossover probability – 80 %, mutation probability – 11 %. The correct answer was achieved on average after 115 generations.

The Chor-Rivest Knapsack attack method described in (Yaseen and Sahasrabudde 1999) uses the GA in the following way: input to the GA is $A = (a_1, \dots, a_n)$, a knapsack of positive integers, and c , an integer representing the cipher. The output is $M = (m_1, \dots, m_n)$, a binary vector such that $c = A \cdot M$. The binary vector A is the chromosome that represents the solution message M . The length of the vector is n , equal to the size of the knapsack. $a_i = 1$ if the element has been chosen and 0 otherwise. The initial population is created randomly, such that each n -bit binary chromosome contains exactly h 1's, and h is the degree of the monic irreducible polynomial $f(x)$ over Z_p . Fitness is evaluated by Eq. 1.4:

$$Fit = 1 - \sqrt{\frac{|Sum - Target|}{FullSum}}, \quad (1.6)$$

where *FullSum* is the sum of all the components in the knapsack. To assist the fitness function, the chromosomes with a Hamming distance of 2 and 4 from a target were the ones used. 30 individual population size is defined. Experiments, both with and without the Hamming distance technique were carried out. Of the 120 experiments done in each case, approximately one third of the experiments without the Hamming distance technique did not reach a solution even after 5000 generations.

1.3. Genetic Algorithms Approach in Intrusion Detection Systems

Intrusion Detection (ID) is the process of identifying and responding to malicious activity targeted at computing and networking resources (Amoroso 1999). It is very important to emphasize that ID is a process, since it must involve technology, people, and tools (Bobor 2006). The study of intrusion detection systems (IDS) provides many challenges. In particular the environment is forever changing, both with respect to what constitutes normal behaviour and abnormal behaviour (Song *et al.* 2003). While application of GAs in Antivirus systems is already in the production and commercialisation phase, e.g. ESET NOD32 Antivirus uses proactive detection technology to identify malware which relies on advanced heuristics as well as generic and genetic algorithms to detect new malware and new mutations of known malware (ESET 2009) application of GAs in IDS systems is more in the scientific research and prototype testing phase, but the results seem promising.

1.3.1. Review of the IDS Technology

According to (Proctor 2000) IDS generated messages can be classified to the following categories, part of which can lead the IDS operator to the correct decision, and another – to the wrong decisions, regarding the attack status against the information system:

- Positive – The system is alarming (true or false);
- Negative– The system is not alarming (true or false);
- True-positive – An alarm was generated in the presence of a condition that should be alarmed;

- False-positive – An alarm was generated and no a condition is present to warrant one;
- True-negative – An alarm was not generated and there is no condition present to warrant one;
- False negative – An alarm was not generated and a present condition should be alarmed.

There are two main categories of IDS: anomaly detection and misuse detection. Anomaly detection systems seek to identify deviations from normal behaviour models, which are built from large training data sets. Misuse detection systems compare the system use behaviour with signatures extracted from known attacks; a match with a high confidence is considered an attack. These two kinds of systems have their own strengths and weaknesses. The former can detect novel attacks but, in general for most such existing systems, have a high false alarm rate because it is difficult to generate practical normal behaviour profiles for protected systems. The latter can detect known attacks with a very high accuracy via pattern matching on known signatures, but cannot detect novel attacks because their signatures are not yet available for pattern matching (Stein *et al.* 2005).

The IDS, according to the environment they are operating, can be classified into: Host-based IDS (HIDS) and Network-based IDS (NIDS). HIDS is installed on a host and the IDS process analyses information generated by the applications and the operating system. It is usually based on revision of logs as application logs, event logs, and kernel logs (Endorf *et al.* 2003). NIDS analyses information from the network traffic and can be considered as a network analyser with an attack signature module. The new trend on the market today is to offer so called Intrusion Prevention System (IPS) which is more or less network based IDS with an inbuilt blocking option (Endorf *et al.* 2003).

There are three fundamental components in IDS's: sensors, agents and management console. Sensors collect data from environment and send it to the agents. The agent contains processes that analyse data, which have been sent from sensors. The last component is management component that provides "an executive or master control capability of IDS" (Endorf *et al.* 2003).

The aim of IDS developers is minimization of false alarms or messages. Other urgent tasks nowadays are: automation of signature generation, response rule automatic generation and attack classification tasks. Different technologies are used to solve them such as Artificial Intelligence (AI), machine learning techniques, decision trees, genetic and other evolutionary algorithms, neural networks, other methods or method combinations (Bobor 2006; Stein *et al.* 2005).

1.3.2. Genetic Algorithm Application Research Trends and Neuro-Genetic Approach

One of the first evolutionary approach applications in IDS systems attempts was proposed in 1995 in (Crosbie and Spafford 1995). The system should be based on the multiagent technology and genetic programming. In the system each agent should monitor only one audited network data parameter. The method advantage is possibility to use a lot of autonomous agents, but their communication is very complex and system training would require high time expenses if agents are not correctly initialised (Selvakani and Rajesh 2007).

Bridges (Bridges and Vaughn 2000) describes the method, which integrates fuzzy logic data mining technologies and GAs into HIDS and NIDS systems. Other authors (Alanni and Sundarajan 2009) propose using the GA for reliable Denial of Service (DOS). (Diaz-Gomez and Hougen 2005a,b) analyse application of multiobjective GA in autonomous Off-line IDS systems.

The main research on GA application in IDS systems is done in area of GA application for automatic IDS rules generation optimisation/training of Artificial Neural Networks (ANN), so-called Neuro-Genetic approach. ANN – is self-training systems, that imitates the human brain structure and operating principles, that has become one of the most popular AI methodics. ANN is formed out of interconnected and interacting neurons, that are allocated into layers: input layer (accepts information for processing), hidden layers (processes the received information) and the output layer (presents the processing results). One of the most important ANN qualities is ability to approximate any non-linear functions without any prior assumption about the processed data. ANN can process the changing information and adapt to changes, which are frequent in computer networks (Sindhu *et al.* 2009), that is why application of ANN in IDS systems are considered to be perspective (Bobor 2006). ANN usage requires time and computational resources that are highly dependant on ANN size.

One of the most perspective research areas – GA application for ANN network size optimisation (Ibrahim and Chris 1994). (David 1995) analyses the ANN weight selection problem with the help of GAs and points that GA usually provides better results than compared with non-optimised ANN.

There is a number of experimental IDS systems proposed, based on the hybrid Neuro-Genetic approach. (Bobor 2006) proposes the theoretical IDS model called Efficient Artificial Network Genetic Algorithm based Intrusion Detection System. Authors of (Sindhu *et al.* 2009) apply the GAs in the HIDS system, where GAs are used for ANN training, which is used for anomalous user behaviour detection. In a training phase GAs are used for knowledge base formation. Training is based on data sets with normal and abnormal behaviour samples. In (Diaz-Gomez and Hougen 2005a) also the Neuro-Genetic based HIDS system is proposed. Authors base the system on (Me 1998) work.

1.3.3. GA Application for Automatic IDS Rule Generation

Rules stored in the IDS rule database are in fact represented as conditional sentences (Sinclair *et al.* 1999):

$$\text{if \{ condition \} then \{ act \},} \quad (1.7)$$

where variable *condition* usually reflects different network traffic parameters, such as source and destination Internet protocol (IP) addresses, port number, connection time, etc., and *act* defines, what actions should be taken by the IDS system, e.g. alarm IDS operator, break the connection, etc. The sample rule can look like (Fig. 1.8):

```
if {the connection has following information:
source IP address 124.12.5.18; destination IP
address: 130.18.206.55; destination port
number: 21; connection time: 10.1 seconds }
then {stop the connection}
```

Fig. 1.8. Sample IDS rule (Sinclair *et al.* 1999)

Li (2004) proposes GA usage for such rule generation. Classified historical data is used as fitness evaluation criteria. The solution chromosome is composed out of 57 genes. For representation simplicity IPs are depicted in HEX format:

- Source IP address;
- Destination IP address;
- Source Port Number;
- Destination Port Number;
- Duration;
- State;
- Protocol;
- Number of Bytes Sent by Originator;
- Number of Bytes sent by Responder.

Sample chromosome is presented on Fig. 1.9. After algorithm run the best rules are selected and uploaded to the IDS database.

(d, 1, 0, b, -1, -1, -1, -1, 8, 2, 1, 2, b, -1, -1, -1, 4, 2, 3, 3,
5, 0, 0, 0, 8, 0, 0, 0, 0, 0, 4, 8, 2, 1, 1, 2, 0, 0, 0,
0, 0, 0, 7, 3, 2, 0, 0, 0, 0, 0, 0, 3, 8, 8, 9, 1)

Fig. 1.9. Chromosome representing IDS rule (Li 2004)

Selection mechanism uses the classified historical data. At first the value of the *Outcome* variable is calculated:

$$Outcome = \sum_{i=1}^{57} Matched * Weight_i, \quad (1.8)$$

where variable *Matched* (possible values 0 or 1) shows, if the corresponding gene has revealed itself in historical data. Variable *Weight* defines the conditional gene's weight, which changes according to the scheme, presented on Fig. 1.10.

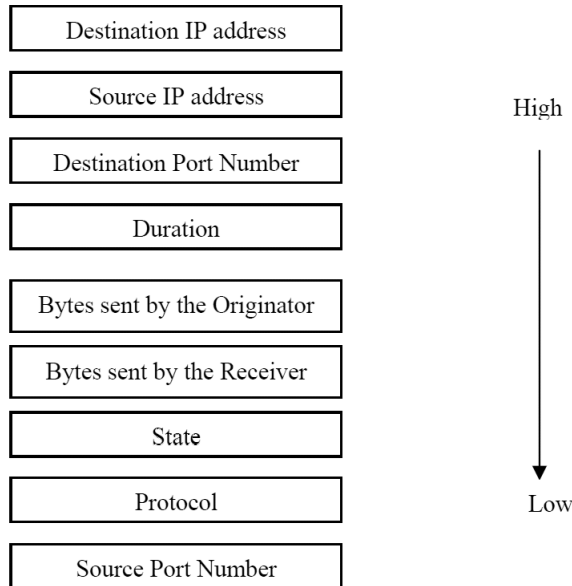


Fig. 1.10. Order of variable *Weight* change order (Li 2004)

After that the modulus difference of the variable *Outcome* and *Suspicious_level* is calculated (Eq. 1.9).

$$\Delta = |\text{outcome} - \text{suspicious_level}|. \quad (1.9)$$

Suspicious level is defined according to historical data. Later the chromosome's penalty points are calculated:

$$\text{penalty} = \frac{\Delta * \text{ranking}}{100}, \quad (1.10)$$

where variable *ranking* defines the difficulty of intrusion detection.

Eq. 1.10 is used to calculate the overall chromosome's fitness, which may vary from 0 to 1:

$$\text{fitness} = 1 - \text{penalty}. \quad (1.11)$$

Similar research is done by authors of (Selvakani and Rajesh 2007). They have optimised the (Li 2004) proposed model for Remote2Local (R2L) and Denial-of-Service (DOS) attacks. The model was tested on the DARPA test intrusion database KDDCUP'99 and has shown low error rate seeking only 0.1 %.

1.4. Genetic Algorithms Application Analysis in Biometric Systems

In today's world there is a growing concern regarding identity theft, national security, and on-line terrorism (Abbazio *et al.* 2009). Biometrics are seen by many researchers as a solution to a lot of user identification and security problems nowadays (Jain *et al.* 1999). Biometric identification is any automatically measurable, robust and distinctive physical characteristic or personal trait that can be used to identify an individual or verify the claimed identity of an individual (Woodward 2003). Biometric science utilizes the measurements of a person's behavioural characteristics (keyboard strokes, mouse movement) or biological characteristics (fingerprint, iris, nose, eyes, jaw, voice pattern, etc). It is these measurements that are then used to create a reference template, which the recognition software uses to identify or authorize an individual as the person they claim to be (Abbazio *et al.* 2009). The most common biometric method to identify the individuals is through fingerprint (Uludag *et al.* 2004) recognition. In recent years, there has been significant interest in using other biometrics for identifying individuals. These include techniques that rely on: DNA, hand geometry, palm print, face (both optical and infrared), iris, retina, signature, ear shape, odor, keystroke entry pattern, gait, and voice (Pankanti *et al.* 2000). Other emerging biometrics such as ear force fields (Hurley *et al.* 2005), heart signals

and electroencephalogram (EEG) (Biel *et al.* 2001.) or brain signals (Ravi and Palaniappan 2006.) have also been proposed in recent years.

Biometric system implementation for person identity verification purposes, terrorist acts prevention measures, authentication process simplification in computer systems and many other tasks has raised significant attention to reliability and efficiency of biometric systems. Modern biometric systems still face many reliability and efficiency related issues such as reference database search speed, errors while recognizing of biometric information or automating biometric feature extraction. These problems will be discussed later in more detail when discussing specific biometric techniques and GA application for overcoming these problems. Scientific investigations show that application of evolutionary algorithms may significantly improve biometric systems.

1.4.1. Genetic Algorithm Application for Fingerprint Biometric Information Processing

Fingerprint has been researched the longest period of time and shows the most promising future in real-world applications (El-Emary and Abd El-Kareem 2008). A fingerprint is formed of a group of curves. The most common characteristics include endpoints and bifurcations called as minutiae (Pires *et al.* 2005). Due to the persistence and individuality of fingerprints, fingerprint recognition has become a popular personal identification technique (Pankanti *et al.* 2002). However, because of the complex distortions among the different impressions of the same finger, fingerprint recognition is still a challenging problem (El-Emary and Abd El-Kareem 2008). Other real world problems related with fingerprint usage in biometric systems are fingerprint registration (El-Emary and Abd El-Kareem 2008), creation and usage of fingerprint test databases (Jain *et al.* 1997) and many others.

Genetic Algorithm Application for Fingerprint Registration

According to (El-Emary and Abd El-Kareem 2008) image registration algorithms fall within two categories: area based methods and feature based methods. The original image is often referred to as the reference image and the image to be mapped onto the reference image is referred to as the target image. For area based image registration methods, the algorithm looks at the structure of the image via correlation metrics, Fourier properties and other means of structural analysis. Feature based methods, tune their mappings to the correlation of image features: lines, curves, points, line intersections, boundaries, etc. Image registration algorithms can also be classified according to the transformation model used to relate the reference image space with the target image space. The first category of transformation models includes linear transformations (Fig. 1.11), which are a combination of translation, rotation, global scaling, shear and per-

spective components. Linear transformations are global in nature, thus not being able to model local deformations. The second category includes “elastic” or “nonrigid” transformations. These transformations allow local warping of image features, thus providing support for local deformations. Nonrigid transformation approaches include polynomial warping, interpolation of smooth basis functions (thin-plate splines and wavelets) and physical continuum models.

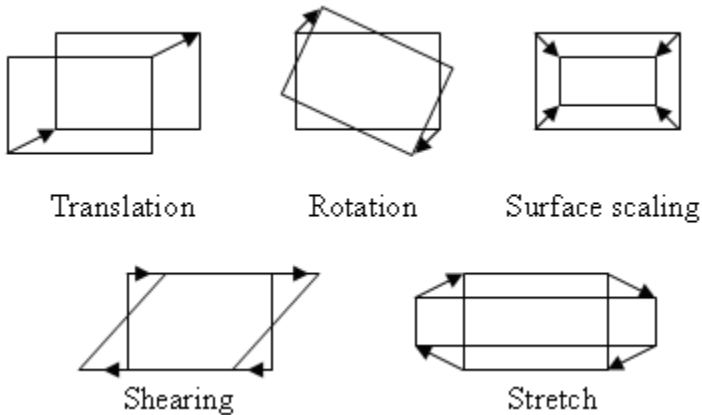


Fig. 1.11. Elementary geometric transforms for planar surface elements (El-Emary and Abd El-Kareem 2008)

In paper (El-Emary and Abd El-Kareem 2008) GA usage for registration of two images is proposed. Two images cannot be compared if one of these images is translated or rotated by some unknown angle. To obtain the registration, the first image must be transformed until gets best matching with the second image. However, the conventional registration methods suffer from misregistration because of the difference in rotation angles. Authors investigate the use of GA in image registration, since the GA is accurate, very fast and now is used very frequently. Image registration is described as the process of overlaying two or more images of the same scene taken at different times, from different viewpoints and/or by different sensors. Search-based image registration methods utilize an iterative procedure to improve the initial guess of the unknown transform parameters. Registration process mainly consists of determining the unknown transformation parameters required to map the input image to the reference image in order to compare and analyse both in a common reference frame.

Automatic registration of the two images requires determining transformation applied and a measure of similarity. The applied transformation is given by affine transformation where an affine transform is a linear coordinate transfor-

mation that includes the elementary transformations. The affine transform can be expressed by vector addition and matrix multiplication as:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}. \quad (1.12)$$

The affine transform can be rewritten by the following:

$$X = t_x + a_{11}x + a_{12}y; \quad (1.13)$$

$$Y = t_y + a_{21}x + a_{22}y, \quad (1.14)$$

where x, y are the two Cartesian spatial dimensions and a_{ij} and t_x, t_y are the adjustable parameters whose values are to be estimated under the following constraints: -size of image = $t_x * t_y$ = size of image and $-1 = a_{ij} = 1$.

The measure of fitness, or success of the transformation is based simply on the point-by-point absolute difference between the two images:

$$f = 1/m \bullet \sum_{j=1}^m |\bullet c(j)|, \quad (1.15)$$

where m -number of points considered; $\bullet c$ -greyscale intensity difference between the same point in the first and the transformed second image. Since the two images may well have different overall intensity distributions an additional unknown parameter a_{31} is required to equalize the distributions:

$$\bullet c(j) = c_1(j) - a_{31}c_2(j), \quad (1.16)$$

where $c_1(j)$ and $c_2(j)$ are the individual greyscale intensities of pixel j for the two images (a_{31} is the same for all j).

The similarity measure provides a quality index of each solution. The choice of the similarity measure is closely related to the selected feature since it measures the similarity between the same features in the reference and the transformed input image typically, similarity measures are the correlation. The sum of absolute differences, the root mean square. The normalized cross-correlation function (CC) can be written as:

$$CC = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} ((R_{ij} - R) \times (I_{ij} - I))}{\sqrt{\sum \sum (R_{ij} - R)^2 \times \sum \sum (I_{ij} - I)^2}}. \quad (1.17)$$

The technique was applied to get some of rotated figures with some known rotation angles. Authors state that they could retrieve the rotation angles successfully up to an accuracy approaching 100 % (Percentage relative error: 0–10%). The time of execution was reasonable. It was noticed that the wider the rotation angle is, the less the relative error will be, i.e. the rotation angle is inversely proportional to the relative error. The method can be suitable for fingerprints registration in large databases. This recommendation comes from the fact that the time consumed in finding the rotation angle is so minute.

Genetic Algorithm Based Fingerprint Matching Techniques

Fingerprint matching depends on the comparison of the characteristic of local ridges and their relationships. Most of the existing automatic fingerprint verification systems are based on minutiae features (ridge bifurcation and ending). Such systems first detect the minutiae (Fig. 1.12) in a fingerprint image and then match the input minutiae set with the stored template (Jain *et al.* 1997; Maio and Maltoni 1997; Prabhakar *et al.* 2003).

Extracting minutiae from fingerprint images is one of the most important steps in automatic fingerprint identification system. A number of publications is available on the topic GA and GP application for fingerprint matching.

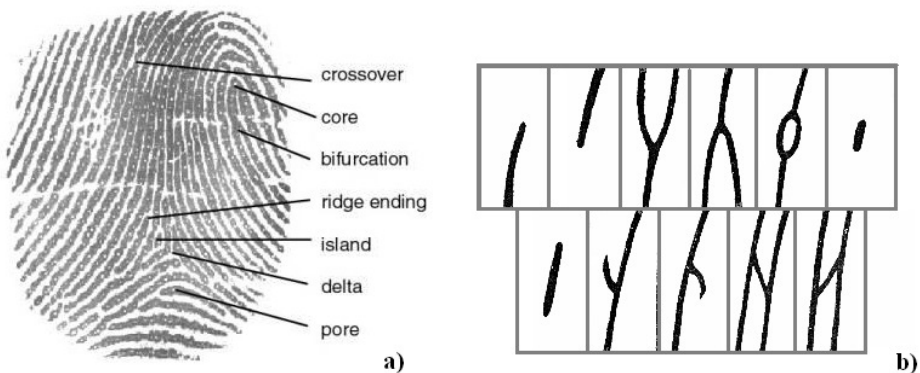


Fig. 1.12. a) Fingerprint sample with minutiae marked (Fingerprint 2004) b) minutiae types (Fingerprint 2005)

Authors of (Tan and Bhanu 2002) propose a fingerprint matching approach based on GA, which finds the optimal global transformation between two different fingerprints. In order to deal with low quality fingerprint images, which introduce significant occlusion and clutter of minutiae features, they design the fitness function based on the local properties of each triplet of minutiae. The experimental results on National Institute of Standards and Technology finger-

print database, NIST-4, not only show that the proposed approach can achieve good performance even when a large portion of fingerprints in the database are of poor quality, but also show that the proposed approach is better than another approach, which is based on mean-squared error estimation.

In (Scheidat *et al.* 2006) parameter optimisation for biometric fingerprint recognition using GA is discussed. The created application is planned so that it can be used without great effort for different biometric systems. Instead of estimating the required parameters as in the case of some methods, here they are determined with the help of GA. The test database used consisted of 1200 fingerprints of 12 people. For the confirmation of the results, which were found out with this test set, the databases of the Fingerprint Verification Contests of the years 2000, 2002 and 2004 were examined in addition. In the best case an improvement in the recognition performance of 38 % was observed.

A Kohonen self-organizing neural network embedded with GA for fingerprint recognition was proposed in (Sudarshana Reddy and Subba Reddy 2004). The GA was embedded to initiate the Kohonen classifiers. By the proposed approach, the ANN learning performance and accuracy were greatly enhanced. In addition, the GA could successfully avoid the ANN from being trapped in a local minimum. The proposed method was tested for the recognition of fingerprints. The results were promising to applications.

(Pires *et al.* 2005) presents a GA aimed at optimising the transformation between two sets of minutiae extracted from two different fingerprints belonging to the same finger. Experiments with FVC2004 image database were performed. Based on preliminary results, one may conclude that the system developed in this work obtained good accuracy rate in the verification process, considering the restriction variation imposed by the threshold.

While most fingerprint matching systems rely on the distribution of minutiae on the fingertip to represent and match fingerprints and the ridge flow pattern is generally used for classifying fingerprints, it is seldom used for matching. The (Girgisa *et al.* 2007) article describes a new method for fingerprint matching based on lines extraction and graph matching principles.

Genetic Algorithm Application for Fingerprints Image Generation

Constructing a fingerprint database is important to evaluate the performance of automatic fingerprint recognition systems. The construction of fingerprint databases requires an enormous effort so that, in practice, it is often too costly or the resulting database is incomplete or unrealistic (Maltoni 2004).

Because of the difficulty in collecting fingerprint samples, there are only few benchmark databases available. Moreover, various types of fingerprints are required to measure how robust the system is in various environments. (Cho *et al.* 2007) presents a method that generates various fingerprint images automati-

cally from only a few training samples by using the GA. Fingerprint images generated by the proposed method include similar characteristics of those collected from a corresponding real environment. When a target environment is given, the proposed method constructs a set of filters that modify an original image so as to become similar to that collected in the environment. A proper set of filters is found by the GA, where fitness evaluation is conducted using various statistics of fingerprints to measure the similarity. In the initialisation step, it sets basic parameters including the population size, the maximum number of generations, the length of chromosomes, etc. The length of chromosomes means the size of a filter composed, where each gene in the chromosome represents the corresponding filter in the pool of filters. Only a few samples are required to calculate several statistics for the target environment to evaluate a chromosome. The fitness of a chromosome is obtained from the similarity between a few real images from the target environment and images generated after filtering. The value of each gene means a filter to apply for images of the training database.

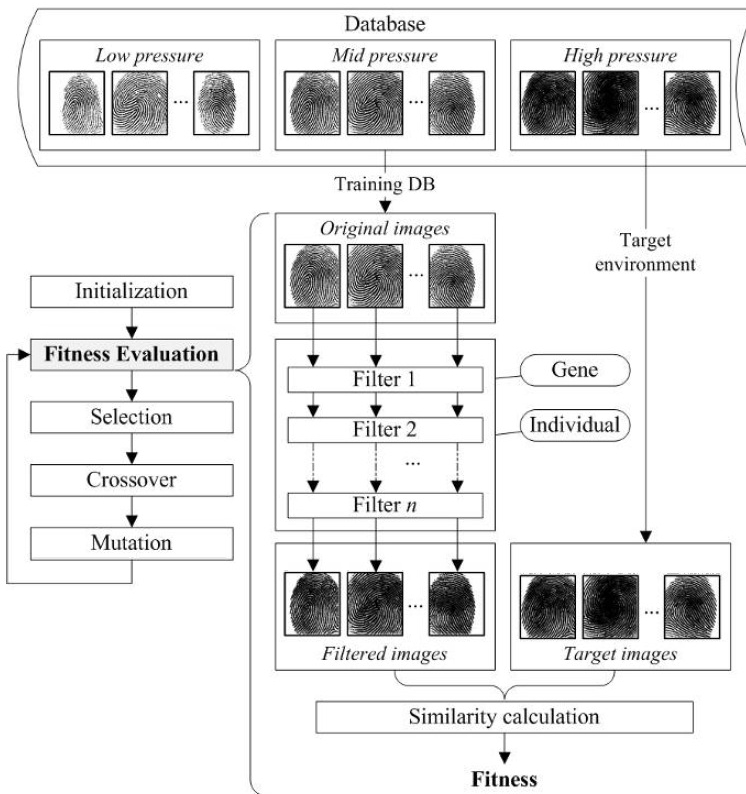


Fig. 1.13. GA-based fingerprint image generation method (Cho *et al.* 2007)

Popular image filters (Brightness (3 values), Contrast (3 values), Stretch, Equalize, Logarithm, Blur (6 masks), Sharper (4 masks), Median (10 masks), Morphology(10 masks) Erosion, Dilation, Opening, Closing) are used to produce similar effects in real environments. The order and type of filters used in the filter set are determined by the GA, because it is practically impossible to test all the cases of composition. The fitness of a filter set is estimated by measuring the similarity between fingerprints collected from the target environment and images generated by the composite filter. Several representative features of fingerprints, such as the mean and variance of images, directional contrasts, average ridge thickness and interval, singularities and minutiae, are used to design the fitness evaluation function, in which weights are heuristically determined:

$$\begin{aligned}
 fitness(i) = & w_1 \times (mean_i - mean_{target}) \\
 & + w_2 \times (variance_i - variance_{target}) \\
 & + w_3 \times \sum_{j=1}^4 (contrast_i^j - contrast_{target}^j) \\
 & + w_4 \times (thickness_i - thickness_{target}) \\
 & + w_5 \times (interval_i - interval_{target}) \\
 & + w_6 \times \sum_{ce \text{ singularityType}} (singularity_i(c) + singularity_{target}(c)) \\
 & + w_7 \times \sum_{ce \text{ minutiaeType}} (minutiae_i(c) + minutiae_{target}(c)).
 \end{aligned} \tag{1.18}$$

The statistics of the target environment is calculated from the environment database and all the values are normalized from 0 to 1. The generated images (Fig. 1.14) might be used to evaluate the fingerprint recognition systems.

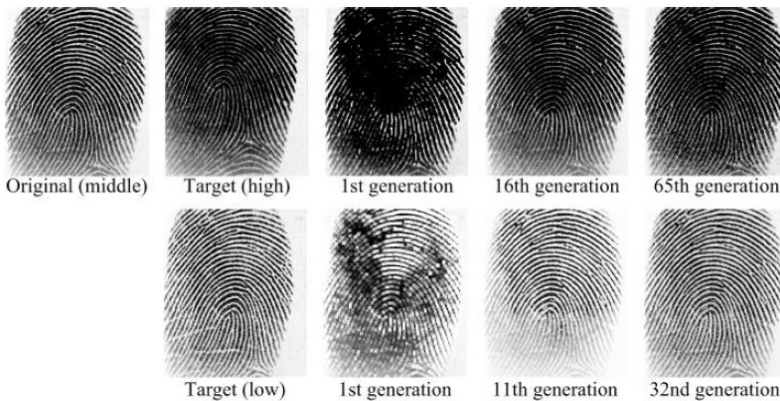


Fig. 1.14. Generated fingerprint images (Cho *et al.* 2007)

The usability of the proposed method was verified by comparing the fingerprints collected from real environments with those generated. Moreover, the proposed method has the applicability to the fingerprint image enhancement by modifying the fitness evaluation module.

1.4.2. Genetic Algorithm Application for Face Feature Processing

Face recognition is a biometric authentication method that has become more significant and relevant in recent years. It is becoming a more mature technology that has been employed in many large scale systems such as Visa Information System, surveillance access control and multimedia search engine (Mohamad 2009). Facial feature processing plays an important role in law enforcement forensic investigation (Brunelli and Poggio 1993), low bit video coding (Chuang *et al.* 2000), security access control systems (Lin and Ling 1999) and other applied and security systems. Generally, there are three categories of approaches for recognition, namely global facial feature, local facial feature and hybrid feature (Mohamad 2009). Systems using facial features can also be classified by source information used – 2D or 3D images and static images or video. Significant research has been done in EA application area for system using facial features improvement. Kirchberg *et al.* (2002) present an optimisation approach that creates and successively improves Hausdorff Distance-Based Face Localization model by means of GA. To speed up the process and to prevent early saturation the researchers use a special bootstrapping method on the sample set and test several initialisation functions. In (Akashi *et al.* 2007) the high tolerance in human head movement and real-time processing that are needed for many applications, such as eye gaze tracking, is discussed. Template matching is used with GA, in order to overcome these problems. A high speed and accuracy tracking scheme using Evolutionary Video Processing for eye detection and tracking is proposed. Usually, a GA is unsuitable for a real time processing, however, authors state that they have achieved real-time processing. The generality of this proposed method is provided by the artificial iris template used. In simulations the 97.9 % eye tracking accuracy and an average processing time of 28 milliseconds per frame was achieved. Further we provide a detailed technical analysis of two research papers on GA application for 2D image and 3D image processing.

2D Face Image Processing by Genetic Algorithms

An automatic facial feature extraction method is presented in (Yen and Nithianandan 2002). The method is based on the edge density distribution of the image. In the pre-processing stage a face is approximated to an ellipse, and GA is applied to search for the best ellipse region match. In the feature extraction stage, GA is applied to extract the facial features, such as the eyes, nose and mouth, in

the predefined sub regions. The normal process of searching for the features is computationally expensive, therefore GA is used as a search algorithm.

All the images used in experiments were head and shoulder images in a frontal view. Smoothing filters (median) were used for noise reduction. The face segmentation process was proceeded under the assumption that the face region can be approximated by an ellipsoid. This method works well even under the environments when the background is complex and the face contains extra features such as spectacles, beard and etc. The general processing scheme is presented on Fig. 1.15.

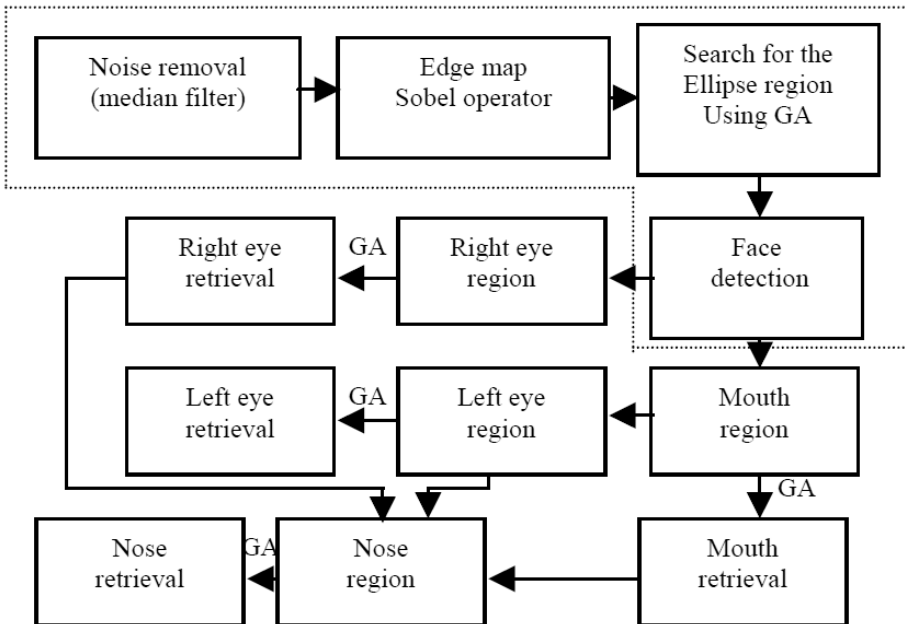


Fig. 1.15. Block diagram of 2D face feature extraction method (Yen and Nithianandan 2002)

Each chromosome in the population during the evolutionary search has five parameters genes, the center of the ellipse (x and y), x directional radius (r_x), y directional radius (r_y) and the angle (θ). The chromosome in binary form for each parameter is coded as shown in Fig. 1.16.

x-8bits	y-8bits	r_x -8bits	r_y - 8bits	θ -7bits
---------	---------	--------------	---------------	-----------------

Fig. 1.16. Chromosome for 2D face image segmentation (Yen and Nithianandan 2002)

The fitness function is defined by the number of edge pixels in the approximated ellipse like face to the actual number of pixels in the actual ellipse. The ratio is large when both ellipses overlap perfectly. It is commonly assumed in literature that the ratio of the length to breath of the face is 1.5:1, therefore the same ratio is used to obtain the face area once the ellipse region is located. In the case of multiple faces in the image, faces are located until a threshold is satisfied. The threshold is based on the fitness value used to locate the faces. In the feature extraction stage, GA is used to search for the global maximum point when the template best matches the feature. The chromosome (Fig. 1.17) represents the position of the feature in the x and y direction.



Fig. 1.17. Chromosome for 2D face image feature extraction (Yen and Nithianandan 2002)

The fitness is evaluated in terms of the density of the template. The best template is selected when the fitness is maximized. The fitness function F is defined as:

$$F = \frac{1}{m \times n} \sum_{x=1}^m \sum_{y=1}^n T(x, y), \quad (1.19)$$

$$\text{where } \begin{cases} T(x, y) = 1, & \text{if the pixel is white;} \\ T(x, y) = 0, & \text{if the pixel is black,} \end{cases}$$

and T is the template, (x, y) are the coordinates of the template, and m, n is the size of the template.

Initially the population is chosen randomly. In each generation 20 % of the population is considered for the reproduction. The Roulette Wheel selection scheme is applied in the selection process.

The proposed facial feature extraction approach has been validated with a large number of images. Some of the images contained more than one person, while others with person oriented at an angle. Simulation results showed that the facial features were extracted successfully. GA was able to search effectively and reduce computational complexity, therefore reduce the search time. The facial features were extracted even in the presence of artificial noise.

3D Face Image Processing by Genetic Algorithm

Research on 3D face recognition has been intensified recently due to the significant advances of the 3D imaging technology. Most of the research focuses on the investigation of 3D range data obtained by a 3D scanner. Although 3D capture systems provide highly accurate 3D face information, it is not trivial to process the large amount of facial surface data. In model described in (Sun and Yin 2007) each individualized facial model consists of 2953 vertices, 3D face model database is generated using 105 pairs of face images from 40 subjects. For each subject, there are two or three pairs of frontal and profile images, which were taken under different imaging conditions. In order to better characterize 3D features of the facial surface, each vertex on the individual model is labelled by one of eight label types. Therefore, the facial feature space is represented by a set of labels. A cubic approximation method is explored to estimate the principal curvatures of each vertex on the model. Then the eight typical curvature types (i.e., convex peak, convex cylinder/cone, convex saddle, minimal surface, concave saddle, concave cylinder/cone, concave pit and planar) are categorized according to the relation of the principal curvatures. Fig. 1.18a shows the labelled original feature space. Among the set of labels, only the labels located in certain regions are of the most interest.

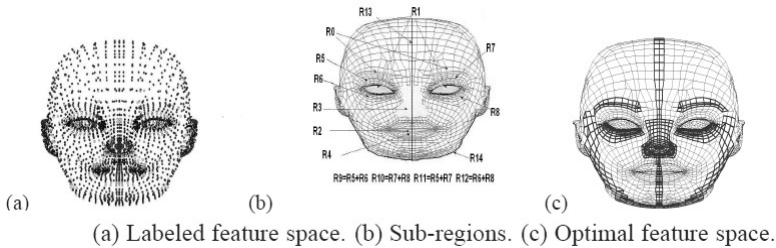


Fig. 1.18. 3D facial feature extraction (Sun and Yin 2007)

Some non-feature labels could be noises that may blur the individual facial characteristics. Therefore, a feature screening process to select features in order to better represent the individual facial traits for maximizing the difference between different subjects while minimizing the size of the feature space is applied.

In order to select the optimal features, the face model is partitioned into 15 sub-regions based on their physical structures (there are overlaps between some of the regions, Fig. 1.18b). Since not all the sub-regions contribute to the recognition task, and not all the vertices within one sub-region contribute to the classification, selection of the best set of vertex labels and the best set of sub-regions is needed. The purpose of the feature selection is to remove the irrelevant or re-

dundant features which may degrade the performance of face classification. The GA is used successfully to address this type of problem.

The GA-based method selects the components that contribute the most to the face recognition task. The procedure for the GA-based feature selection consists of two parts:

1. vertices selection in each sub-region and
2. the integration of sub-regions.

In the first stage, equal error rate (EER) is used as the fitness function and those resulting in a higher EER rate are selected as good features. In the second stage, the sub-regions whose EER rate are higher than the mean EER rate value are integrated together as the final optimal feature spaces. Fig. 1.18c shows the optimised feature space (dark colour).

Two sets of databases were tested. One set consisted of 105 3D facial models and about 92 % rank-four correct recognition rate was achieved. The other set had 387 models, the correct recognition rate was up to 87.6 %. The experimental results showed that the features obtained from the 3D individualized model is feasible to classify and can be used to identify individual faces.

1.4.3. Other Biometric Information Processing by Genetic Algorithms

In this chapter a short overview of other biometric information processing techniques by the means of GAs, which seem either promising for practical implementation in the nearest future (e.g., speaker recognition) or interesting from the scientific and novelty point of view (e.g., brain signals and activity style) is provided.

Neuro-Genetic Approach for Speaker Recognition

Speaker identification is one of the most important areas where biometric techniques can be used. Most published works in the areas of speech recognition and speaker recognition focus on speech under the noiseless environments and few published works focus on speech under noisy conditions. Learning systems in speaker identification that employ hybrid strategies can potentially offer significant advantages over single-strategy systems. In paper (Islam and Rahman 2009), Neuro-Genetic Hybrid algorithm with cepstral based features has been used to improve the performance of the text dependent speaker identification system under noisy environment.

At first algorithm performs the acquisition of speech utterances from speakers. To remove the background noises from the original speech, Wiener filter is used. Then the detection algorithm is used to detect the start and end points from

each speech utterance, after which the unnecessary parts are removed. Pre-emphasis filtering technique is used as a noise reduction technique to increase the amplitude of the input signal at frequencies where signal-to-noise ratio (SNR) is low. The speech signal is segmented into overlapping frames. The purpose of the overlapping analysis is that each speech sound of the input sequence would be approximately centered at some frame. After segmentation, windowing technique is used. Features are extracted from the segmented speech. The extracted features are then fed to the Neuro-Genetic hybrid techniques for learning and classification. Fig. 1.19 shows the working process of neuro-genetic hybrid system.

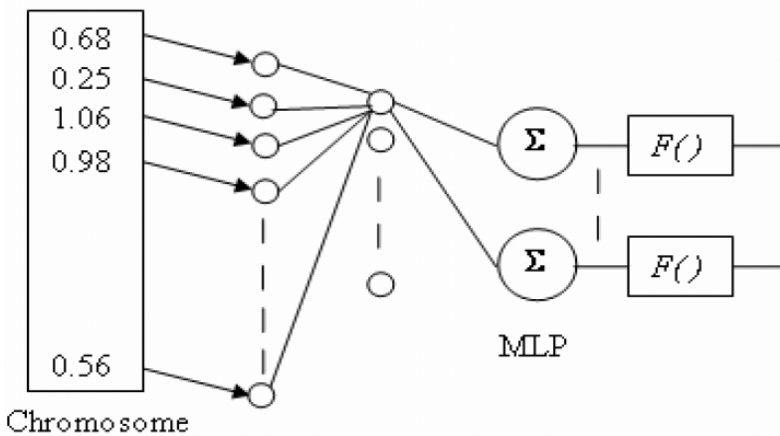


Fig. 1.19. Learning and recognition model for the Neuro-Genetic hybrid system (Islam and Rahman 2009)

The structure of the multilayer neural network does not matter for the GA as long as the ANNs parameters are mapped correctly to the genes of the chromosome the GA is optimising. Basically, each gene represents the value of a certain weight in the ANN and the chromosome is a vector that contains these values such that each weight corresponds to a fixed position in the vector as shown in Fig. 1.19. The fitness function can be assigned from the identification error of the ANN for the set of frames used for training. The GA searches for parameter values that minimize the fitness function, thus the identification error of the ANN is reduced and the identification rate is maximized.

The experimental results have shown the versatility of the Neuro-Genetic hybrid algorithm based text-dependent speaker identification system. The critical parameters such as gain term, speed factor, number of hidden layer nodes, crossover rate and the number of generations have a great impact on the recognition performance of the proposed system. The optimum values of the above pa-

rameters have been selected effectively to find out the best performance. The highest recognition rate of ANN and GA have been achieved to be 94 % and 95 % respectively. According to VALID speech database, 100 % identification rate in clean environment and 82.33 % in office environment conditions have been achieved in Neuro-Genetic hybrid system. Therefore, this proposed system can be used in various security and access control purposes.

Genetic Algorithm Inspired Activity Style Identification and Brain Signal Processing

An online biometric verification system for use over the Internet that requires no specific equipment was presented in (Everit and McOwan 2003). Combining two distinct tests to ensure authenticity, a typing style test and a mouse-based signature test, the fraudulent access rate was equal to 4.4 %, while authentic users could access the system with a rate of 99 %.

Signal recording from the brain is rather complicated biometrics, based on brain signals. It has not been studied extensively yet though it is one of the most fraud resistant biometrics. It is unlikely that different persons will have similar activity in all parts of the brain. The (Ravi *et al.* 2009) paper describes the electroencephalogram (EEG) based method which uses features computed from 61 Visual Evoked Potential (VEP) signals and states that VEP signals are the most suitable for identification of individuals. Fischer Discriminant Ratio (FDR) has been used to find the optimal EEG channels to reduce the computational time. However, the fusion of GA with Linear Discriminant Analysis (LDA) classifier shows that the identification performance is improved compared to FDR.

1.5. Research on Information Security in Lithuania

The importance of information security was recognized both by Lithuanian academic staff and business representatives. A number of specialized companies exist that provide security audit, penetration testing, security technology installation and support services. Information security courses are included in a training set of almost all universities and several textbooks were published on the topic (Kazanavičius *et al.* 2008, Stakenas 2007; Vasilecas *et al.* 2008). Scientific research is performed by several groups of scientists. Alisauskaitė and Rimkus have reviewed the computer network system security problematics. Venckauskas *et al.* have published the analysis on measuring the information security measure effectiveness applied on a company level. Stakėnas has performed study on number and probabilistic methods in cryptography (Stakenas 2006a; Stakenas 2006b). Sakalauskas and Burba have proposed the digital signature scheme based on action of infinite ring (Sakalauskas and Burba 2004). Sa-

kalauskas with co-authors (Sakalauskas *et al.* 2005) has also published an article on development of the cryptographic (FS) system. Katvickis *et al.* has performed study on one-way hash functions based on system identification theory. Kazanavicius and Pakalniskis have analyzed design and implementation of DES algorithm. Puniskis and Laurutis has published a number of articles (Puniskis and Laurutis 2007; Puniskis *et al.* 2006; Puniskis and Laurutis 2005; Laurutis 2003) on ANN application for malware epidemics detection and spam filtering. Skudutis, Garsva and Paulauskas performed the research on computer attacks, their detection methods (Paulauskas and Garsva 2006; Paulauskas 2009; Paulauskas *et al.* 2009; Paulauskas and Skudutis 2008) and attack simulation, attack modeling by stochastic networks and attack classification problem area (Garsva 2006b; Paulauskas and Garsva 2008). Research Laboratory of Security of Information Technologies, (headed by prof. Cenys) was established in 2006 at Vilnius Gediminas Technical University, that performs research on security level monitoring, malware research and modeling, computer attack classification, secure system design and programming, performs international projects with academic and business organizations in Germany, Cyprus, Portugal, Slovenia, Czech Republic and S. Korea. Laboratory provides virtual environments for information security experiments and student training.

1.6. Conclusions of Chapter 1 and Formulating Tasks for the Dissertation

Malware is considered to be among the biggest threats to modern information technology infrastructure and harm done by it is constantly increasing. The negative malware economic effect is extremely high and countermeasures system development is a constantly evolving process. Malware risk level evaluation and forecasting evolution tendencies is important for scientific substantiation of expert prediction evaluation, understanding of malware development tendencies, forecasting of malware evolution by separate parameters or their complexes, development of countermeasure techniques and prevention of malware epidemic outbreaks by implementing quick response mechanisms.

It can be expected, that application of GA can be successful for the task specified taking into consideration its ability to simulate natural evolution processes, efficiency while solving optimisation, and modelling problems with large solution space. The literature survey has shown that currently the main scientific research in area of GA application in information security systems concentrates on their application in cryptology, optimisation of IDS systems and biometric authentication systems and only a few papers exist on information security process modelling tasks.

Cryptographical GA applications can be considered not of a big practical value, since only non-standard improvements for modern ciphers were proposed and GA-based attacks proposed are against classic (transposition, substitution) or already evaluated as unsecure ciphers ("light" and Merkle-Hellman knapsacks), attack efficiency is not high (especially against classic ciphers). The only effective and novel cryptological attack is against Chor-Rivest Knapsack, but the specified cipher is not widespread. Research on GA application in IDS systems is carried out actively, but systems proposed are in scientific research or prototype testing phase, prototypes are highly specialized, no production systems exist. On the other hand research in this area can be considered as perspective, since prototypes demonstrate high efficiency rate and still can be improved by optimising GA parameters used. Application of GA for biometric security systems is also perspective and may insure qualitative increase of biometric system parameters, such as speed, error rate and flexibility. The biggest research currently done is in sphere of fingerprint and facial feature processing, several works on other biometric information types also exist, but most of the proposed systems are also of a prototype level and production tests are necessary to be certain in their reliability and suitability for practical application.

The review has shown, that none of the existing GA applications in information security systems can be directly used for the fulfilment of the specified objective and experience currently collected can be used only for understanding of GAs representation methods and operating conditions in the specific information security area. Taking this into consideration the following basic issues should be raised:

1. Reviewing of the existing malware models.
2. Analysing and describing features of selected modern malware types in a way suitable for representation in GA model, dedicated to risk level evaluation and evolution forecasting.
3. Deriving fitness evaluation criteria for the GA based malware risk level evaluation model.
4. Deriving a fitness functions for 2–3 parameters evolution tendencies forecasting of several malware types in a friendly (no countermeasures applied) and, where applicable, hostile (countermeasures applied) environment.
5. Evaluating correctness of the fitness functions and fitness evaluation criteria by applying them to historical data.
6. Defining GA operation conditions such as mutation, crossover operators, population size, etc.
7. Developing the model prototype, performing experiments on several malware types, collecting and evaluating modelling results.

Malware Technical Analysis and Malware Models

This Chapter describes the features of the three selected modern malware types: Internet worms, which can be classified as the most aggressive malware type, relatively new mobile malware and rapidly evolving botnets, which may be considered as the biggest threat to modern information infrastructure. Understanding of malware propagation, survivability and other mechanisms is important for representation in a GA suitable format. Technical analysis covers an overview of the malware type and more in-deep description of the most typical malware type representatives. Existing malware models are analysed in order to compare their features and efficiency with those proposed in the genetic algorithm based malware risk evaluation and evolution forecasting model.

The analysis and research done in this chapter was published in (Goranin and Cenys 2008a; Goranin *et al.* 2010; Juzonis *et al.* 2010; Goranin and Cenys 2008c^{*}).

^{*} The references are given in the list of publications by the author on the topic of the dissertation

2.1. Malware Strategy Definition

Nowadays malware, i.e. software created with malicious purposes in order to harm the computer software or to be installed on computer without allowance of the legal user (Monga 2009), is considered to be one of the major threats to information security, information systems and modern communication methods. The number of malware in the wild and rate of malware usage by e-criminals has the tendency to increase making protection against it a crucial task (Turner 2008). Significant shift in motivation for malicious activity has taken place over the past several years: from vandalism and recognition in the hacker community, to attacks and intrusions for financial gain. This shift has been marked by a growing sophistication in the tools and methods used to conduct attacks, thereby escalating the network security arms race (Barford and Yegneswaran 2007).

We define the malware strategy as a combination of methods and techniques, used by the worm to achieve tasks assigned to it by the worm creator. So the strategy suitable to achieve one specific task (e.g., creating the botnet) may be not useful for another (e.g., disrupting Internet functioning). Strategy definition is important in context of evaluating evolution tendencies and describing malware creation aims.

2.2. Internet Worms

Worms is a network malware, primarily replicating on networks. Usually a worm will execute itself automatically on a remote machine without any extra help from a user. However, there are worms, such as mailer or mass-mailer worms, that will not always automatically execute themselves without the help of a user (Szor 2005). Here we analyze Internet worm propagation strategies, since their replication mechanisms differ significantly from mailer and mass-mailer worms.

Propagation of most worms is rapid (compared with classical computer viruses) and aggressive. Worms such as Code Red and Nimda have been persistent for longer than 8 months since their introduction date. As worms spread through nearly all networks, they find nearly all of the weakest hosts accessible and begin their lifecycle anew on these systems. This then gives worms a broad base of installation from which to act (Nazario 2004).

Modern worms are usually created on a modular basis and may contain all or some of the following parts (Nazario 2004): a reconnaissance module, that scans the Internet for vulnerable hosts; an attack module, that may exploit from one to many known vulnerabilities at potentially vulnerable host; a communication module that allows worms to communicate between themselves or to trans-

fer information to the worm management center; a command module, that allows to accept commands; and an intelligence module, that insures functioning of the communication module, since it contains information how to find a neighbour worm for communication. Specific methods used in each of the modules are called patterns and a strategy can be also defined as a combination of patterns.

A strategy is also dependent on worm introduction techniques, i.e. method used to release worm to the wild, connection protocol used (e.g., TCP or UDP), etc. Since the number of existing and historic worms is high, we will describe only two propagation strategies used by CodeRed and Ramen, since they represent two different attitudes in complexity, vulnerable platform and functionality and can provide an understanding of strategies used in the wild.

2.2.1. Internet Worm – CodeRed

On June 18th 2001 a serious Windows IIS vulnerability was discovered. On July 13th 2001 Code Red worm version 1 that exploited this single vulnerability was released. Due to a code error in its random number generator, it did not propagate well. 10:00 UTC of July 19th Code Red version 2 was released with the corrected random generator. It generated 100 threads. Each of the first 99 threads randomly chose one IP address and tried to set up connection on port 80 with the target machine (If the system was an English Windows 2000 system, the 100th worm thread would deface the infected system's web site, otherwise the thread was used to infect other systems, too) (Zou *et al.* 2002).

Worm was programmed to scan hosts in /8 with a 50 % probability, /16 – with 37.5 % probability and with 12.5 % probability it would scan a totally random network (Nazario 2004). Subnetworks 127.0.0.0/8, loopback, 224.0.0.0/8, multicast were excluded (Serazzi and Zanero 2004). If the connection was successful, the worm would send a copy of itself to the victim web server to compromise it and continue to find another web server. If the victim was not a web server or the connection could not be setup, the worm thread would randomly generate another IP address to probe. The timeout of the Code Red connection request was programmed to be 21 seconds.

Netcraft web server survey showed that there were about 6 million Windows IIS web servers at the end of June 2001 (Zou *et al.* 2002). More than 350 000 of them were infected in several hours (Staniford *et al.* 2002).

2.2.2. Internet Worm – Ramen

The Ramen worm appeared in January 2001. Ramen attacked RedHat Linux 6.0, 6.1, 6.2, and 7.0 installations, taking advantage of the default installation and three known vulnerabilities: FTPd string format exploits against wu-ftp 2.6.0, RPC.statd Linux unformatted strings exploits, and LPR string format attacks. This vulnerable software could be installed on any Linux system, meaning the Ramen worm can affect other Linux systems, as well. The worm acted in the following way: defaced any Web sites it found; disabled anonymous FTP access to the system; disabled and removed the vulnerable rpc.statd and lpd daemons, and ensured the worm would be unable to attack the host again; installed a Web server on TCP port 27374, used to pass the worm payload to the child infections; removed any host access restrictions and ensured that the worm software would start at boot time; notified the owner (worm creator) of two e-mail accounts of the presence of the worm infection. Worm then began scanning for new victim hosts by generating random class B (/16) address blocks (scans were restricted from 128/8 to 224/8, the most heavily used section of the Internet). Web server acted as a small command interface with a very limited set of possible actions. The mailboxes served as the intelligence database, containing information about the nodes on the network. This allowed the owners of the database to be able to contact infected systems and operate them as needed (Nazario 2004).

2.3. Mobile Malware

Mobile malware is defined as viruses, worms, Trojans or other types that spread on the Smartphones or other mobile devices running mobile operating system (OS). Although it is a relatively new malware type and not very common in the wild yet its portion is highly expected to increase with the increase of the smart mobile device market. IDC (Shah 2009) predicts that 1 billion mobile devices will go online by 2013. Protection against malware on mobile platforms is not very common, compared to traditional computer systems, making them especially attractive for e-criminals. Mobile devices can also provide a variety of services to e-criminals, the traditional systems cannot do: SMS-spam, MMS-spam, call-proxy, etc.

According to Kaspersky labs (Kaspersky 2009) the first mobile virus to appear was the “*Cabir*” virus which appeared on the 15th of June 2004, infected mobile phones running the Symbian OS and used Bluetooth wireless network as a propagation channel. After the successful infection the virus appended the telephone software with its code, activated the Bluetooth and started searching for another Bluetooth device to forward the infected file. Since Bluetooth network coverage is limited the propagation rate of the first mobile virus was rather

limited also. The first Trojan malware („*Skulls*“) also appeared in 2004, November (Naraine 2004; Niemela 2005). It infected NOKIA mobile phones, running Symbian operating system. „*Skulls*“ propagated by pretending to be a software update, usually as Macromedia Flash update file with .*sis* extension. When the phone user activated the Trojan it changed the phone configuration settings and depicted the skulls on the screen. It also blocked many functions, such as SMS, MMS, calendar, camera, etc. The phone user could only perform telephone calls. The mobile Trojan evolution continued in 2005. A new Trojan „*Locknut.A*“ was detected (Jarno 2005). Also created for the Symbian platform it was particular in size. The „*patch.sis*“ file that contained the infection was only 2KB size, making it the smallest known Trojan for mobile platform.

The first mobile malware that started using propagation methods, other than Bluetooth, was the „*Commwarrior.A*“ virus, also running on the Symbian platform (F-Secure 2006; Sundgot 2005). It was using much quicker propagation by MMS, since this method does not have limitations by distance, although Bluetooth was also supported. MMS message included text in English, which proposed the phone user a new game, update for antivirus software or similar. The message was sent to all contacts, found in the phone address book. In this case virus authors have relied on the social engineering since when the recipient receives the message from his friend or familiar person the probability of opening it is higher than when it comes from the unknown number. An interesting thing is that Bluetooth was activated during the working hours and MMS were sent in the evening and at night. After each successful infection the virus makes a one minute delay and after that starts searching for a new victim. In 2009 the Kaspersky Labs has discovered a new mobile malware named „*sms.python.flocker*“, written in Python language and designed to manipulate the mobile phone accounts. The main malware functionality is dedicated to financial gain. Virus sends SMS messages to the specific number, which allows transferring money from the account of the infected phone to the account of the malware author (Kaspersky 2009).

2.4. Botnets

The term bot describes a remote control program loaded on a computer, usually after a successful invasion (compromise can be achieved with the help of a worms, Trojan horses or “backdoor” software (Brand *et al.* 2009)), that is often used for nefarious purposes (Provos and Holz 2007) usually against the computer owners’ intentions and without their knowledge (Lee *et al.* 2007). A botnet is a network of computers on which a bot has been installed, and is usually managed remotely from a Command & Control (C&C) server. The main purpose of

botnets is to use hijacked computers for fraudulent online activities (Barroso 2007): identity theft (Barford and Yegneswaran 2007), sending spam, performing massive, sophisticated, tactically agile, difficult to trace denial-of-service attacks (Lee *et al.* 2007; Banks and Stytz 2008) which due to botnet development have gone evolution from theoretical to real informational weapons (Fultz 2008), click fraud, key cracking, phishing, distribution of new malware to the wild (Lee *et al.* 2007), distribute pirated media and other tasks (Karasaridis *et al.* 2007). Botnets are managed by a criminal, a group of criminals or an organized crime syndicate (Barroso 2007) which are called botmasters or botherders (Banks and Stytz 2008).

Botnets dominate today's attack landscape (Li *et al.* 2009a) and it is widely accepted that botnets pose one of the most significant and steadily increasing threats to the Internet with devastating consequences (Barroso 2007; Rajab *et al.* 2007). Bot technology has accelerated in its development in the last few years (Banks and Stytz 2008). Bot armies are effective for two reasons: they can execute multiple overt actions against targets and can, alternatively, provide multiple coordinated and covert listening points within targeted networks and computer systems (Banks and Stytz 2008). It is a main weapon used on targeted computers and also a significant threat even on a whole country scale, when botnets are used in cyber warfare as brute force army performing Distributed Denial of Service attacks (DDoS) (Juknius and Čenys 2009).

The deployment and operation of bot armies are aided by the security vulnerabilities that exist in contemporary software; vulnerabilities that are likely to increase in number commensurately with the increase in the size of software products (Banks and Stytz 2008). The trend toward an economic motivation is likely to catalyze the development of new capabilities in botnet code making the task of securing networks against this threat much more difficult (Barford and Yegneswaran 2007). According to ENISA report (Barroso 2007) the objective of criminals using botnets will be twofold: to increase the number of infected computers and increase the stealth (or as we propose using the term survivability) by different methods – polymorphism, peer-to-peer (P2P), active protection mechanisms.

Over the years botnet capability has increased substantially to the point of blurring the lines between traditional categories of malware (Barford and Yegneswaran 2007) and in fact possesses characteristics are those of a virus, a worm, and a Trojan (Banks and Stytz 2008). While bots belonging to a certain botnet are expected to have some distinct modes of operation, individual bots are also expected to have unique behaviours due to variability in the software or hardware they run on, phase difference in their states, different background applications running simultaneously etc. (Karasaridis *et al.* 2007). Botnets are different from traditional discrete infections in that they act as a coordinated attack-

ing group (Dagon *et al.* 2005). We would even like to propose understanding of a botnet as a single artificial organism, not a combination of small organisms (bots), since the task being assigned is assigned to the whole botnet. Botnet has similarities in organization with termite or bee families, where a single individual is not very important (except dominant), but the whole family forms a real power, which some scientists are ready to name as a super-organism.

The overall architecture and implementation of botnets is complex, and is evolving toward the use of common software engineering techniques such as modularity (Barford and Yegneswaran 2007). The taxonomy proposed in (Barford and Yegneswaran 2007) classifies botnets by key mechanisms used:

1. architecture;
2. botnet control mechanisms;
3. host control mechanisms;
4. propagation mechanisms;
5. target exploits and attack mechanisms;
6. malware delivery mechanisms;
7. obfuscation methods;
8. deception strategies.

On the other hand according to (Banks and Stytz 2008) the botnet creation process can be described in 7 steps:

1. malware creation;
2. command and control creation;
3. malware propagation;
4. malware infestation;
5. command and control setup;
6. further malware download;
7. malware check-in for further instructions via the command and control setup.

Most botnets that have appeared prior to 2005 have had a common centralized architecture. That is, bots in the botnet connect directly to some special hosts (called "command-and-control" servers, or "C&C" servers) (Wang, P. *et al.* 2007) and were based on IRC due to its ability to scale to thousands of clients easily (Karasaridis *et al.* 2007). According to ENISA report (Barroso 2007) IRC is still being used by some botnets, but HTTP is now more widespread, since it is even easier to implement and can be hidden in normal user navigation. There

are other methods of communication that use covert channels (e.g., in DNS, ICMP etc.) (Barroso 2007). The move from IRC-based architecture has happened because it was rather easy for ISP to disrupt botnet by blocking the central C&C. Considering the above weaknesses inherent to the centralized architecture of current C&C botnets, it is a natural strategy for botmasters to design a peer-to-peer (P2P) control mechanism into their botnets and in fact different kinds of P2P control architectures were implemented (Wang, P. *et al.* 2007). In a P2P architecture (Grizzard *et al.* 2007), there is no centralized point for C&C. Nodes in a peer-to-peer network act as both clients and servers such that there is no centralized coordination point that can be incapacitated. If nodes in the network are taken offline, the gaps in the network are closed and the network continues to operate under the control of the botmaster. One more problem posed by P2P botnets to security specialists is the difficulty in estimating the size of the P2P botnet (Dittrich, D. and Dittrich, S. 2008).

Botnets usually do not rely only on a single method of propagation but make use of a combined approach. Methods include scanning for vulnerable hosts (Li *et al.* 2007), network shares, spam or unsolicited e-mail, P2P (Barroso 2007), net news, web blogs, other WEB resources (Dagon *et al.* 2005), social engineering via an enticement 'lure' e-mail, browser exploit and malicious file download (Barroso 2007), via instant messenger (Dittrich, D. and Dittrich, S. 2008) and other common malware propagation methods. Separate botnet parts can use different propagation methods. In case botnet uses scanning for search of vulnerable hosts (Li *et al.* 2007) three types of scanning can be separated: localized scanning (each bot chose the scanning range based on their own IP prefixes), targeted scanning (botmaster specifies a particular IP prefix for bots to scan) and uniform scanning (scanning the whole Internet).

Since information security community has accepted botnets as a threat and the use of countermeasures has been started the botnet creators were forced to use different obfuscation and deception methods to protect botnets and escape from punishment for illegal activities. The advanced modern botnet examples rely on a wide range of complex methods such as extremely resilient random topologies (including structured P2P networks), traffic anonymization (Dagon *et al.* 2005), load balancing, reverse proxies for the C&C servers (making it harder to track down the attacks), fast-flux services (networks of compromised computer systems with public DNS records that are constantly changing), Rock Phish (compromised computers and thousands of DNS sub-domains are used in order to set up phishing scenarios that hide the real phishing site) (Barroso 2007), encrypted/obfuscated control channel (Wang, P. *et al.* 2007) and many others (Dagon *et al.* 2005; Wang, P. *et al.* 2007). The recent trend is toward smaller botnets with only several hundred to several thousand zombies since big botnets are bad from the standpoint of survivability. It has also been suggested

that the wider availability of broadband access makes smaller botnets as capable as the larger botnets earlier (Vogt *et al.* 2007).

For better understanding of modern botnets further we provide a technical description of two botnet types: centralized, IRC based Agobot (description based on (Gordon 2004; Zhaosheng *et al.* 2002; Symantec 2004; Stewart 2004), other sources as cited and of the most famous P2P botnets STORM (description based on (Zhang 2008; Kaspersky 2008).

2.4.1. IRC Botnet – Agobot

Originally botnets were based on IRC and many of them still are. A botmaster sets up an IRC server and clients connect to it. The server acts as the C&C from where the bots get their orders (Fig. 2.1). All communication is based on the IRC protocol (Mukamurenzi 2008). IRC is the most common botnet type because it is scalable and easy to hide within.

While instances of botnets with looser control structures, such as those that use peer-to-peer networks, are increasing, IRC-style is still the most prevalent because it is scalable and provides instantaneous control over the bots (Canavan 2005). The attacker and the zombie hosts subscribe to the same IRC channel. The attacker issues commands and the bots respond through that channel (Lee *et al.* 2007).

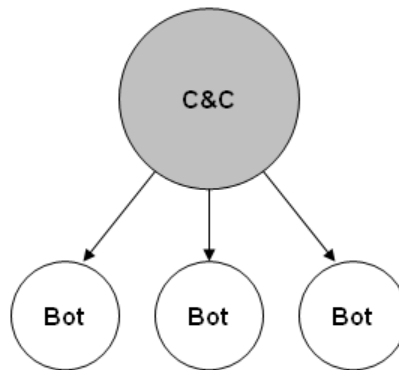


Fig. 2.1. Botnet with centralized architecture

In June 1999 the first worm emerged to make use of IRC as a means of remote control. Written in Delphi, PrettyPark.Worm connected to a remote IRC server and allowed the attacker to retrieve a variety of information about the system. It also had a basic update mechanism, which allowed it to download and execute a file from IRC.

One of the most typical and widespread IRC botnets is the Agobot family (also known as Gaobot, Forbot, Phatbot, Urxbot, Rxbot, Rbot and compilations by individuals), which is among the most widespread in the number of variants created according to (Gordon 2004). In some cases, there is an overlapping name, making the study of them very difficult (Gordon 2004). The (Gordon 2004) paper presents a detailed overview of the Agobot family history and changes in functionality and description of them. Agobot is created on a modular basis and mainly affects computers, running MS Windows platform. Bot development kit is distributed under the GNU GPL license (Gordon 2004). Agobot can exploit many well known OS vulnerabilities (e.g. buffer overflow) and back doors left by other viruses (a large collection of target exploits) (Barford and Yegneswaran 2007).

Exploits and delivery functions are separated. Once the first step exploits succeed, it opens a shell on the remote host to download bot binary. The binary is encoded to avoid network-based signature detection. The bot has the module to test for debuggers (e.g. SoftIce) and VMware once it is installed. If it detects VMware it stops running. So VMware based Honeypot cannot run Agobot (Zhaosheng *et al.* 2002). The bots' functionality may include (depending on compilation) but is not limited to the following:

- credentials stealing, key-logging;
- self-protection against Firewall/Antivirus processes by stopping them;
- self-protection by blocking Antivirus Updates (by modification of HOSTS file);
- backdoor opening (using various listening ports), execution of commands and programs;
- author notification about the Compromise via IRC;
- commands acceptance via IRC;
- IRC Client control interface protection by password;
- remote update and deinstallation of the installed bot;
- port scanning for detection of other vulnerable hosts;
- DDoS functionality;
- packet sniffing;
- etc.

2.4.2. P2P Botnet – STORM

As it was written earlier the STORM botnet which was linked to a single network with the help of the Storm Worm, special crafted Trojan horse, uses the decentralized hierarchy, as shown on Fig. 2.2. We do not provide estimations on the number of infected hosts, since they differ a lot and are exaggerated in many cases, but Sans (2008) has named the Storm botnet as biggest security issue in 2008.

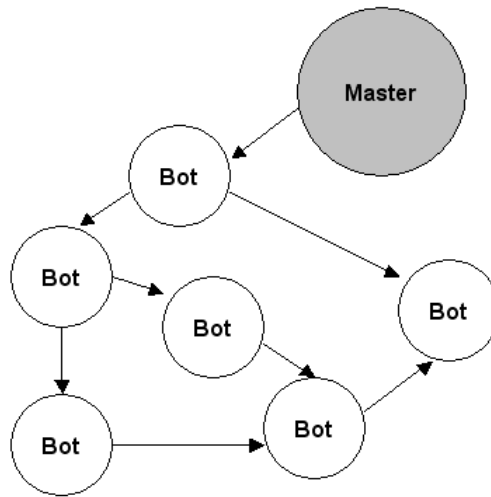


Fig. 2.2. Decentralized P2P botnet

There are no significant differences from the IRC botnets in malicious functionality, but it differs in resistance and self-protection mechanisms. The P2P-based botnet is very hard to trace and to shut down, because the botnet has robust network connectivity, uses encryption, and controls traffic dispersion. Each bot influences only a small part of the botnet, and upgrade/recovery is accomplished easily by its botmaster (Zhang 2008). It has aggressive defences (it attacks anyone (DDoS) who tries to analyse it or reverse engineer it). It uses a clique architecture where each clique has its own 40-bit encryption key. There are no file exchanges between infected hosts which makes it difficult to track. Storm Worm is now employing the same tactics as terrorist organisations. Each node belongs to a cell and knows only the members of the cell. If a cell is taken down, it does not put the whole botnet at risk. When the botmaster needs to send information to all the nodes, he notifies one member of each cell who then passes on the information. This creates less traffic than if he were broadcasting

to all the bots. Another advantage is that even if traffic is being monitored on a certain bot, suspicion is not raised if it keeps receiving messages from the same source (Mukamurenzi 2008).

Dissemination and functionality mechanisms are not so specific. Storm botnet first emerged in early 2007 and spread by sending e-mails with currently relevant subjects such as natural disasters and other topics in the news with attached or links to videos/images (Mukamurenzi 2008; Porras *et al.* 2007). It was spreading mainly using social engineering methods, like tricking people into downloading it from e-mails or websites. Storm was using fast-flux service networks. The website's DNS records changed every few minutes (Barroso 2007; Mukamurenzi 2008). It does not roam the Internet looking for vulnerabilities in machines that it can exploit (Mukamurenzi 2008).

After the executable has been downloaded and executed, it adds the system driver "*wincom32.sys*" to the Windows process "*services.exe*". Part of the installation involves hard-coding a peer list on the bot and saving it in a file called *windir/system32/wincom32.ini*. The Windows firewall is then disabled and several TCP ports are opened. The worm then bootstraps the bot onto the peer-to-peer Overnet network based on the Kademlia (eDonkey) algorithm so that it can contact its peer list if they are online (Mukamurenzi 2008). Storm botnet, uses UDP-port 4000 for communication between peers. Such protocol makes closing down C&Cs – which would normally be an effective countermeasure against IRC botnets – useless (Barroso 2007). After connecting to network and contacting its peer list, the bot is then ready for the secondary injections such as Rootkit component, SMTP spamming component DDoS tool and etc. (Mukamurenzi 2008).

The success of STORM is partially due to the lack of security awareness in the average computer user. The other part of its success is the use of state-of-the-art technologies and a reputation for aggressiveness (Mukamurenzi 2008).

2.5. Malware Models

Existing malware models mainly concentrate on malware epidemic consequences modelling, i.e. forecasting the number of infected computers, simulating malware behaviour or economic propagation aspects and are based only on current malware propagation strategies. Only one paper (Noreen *et al.* 2009) except our papers published on the topic of this was released, almost one year later than

our first publication (Goranin and Cenys 2008a*), that demonstrates the evolution concept on separate virus family.

2.5.1. General or Malware Specific Models

Existing malware propagation models concentrate to forecasting the number of infected computers in the initial propagation phase (Fig. 2.3).

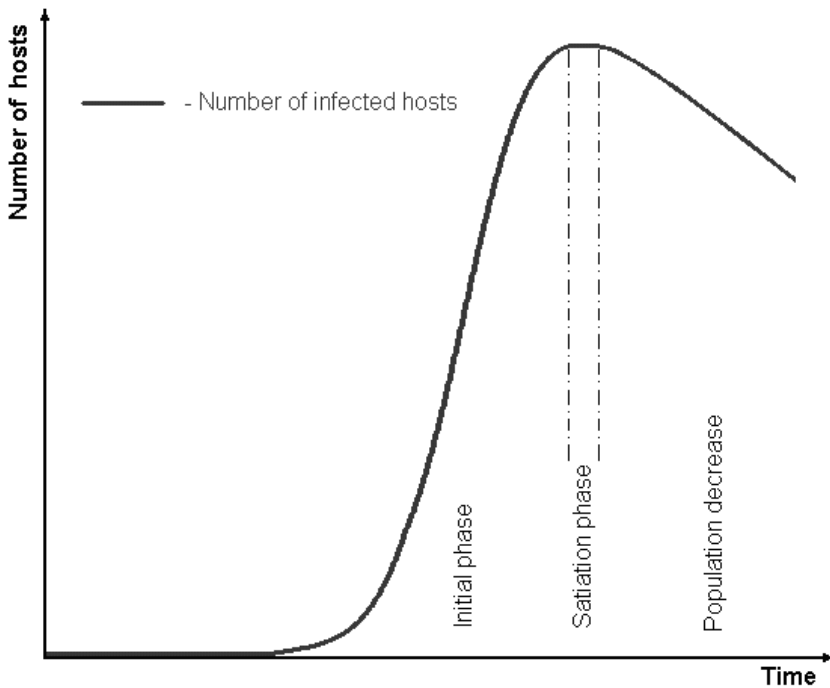


Fig. 2.3. Malware propagation graph

The first epidemiological model to computer virus propagation was proposed by (Kephart and White 1991). Epidemiological models abstract from the individuals, and consider them units of a population. Each unit can only belong to a limited number of states. A *SIR* model assumes the *Susceptible-Infected-Recovered* state chain and *SIS* model – the *Susceptible-Infected-Susceptible* chain.

* The reference is given in the list of publications by the author on the topic of the dissertation

The Random Constant Spread (RCS) model (Staniford *et al.* 2002) was developed using empirical data derived from the outbreak of the CodeRed worm. It assumes that the worm has a good random number generator that is properly seeded. The model assumes that a machine cannot be compromised multiple times and operates several variables: K is the constant average compromise rate, which is dependant on worm processor speed, network bandwidth and location of the infected host; $a(t)$ is the proportion of vulnerable machines which have been compromised at the instant t , $N \cdot a(t)$ is the number of infected hosts, each of which scans other vulnerable machines at a rate K per unit of time. But since a portion $a(t)$ of the vulnerable machines is already infected, only $K \cdot (1 - a(t))$ new infections will be generated by each infected host, per unit of time. The number n of machines that will be compromised in the interval of time dt (in which a is assumed to be constant) is thus given by:

$$n = (Na) \cdot K(1 - a)dt. \quad (2.1)$$

From this

$$\frac{da}{dt} = Ka(1 - a), \quad (2.2)$$

where

$$a = \frac{e^{K(t-T)}}{1 + e^{K(t-T)}}. \quad (2.3)$$

So the model can predict the number of infected hosts at time t if K is known. The higher is K , the quicker the satiation phase will be achieved by worm.

As Nazario (2004) states, that although more complicated models can be derived, most network worms will follow this trend. Other authors (Chen *et al.* 2003) propose the discrete time model, in the hope to better capture the discrete time behavior of a worm. However, according to (Serazi and Zanero 2004) continuous model is appropriate for large scale models and the benefits of using a discrete time model seem to be very limited. On the other hand (Serazi and Zanero 2004) propose a sophisticated compartment based model, which treats Internet as the interconnection of autonomous systems, i.e. subnetworks. Interconnections called “bottlenecks”. The model assumes that inside a single autonomous system the worm propagates unhindered, following the RCS model. The authors motivate the necessity of their model via the fact that “bottlenecks” can be flooded by the worm scans. (Zou *et al.* 2002) propose a two-factor propagation model, which is more precise in modelling the satiation phase taking into

consideration the human countermeasures and the decreased scan and infection rate due to the large amount of scan-traffic. The same authors have also published an article on modelling worm propagation under dynamic quarantine defence (Zou *et al.* 2003) and evaluated the effectiveness of several existing and perspective worm propagation strategies (Zou *et al.* 2005).

In a technical report (Zou *et al.* 2004) authors described a model of e-mail worm propagation. The authors model the Internet e-mail service as an undirected graph of relationship between people. In order to build a simulation of this graph, they assume that each node degree is distributed on a power-law probability function. Malware propagation in Gnutella type P2P networks was described in (Ramachandran and Sikdar 2006). The study revealed that the existing bound on the spectral radius governing the possibility of an epidemic outbreak needs to be revised in the context of a P2P network. An analytical model that emulates the mechanics of a decentralized Gnutella type network was formulated and the study of malware spread on such networks was performed. Authors of (Ruitenbeek *et al.* 2007) simulate virus propagation using parameterized stochastic models of a network of mobile phones, created with the help of Mobius tool and provides insight into the relative effectiveness of each response mechanism.

Several botnet-oriented models were proposed, but they all concentrate to tasks other than botnet evolution forecasting. Lelarge (Lelarge 2009) introduces an economic approach to malware epidemic modelling (including botnets). In his model computers on the network face epidemic risks. Epidemic risks which depend on the behaviour of other in the network. The model is based on graph theory that quantifies the impact of such externalities on the investment in security features in a network. (Li *et al.* 2009b) model botnet-related cybercrimes as a result of profit-maximizing decision-making from the perspectives of both botnet masters and renters/attackers. From this economic model, they derive the effective rental size and the optimal botnet size. Fultz in (Fultz, 2008) describes distributed attacks organized with the help of botnets as economic security games.

Banks and Stytz (2008) use the epidemiological model as a basis for botnet modelling. The model is modified from this general model based upon the type of infection, transfer modality, and potential for re-infection and can be represented as a $M-S-E-I-R$ (M is the class of computers who are not infected with malware that can be exploited to enable bot infestation; S represents the class of computers that are infected during manufacture with malware that can be exploited to enable bot infestation; E is the set of computers that have been infected, are not transmitting the infection, and in whom the infection has not been detected; I is the set of computers that have been infected, are transmitting the infection, and in whom the infection has not been detected; R is the set of com-

puters that have been infected, whose infection has been detected, and that have had their bot removed).

In (Zou *et al.* 2008) authors suggest using botnet propagation model via vulnerability exploitation and notice some similarities of bot and worm propagation. Botnet propagation modelling using time zones was proposed in (Dagon *et al.* 2006). Authors of (Ruitenbeek and Sanders 2008) have developed a stochastic model of P2P botnet formation to provide insight on possible defence tactics and examine how different factors impact the growth of the botnet.

2.5.2. Genetic Algorithm Based Models

The idea of GA usage for malware evolution forecasting was first proposed by Goranin and Čenys in an article (Goranin and Cenys 2008a^{*}), dedicated to Internet worm propagation strategy evolution forecasting and was further developed in (Goranin and Cenys 2009^{*}), where model changes for hostile environment were described. In 2010 we have proposed model extensions for mobile malware and botnet evolution propagation and survivability parameters' evolution forecasting (Goranin *et al.* 2010; Juzonis *et al.* 2010^{*}). We have also proposed the automated genetic algorithm based decision tree generation model, that can be used for risk level evaluation of a newly appearing malware samples (Goranin and Cenys 2008b^{*}).

Since these models are described in Chapters 3 and 4, here we provide an analysis of (Noreen *et al.* 2009) article, which validates the notion of evolution in viruses on a well-known Bagle virus family. The model proposed consists of three modules:

1. a code analyzer that generates a high-level genotype representation of a virus from its machine code;
2. a GA that uses the standard selection, cross-over and mutation operators to evolve viruses;
3. the code generator converts the genotype of a newly evolved virus to its machine level code

The general model description is presented on Fig. 2.4. The results of the study showed that new viruses—previously unknown of Bagle family have successfully evolved starting from a random population. The idea of high level feature representation is rather similar to that proposed in our research papers, but authors (Noreen *et al.* 2009) do not try predicting the evolution trends and simply uses the mode for evolution possibility demonstration. This paper is more

^{*} The reference is given in the list of publications by the author on the topic of the dissertation

malware specific compared to our articles* since its characteristic representation is created for the specific malware type (Bagle virus), code-dependent, mainly demonstrates evolution concept and is not specialized for evolution forecasting.

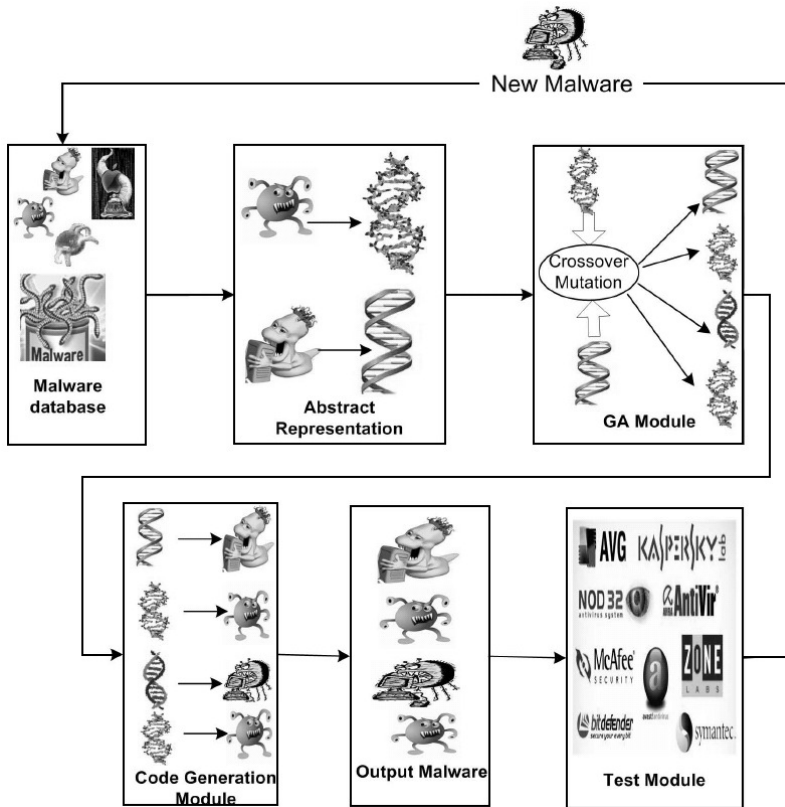


Fig. 2.4. Architecture of the malware evolution model (Noreen *et al.* 2009)

The first step in malware evolution is the high-level abstract representation (or genotype) of a given malware (Table 2.1).

The second step is the application of evolutionary algorithms to the high-level representation. After the application of GA, new individuals in the population are translated back from high-level representation to machine-level code. It is effectively a code generator that converts an abstract representation to a machine-level code. The other module of the proposed model translates the high-level representation back to the machine-level code. Finally, the generated virus files are tested using commercial antivirus software to check if they are known variants of the given malware.

Table 2.1. Abstract Representation of Bagle (Noreen *et al.* 2009)

Feature	Description	Examples
Date	The date checked by Bagle to (de-)activate its process	28 January, 2004
Application	The applications used to conceal Bagle	calc.exe, notepad.exe, sndrec32.exe
Port Number	Port opened by Bagle to send or receive commands	2475, 6777, 2556
Attachment	Name of the attachment used by Bagle	Random characters
Attachment Extension	It specifies the extension of the attachment	.rar, .exe, .pif, .zip
Websites	Bagle contact the websites to inform about the infection	http://www.it-msc.de/1.php, http://www.getyourfree.net/1.php
Domain	Bagle ignores to email itself to the domains specified	@hotmail.com, @msn.com
Email Body	Contains the email body of Bagle	Test=), YoursID<Random Characters>
Email Subject	Specifies the subject of the email	Hi, Subject:ID <Random Characters>
Registry Variable	Contains the name of registry variables used by Bagle	au.exe, d3dupdate.exe
Virus Name	Name of the Bagle shown in the task manager	bbeagle.exe, au.exe, readme.exe
File Extension	File extensions to be searched in the fixed directories	.wab, .txt, .htm, .php
Process Terminated	Processes terminated by Bagle	atupdater.exe, aupdate.exe
P2P Propagation	Names used by Bagle to copy itself to peer computers	ACDSee 9.exe, Ahead Nero 7.exe

The Bagle samples are divided into two categories: training and testing. The samples in the training category are utilized to guide the evaluation process: the fitness of the offspring is a function of the similarity measure of their chromosomes with that of training viruses.

The fitness of an individual is directly proportional to its resemblance to the malware samples in the training set. The resemblance is calculated by comparing each parameter of the genotype to the respective genotype parameter of the training set. Moreover, the fitness is normalized by assigning a weight to every gene of the representation. For k genes, the fitness F is mathematically given by:

$$F = \sum_{i=1}^k \frac{f_i}{k}, \quad (2.4)$$

where f_i is determined by the resemblance of respective genes of an offspring and a sample virus in the training set. The generated offspring are also compared to the testing set and the statistics archived. An exact match results in a fitness of 1. Moreover, once new individuals have evolved that do not match with the training samples then three conclusions can be formulated:

1. the new individual is a malware in the testing category;
2. the new individual is an unknown Bagle virus;
3. the new individual is not a Bagle virus.

The (Noreen *et al.* 2009) article demonstrates that evolution forecasting is an important task and other researcher also work in a similar direction.

2.6. Conclusions of Chapter 2

1. In Chapter 2 technical analysis of three modern malware types was provided. The information collected will be used for GA-ready representation of malware features in the malware risk evaluation and evolution forecasting model. It was noticed, that almost all modern malware is created on a modular basis. Features included in malware depend on the target systems and tasks assigned to malware by its creator. Usually malware developers base their design intuitively, taking into consideration factors of the outer environment, where malware will have to operate.
2. Existing malware models mainly concentrate on malware epidemic consequences modelling, i.e. forecasting the number of infected computers, simulating malware behaviour or economic propagation aspects and are based only on current malware propagation strategies.
3. The only existing GA-based model is dedicated to evolution concept demonstration, i.e. showing that new previously unobserved malware can evolve, and can not be used for malware type evolution forecasting since it is code dependant and limited to a single virus family. Still, some concepts proposed, such as high level of genotype representation, are very similar to our research done and demonstrate, that other researchers also perform investigation in the direction and treat them as perspective.

Automatic Malware Risk Evaluation Model

This Chapter presents malware risk evaluation model and experimental investigation of its application to evaluation of Internet worm population stability threat after the satiation phase. Risk evaluation model is based on decision trees, composed out of the classified historical statistical data and generated with the help of genetic algorithms. The model performance tests on Internet worms with known population stability after the satiation phase are described. Decision trees are generated with the help of the GAtree software.

The experiments and results presented in this chapter were published in (Goranin and Cenys 2008b^{*}).

3.1. Risk Definition and General Model Requirements

According to ISO/IEC 27005 standard definition risk is defined as a combination of event probability and its consequences. Event in context of information security is defined as a utilized threat scenario, initiated by some agent. Event makes some harm to information resource or system part. Risk term is usually

^{*} The reference is given in the list of publications by the author on the topic of the dissertation

used while speaking about organizational security, risk management and risk analysis processes. Risk evaluation is mainly used as a business decision tool in order to estimate the necessity of countermeasures and their level. Risk evaluation methods can be classified into quantitative (the approximate harm level in a measurable value is calculated) and qualitative (only a harm category, such as low, medium or high, is estimated). In many cases the second approach is the most appropriate, since quantitative calculations make the risk evaluation process very complicated and resource-demanding (Vageris 2005). Risk evaluation can be based either on statistical evaluation of historical data or expert knowledge.

The malware risk level evaluation can be defined as a process of evaluating potential threat caused by specific malware to global IT infrastructure. Examples of such threats could be propagation speed and corresponding network flood, stability of malware population allowing attackers to use the created infrastructure of infected computers for further deployment of Botnets, increase of confidential data leakage, etc.

Malware risk level evaluation is especially important for effective and quick response to newly appearing malware. Taking into consideration the number of malware released to the wild every day and propagation speed shown by certain malware types, that should be the automatic and not an expert-based system. Deployment of such scheme could be based on decision trees, composed on historical statistical data. Creation of such trees requires reliable statistical data, tree optimization and update, as new malware samples appear and are added to the statistical database.

Since currently Internet worms are considered to be the most aggressive malware type with particularly high propagation rate here we propose the model for evaluation of Internet worm population stability threat after the satiation phase.

3.2. Decision Trees Optimization by Genetic Algorithms

A decision tree is made of decision nodes and leaf nodes. Each decision node corresponds to a test X over a single attribute of the input data and has a number of branches, each of which handles an outcome of the test X . Each leaf node represents a class that is the result of decision for a case. The problem here is how to choose the best attribute for each decision node during construction of the decision tree. The basic idea of this criterion is to, at each splitting step, choose an attribute which provides the maximum information gain while reducing the bias in favour of tests with many outcomes by normalization (Stein *et al.* 2005).

The classifier learning contest, organized by the KDD conference in 1999 has shown, that top three classifiers were all decision tree classifiers (Stein *et al.* 2005). That shows that decision trees can be considered as the most appropriate solution for the raised risk evaluation task. Decision trees or combined decision tree and GA approach is already widely used for attack classification in IDS systems.

In (Stein *et al.* 2005) article the hybrid GA and decision tree method is proposed, which optimizes the decision tree structure and insures effective attack classification into four categories: Probe, DOS, Remote2Locan and User2Root. The average Hybrid results are typically better than those of decision trees (higher detection and lower error rate) since the hybrid approach was able to focus on relevant features and eliminate unnecessary or distracting features. GAs are used for optimal parameter set formation. Parameters are taken from the KDDCUP'99 database. The hybrid model is based on (Kohavi *et al.* 1995) model, where the search component is a GA and the evaluation component is a decision tree.

The initial population is randomly generated. Every individual of the population has 41 genes, each of which represents a feature of the input data and can be assigned to 1 or 0. 1 means the represented feature is used during constructing decision trees; 0 means it is not used. As a result, each individual in the population represents a choice of available features. For each individual in the current population, a decision tree is built using the C4.5 program. This resulting decision tree is then tested over nine validation data sets, which generate nine classification error rates.

3.3. GAtree Modelling Tool

For our experiments we use the GAtree program (Papagelis and Kalles 2001), which is able to perform decision tree classification. GAtree acts according to a classical genetic algorithm only modifying the chromosome representation from binary to tree.

Decision tree fitness is evaluated by applying it to the test data. The mutation and crossover operations are performed with tree structures as shown on Fig. 3.1 and 3.2.

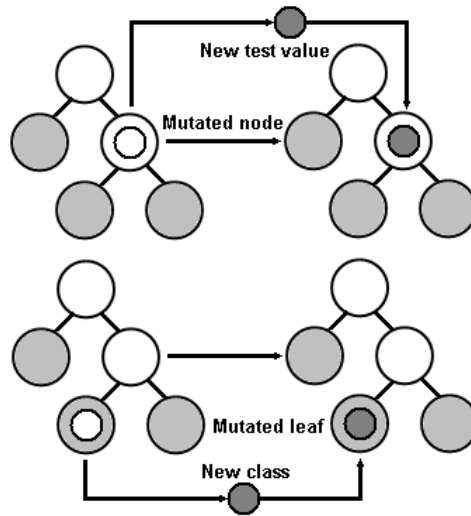


Fig. 3.1. Mutation on tree data structure (Papagelis and Kalles 2001)

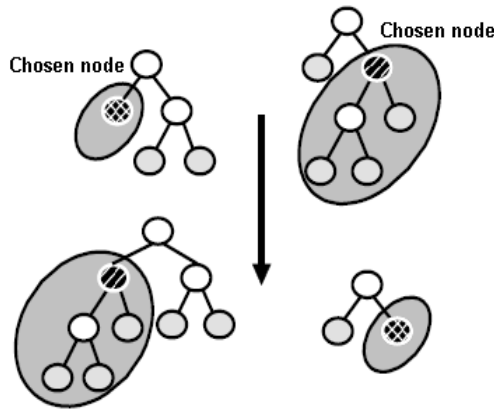


Fig. 3.2. Crossover on tree data structure (Papagelis and Kalles 2001)

The GAtree program uses the classified data files in order to generate the decision trees, allows the user to select the characteristics of the resulting decision tree, can provide a set of totally different decision trees that are close matches to the solution space. All those trees can be used alternatively to the best-fit one.

3.4. Data Representation for Decision Tree Generation

Propagation strategies of Internet worms, selected for the training data set was presented in the way, described in Table 3.1.

Table 3.1. Representation of the Internet Worm strategies for risk evaluation
3.1 lentelė. Interneto kirminų strategijų atvaizdavimas rizikos vertinimui

Gene's name (Parameter)	Description
OS_PLATF	Describes the OS platform the worm can function on. Compulsory gene.
EXPL_1 ... EXPL_N	Describes the exploits included in worm's body. Up to 8 exploits can be described in the model. The first exploit is compulsory, since at least one exploit is necessary for worm's propagation. Exploits from 2 to 8 may or may not be included.
IP_GEN	Describes potential victim's IP address generation algorithm. Compulsory gene.
TRANSF	Describes the worm's body transfer mechanism. Compulsory gene
MEM	Describes type of memory the worm uses. Not compulsory gene.
HIER	Describes worm's network hierarchy. Not compulsory gene.
COM	Describes worms' communication algorithm. Not compulsory gene.
EXEC	Describes remote worm management features. Not compulsory gene.
ADD	Describes additional worm functionality features. Not compulsory gene.
EVOL	Describes worm's evolution mechanisms. Not compulsory gene.

A sample strategy representation for a worm with a complex propagation strategy Ramen is shown on Fig. 3.3. ("-" marks attributes that were not included).

```

S(Ramen)=("Linux", EXPL_1="FTP port", EXPL_2="RPC port",
EXPL_3="LPR attack", EXPL_4="-", EXPL_5="-", EXPL_6="-",
EXPL_7="-", EXPL_8="-", IP_GEN="Random, range from 128/8
to 224/8", TRANSF="TCP/IP", MEM="-", HIER="Centralized
hierarchy", COM="child-management host", EXEC="http
interface", ADD="deface, protect against reinfection",
EVOL="-")

```

Fig. 3.3. Sample Internet worm strategy representation for risk evaluation

As it can be seen from Figure 3.3 the strategy described is rather complex for this malware type. This is done for the sake of visibility. The Internet worm using this strategy (Ramen) infects Linux operating system, uses 3 exploits, uses rather simple IP generation method, propagates by the means of TCP/IP protocol, supports hierarchy, communication channels, management interface and protects infected host from reinfection.

3.5. Experiment Conditions and Results

A data file with records of the following structure "*S(worm name), population decrease*" was created of 100 data records size. „*Population decrease*“ values described the change of population size of infected hosts compared to population size at satiation phase during one month. Value assignment was based on worm descriptions from different AntiVirus software vendors, such as Symantec, MacAfee, etc. and information collected from security forums. Decrease rate was divided into 5 categories, presented in Table 3.2.

Table 3.2. Internet Worm population decrease categories

Category	Decrease rate
Low	0 %–20 %
Medium-low	20 %–40 %
Medium	40 %–60 %
Medium-high	60 %–80 %
High	80 %–100 %

The created data set cannot be treated as absolutely precise, since propagation rates from different information sources were used and this is challenge for further model improvements.

The lower is the decrease rate, the more stable population of infected hosts is created by the corresponding Internet worm strategy. The generated data file was supplied to the GAtree program for decision tree generation. Best genome score received was equal to 0.571, average genome score was equal to 0.517. The best generated decision tree was of size 43. Decision tree fragment is presented on Fig. 3.4.

The decision tree efficiency was tested against 5 worms with known population stability values that were not included in the training data file. All the test examples were classified correctly. Although the test data set is rather limited we can state, that approach used is effective.

```

...
| | |-'Low'
| | +-if HIER='Centralized hierarchy' then
| |   |-if COM='child-parent' then
| |     | |-'Medium'
| |     +- 'Medium-low'
| +- 'Low'
+-if TRANSF='UDP' then
  |-'Low'
  +-if MEM='- ' then
    |-'Low'
    +- 'Medium-low'
...

```

Fig. 3.4. Risk evaluation decision tree fragment

The main task of the decision tree created in our experiment is to assign the decrease rate value to the supplied for estimation propagation strategy of the Internet worm. This task is very important when a new virus epidemic starts, the worm's propagation strategy is already known and the estimation of worm's population stability after satiation phase is needed for countermeasures planning.

The training data set used cannot be treated as absolutely precise, since decrease rates from different information sources were used and this is a challenge for further model improvements. Still, this does not minimize the value of the model proposed for automatic malware caused risk evaluation since model approach was tested to be correct. The approach can be also expanded for other risk type evaluation.

3.6. Conclusions of Chapter 3

Present experimental research has been pursuing the following objectives: 1) creating the training data set, suitable for processing by the GAtree and dedicated to automatic evaluation risk, posed by newly appearing Internet worms; 2) evaluating if generated decision trees can evaluate the risk posed by a newly appearing Internet worms and proving the concept correctness.

The experiment resulted in generating of a population of decision trees, the best of which was used for test data evaluation. The proposed GA model can be characterized by the following statements:

1. The automated model of malware risk evaluation, based on decision tree generation by genetic algorithm and embracing data representation requirements, was proposed, which can be considered as the first fully automatic approach for the task specified, on the contrary to expert evaluations which are used now.
2. The model was applied for evaluating the Internet worm population stability threat. Model can be easily applied for other threats, posed by Internet worms, evaluation, since the strategy representation would remain untouched and only corresponding classification should be updated in the training data set and the decision tree should be automatically regenerated.
3. Model tests were performed against Internet worm samples that were not included in the training data set. The test samples were classified correctly and have proved the correctness of model approach.

Malware Evolution Forecasting Model

This Chapter provides the description of malware evolution forecasting model, based on genetic algorithms, that simulate the process of natural evolution to malware, which can be considered as a form of artificial life. Model is applied for three malware types: Internet worms, mobile malware and botnets. Evolution of two malware parameters: propagation techniques (for Internet worms, mobile malware, botnets) and survivability (for botnets) is being modelled. Model for Internet worm propagation technique evolution forecasting includes both friendly and hostile environment cases.

The experiments and results presented in this chapter were published in (Goranin and Cenys 2008a; Goranin and Cenys 2009; Goranin *et al.* 2010; Juzonis *et al.* 2010^{*}).

^{*} The references are given in the list of publications by the author on the topic of the dissertation

4.1. General Model Assumptions

Malware evolution forecasting model is important for scientific substantiation of expert prediction evaluation, understanding of malware development tendencies, forecasting of malware evolution by separate parameters or their complexes, development of countermeasure techniques and prevention of malware epidemic outbreaks by implementing quick response mechanisms.

The model is a physical, mathematical or logical representation of system entities, phenomena or processes (Defense 2001). Security models are useful when modelling the impact on the system and permits binding of the modelled computer system states with a real computer system (Garšva 2006a). Modelling allows forecasting the malware propagation consequences damage (Zou *et al.* 2002) and evolution trends (Goranin and Cenys 2008a*), understand the behaviour of malware, including spreading characteristics (Garetto *et al.* 2003), understand the factors affecting the malware spread, determine the required effectiveness of countermeasures in order to control the spread and facilitate network designs that are resilient to malware attacks (Ramachandran and Sikdar 2006), predict the failures of the global network infrastructure (Serazzi and Zanero 2004) and many other tasks that cannot be investigated without harm to production systems in the wild.

When we try to predict the malware evolution trends, first of all we have to stand on the position of cybercriminals and evaluate their aims while developing malware. The turn point to malware commercialization was discussed previously. Shift in malware development tasks has also changed its development style. The developers now should evaluate the requirements of their “clients” such as robustness, disclosure prevention, ease of use, protection of malware creator and operators, such as botowners, insuring botnet growth and stability, functionality and others.

Simulation environments serve many purposes, but they are only as good as their content (Banks and Stytyz 2008). While designing the model it is necessary to select main factors out of many and reject those that are not important or may cause result distortion. In case of GA modelling the main task consists of three parts: appropriate selection of chromosome structure, which represents the solution, definition of the fitness function and GA operating conditions, such as population size, mutation rates, parent selection, etc.

The model we propose is composed in a way to make it scalable and ease adjustable for different malware types and evolution forecasting parameters. In order to demonstrate this statement model is applied to several malware types,

* The reference is given in the list of publications by the author on the topic of the dissertation

environment conditions (friendly and hostile) and parameters, such as propagation techniques and survivability mechanisms. In case of the same malware type changes only to the fitness function are needed. Due to this reason high level representation of malware types is sometimes overloaded with genes that are not typical to the specific malware type, still they are left in representation, since evolution trends may change and inactive genes may be activated in future, so giving the malware new features.

4.2. Model Correctness Evaluation

Modelling results in case of genetic algorithm approach mainly depend on the correctness of the fitness function proposed and fullness of features represented in a chromosome. Mistakes in fitness function may lead to incorrect evaluation of solutions and insufficiency in methods presented can make the modelling results inappropriate to the tasks specified. Other model parts, such as operation parameters, mainly influence modelling speed or may lead to convergence to a local solution.

Since chromosome representation proposed here is based on the thorough malware technical analysis, provided in Chapter 2, and assumption is made that nearly all possible functionality is included, only the fitness function evaluation is selected to check the model correctness. GA operation parameters were selected on the “best-practice” basis, and several algorithm runs were performed in each case to prevent results from convergence to local minimum/maximum.

As in case with almost all forecasting models that can not be evaluated experimentally (climate change, catastrophe consequences, demographic, etc.) the malware evolution forecasting correctness can be evaluated only if applied to historic data. The correctness of fitness functions used for Internet worm and mobile malware evolution forecasting was tested on historical data, by applying for fitness evaluation some malware samples with known fitness, observed experimentally. In each case two samples were tested:

- CodeRed v.2 and Slammer in case of Internet worms;
- Locknut.A and Cabir in case of mobile malware.

Although some minor differences between obtained and observed values were found (less than 0.01), they were classified as not having significant influence on the final result.

Fitness functions for botnet evolution forecasting were based on the solution proposed for mobile malware, but no real tests were performed due to lack of reliable statistical data.

4.3. Internet Worm Evolution Modelling

Worms remain a significant part of all malware and may be defined as one of the major threats. The main issues faced in worm evaluation include the scale and propagation of the infections (Nazario 2004), that is why the Internet worms' propagation strategy was selected for its evolution forecasting.

4.3.1. Internet Worm Features Representing Chromosome

Each strategy is represented as a chromosome, which is combined of genes, i.e. combination of techniques and methods.

Genes are divided into compulsory (vitally necessary for worm to propagate, e.g., scanning gene, exploit gene or controlling non-compulsory genes) and non compulsory (i.e. giving a worm some functionality, that may or may not result in additional worm efficiency, e.g., remote administration function).

Compulsory genes are active in all chromosomes. Genes that are not compulsory have an additional activation gene (*AA* – always active genes, *AE* – active if enabled by activation gene). Such an assumption allows modelling any combination of methods and techniques and insures the fixed length of the chromosome.

Method grouping into genes is performed according to method tasks assigned, method relationship. Activation gene always goes before the gene being activated by it.

Chromosome structure is described in Table 4.1. As it can be seen from Table 4.1, many genes are related, for example disabling memory gene will effect in making hierarchy and communication genes ineffective even if they were enabled.

Despite this the proposed propagation strategy representation remains universal even in case such gene combinations are created in the initial population generation phase due to the nature of genetic algorithms, since lifeless combinations will be eliminated depending on the fitness function evaluation criteria during the evolutionary process and only the fittest will remain.

Chromosome structure and method/technique ascribing to the specific gene is based on technical Internet worm analysis provided in Chapter 2.2. In fact the representation provided simulates the modern malware creation principal, which highly depends on general software engineering techniques, such as modularity.

Table 4.1. Internet Worm feature representing chromosome

Nr. / Gene code (AA^a / AE) / Gene description	Range of values or sample values	Comments
1 / IP_GEN (AA) / Defines potential victim's IP address generation algorithm.	Random; Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast; Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast and LAN addresses; Differentiated random: /8 with X % probability, /16 - with Y % probability, Z % - fully random; Random /16 addresses in range from 128/8 to 224/8; Random addresses from networks reserved for home user networks (DSL, etc.); ...	X %, Y %, Z % - are generated randomly at initial population generation phase, in case a differentiated random algorithm was selected for the generated individual, representing propagation strategy. Since it is not possible to present all IP generation algorithms, used in the experiment, only a short representative selection was presented in "Range of values" column of IP_GEN gene. In a chromosome representing propagation strategy only a reference number to the IP generation algorithm, stored in an external array, is provided.
2 / OS_PLATF (AA) / Defines the OS platform the worm can function on.	DOS, *nix (Unix, Solaris, Linux), Win 9x, Win NT (NT, 2000, XP, Vista), Apple OS.	The proposed list of values is not finite. It only represents values that were used for the experiments.
3 / TRANSF (AA) / Defines worm's body transfer mechanism	Connectionless (also called "Fire and forget") Connection oriented	Connectionless mechanism uses UDP protocol for worm body transfer. Assumption is done, that worm can fit in one datagram. Connection oriented mechanism uses TCP protocol for worm transfer to the target host. For simplicity reasons we make an assumption that worm can be transferred in both methods, as for example attacking DNS (53/TCP, UDP) server.
<i>To be continued.</i>	<i>To be continued.</i>	<i>To be continued.</i>

Nr. / Gene code (AA ^a / AE) / Gene description	Range of values or sample values	Comments
<i>Continuation of Table 4.1.</i>		
4 / EXPL_1 (AA) / Defines the first exploit to be included in worm's body. The first exploit is compulsory, since at least one exploit is necessary for worm's propagation.	Random exploit from the array of exploits for the selected OS platform, for example exploit based on CVE-2004-0297	Array of exploits for each platform consists of 20 exploits with "remote" exploitation feature and is based on a list of Common Vulnerabilities and Exposures (CVE). Each exploit is assigned with a random percent number (due to the lack of statistical information), which defines the part of potentially vulnerable hosts (i.e. hosts running this software type) among all hosts running this OS platform.
5, 6, 7, 8, 9, 10, 11 / EN_EXPL_N (N=2-8) (AA) / Exploit EXPL_N activation gene.	True/False	Enables EXPL_N gene, if gene value is True. The maximum number of exploits (including compulsory) is limited to 8 for simplicity reasons.
12, 13, 14, 15, 16, 17, 18 / EXPL_N (N=2-8) (AE) / See EXPL_1 for description	See EXPL_1 for description.	See EXPL_1 for description.
19 / EN_MEM (AA) / MEM activation gene.	True/False	Enables MEM gene, if gene value is True.
20 / MEM (AE) / Defines type of memory the worm uses.	RAM, file, DB	Memory module is used to store information about communication paths. Additional information may also be stored.
21 / EN_HIER (AA) / HIER activation gene.	True/False	Enables HIER gene, if gene value is True
22 / HIER (AE) / Defines worm network hierarchy	Autonomous Centralized hierarchy; Decentralized hierarchy.	Autonomous – each infected host can act as a management source; Centralized hierarchy – there exists only one management center; Decentralized hierarchy – several management centers exist on a network.
<i>To be continued.</i>	<i>To be continued.</i>	<i>To be continued.</i>

Nr. / Gene code (AA ^a / AE) / Gene description	Range of values or sample values	Comments
<i>Continuation of Table 4.1.</i>		
23 / EN_COM (AA) / COM activation gene.	True/False	Enables COM gene, if gene value is True.
24/ COM (AE) / Defines worms' communication algorithm.	Communicate via child/parent chain; Communicate via direct connection between infected and management hosts.	In a child/parent chain each individual worm knows only the connection to its parent, i.e. the host from which it was infected and parent knows only these hosts that were infected by him. This mechanism may be used in autonomous, centralized and decentralized hierarchies. Direct communication assumes that each infected host stores information, that is used to achieve management host (in case of centralized h.), some or part of management hosts (in case of decentralized h.)
25 / EN_EXEC (AA) / EXEC activation gene.	True/False	Enables EXEC gene, if gene value is True
26 / EXEC (AE) / Defines remote worm management features.	Standard functionality Update functionality Standard + Update functionality	Standard functionality – supports remote execution of functions, included in the worm's body or functions of a compromised host's OS; Update functionality – supports update or change of any worm module. Standard + Update – combined functionality of the two previous techniques.
27 / EN_ADD (AA) / ADD activation gene.	True/False	Enables ADD gene if gene value is True.
28 / ADD (AE) / Defines additional worm functionality features.	Deface local host; write to MBR to remain after reboot; DDoS support; or any combination of these techniques.	Assumption is done, that additional functionality is performed prior to begging of IP generation and exploit transfer.
<i>To be continued.</i>	<i>To be continued.</i>	<i>To be continued.</i>

Nr. / Gene code (AA ^a / AE) / Gene description	Range of values or sample values	Comments
<i>Continuation of Table 4.1.</i>		
29 / EN_EVOL (AA) / EVOL activation gene.	True/False	Enables EVOL gene if gene value is True.
30 / EVOL (A) / Defines worm's evolution.	... If only one exploit is used, then after 100 generations enable EXEC gene with Update functionality, enable additional exploits, update the worm's body with additional exploits. ...	Evolutionary gene allows the change of worm's propagation strategy after defined number of worm generations or according to propagation conditions analysis (propagation slow-down). Since in this experiment we model worm's propagation in the initial stage, evolutionary genes do not play an important role.

A sample strategy, generated during the initial population generation is presented on Fig. 4.1.

```

Si={IP_GEN="Random,    excluding    127.0.0.0/8,    loopback,
224.0.0.0/8,    multicast";    OS_PLATF="Apple    OS";
TRANSF="Connection    oriented";    EXPL_1="    CVE-2007-3876";
EN_EXPL_2="False";    EN_EXPL_3="False";    EN_EXPL_4="False";
EN_EXPL_5="True";    EN_EXPL_6="False";    EN_EXPL_7="False";
EN_EXPL_8="False";    EXPL_2="-";    EXPL_3="-";    EXPL_4="-";
EXPL_5="    CVE-2004-0485";    EXPL_6="-";    EXPL_7="-";
EXPL_8="-";    EN_MEM="False";    MEM="-";    EN_HIER="True";
HIER="Autonomous";    EN_COM="False";    COM="-";
EN_EXEC="True";    EXEC="Update    functionality";
EN_ADD="True";    ADD="Write to MBR to remain after reboot";
EN_EVOL="False";    EVOL="-"}

```

Fig. 4.1. Sample Internet worm strategy randomly generated

The strategy generated may be composed of any possible combination of genes, since its vitality is evaluated during the fitness evaluation process. During the mutation process the gene selected for mutation changes its current value to a random from the range of possible values.

4.3.2. Genetic Algorithm Operation Parameters

Initial population is generated on a random basis, i.e. each individual, representing separate worm propagation strategy is combined of random genes' values. Population size N is equal to 50. Population size remains constant after each new generation. The combined termination condition was selected. Algorithm would stop producing new generations in two cases: either number of generations have reached 100, or the fitness evaluation of the fittest individual in a population remains constant for 10 generations. Crossover point for each pair of parents is selected randomly and defines the number of gene, after which the crossover operation is performed. The mutation operator defines the gene number of a newly generated individual that should change value from that prior to mutation to any other random value from the range of possible gene values. Mutation operator is activated to each newly generated individual with a 0.005 probability. Fitness proportionate selection was used. After the assignment of fitness value to an individual its selection probability is calculated according to Eq. 4.1, that defines the individual's probability to be selected as a parent.

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j}, \quad (4.1)$$

where p_i is the probability and f_i is the fitness of individual i in the population in the population consisting of N (50 in our case) individuals. In such a way more individuals with higher fitness value get the higher chance to leave the offspring then those with lower.

4.3.3. Fitness function

The task of our experiment is to create a worm's propagation strategy that aims infecting the largest amount of computers in a limited period of time. From (Staniford *et al.* 2002) we can say that the propagation strategy efficiency can be evaluated by value K , i.e. the number of computers the first worm individual in the wild can infect in a fixed time period. That means that the higher is K , the higher is the fitness of a propagations strategy. Our K calculations by fitness function are based on combined statistical or empirical (in case of statistical data lack) evaluation of time expenditures of strategy's functionality and probabilistic evaluation of strategy's functionality efficiency. The fitness function we used is the following:

$$K(S) = k \cdot p_1^S \cdot p_2^S \cdot p_3^S \cdot \sum_{i=4}^{30} p_i^S, \quad (4.2)$$

where: S – evaluated strategy; p_1^S – probability, that the generated IP address exists and alive, p_2^S – probability that host is running the OS platform that the worm supports, p_3^S – probability, that worm will be successfully transferred to the potential victim, p_i^S – probability, that gene (Nr. 4–30) will result in an infected host; k – the number of cycles the worm using the evaluated strategy can perform in one second time interval which is calculated according to Eq. 4.3:

$$k = \frac{1}{\sum_{j=1}^{30} t_j^S}, \quad (4.3)$$

where t_j^S are time expenditures needed for gene functionality.

In other words the following maximization task is being solved:

$$\max_S (K(S)). \quad (4.4)$$

So the fitness function can be read as: “Strategy S can perform k cycles per second. During each cycle the worm using this strategy will infect a host in case the generated IP address exists, the host is up and running the OS platform the worm supports, worms is successfully transferred to the target and any other gene results in host infection. The received calculated value of the evaluated strategy is its K value.”

The fitness value for the sample strategy, presented on Fig. 4.1. would be equal to $K(S_i) = 0.0058$ ($k = 5.13$). The values used for calculation are presented in Table 4.2:

Table 4.2. Sample S_i Internet worm strategy fitness calculation parameters

Gene #	1	2	3	4	5	6	7	8
p	0.6	0.04	0.95	0.04	0	0	0	0
t	0.01	0.005	0.15	0.005	0	0	0	0
Gene #	9	10	11	12	13	14	15	16
p	0	0	0	0	0	0	0.01	0
t	0	0	0	0	0	0	0.005	0
Gene #	17	18	19	20	21	22	23	24
p	0	0	0	0	0	0	0	0
t	0	0	0	0	0	0.01	0	0
Gene #	25	26	27	28	29	30		
p	0	0	0	0	0	0		
t	0	0.005	0	0.005	0	0		

For comparison the received K value for CodeRed v.2 worm in case it would use only one thread instead of 100 is much higher and equal to 0.015. In case of 100 threads K would reach 1.5, which is rather similar to the observed CodeRed v.2 K value of 1.6 (Serazzi and Zanero 2004). Differences between calculated and observed values of K can be explained by not exact time consuming estimations.

4.3.4. Model Limitations

Simplification is done while evaluating exploits' probabilities: probabilities are assumed to be rather low, i.e. less than 0.05; probabilities of all exploits are independent, i.e. host can be infected only by one of the exploits. It is also obvious that genes Nr. 19–30 do not provide additional probability to host infection (i.e. $p=0$) and differ only in time consumption. It is assumed, that exploits can be transferred both by TCP and UDP protocol. An Internet worm is supposed to propagate in a friendly environment, facing no countermeasures.

4.3.5. Experiment Results

Algorithm was implemented using the Object Pascal (Delphi) programming language. While modelling the evolution of propagation strategies, that aim infecting the maximum number of hosts in a limited period of time, 10 algorithm runs were performed.

The highest K value obtained was equal to 1.18 (equivalent to 118 in case of 100 threads) and corresponded to the rather simple strategy that got use of Windows platform ($p=0.9$), random (excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast) IP generation mechanism, connectionless UDP based transfer mechanism ($p=0.8$, $t=0.01$) and 4 exploits with rather high to high probabilities (0.05; 0.05; 0.045; 0.035). All other genes were disabled. The change in the fitness of the whole population and the fitness of the best individual in case when the highest K value was obtained, can be seen on Fig. 4.2.

Calculated fitness of the best strategy obtained during the experiment was 79 times higher than that of the most famous Code Red v.2 worm. This allow saying that consequences of future Internet worm epidemic outbreaks can be more devastating than caused by modern Internet worm samples.

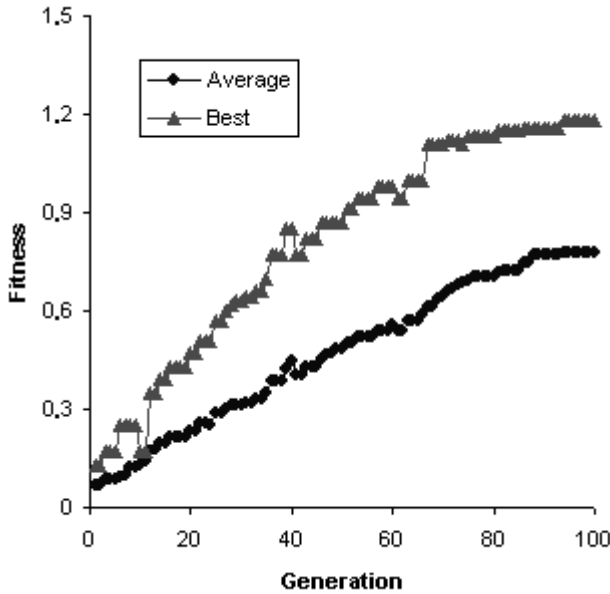


Fig. 4.2. Change in fitness of the whole population and the best individual (Internet worms)

Although we cannot claim that the found solution is optimal, the strategy of using simple “fire and forget” mechanism on a popular platform and a combination of several exploits with high probability seems to be very effective and challenging for countermeasure planning. Partially the proof of this was provided by the creators of the Slammer worm, which used only one exploit but still was much quicker than connection oriented CodeRed v.2 (Slammer doubling time was 8.5 s compared to CodeRed v.2 37 min (Serazzi and Zanero 2004)).

4.3.6. Fitness Function and Modelling in Hostile Environment

In order to evaluate countermeasures efficiency on worm propagation it is necessary to classify them. We propose using the countermeasures taxonomy proposed in (Brumley *et al.* 2005), which contains several countermeasure types: Reactive Antibody Defence (signatures, patching after worm break-out); Reactive Address Blacklisting (blocking the connections from known infected hosts); Proactive Protection (universal system hardening based on worm disorientation); Local Containment (“good neighbour” blocking the outgoing worm scans if infected).

We do not evaluate the technical problems related with the deployment of each countermeasure type, but it is obvious that their deployment is time dependant, since it takes time to prepare signatures, disseminate and constantly update blacklist, etc. It is also unarguable that worm spread becomes time dependant and the rate will decrease not when satiation phase is reached but much earlier. In that case Eq. 4.3 can be rewritten as

$$K_S(t) = k \cdot p_1^S(t) \cdot p_2^S \cdot p_3^S(t) \sum_{i=4}^{30} p_i^S(t). \quad (4.5)$$

Variable p_2^S is not time-dependant since we assume that the number of computers running the OS is constant (negligible percent of users will change OS for example from Windows to Linux or vice versa in case a new worm appears) and the disorientation measures will effect in exploit efficiency. Each $p^S(t)$ can be described as a curve which shows the decrease of probability. In real life all countermeasures would be used in combination. Due to that fact the function $p(t)$ representing the probability decrease in time would be an approximation of the real statistical data. Currently no such data is available and systematic data collection is needed in order to create such curves.

Eq.4.4 could be used to draw the curve of a specific worm propagation strategy that would be decreasing in time. In order to compare efficiency of different strategies under pressure of countermeasures we propose the new fitness function:

$$F_{SC}(K_S(t)) = \frac{dK_S(t)}{dt}, \quad (4.6)$$

which is equal to time derivative of K_S . The derivative shows the strategy's efficiency decrease rate. The lower is the decrease the more efficient the strategy is, i.e. the minimization task is being solved:

$$\min_S (F_{SC}(K_S(t))). \quad (4.7)$$

All other model assumptions and limitations do not change. We could not check the efficiency of the proposed model extension due to the lack of statistical data but the model proposed allows modelling of Internet worm evolution under pressure of countermeasures. It is also important to note that different countermeasure proportions may lead to different probability curves and worm strategy evolution.

4.4. Mobile Malware Evolution Modelling

The model proposed for mobile malware evolution forecasting is based on the model proposed for Internet worms and described in Chapter 4.2 with some modifications, adapting it for mobile malware specifics, and improvements, related to model limitations and fitness function. Mobile malware propagation strategy evolution is being evaluated.

4.4.1. Mobile Malware Features Representing Chromosome

Strategy representation is rather similar to that proposed for Internet worms and presented in Table 4.3. Chromosome structure and method/technique ascribing to the specific gene is based on technical mobile malware analysis provided in Chapter 2.5.

Compared to Internet worms five propagation methods are possible (by MMS, SMS, Bluetooth, e-mail and WI-FI). Additional genes that implement mobile malware ability to hide themselves by some period of inactivity are introduced (NR_TIME, BT_TIME, WIFI_TIME). Genes, that describe telephone platforms (mobile devices) are also introduced, operating system defining gene is added with values, that represent operating systems running on mobile device (Symbian, Win Mobile, etc.)

Table 4.3. Mobile malware feature representing chromosome

Gene number / Name / Type / Description / Comments	Value range or sample values
1 / TRANSF1 / AA / Defines the 1st supported propagation type / Enables NR	MMS
2 / TRANSF2 / AA / Defines the 2nd supported propagation type / Enables NR	SMS
3 / TRANSF3 / AA / Defines the 3rd supported propagation type / Enables BT	Bluetooth
4 / TRANSF4 / AA / Defines the 4th supported propagation type / Enables EMAIL	e-mail
5 / TRANSF5 / AA / Defines the 5th supported propagation type / Enables WIFI	Wi-Fi
6 / NR / AE / Telephone number search or generation module / Effective if SMS or MMS transfer methods.	Address book; Accepted/ Dialed numbers; Random.
<i>To be continued.</i>	<i>To be continued.</i>

Gene number / Name / Type / Description / Comments	Value range or sample values
<i>Continuation of Table 4.3.</i>	
7 / BT / AE / Scanner module, that searches for mobile devices with Bluetooth support.	Scan
8 / EMAIL / AE / E-mail sending module	Address book; e-mail address DB.
9 / WIFI / AE / Scanner module, searches for mobile devices with WIFI support.	Scan
10 / OS_PLATF / AA / OS platform affected by malware	Win Mobile; Symbian; etc.
11 / TEL / AA / Telephone models, affected by malware	NOKIA, SAMSUNG, Apple,...
12,13,14 / EN_EXPL_N / AA / EXPL_N (N=1-3) activation gene	ON=ExploitRef / OFF
15,16,17/ EXPL_N(N=1-3) / AE / Defines the exploit used for propagation	Random exploit out of suitable exploit array
18 / NR_TIME / AA / Defines the NR gene's activity hours	Always; 10:00-20:00; 20:00-10:00
19 / BT_TIME / AA / Defines BT gene's activity	Always; 10:00-20:00; 20:00-10:00
20 / WIFI_TIME / AA / Defines WIFI gene's activity	Always; 10:00-20:00; 20:00-10:00
21 / EXEC / AA / Defines additional malware functionality / Activates EXEC_CHAN	None; Manage; Update; Manage+Update
22 / EXEC_CHAN / AE / Defines malware update type	e-mail; WI-FI; web-update

4.4.2. Genetic Algorithm Operation Parameters

GA operation parameters are almost similar to Internet worms' model. Initial population is generated on a random basis. Population size N is equal to 50. Population size remains constant after each new generation. The algorithm would stop producing new generations in case the number of generations have reached 100. Fitness proportionate selection was used.

4.4.3. Fitness Function

The same fitness evaluation method, based on the Random Constant Spread model (Staniford *et al.* 2002) is used. We solve the maximization task (Eq. 4.8):

$$\max_S (K(S)). \quad (4.8)$$

K calculations for mobile malware propagation strategy evolution forecasting are performed by Eq. 4.9:

$$K(S) = k \cdot \left(1 - \left(1 - p_6^S(NR_TIME) \right) \cdot \left(1 - p_7^S(BT_TIME) \right) \right) \cdot \left(1 - p_8^S \right) \cdot \left(1 - p_9^S(WIFI_TIME) \right) \cdot p_{10}^S \cdot p_{11}^S \cdot \left(1 - \prod_{i=15}^{17} \left(1 - p_i^S \right) \right), \quad (4.9)$$

where: S – evaluated strategy; p_6^S – p_9^S – probability, that exploits will be successfully transferred to the target device (p_6^S , p_7^S and p_9^S are time dependant); p_{10}^S – probability, that the target device will run the supported OS; p_{11}^S – probability, that device hardware is compatible; p_{15}^S – p_{17}^S – probabilities, that exploit will result in infection; k – the number of cycles the virus, using the evaluated strategy, can perform in one second time interval. k is calculated by Eq. 4.6:

$$k = \frac{1}{\sum_{j=1}^{22} t_j^S}, \quad (4.10)$$

where t_j^S are time expenditures needed for j^{th} gene functionality.

The fitness function can be read as: “The evaluated strategy S can perform k cycles per second. During each cycle the virus, using this strategy, will infect a target host in case at least one of the transfer methods successfully transfers the exploits to the target, the target runs the supported OS on the supported platform and at least one of exploits result in target infection.

Compared to model for Internet worms limitations for probabilities’ size were removed, since the fitness function of a different logic is proposed.

4.4.4. Experiment Results

Algorithm was implemented using the MATLAB program. The best fitness result achieved during algorithm test was equal to $K(S_d) = 0.023$. Compared to fitness of a sample strategy $K(S_p) = 0.017$ of the current mobile malware

(Transfer method – MMS only; OS platform – Symbian; Telephone platform – NOKIA; activity hours – Always; Numbers used – Address book; one exploit) fitness of the predicted mobile virus has increased almost 1.7 times.

The fitness change during evolution of the best individual is shown on Fig. 4.3, average population fitness change – Fig. 4.4. It should be noticed, that general population fitness also increases in time and that the number of individuals with “better” strategies increase even though the best individual evolution stops after the 42 generation.

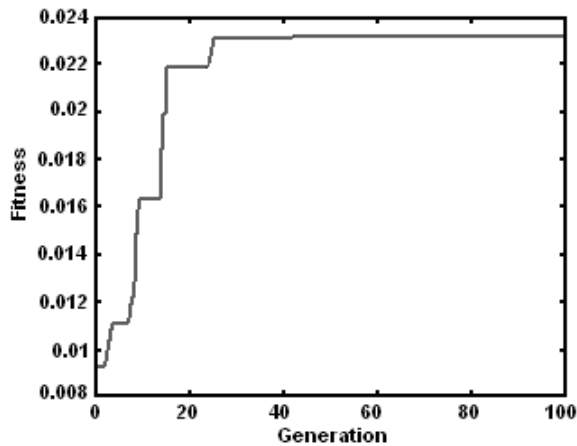


Fig. 4.3. Best strategy fitness change graph (Mobile malware)

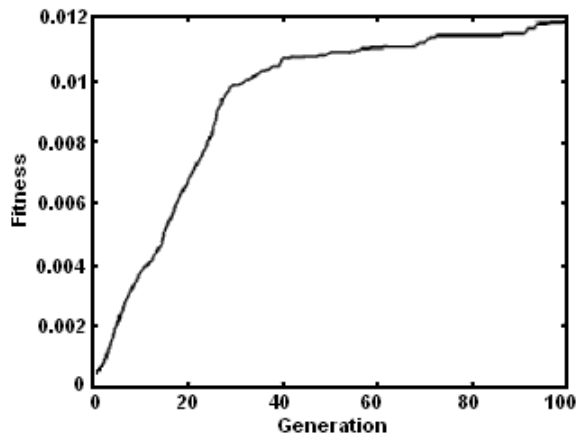


Fig. 4.4. Average population fitness change graph (Mobile malware)

Compared to the sample strategy the following functionality (genes) was enabled in the best strategy during evolution: Windows mobile support, WI-FI transfer method support. We can make an assumption that these methods were included since they provide rather high infection efficiency (additional popular OS and W-FI with relatively high network coverage). Other potentially efficient methods were not included since their added-value to propagation efficiency was neglected by time consumption, other methods do not result in infection at all (additional functionality) or even minimize the propagation rate (e.g. limitation by hours).

The fitness increase was not as impressive as in Internet worms' case (only 1.7 times compared to a sample modern sample). This can be explained by technological differences (e.g. Bluetooth area coverage, connection approval necessity), limited number of exploits available, low infection probabilities, compared to computer networks (this is especially important due to effective countermeasures, such as centralized antivirus solutions, that can monitor MMS traffic at mobile operator host), and rather limited value of additional propagation methods.

4.5. Botnet Evolution Modelling

It is difficult to measure the extent of damage caused on the Internet by botnets, but it is widely accepted that the damage done is significant (Grizzard *et al.* 2007). Rajab *et al.* assumes that botnets are a major contributor of unwanted Internet traffic and 27 % of all malicious connection attempts can be directly attributed to botnet-related spreading activity (Rajab *et al.* 2006; Rajab *et al.* 2007). The botnet size estimation vary a lot, especially due to the fact that hackers frequently attack large numbers of easy-to-compromise home computers (Wash 2008), the number of which can not be measured accurately (they are not constantly on-line, no registration in DNS servers, dynamic IP's, etc.) (Zhuge *et al.* 2007). Zhuge *et al.* state that they tracked 3 290 IRC-based botnets during the measurement period between June 2006 and June 2007 and in total observed about 700 700 distinct IP addresses, the biggest botnet observed they tracked has controlled more than 50 000 hosts. According to ENISA report (Barroso 2007) each botnet has an average of 20.000 compromised computers (bots): some C&C servers manage just a few infected computers (~10), large ones manage thousands of bots (~300 000).

As stated in (Banks and Stytz 2008) the development of botnet simulation and modelling capabilities requires advances in improving understanding of botnet technologies and the development of standards that support the simulation of bot army operations, but these tasks are complex for a variety of reasons, such as

wide variety of botnets and their manner of propagation, challenge posed by modelling the amount of time and patterns of their infestation. That is why we say that GA approach for modelling botnets is extremely efficient due to its ability to solve complex problems with large solution space. On the other hand, due to lack of reliable statistical data on botnet sizes, expansion speeds and effectiveness of different techniques used by botnets here we provide only the model as proof-of-concept, which includes botnet strategy representation chromosome description, which may be used to evaluate different modelling characteristics (propagation, survivability, manageability, economic effectiveness evaluation, etc.) and the fitness function which may be used for botnet propagation and survivability mechanism evolution forecasting. The model may be used and tuned when effective data collection methods about botnets will appear.

4.5.1. Botnet Features Representing Chromosome

The proposed strategy representation via chromosomes is provided in Table 4.4. Chromosome structure and method/technique ascribing to the specific gene is based on technical botnet analysis provided in Chapter 2.4.

Since compared to Internet worms botnets are more flexible (use parallel propagation methods, complex functionality, etc.) we propose using a reference system for gene activation and method definition (selection from maximum 9 methods or gene deactivation).

In Table 1 gene (fixed-length number – 10 positions) in a “Gene code” column just activates one or several methods from a reference database (samples are provided in the “Reference database (or sample)” column). “0” marks references to methods that do not exist, and other digits (1–9) references to the reference database.

The number of non-zero digits shows the number of methods activated. There can not be the same non-zero digits in one gene (e.g., 4510000700 is “OK”, but 4550000700 is not). This check is performed during the initial population generation phase. Non-zero digit order is not important (that means that in case gene is 4510000700, then method 4, 5, 1 and 7 are active).

In case of a 0000000000 gene this means that gene is not active and no methods are used. If gene is compulsory for botnet but is not active such an individual (strategy) will be simply eliminated by evolutionary selection process.

Table 4.4. Botnet feature representing chromosome

No./ Gene code Gene description	Reference database (or sample)	Comments
1 / TARGET_SEA RCH/ Defines methods used by botnet for potential victim search	1) Scan – Random; 2) Scan - Random, excluding 127.0.0.0/8, loopback, 224.0.0.0/8, multicast, LAN addresses; 3) Scan - Random addresses from networks reserved for home user networks (DSL, etc.); 4) Non-automatic infection (i.e. manual); 5) E-mail spam – Malware / Link to malware; 6) Instant messaging; 7) Infected site; 8) Removable media; 9) P2P	Each method may use from 1 to 9 exploits. The necessary number of exploits with referencing exploiting probability are selected from the list of exploits. Limitations: exploits used for one method should run on a single platform.
2 / TRANSF / Defines exploit body transfer mechanism	1) Connectionless (also called “Fire and forget”) 2) Connection oriented	Connectionless mechanism uses UDP protocol for exploit body transfer. Assumption is done, that it can fit in one datagram. Connection oriented mechanism uses TCP protocol for exploit transfer to the target host. For simplicity reasons, we make an assumption that both methods can be used, as for example attacking DNS (53/TCP,UDP) server. This is important in case botnet scans network for vulnerable hosts. In case other methods are used then they are always considered to be “Connection oriented”.
3 / EXPL_PLATF / Defines the OS platform used for each method	1) DOS 2) *nix (Unix, Solaris, Linux) 3) Win9x 4) Win NT (NT, 2000, XP, Vista) 5) Mobile OS 6) Apple OS 7) Multi platform 8) Other OS 9) WEB application exploit	Despite the number of exploits used they all should run on a single platform. Different methods may run on different OS platforms. WEB based exploits are those that exploit vulnerabilities of WEB-based applications that do not depend on the OS. Mobile OS covers a variety of mobile OS platforms.
<i>To be continued</i>	<i>To be continued.</i>	<i>To be continued.</i>

No./ Gene code Gene description	Reference database (or sample)	Comments
<i>Continuation of Table 4.4.</i>		
4 / EXPL_NUM / Defines number of exploits used by each method	[0..9] (random number)	A random number of exploits used by each method, activated in TARGET_SEARCH. For example if TARGET_SEARCH gene is 4510000700 and EXPL_NUM is 1210000300, that means that the 4 th method is activated and it uses only one exploit, 5 th method uses 2 exploits, etc.
5 / HIERARCHY / Defines the hierarchy used for communication	1) Central – one management host; 2) Central – one management host – botnet is splitted, when certain number of bots is achieved. After that operates as two (or more botnets) with different manament hosts, but managed by the same botmaster. The total number of bots in all botnets is calculated when evaluated; 3) Central – several (2-9) management hosts – independent; 4) Central – several (2-9) management hosts – load balancing; 5) Central – several (2-9) management hosts – fast-flux protection; 6) Central – several (2-9) management hosts – fast-flux protection – load balancing; 7) Belongs to several botnets (2–9) with central management hierarchy; 8) Decentralized – P2P principal; 9) Decentralized – P2P – fast-flux technology	In this gene only the first number out of 10 has sense and defines the hierarchy used for management. Other are just “0”.
<i>To be continued</i>	<i>To be continued</i>	<i>To be continued</i>

No./ Gene code Gene description	Reference database (or sample)	Comments
<i>Continuation of Table 4.4.</i>		
6 / FUNCTIONALITY / Defines the functionality that botnet provides for botnet owners.	1) Information collection (credentials stealing, keylogging); 2) Backdoor opening (using various listening ports), execution of commands and programs; 3) Botnet owner notification about the Compromise; 4) Packet sniffing; 5) DDoS functionality; 6) Spam sending; 7) Remote update and deinstallation of the installed bot; 8) Botnet rental tools; 9) Botnet management tools (ease-of-use)	This list is not complete since limited to the number of 9, which is selected for the reason of simplicity.
7 / SELF_PROTECT / Defines methods that a single bot can use to protect itself against deinstallation, management blocking or analysis	1) Blocking Firewall/Antivirus processes; 2) Blocking Antivirus Updates; 3) Blocking OS updates; 4) Deinstallation imitation, if detected by Antivirus with no real deinstallation; 5) Imitation of usefulness (for e.g., imitation of Antivirus – “Antivirus 2009”); 6) Period of inactivity; 7) Low activity; 8) De-installation if “honeypot” or “sand-box” or analysis is suspected; 9) Social engineering (language selection, advertisements, related with latest or known events)	This list is not complete since limited to the number of 9, which is selected for the reason of simplicity.

It is necessary to note that compared to Internet worms and mobile malware there is a limited number of genes that can be “inactive”. In *TRANSF_METHODS* at least one method should be enabled, *TRANSF* – one method should be enabled, *EXPL_NUM* – at least one should be enabled, *EXPL_PLATF* – at least one should be enabled, *HIERARCHY* – one method should be enabled, *FUNCTIONALITY* – at least “*Remote update and deinstallation of the installed bot*” should be enabled or any other method should be enabled, *SELF_PROTECT* – not compulsory, but such strategies will be eliminated during the evolution process since such strategy will be non-vital if botnet survivability mechanisms are evaluated.

4.5.2. Genetic Algorithm Operation Parameters

The main difference from the Internet worm and mobile malware models is that due to lack of reliable statistical data on botnet sizes, expansion speeds and effectiveness of different techniques used by botnets here we provide only the model description with no real data tests. No changes to GA operation conditions (population size, termination conditions, crossover conditions, mutation frequency, selection strategy) are proposed. This is mainly due to the fact that tuning these parameters may be effective only if using real statistical data. Here we provide only a proof of concept or a model that can be used later, when effective data collection methods about botnets will appear in scientific press.

4.5.3. Fitness Function – Propagation Technique Evolution Forecasting

Here we propose fitness function that allows forecasting botnet strategy evolution of population increase characteristics (Eq. 4.11.). As with Internet worms and mobile malware the fitness function for botnet propagation technique evolution forecasting solves the maximization task:

$$\max_S (K(S)), \quad (4.10)$$

but more complex, since it represents much more complex malware type.

$$\begin{aligned} K(S) = & k \cdot k_1 \cdot k_2 \cdot k_3 \cdot \left(1 - \prod_{i=1}^9 (1 - p_i^{1,S}) \right) \cdot \left(1 - \prod_{i=1}^9 (1 - p_i^{2,S}) \right) \\ & \cdot \left(1 - \prod_{i=1}^9 (1 - p_i^{3,S}) \right) \cdot \left(1 - \prod_{i=1}^9 \prod_{j=1}^{E[i]} (1 - p_{ij}^{4,S}) \right), \end{aligned} \quad (4.11)$$

where: S – strategy being evaluated; p_i^1 – probability, that the i^{th} method (*TARGET_SEARCH*) will find the live host; p_i^2 – probability, that the i^{th} method (*TRANSFER* method, corresponding to the i^{th} *TARGET_SEARCH* method) will successfully transfer the exploit to the target host; p_i^3 – probability, that the i^{th} method's (*EXPL_PLAT*) supported platform will be the same with the target host; p_{ij}^4 – probability, that the j^{th} exploit (out of $E[i]$ exploits enabled by the i^{th} method (*EXPL_NUM*) will infect the target computer; k – the number of cycles the bot, using the evaluated strategy, can perform in one second time interval; k_1 – coefficient of the hierarchy effectiveness on the propagation rate; k_2 – coefficient of the functionality effectiveness on the propagation rate; k_3 – coefficient of the self protection functionality on the propagation rate. k is calculated according to the following expression:

$$k = \frac{1}{\sum_{i=1}^7 \sum_{j=1}^9 t_{ij}^S}, \quad (4.12)$$

where t_{ij} are time expenditures needed for j^{th} method of the i^{th} gene, $i=1,...,7$, $j=1,...,9$.

Methods activated by genes 5–7 do not directly influence the propagation rate (except the case, that time is needed for their transfer and changes the k coefficient), since they do not carry any payload or provide any direct actions for propagation. But since we do not have enough statistical data regarding botnet propagation and influence of different functionality and organizational structure on the propagation rate we cannot state, that these genes do not influence propagation at all.

It is possible that some functions may increase or decrease the propagation rate due to some indirect qualities, for example hierarchy may be used to optimize the scan target, or blocking the antivirus would minimize the period of inactivity, etc. That is why we have introduced k_1 , k_2 , k_3 coefficients for *HIERARCHY*, *SELF_PROTECTION* and *FUNCTIONALITY*. These coefficients should be based on statistical evaluations and would be equal to 1, if no influence is detected, < 1 if influence is negative (slow the propagation rate) and > 1 if influence is positive.

So the fitness function can be read as: “The bot, using strategy S for propagation can perform k cycles per second, propagation rate is influenced by coefficients k_1 , k_2 and k_3 , which correspond to influence of hierarchy, functionality and self-protection features respectively. Propagation is successful if at least one of the target search methods finds the “live” target, the bot is successfully transferred to the target, at least one of the platforms, supported by the bot is found on the target, and at least one of all exploits at any supported platform resulted in infection.”

When the botnet faces the satiation phase or countermeasures it has to change the propagation strategy to overcome the stagnation or decrease in population and start growing again. For each stage (initial, satiation, population decrease) we propose performing a separate experiment. In order to insure inheritance between phases we propose using rather high rate of elitism (25 % of the most effective strategies should be moved from one phase to another). This is a concept of elitism between phases, not generations.

4.5.4. Fitness Function – Survivability Technique Evolution Forecasting

For botmasters it is necessary to insure that botnet will be stable (functional, manageable and of a relatively fixed size) for a time period T , necessary to fulfil the botmasters tasks (spam sending, DDoS, rental, etc.). T is task dependant and survivability mechanisms activated in botnet to insure the necessary stability level in each case will be different.

The botnet stability may be discussed in two aspects: hierarchy stability, which insures botnets overall functionality and stability (e.g., if C&C is blocked the bots become useless, even if not detected by antivirus programs) and stability of botnet nodes – bots, i.e. probability, that the bot will not be removed from the botnet network by different countermeasures. To insure stability botnet creators implement different survivability (or protection) mechanisms.

In case we want to evaluate the botnets' evolution only in case of survivability, we can say that hierarchy stability is directly proportional to the number of C&C used and corresponding protection mechanisms and can be calculated according to Eq. 4.13 and GA modelling is not needed since the evolution trend is clear and the number of possible trends is very limited:

$$N_{CC} = \frac{T_s}{t_{CC_block}(hierarchy_Nr)}, \quad (4.13)$$

where N_{CC} – number of C&C, T_s – time interval necessary for botnet to remain stable, t_{CC_block} – the average time needed for botnet fighters to block the C&C of a specific hierarchy ($hierarchy_Nr$), using or not using some self-protection measures. In reality the botnet creator's decision would be based on hierarchy implementation economic evaluation or realization complexity.

Another botnet stability part – bot or node stability has much more survivability mechanisms combinations possible which are dependant on a variety of function bot has to perform and evolution trends are not so clear.

That is why we propose applying the GA model and survivability mechanism evaluation via fitness-based process. We state that the bot's survivability on the node's level can be evaluated by Eq. 4.14, which uses probabilistic and time consumption parameters for methods activated by genes for each of the strategies:

$$F(S) = k \cdot \prod_{i=1}^9 \left(1 - p_i^S (self_protect) \right), \quad (4.14)$$

where S – botnet strategy being evaluated; p_i^S – probability, that the i^{th} method (*SELF_PROTECT*) will protect the botnet node bot from detection and removal; k – bot’s activity level (e.g., sniffing, spam sending, DDoS performing, etc.), which directly influences the efficiency of self-protection measures, employed by the bot, since the higher is the bot activity level, the bigger is probability, that it is noticed and removed. k is calculated by Eq. 4.11:

$$k = \frac{t(S)}{T_S}, \quad (4.15)$$

where:

$$t(S) = \sum_{j=1}^7 \sum_{i=1}^9 t_i^S \cdot \frac{CPU_LOAD_i}{100\%} \quad (4.16)$$

is the summary of strategy’s S time consumption in the evaluated time period T_S , t_i^S – time consumption of a specific method (j – gene’s index, i – method’s index of the j^{th} gene), CPU_LOAD_i – average method’s load on the infected computer’s CPU during time t_i^S .

CPU load is selected as one of the most descriptive computer process activity parameters (compared to network or hard disk (HDD) load). Model limitation is introduced, that all computers included in the botnet run almost similar CPUs by the processing power. Here we maximize the $F(S)$ by S (Eq. 4.17):

$$\max_S (F(S)). \quad (4.17)$$

So the fitness function can be read as: “The bot, using strategy S will be detected and removed depending on its activity level k and none of the self protection methods will be effective (inverse probability to the effectiveness of survivability mechanisms). According to such definition the higher is the fitness function value for the evaluated strategy the less is its fitness in sense of the bot’s survivability and stability of the whole botnet. We do not analyze the botnet size influence on the stability, since it is not absolute but a relative value.

Destruction of a big botnet’s nodes will certainly take more time than that of a small one, but sometimes for botnet owners creation of huge botnets is not necessary and stability of a small or mid-size botnet is very important. Here we do not analyze the case when some part of the botnet population (e.g. 35 %) uses one survivability mechanisms and another (65 %) a bit or absolutely different.

4.5.5. Model Limitations

It is necessary to notice that compared to the Internet worm and mobile malware the model described for botnets is more flexible and compact (possibility to use several target search methods; support of OS platforms in any combinations; increase of supported exploits to up to 81; the chromosome representation by digits is more universal and nearer to the “classical” GA representation; methods used by different genes are grouped in a compact manner), the model can be used to model botnet characteristics not only at different stages (propagation, satiation and resistance), but also with some limitations during the full lifetime period, i.e. not separated but interconnected phases due to introduction the elitism concept between phases. Although some limitations still exist (number of possible target search methods/exploits, assumptions that bot may use both TCP and UDP for transportation) other limitations, such as for probability rate limitations were removed.

4.6. Concluding Remarks of Chapter 4

In this Chapter the malware evolution forecasting model is described. It consists of malware strategy representation structure, genetic algorithm acting under specified conditions and a fitness function for propagation strategy evaluation.

The model can be used for different malware parameters evolution modelling. In case of the same malware type in order to use the model for different characteristic or characteristic complexes evolution modelling only the fitness functions should be changed, leaving the chromosome structure untouched. In case of different malware types chromosome structure should be updated to represent specific malware features.

The correctness of fitness functions used for Internet worm and mobile malware evolution forecasting was tested on historical data, by applying for fitness evaluation some malware samples with known fitness, observed experimentally. Although some minor differences between obtained and observed values were found, they were classified as not having significant influence on the final result.

The modelling of perspective propagation strategies of Internet worms and mobile malware, aiming to infect the highest number of computers in a limited period of time has shown the following tendencies:

1. Internet worms will tend to evolve to rather simple solutions, making use of a popular OS platform, quick connectionless UDP based transfer mechanisms and a combination of several (4–5) exploits with high infection probability. Fitness of the best strategy obtained dur-

ing the experiment was equal to 1.18, compared to 0.015 of the most famous Code Red v.2 worm.

2. Mobile malware evolution will develop in the direction of inclusion of additional OS platforms and propagation by WI-FI networks. The forecasted propagation strategy tends not to be function overloaded due to time consumption increase. The fitness increase was not as impressive as in Internet worms' case (only 1.7 times compared to a sample modern sample). This can be explained by technological differences, limited number of exploits available, low infection probabilities, compared to computer networks, and rather limited value of additional propagation methods.

Due to lack of the statistical data the model for botnet evolution forecasting (propagation and survivability techniques) and Internet worm propagation under pressure of countermeasures was described as a "proof-of-concept" with no real data tests.

General Conclusions

Conclusions

1. According to the test results the principal correctness of the proposed genetic algorithm based malware risk evaluation model, using decision trees generated by the means of genetic algorithms for newly appearing malware risk evaluation, was approved. The proposed model can be recommended for automating the malware threat warning systems.
2. The modelling of perspective propagation strategies of Internet worms aiming to infect the highest number of computers in a limited period of time has shown the following evolution tendencies: Internet worms will tend to evolve to rather simple solutions, making use of a popular OS platform, quick connectionless UDP based transfer mechanisms (“fire-and-forget” principal) and a combination of 4–5 exploits with high infection probability. Calculated fitness of the best strategy obtained during the experiment was equal to 1.18, which is 79 times higher than 0.015 of the most famous Code Red v.2 worm.
3. The modelling of perspective mobile malware propagation strategies, aiming to infect the highest number of computers in a limited period of time has shown that evolving mobile malware will tend to inclusion of

several operating system platforms and propagation by WI-FI networks, compared to dominating nowadays MMS and Bluetooth. The forecasted propagation strategy tends not to be function overloaded. The fitness increased 1.7 times compared to a sample mobile virus.

4. Estimated evolution tendencies allow saying that consequences of future malware epidemic outbreaks can be more devastating than caused by modern malware, due to that novel methods, such as malware detection and blocking in network traffic, anomaly based detection mechanisms, network infrastructure load balancing, “bottleneck” prevention and others, should be developed and implemented to minimize the epidemic outbreak effect.

Suggestions for Further Research

The model proposed is highly dependant on reliable statistical data on malware descriptive characteristic, such as population sizes and change dynamics, propagation speeds, etc. That is why such data collection and systematization via honeynet system can be considered as the main task in the nearest future. Systemized information could be used for model tuning and selecting the most appropriate genetic algorithm operating conditions, since current model tests were performed with rather limited test data sets or provided only as a „proof-of-concept“.

Minimization of model limitations can be named as the second area of model improvement, although many of them were removed while modelling mobile malware and describing the botnet evolution „proof-of-concept“ model, compared to Internet worms’ evolution model.

Thirdly, the model should also be expanded to represent the possibility of modelling the malware population, where one part of individuals uses one strategy, and other or others – a bit or absolutely different, but fitness is evaluated against the whole population.

Finally, although possibility to analyse malware evolution under pressure of countermeasures was proposed on the Internet worm sample, it would be interesting and important both from practical and scientific point of view to model the “arms race” between malware and countermeasures, by adapting model for two co-evolving populations: malware and countermeasures strategies. In such a model efficiency of each strategy will be evaluated against the array of opposing strategies (i.e. countermeasure application strategy’s efficiency would be evaluated in measure of malware propagation strategy decrease rate and vice versa).

References

- Abbazio, J.; Perez, S.; Silva, D.; Tesoriero, R.; Penna, F.; Zack, R. 2009. Face Biometric Systems, in *Proc. of Student-Faculty Research Day, CSIS, Pace University, 2009*. New York, C1.1–C1.8
- Ahmad, R.; Samy, G. N.; Ibrahim, N. K.; Bath, P. A.; Ismail, Z. 2010. Threats Identification in Healthcare Information Systems Using Genetic Algorithm and Cox Regression, *Journal of Information Assurance and Security* 5(1): 154–161.
- Akashi, T.; Wakasa, Y.; Tanaka, K. 2007. Using Genetic Algorithm for Eye Detection and Tracking in Video Sequence, *Systematics, Cybernetics and Informatics* 5(2): 72–78.
- Alanni, M. K.; Sundarajan, V. 2009. Detecting a Denial of Service Using Artificial Intelligent Tools, Genetic Algorithm, *Indian Journal of Science and Technology* 2(2): 16–21.
- Alisauskaitė, V.; Rimkus, D. 2007. Computer Network Security Systems Problematics Analysis, in *Proc. of Information Technologies 2007*. Kaunas, 239–241.
- Almarimi, A.; Kumar, A.; Almerhag, I. 2008. A New Approach for Data Encryption Using Genetic Algorithms, in *Proc. of the International Arab Conference on Information Technology 2008*. Zarqa, 5 pp.
- Amoroso, E. G. 1999. *Intrusion Detection – An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response*. 1st ed. Scranton: Intrusion.net Books. 224 p.

- Bäck, T. 1996. *Evolutionary Algorithms in Theory and Practice*. Oxford: Oxford University Press. 120 p.
- Baker, J. E. 1987. Reducing Bias and Inefficiency in the Selection Algorithm, in *Proc. of the Second International Conference on Genetic Algorithms and their Application*. Hillsdale, New Jersey: L. Erlbaum Associates, 14–21.
- Banks, B. S.; Stytz, R. M. 2008. Challenges Of Modeling BotNets For Military And Security, in *Proc. of SimTecT 2008*. Melbourne, 6 pp.
- Barford, P.; Yegneswaran V. 2007. An Inside Look at Botnets, *Advances in Information Security* 27: 171–191.
- Barricelli, N. A. 1963. Numerical Testing of Evolution Theories. Part II. Preliminary Tests of Performance, Symbiogenesis and Terrestrial Life, *Acta Biotheoretica* 16(3-4): 99–126.
- Barricelli, N. A. 1957. Symbiogenetic Evolution Processes Realized by Artificial Methods, *Methodos* 9(35-36):143–182.
- Barroso, D. 2007. *Botnets – The Silent Threat*: ENISA Position Paper. No. 3. Heraklion: ENISA. 12 p.
- Biel, L.; Pettersson, O.; Philipson, L.; Wide, P. 2001. ECG Analysis: a New Approach in Human Identification, *IEEE Transactions on Instrument and Measurement* 50(3): 808–812.
- Bobor, V. 2006. *Efficient Intrusion Detection System Architecture Based on Neural Networks and Genetic Algorithms*. Stockholm University / Royal Institute of Technology. Stockholm. 72 p.
- Brand, M.; Champion, A.; Chan, D. 2009. Combating the Botnet Scourge. [online]. [cited 26 November 2009]. Available from Internet: <http://www.cse.ohio-state.edu/~champion/research/Combating_the_Botnet_Scourge.pdf>.
- Bridges, S. M.; Vaughn, R. B. 2000. Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection, in *Proc. of 12th Annual Canadian Information Technology Security Symposium*. Ottawa, 109–122.
- Brumley, D.; Liu, L.; Poosankam, P.; Song, D. 2005 *Taxonomy and Effectiveness of Worm Defense Strategies*: Tech. Rep. CMU-CS-05-156. Carnegie Mellon University, School of Computer Science. Pittsburgh.
- Brunelli, R.; Poggio, T. 1993. Face Recognition: Features Versus Templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(10): 1042–1052.
- Canavan, J. 2005. The Evolution of Malicious IRC Bots, in *Proc. of Virus Bulletin Conference 2005*. Dublin, 104–114.
- Chen, Z.; Gao, L.; Kwiat, K. 2003. Modeling the Spread of Active Worms, in *Proc. of IEEE INFOCOM 2003*. San Francisco, 1890–1900.
- Cho, U. K.; Hong, J. H.; Cho, S. B. 2007. Automatic Fingerprints Image Generation Using Evolutionary Algorithm, *Lecture Notes in Artificial Intelligence* 4570: 444–453.

- Chuang, M.; Chang, R.; Huang, L. 2000. Automatic Facial Feature Extraction Model Based Coding, *Journal of Information Science and Engineering* 16: 447–458.
- Clark, A. 1994. Modern Optimisation Algorithms for Cryptanalysis, in *Proc. of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems*. Brisbane, 258–262.
- Clark, A.; Dawson, E. 1997. A Parallel Genetic Algorithm for Cryptanalysis of the Polyalphabetic Substitution Cipher, *Cryptologia* 21(2): 129–138.
- Crosbie, M.; Spafford, E. 1995. Applying Genetic Programming to Intrusion Detection, in *Proc. of the AAAI Fall Symposium*. Massachusetts, 1–8.
- Crosby, J. L. 1973. *Computer Simulation in Genetics*. London: John Wiley & Sons.
- Dagon, D.; Gu, G.; Zou, C.; Grizzard, J.; Dwivedi, S.; Lee, W.; Lipton, R. 2005. A Taxonomy of Botnets, in *Proc. of CAIDA DNS-OARC Workshop*. San Jose, 1–16.
- Dagon, D.; Zou, C.; Lee, W. 2006. Modeling Botnet Propagation Using Time Zones, in *Proc. of the 13 th Network and Distributed System Security Symposium*. San Diego, 15 pp.
- David, J. M. 1995. *Neural Network Weight Selection Using Genetic Algorithms* [online]. Cambridge [cited 11 March 2009]. Available from Internet: <<http://vishnu.bbn.com/papers/hybrid.pdf>>.
- Defense Dept., Defense Acquisition University. 2001. *Systems Engineering Fundamentals*: January 2001. Defense Acquisition University. Fort Belvoir. 222 p.
- Delman, B. 2004. *Genetic Algorithms in Cryptography*. Rochester Institute of Technology. Rochester. 97 p.
- Diaz-Gomez, P. A.; Hougen, D. F. 2005a. Analysis of an Off-Line Intrusion Detection System: A Case Study in Multi-Objective Genetic Algorithms, in *Proc. FLAIRS 2005 conference*. Massachusetts, 822–823.
- Diaz-Gomez, P. A.; Hougen, D. F. 2005b. *Improved off-line Intrusion Detection Using a Genetic Algorithm* [online]. [cited 8 March 2009]. Available from Internet: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.1321&rep=rep1&type=pdf>>
- Dimovski, A.; Gligoroski, D. 2003. Attacks On the Transposition Ciphers using Optimization Heuristics, in *Proc. of ICEST 2003*. Sofia, Bulgaria, 1–4.
- Dittrich, D.; Dittrich, S. 2008. P2P as Botnet Command and Control: A Deeper Insight, in *Proc. of the 2008 3rd International Conference on Malicious and Unwanted Software*. Fairfax, 41–48.
- El-Emary, I. M. M.; Abd El-Kareem, M. M. 2008. On the Application of Genetic Algorithms in Finger Prints Registration, *World Applied Sciences Journal* 5(3): 276–281
- Endorf, C.; Schultz, E.; Mellander, J. 2003. *Intrusion Detection & Prevention*. 1st ed. UK: McGraw-Hill Osborne Media. 500 p.

- ESET, LLC. 2009. *ESET NOD32 Antivirus 4 Business Edition, Key Features* [online] 2009. [cited 11 March 2009]. Available from Internet: <<http://www.eset.com/download/datasheets/Datasheet-ENA-BE-4.pdf>>.
- Everit, R. A. J.; McOwan, P. W. 2003. Java-Based Internet Biometric Authentication System, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25: 1166–1172.
- Fingerprint minutiae types [online] 2005. [cited 12 February 2010]. Available from Internet: <<http://www.emeraldinsight.com/fig/0870210403001.png>>.
- Fingerprint with minutiae marked [online] 2004. [cited 12 February 2010]. Available from Internet: <http://www.uh.edu/engines/fingerprint_minutiae.jpg>.
- Fraser, A. 1957. Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction, *Australian Journal of Biological Sciences* 10: 484–491.
- Fraser, A.; Donald, B. 1970. *Computer Models in Genetics*. New York: McGraw-Hill.
- F-Secure. 2006. Worm:SymbOS/Commwarrior. *F-Secure Corporation*. [online]. [cited 12 November 2009]. Available from Internet: <<http://www.f-secure.com/v-descs/commwarrior.shtml>>.
- Fultz, N. 2008. *Distributed Attacks as Security Games*. US Berkley School of Information. Berkley.
- Garetto, M.; Gong, W.; Towsley, D. 2003. Modeling Malware Spreading Dynamics, in *Proc. INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, vol. 3. San Francisco, 1869–1879.
- Garg, P.; Shastri, A. 2007. An Improved Cryptanalytic Attack on Knapsack Cipher using Genetic Algorithm, *International Journal of Information Technology* 3(3): 145–152.
- Garsva, E. 2006a. Computer System Survivability Modelling by Using Stochastic Activity Network, *Lecture Notes in Computer Science* 4166(2006): 71–84.
- Garšva, E. 2006b *Modelling of Computer System Security*: Summary of Doctoral Dissertation. Vilnius Gediminas Technical University. Vilnius: Technica. 24 p.
- Girgisa, M. R.; Sewisyb, A. A.; Mansour, R. F. 2007. Employing Generic Algorithms for Precise Fingerprint Matching Based on Line Extraction, *GVIP Journal* 7(1): 51–59.
- Gordon, J. 2004. Agobot and the the „Kit“ chen Sink [online]. [cited 6 April 2009]. Available from Internet: <http://www.infectionvectors.com/vectors/Agobot_&_the_Kitchen_Sink.pdf>.
- Grizzard, B. J.; Sharma, V.; Nunnery, C.; Kang, B. B.; Dagon, D. 2007. Peer-to-Peer Botnets: Overview and Case Study, in *Proc. of the first conference on First Workshop on Hot Topics in Understanding Botnets, HotBots'07*. Cambridge, 1–1.
- Holland, J. 1975. *Adoption in Natural and Artificial Systems*. Cambridge: The MIT press. 211 p.

- Huang, J.; Wechsler, H. 1999. Eye Location Using Genetic Algorithm, in *Proc. of 2nd International Conference on Audio and Video-Based Biometric Person Authentication (AVBPA)*. Washington, DC, 8 pp.
- Hurley, D.; Nixon, M.; Carter, J. 2005. Force Field Feature Extraction for Ear Biometrics, *Computer Vision and Image Understanding* 98(3): 491–512.
- Ibrahim, K.; Chris, T. 1994. *Design of Artificial Neural Networks Using Genetic Algorithms: Review and Prospect* [online]. School of Cognitive and Computing Sciences, University of Sussex [cited 8 March 2009]. Available from Internet: <<http://citeseer.ist.psu.edu/article/kuscu94design.html>>.
- Islam, R.; Rahman, F. 2009. Improvement of Text Dependent Speaker Identification System Using Neuro-Genetic Hybrid Algorithm in Office Environmental Conditions, *International Journal of Computer Science* 1: 42–47.
- ISO/IEC 27005:2008 *Information technology - Security techniques - Information security risk management*. International Organization for Standardization, 2008.
- Jain, A.; Bole, R.; Pankanti, S. 1999. *BIOMETRICS Personal Identification in Networked Society*. Boston: Kluwer Academic Press.
- Jain, A.; Hong, L.; Pankanti, S.; Bolle, R. 1997. An Identity Authentication System Using Fingerprints, *IEEE Transaction* 85(9): 1365–1388.
- Jarno, U. 2005. Disinfection tool for SymbOS/Locknut.A (Gavno.A and Gavno.B). *F-Secure Corporation*. [online].[cited 12 November 2009]. Available from Internet: <<http://www.f-secure.com/>>.
- Juknius, J.; Čenys, A. 2009. Intelligent Botnet Attacks in Modern Information Warfare, in *Proc. of 15th International Conference on Information and Software Technologies*. Kaunas, 39–42.
- Karasaridis, A.; Rexroad, B.; Hoefling, D. 2007. Wide-Scale Botnet Detection and Characterization, in *Proc. USENIX Workshop on Hot Topics in Understanding Botnets*. Cambridge, 7–7.
- Kaspersky Lab. 2008. The Botnet Business [online]. [cited 12 March 2009]. Available from Internet: <<http://www.viruslist.com/en/analysis?pubid=204792003>>.
- Kaspersky Lab. 2009. Kaspersky Lab Reports [online]. [cited 26 November 2009]. Available from Internet: <<http://www.kaspersky.com/>>.
- Katvickis, A.; Raulynaitis, A.; Luksys, K.; Burba, T.; Dosinas, G. S.; Sakalauskas, E. 2005. Kriptografinių vienkrypčių funkcijų konstravimas remiantis sistemų identifikacijos teorija [Constructing the Cryptographic One-Way Hash Functions Based on System Identification Theory], *Matematika ir matematinis modeliavimas* [Mathematics and Mathematical Modelling] 1: 160–165.
- Kazanavicius, E.; Pakalniskis, K. 2001. Design and Implementation of a DES algorithm, in *Proc. of Information technologies 2001*. Kaunas, 391–394.

- Kazanavicius, E.; Venckauskas, A.; Liutkevicius, A.; Vrubliauskas, A. 2008. *Informacijos saugos vadyba: mokomoji knyga* [Information Security Management: Textbook]. Kaunas. 169 p.
- Kephart, O. J., White, R. S. 1991. Directed-Graph Epidemiological Models of Computer Viruses, in *Proc. of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*. Oakland, 342–359.
- Kirchberg, K. J.; Jesorsky, O.; Frischholz, R. W. 2002. Genetic Model Optimization for Hausdorff Distance-Based Face Localization, *Lecture Notes in Computer Science: Biometric Authentication* 2359/2002: 103–111.
- Kohavi, R.; John, G. 1995. Automatic Parameter Selection by Minimizing Estimated Error, in *Proc. of the Twelfth International Conference on Machine Learning*. Tahoe, 304–312.
- Kosorukoff, A. 2001. Human-based Genetic Algorithm, in *Proc. of IEEE International Conference on Systems, Man, and Cybernetics 2001*. Tucson, 3464–3469.
- Kumar, A.; Ghose, M. K. 2009. Overview of Information Security Using Genetic Algorithm and Chaos, *Information Security Journal: A Global Perspective* 18(6): 306–315.
- Laurutis, R. 2003. Application of Neural Networks for Data Protection Research, *Electronics and Electrical Engineering* 4(46): 61–64.
- Lee, W.; Wang, C.; Dagon, D. 2007. *Botnet Detection : Countering the Largest Security Threat*. Heidelberg: Springer. 168 p.
- Lelarge, M. 2009. The Economics of Malware: Epidemic Risks Model, Network Externalities and Incentives, in *Proc. of Fifth bi-annual Conference on The Economics of the Software and Internet Industries*. Toulouse, 19 pp.
- Li, W. 2004. Using Genetic Algorithm for Network Intrusion Detection, in *Proc. of the United States Department of Energy Cyber Security Group 2004 Training Conference*. Kansas City, 24–27.
- Li, Z.; Goyal, A.; Chen, Y. 2007. HoneyNet-based Botnet Scan Traffic Analysis, *Advances in Information Security: Botnet Detection* 36: 25–44.
- Li, Z.; Goyal, A.; Chen, Y.; Paxson, V. 2009a. Automating Analysis of Large-Scale Botnet Probing Events, in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS '09*. Sydney, 11–12.
- Li, Z.; Liao, Q. C.; Striegel, A. 2009b. Botnet Economics: Uncertainty Matters, In: *Managing Information Risk and the Economics of Security*. US: Springer, 245–267.
- Lin, C.; Ling, W. 1999. Automatic Facial Feature Extraction by Genetic Algorithms, *IEEE Transactions on Image Processing* 8(6): 834–845.
- Maio, D.; Maltoni, D. 1997. Direct Gray-Scale Minutiae Detection in Fingerprints, *IEEE Transactions: Pattern Analysis Machine Intelligence* 19(1): 27–40.
- Maltoni, D. 2004. *Generation of Synthetic Fingerprint Image Databases: Automatic Fingerprint Recognition Systems*. Berlin: Springer Heidelberg.

- Matthews, R. A. J. 1993. The Use of Genetic Algorithms in Cryptanalysis, *Cryptologia* 17(4): 187–201.
- Me, L. 1998. GASSATA, a genetic algorithm as an alternative tool for security audit trail analysis [online], in *Proc. of the First International Workshop on the Recent Advances in Intrusion Detection. Belgium* [cited 8 March 2009]. Available from Internet: <http://windowsecurity.com/whitepapers/GASSATA_A_Genetic_Algorithm_as_an_Alternative_Tool_for_Security_Audit_Trails_Analysis.html>.
- Miller, B. L.; Goldberg, D. E. 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise, *Complex Systems* 9:193–212.
- Mohamad, D. 2009. Multi Local Feature Selection Using Genetic Algorithm for Face Identification, *International Journal of Image Processing* 1(2): 1–10.
- Monga, R. 2009. MASFMMs: Multi Agent Systems Framework for Malware Modeling and Simulation, *Lecture Notes in Computer Science* 5269/2009: 97–109.
- Mühlenbein, H.; Schlierkamp-Voosen, D. 1993. Predictive Models for the Breeder Genetic Algorithm. Pt.I: Continuous Parameter Optimization, *Evolutionary computation* 1: 25–49.
- Mukamurenzi, N. M. 2008. *Storm Worm: A P2P Botnet*. Norwegian University of Science and Technology. Trondheim.
- Naraine, R. 2004. Cell Phone Security: New Skulls Mutant Comes with Virus Extras [online]. [cited 26 November 2009]. Available from Internet: <<http://www.eweek.com/c/a/Mobile-and-Wireless/Cell-Phone-Security-New-Skulls-Mutant-Comes-with-Virus-Extras/>>.
- Nazario, J. 2004. *Defense and Detection Strategies against Internet Worms*. Norwood: Artech House, Inc. 319 p.
- Niemela, J. 2005. F-Secure Virus Descriptions : Skulls.D. *F-Secure Corporation*. [online]. [cited 12 November 2009]. Available from Internet: < http://www.f-secure.com/v-descs/skulls_d.shtml#summary>.
- Noreen, S.; Murtaza, S.; Shafio, M. Z.; Faroo, M. 2009. Evolvable Malware, in *Proc. of the 11th Annual conference on Genetic and evolutionary computation*. Montreal: ACM. 1569–1576.
- Pankanti, S.; Bolle, R. M.; Jain, A. 2000. Biometrics, the Future of Identification, *IEEE Computer: Special Issue on Biometrics* 33(2):46–49.
- Pankanti, S.; Prabhakar, S.; Jain, A. 2002. On the individuality of fingerprints, *IEEE Transaction on Pattern Analysis and Machine Intelligence* 24(8): 1010–1025.
- Papagelis, A.; Kalles, D. 2001. Breeding Decision Trees Using Evolutionary Techniques, in *Proc. of the Eighteenth International Conference on Machine Learning*. San Francisco, 393–400.

- Paulauskas, N. 2009 *Analysis of Computer System Incidents and Security Level Evaluation*: Doctoral Dissertaion. Vilnius Gediminas Technical University. Vilnius: Technica. 24 p. 135 p.
- Paulauskas, N.; Garšva, E. 2006. Computer System Security Incident Analysis, in *Proc. of Seventh International Baltic Conference on Databases and Information Systems: Communications, Materials of Doctoral Consortium*. Vilnius, 336–339.
- Paulauskas, N.; Garsva, E. 2008. Attacker Skill Level Distribution Estimation in the System Mean Time-to-Compromise, in *Proc. of the 1st International conference on information technology*. Gdansk, 463–466.
- Paulauskas, N.; Garsva, E.; Skudutis, J. 2009. Network Scan Detection Simulation, *Electronics and Electrical Engineering* 2(90): 43–46.
- Paulauskas, N.; Skudutis, J. 2008. Investigation of the Intrusion Detection System "Snort" Performance, *Electronics and Electrical Engineering* 7(87): 15–18.
- Pires, M. G.; Duarte, F. V.; Gonzaga, A. 2005. *Genetic Optimization for Fingerprint Verification* [online]. [cited 12 February 2010]. Available from Internet <http://iris.sel.eesc.usp.br/lavi/pdf/D__poster_22526.pdf>.
- Porras, P.; Saidi, H.; Yegneswaran, V. 2007. *A Multi-perspective Analysis of the Storm (Peacomm) Worm*: Technical report. Computer Science Laboratory, SRI International.
- Prabhakar, S.; Jain, A.; Pankanti, S. 2003. Learning Fingerprint Minutiae Location and Type, *Pattern Recognition* 36(8): 1847–1857.
- Proctor, P.E. 2000. *The Practical Intrusion Detection Handbook*. New Jersey: Prentice Hall. 384 p.
- Provos, N.; Holz, T. 2007. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Boston: Addison-Wesley Professional. 440 p.
- Puniskis, D.; Laurutis, R. 2005. The Use of Neuron Networks for the Performance of Epidemics caused by computer viruses, *Electronics and Electrical Engineering* 4(60): 28–32.
- Puniskis, D.; Laurutis, R. 2007. Behavior Statistic based Neural Net Anti-spam Filters, *Electronics and Electrical Engineering* 6(78): 35–38.
- Puniskis, D.; Laurutis, R.; Dirmeikis, R. 2006. An Artificial Neural Nets for Spam e-mail Recognition, *Electronics and Electrical Engineering* 5(69): 73–76.
- Qi-Rong, Q.; Yong-Fang, Z.; Lu, H. 2008. An Optimization Model of Product Selection in Information Security Technology System, in *Proc. of 2008 International Conference on Machine Learning and Cybernetics*. Kunming, 1141–1146.
- Rajab, M. A.; Zarfoss, J.; Monrose, F.; Terzis, A. 2006. A Multifaceted Approach to Understanding the Botnet Phenomenon, in *Proc. of 6th ACM SIGCOMM conference on Internet Measurment, 2006*. Rio de Janeiro, 41–52.
- Rajab, M. A.; Zarfoss, J.; Monrose, F.; Terzis, A. 2007. My Botnet is Bigger Than Yours (Maybe, Better Than Yours): Why Size Estimates Remain Challenging, in *Proc.*

of the first conference on First Workshop on Hot Topics in Understanding Botnets, Hot-Bots'07. Cambridge, 5–5.

Ramachandran, K.; Sikdar, B. 2006. Modeling Malware Propagation in Gnutella Type Peer-to-Peer Networks, in *Proc. of Parallel and Distributed Processing Symposium*, vol. 20. Rhodes Island, 8 pp.

Ravi, K. V. R.; Palaniappan, R. 2006. Neural Network Classification of Late Gamma Band Electroencephalogram Features, *Soft Computing* 10(2): 163–169.

Ravi, K. V. R.; Palaniappan, R.; Eswaran, C.; Phon-Amnuaisuk, S. 2009. Individual Identification and Biometric Cryptosystem Using Visual Evoked Potential Signals [online]. [cited 3 February 2010]. Available from Internet <http://fit.mmu.edu.my/news/newsattach/EXTENDED%20ABSTRACT_1.1_Ravi.pdf>.

Reeves, R. C.; Rowe, E. J. 2003. *Genetic Algorithms: Principles and Perspectives: A Guide to GA Theory*. Massachusetts: Kluwer Academic Publishers. 327 p.

Ruitenbeek, V. E.; Courtney, T.; Sanders, H. W.; Stevens, F. 2007. Quantifying the Effectiveness of Mobile Phone Virus Response Mechanisms, in *Proc. of Dependable Systems and Networks, 2007 // 37th Annual IEEE/IFIP International Conference*. Edinburgh, 790–800.

Ruitenbeek, V. E.; Sanders, H. W. 2008. Modeling Peer-to-Peer Botnets, in *Proc. of Quantitative Evaluation of Systems, 2008. // Fifth International Conference*. St Malo, 307–316.

Sakalauskas, E.; Burba, T. 2004. Digital Signature Scheme Based on Action of Infinite Ring, *Information Technology and Control* 2(31): 60–64.

Sakalauskas, E.; Dumbliauskas, T.; Luksys, K. 2005. Kriptografinės sistemos (FS) struktūros sukūrimas [Development of the Cryptographic (FS) System], in *Proc. Transport means-2005: 9th International Conference*. Kaunas, 19–24.

Sans Institute. 2008. The Sans Institutes Top Ten Cyber Security Menaces for 2008 [online]. [cited 20 May 2009]. Available from Internet: <<http://www.itu.int/osg/csd/newslog/The+SANS+Institutes+Top+Ten+Cyber+Security+Menaces+For+2008.aspx>>.

Scheidat, T.; Engel, A.; Vielhauer, C. 2006. Parameter Optimization for Biometric Fingerprint Recognition Using Genetic Algorithms, in *Proc. of the 8th Workshop on Multimedia and Security*. Geneva, 130–134.

Selvakani, S.; Rajesh, R. S. 2007. Genetic Algorithm for Framing Rules for Intrusion Detection, *International Journal of Computer Science and Network Security* 7(11): 285–290.

Serazzi, G.; Zanero, S. 2004. Computer Virus Propagation Models, *Lecture Notes in Computer Science* 2965/2004: 26–50.

Shah, A. 2009. IDC: 1 Billion Mobile Devices Will Go Online by 2013, in *IDG News Service*, [online]. [cited 11 December 2009]. Available from Internet: <

http://www.pcworld.com/article/184127/idc_1_billion_mobile_devices_will_go_online_by_2013.html>.

Sinclair, C.; Pierce, L.; Matzner, S. 1999. An Application of Machine Learning to Network Intrusion Detection, in *Proc. of 1999 Annual Computer Security Applications Conference*. Arizona: Phoenix, 371–377.

Sindhu, S. S. S.; Geetha, S.; Marikannan, M.; Kannan, A. 2009. A Neuro-Genetic Based Short-Term Forecasting Framework for Network Intrusion Prediction System, *International Journal of Automation and Computing* 6(4): 406–414.

Song, D.; Heywood, M. I.; Zincir-Heywood, N. A. 2003. A Linear Genetic Programming Approach to Intrusion

Spillman, R. 1993. Cryptanalysis of Knapsack Ciphers Using Genetic Algorithms, *Cryptologia* 17(4): 367–377.

Spillman, R.; Janssen, M.; Nelson, B.; Kepner, M. 1993. Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers, *Cryptologia* 17(1): 31–44.

Stakenas, V. 2006a. Analytic and Probabilistic Methods in Number Theory, in *Proc. of the fourth international conference in honour of J. Kubilius*. Palanga, 213–224.

Stakenas, V. 2006b. On Some Inequalities of Probabilistic Number Theory, *Lietuvos matematikos rinkiny* 46(2): 256–266.

Stakenas, V. 2007. *Kodai ir šifrai : informacijos kodavimo ir kriptografijos pagrindai* [Codes and Ciphers: Information Encryption and Cryptography Basics]. Vilnius. 352 p.

Staniford, S.; Paxson, V.; Weaver, N. 2002. How to Own the Internet in Your Spare Time, in *Proc. of the 11th USENIX Security Symposium*. , San Francisco: 149–167.

Stein, G.; Chen, B.; Wu, A. S.; Hua, K. A. 2005. Decision Tree Classifier for Network Intrusion Detection with GA-based Feature Selection, in *Proc. of the 43rd annual South-east Regional Conference*. New York, 136–141.

Stender, J.; Hillebrand, E.; Kingdon, J. 1994. *Genetic Algorithms in Optimization, Simulation and Modeling*. 1st ed. Amsterdam: IOS Press. 272 p.

Stewart, J. 2004. Phatbot Trojan Analysis [online]. [cited 12 March 2009]. Available from Internet: <<http://www.secureworks.com/research/threats/phatbot>>.

Sudarshana Reddy, H. R.; Subba Reddy, N. V. 2004. Development of Genetic Algorithm Embedded KNN for Fingerprint Recognition, *Lecture Notes in Computer Science: Applied Computing* 3285/2004: 9–16

Sun, Y.; Yin, L. 2007. A Genetic Algorithm Based Approach for 3D Face Recognition, *Computational Imaging and Vision: 3D Imaging for Safety and Security* 35: 95–118.

Sundgot, J. 2005. First Symbian OS virus to replicate over MMS appears. [online]. [cited 1 December 2009]. Available from Internet: <<http://www.infosyncworld.com/news/n/5835.html>>.

- Symantec Corp. 2004. W32.Gaobot.DX | Symantec [online]. [cited 20 May 2009]. Available from Internet: <http://www.symantec.com/security_response/writeup.jsp?docid=2004-051816-5418-99>.
- Szor, P. 2005. *The Art of Computer Virus Research and Defense*. Boston: Addison Wesley Professional. 744 p.
- Tan, X.; Bhanu, B. 2002. Fingerprint Verification Using Genetic Algorithms, in *Proc. of Sixth IEEE Workshop on Applications of Computer Vision*. Orlando, 79–85.
- Tang, K. S.; Man, K. F.; Kwong, S.; He, Q. 1996. Genetic Algorithms and Their Applications, *IEEE Signal Processing Magazine* 13(6): 22–37.
- Tragha, A.; Omary, F.; Mouloudi, A. 2006. ICIGA: Improved Cryptography Inspired by Genetic Algorithms, in *Proc. International Conference on Hybrid Information Technology*. Cheju Island, 335–341.
- Turner, D. 2008. *Symantec Global Internet Security Threat Report*. Symantec Corporation.
- Uludag, U.; Pankanti, S.; Prabhakar, S.; Jain A. K. 2004. Biometric Cryptosystems: Issues and challenges, *Proceedings of the IEEE* 92(6): 948–960.
- Vageris, R. 2005. *Rizikos analizės vadovas*. Vilnius: VAGA. 160 p.
- Vasilecas, O.; Cenys, A.; Sosunovas, S.; Goranin, N. 2008. *Informacinių sistemų sauga* [Information System Security]. Vilnius. 274 p.
- Venckauskas, A.; Mikuckienė, I.; Mikuckas, A. 2003. Įmonės informacinės saugos efektyvumo vertinimas [Evaluating the Effectiveness of Company's Information Security], *Informacijos mokslai* [Information Sciences] 26: 90–93.
- Vogt, R.; Aycock, J.; Jacobson, M. J. Jr. 2007. Army of Botnets, in *Proc. of Network and Distributed System Security Symposium*. San Diego, 111–123.
- Wang, L.; Wang, Y.; Yao, R.; Zhang Z. 2006. Hardware Implementation of AES Based on Genetic Algorithm, *Lecture Notes in Computer Science: Advances in Natural Computation* 4222/2006: 904–907.
- Wang, P.; Sparks, S.; Zou, C. C. 2007. An Advanced Hybrid Peer-to-Peer Botnet, in *Proc. of USENIX Workshop on Hot Topics in Understanding Botnets*. Cambridge, 2–2.
- Wash, R. 2008. Mental Models of Home Computer Security [online]. [cited 12 March 2009]. Available from Internet: <<http://cups.cs.cmu.edu/soups/2008/posters/wash.pdf>>.
- Whitley, D. 1993. *A Genetic Algorithm Tutorial*. Colorado State University. Colorado. 37 p.
- Woodward, J. D. 2003. *Biometrics: A Look at Facial Recognition*. US: RAND Corporation. 30 p.
- Yaseen, I. F. T.; Sahasrabudhe, H. V. 1999. A Genetic Algorithm for the Cryptanalysis of Chor-Rivest Knapsack Public Key Cryptosystem (PKC), in *Proc. of Third Interna-*

tional Conference on Computational Intelligence and Multimedia Applications. New Delhi, 81–85.

Yen, G. G.; Nithianandan, N. 2002. Facial Feature Extraction Using Genetic Algorithm, in *Proc. of the 2002 Congress on Evolutionary Computation CEC'02*. Washington, 1895–1900.

Zhang, J. 2008. Storm Worm & Botnet Analysis [online]. [cited 11 April 2009]. Available from Internet: <http://securitylabs.websense.com/content/Assets/Storm_Worm_Botnet_Analysis-June_2008.pdf>.

Zhaosheng, Z.; Guohan, L.; Yan, F. C.; Zhi, J. R.; Han, P. 2002. Botnet Research Survey, in *Proc. of the 2008 32nd Annual IEEE International Computer Software and Applications Conference, COMPSAC '08*. Turku, 967–972.

Zhuge, J.; Holz, T.; Han, X.; Guo, J.; Zou, W. 2007. *Characterizing the IRC-based Botnet Phenomenon*: Technical Report. Peking University & University of Mannheim. Beijing.

Zou, C. C.; Dagon, D.; Lee, W. 2008. Modeling and Measuring Botnets [online]. [cited 11 November 2009]. Available from Internet: <http://www.gtisc.gatech.edu/aroworkshop/ppt/BotnetModel_Zou.ppt>.

Zou, C. C.; Gong, W.; Towsley, D. 2002. Code Red Worm Propagation Modeling and Analysis, in *Proc. of the 9th ACM conference on Computer and Communications Security*. Washington, 138–147.

Zou, C. C.; Gong, W.; Towsley, D. 2003. Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense, in *Proc. of WORM'03*. Washington, 51–60.

ZOU, C. C.; Gong, W.; Towsley, D. 2005. On the Performance of Internet Worm Scanning Strategies, *Performance Evaluation* 63(7):700–723.

Zou, C. C.; Towsley, D.; Gong, W. 2004. *Email Virus Propagation Modeling and Analysis*: Technical report TRCSE- 03-04. University of Massachussets. Massachussets.

List of Publications by the Author on the Topic of the Dissertation

Papers in the Reviewed Scientific Journals

Goranin, N.; Cenys, A. 2008a. Genetic Algorithm Based Internet Worm Propagation Strategy Modeling, *Information Technology and Control* 37(2): 133–140. ISSN 1392-124X. (ISI Web of Science)

Goranin, N.; Cenys, A. 2008b. Malware Propagation Modeling by the Means of Genetic Algorithms, *Elektronika ir elektrotechnika* 6(86): 23–26. ISSN 1392-1215. (ISI Web of Science)

Goranin, N.; Cenys, A. 2009. Genetic Algorithm Based Internet Worm Propagation Strategy Modeling Under Pressure of Countermeasures, *Journal of Engineering Science and Technology Review* 2(1): 43–47. ISSN 1791-2377.

Other Papers

Goranin, N.; Cenys, A.; Juknius, J. 2010. Extension of the Genetic Algorithm Based Malware Strategy Evolution Forecasting Model for Botnet Strategy Evolution Modeling, in *Proc. of NATO RTO Information Systems Technology Panel Symposium, Information Assurance and Cyber Defense (IST-091 / RSY-021)*, Antalya. Turkey. RTA-NATO, P8-1–P8-20.

Juzonis, V.; Goranin, N.; Cenys, A. 2010. Genetic Algorithm Modelling Approach for Mobile Malware Evolution Forecasting, in *Proc. of – The 16th International Conference on Information and Software Technologies*, Kaunas, 259–264. ISSN 2029-0063.

Goranin, N.; Cenys, A. 2008c. Analysis of Malware Propagation Modeling Methods, in *Proc. of the Eleventh Lithuanian Conference of Young Scientists Science – Future of Lithuania*. Vilnius: Technika, 428–434. ISBN 978-995-52830-2-7.

Goranin, N.; Cenys, A. 2007. Genetic Algorithm Application in Cryptography and Cryptology (Genetinių algoritmų taikymas kriptografijoje ir kriptanalizėje), in *Proc. of the Tenth Lithuanian Conference of Young Scientists Science – Future of Lithuania*. Vilnius: Technika, 527–533 (in Lithuanian). ISBN 978-995-52814-4-3.

Nikolaj GORANIN

GENETIC ALGORITHM APPLICATION IN
INFORMATION SECURITY SYSTEMS

Doctoral Dissertation

Technological Sciences,
Informatics Engineering (07T)

Nikolaj GORANIN

GENETINIŲ ALGORITMŲ TAIKYMAS
INFORMACIJOS SAUGOS SISTEMOSE

Daktaro disertacija

Technologijos mokslai,
informatikos inžinerija (07T)

2010 04 30. 12,00 sp. l. Tiražas 20 egz.
Vilniaus Gedimino technikos universiteto
leidykla „Technika“,
Saulėtekio al. 11, 10223 Vilnius,
<http://leidykla.vgtu.lt>
Spausdino UAB „Biznio mašinų kompanija“,
J. Jasinskio g. 16A, 01112 Vilnius
<http://www.bmk.lt>