MODERN TECHNOLOGIES USED FOR SECURE DATA COMMUNICATIONS

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea „Politehnica” din Timişoara
în domeniul INGINERIE ELECTRONICĂ ŞI
TELECOMUNICAŢII
de către

Ing. Tatiana Hodorogea

2012

Conducător ştiinţific: Prof.univ.dr.ing. Corneliu Ioan Toma

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt

Cuvânt înainte

Teza de doctorat a fost elaborată pe parcursul activităţii didactice şi de cercetare desfăşurate în cadrul Departamentului de Comunicaţii al facultăţii de Electronică şi Telecomunicaţii, Universitatea „Politehnica” din Timişoara.

Lucrarea este dedicată unui domeniu de mare interes şi cu o dezvoltare semnificativă în ultimii ani: tehnologii moderne folosite pentru securitatea comunicaţiilor de date. Se acordă o atenţie deosebită tehnologiilor şi metodelor de securitate care îmbunătăţesc în mod direct sau indirect securitatea comunicaţiilor de date.

Prezenta lucrare cuprinde o serie de studii critice cu privire la stadiul actual al securităţii datelor. Domeniul securităţii datelor şi aplicaţiilor reprezintă o tematică de mare actualitate in zilele noastre. Tehnologiile clasice par a fi limitate şi de aceea au început să fie integrate unele tehnologii alternative, cum ar fi cea bazată pe genomul uman şi bioinformatica.

Rezultatele obţinute şi originalitatea tezei constau în faptul că s-a reuşit să se îmbine activitatea de cercetare teoretică şi metodologică cu cea practică, oferind un mecanism capabil a fi utilizat în procesul de securitate a datelor folosind tehnologii alternative.

Tehnologiile abordate, aplicaţiile corespunzătoare şi interpretarea rezultatelor sunt menţionate pe parcursul tezei.

Timişoara, aprilie 2012 Tatiana Hodorogea

Hodorogea, Tatiana

Modern Technologies Used for Secure Data Communications

Teze de doctorat ale UPT, Editura Politehnica, 2012, 180 pagini, 55 figuri, 4 tabele.

Cuvinte cheie:
Securitatea transmisiei de date, JCE, criptografie, chei criptografice public/privat, PKI, ADN, genom uman, modele evoluţioniste, bioinformatică

Rezumat,

Teza de doctorat este dedicată unui domeniu de mare interes şi cu o dezvoltare semnificativă în ultimii ani: tehnologii moderne folosite pentru securitatea transmisiei de date. Se acordă o atenţie deosebită tehnologiilor şi metodelor de securitate care îmbunătăţesc în mod direct sau indirect securitatea comunicaţiilor de date.

Prezenta lucrare cuprinde o serie de studii critice cu privire la stadiul actual al securităţii datelor. Domeniul securităţii transmisiei de date reprezintă o tematică de mare actualitate în zilele noastre. Tehnologiile clasice sunt limitate şi de aceea am integrat tehnologii alternative de securitate bazate pe genomul uman şi bioinformatică.

Rezultatele obţinute şi originalitatea tezei constau în faptul că s-a reuşit să se îmbine activitatea de cercetare teoretică şi metodologică cu cea practică, oferind un mecanism capabil a fi utilizat în procesul de securitate a transmisiei de date folosind tehnologii alternative.

Tehnologiile abordate, aplicaţiile corespunzătoare şi interpretarea rezultatelor sunt menţionate pe parcursul tezei.

Table of Content

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AP	Access points
BAS	Broadband Access Server
BMC	Bio Molecular Computation
CA	Certification Authority
CBC	Cipher Block Chaining
CD	Certificate Directory
CDMB	Central Dogma of Molecular Biology
CFB	Cipher Feedback
CRL	Certificate Revocation Lists
CRT	Chinese Reminder Theorem
DHCP	Dynamic Host Configuration Protocol
DMZ	Demilitarized Zone
DN	Distinguish Name
DNA	Deoxyribo Nucleic Acid
DNAE	Deoxyribo Nucleic Acid Encryption System
DNS	Domain Name System
ECB	Electronic Code Book
EHCR	Electronic Healthcare Record
EPR	Electronic Patient Record
ES	Encryption System
GPRS	General Packet Radio Service
HMM	Hidden Markov Model
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
IPsec	IP Security
JCE	Java Cryptography Extension
LKG	Local Central Key Generator
NAS	Network Access Server
NAT	Network Address Translation
OFB	Output Feedback
PAA	Profession Authentication Authority
PCBC	Propagating Cipher Block Chaining
PKI	Public Key Infrastructure
PKRA	Public Key Registration Authority
QAA	Qualification Authentication Authority
RA	Registration Authority
SSL	Secure Socket Layer
TLS	Transport Layer Security
TKIP	Temporal Key Integrity Protocol
TTP	Trusted Third Party
VLAN	Virtual Local Area Network
VPN	Virtual Private Network
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network

1. Motivation and challenges

1.1 General Consideration

Nowadays information systems involve more complexity because of their heterogeneity involving very big threats on networks which are widely spread, open and interconnected.

The security attacks and the technologies to exploit security attacks are growing continuously. The importance and need of providing and maintaining the data and information security across networks is a major motivation and challenge to ensure and maintain information security.

With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today’s web-based Software Applications.

A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory, [59].

A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output.

If an algorithm's behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*.

When an attacker is theoretically able to intrude, but it is computationally infeasible with available resources, the cryptographic system is said to be *conditionally secure.* The mathematics in such systems is based on computational complexity theory, [57].

The design of a secure cryptographic system is a very challenging task. A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output. If an algorithm's behavior is completely determined by the input, it is called *deterministic,* and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic,* [57].

A distributed algorithm in which two or more entities take part is defined as a protocol and includes a set of communicational steps which requires data to be transferred from one side to the other.

The challenge and the goal of a cryptographer are to reduce to zero the probability of a successful attack against the security of an Encryption System (ES).

Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing me to extend the JCE by implementing faster or more secure cryptographic algorithms.

By the same means I provided my independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB), [59].

My thesis work, motivation and challenge were based on the complexity of developing an unconditionally-secure DNA Encryption System as part of my DNA Provider.

The aim motivation and challenge of my thesis consisted in the development of the novel modern encryption techniques based on bioinformatic science.

I provided an independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB).

I proposed and implemented a new, novel idea of a development of a cryptographic package provider; named *DNAProvider as Java Cryptographic Extension (JCE)*.

I extend the JCE by implementing faster and more secure *DNA cryptographic algorithm* based on bioinformatic science.

I proposed, implemented and tested a novel idea and novel algorithm to derive the asymmetric public/private keys pairs used in Public-Key Cryptography, based on bioinformatics and mathematics of the *Evolutionary Models,* which I called DNA *Cryptographic Keys.*

I developed a Unique Process System Pipeline Evolutionary Models of deriving DNA Cryptographic Keys Sequences by deriving the DNA private/public keys from human genome analysis by computing the philogenetic tree and the branches length during evolution for chosen species.

With every year the relevance and the importance of information security is even higher compared to the previsious year, resulting from the following facts and data:

Exponential increase of information volume, used by society, operated through computers and other technical devices.
There were a significant increase of software applications and instruments which don't fulfill the minimal security normative.

As every activity domain is based on an increasing amount of information nowadays cryptography has a very important role in ensuring information security.

Solving the problems related to information security is the aim motivation and challenge of cryptography, [103]. Cryptography is the branch of modern mathematics, with the aim to elaborate mathematical methods to ensure confidentiality, integrity, authentication and non-repudiation of the data.

1.2 Document Structure

The author's thesis contains an introduction part, eight chapters followed by conclusions and perspectives. The thesis contains the bibliography made from 113 titles up to the date of 2012 and a list of research activity up to March 2012, when the major results and contribution of this thesis have been published in the book called "Modern Cryptography".

The originality and scientific novelty of the thesis and obtained results exceed current available methods by security of encryption, consisting in the development and elaboration of the novel and modern methods for data protection based on bioinformatics, novel ideas and novel algorithms which I implemented and tested.

The thesis consists of an important theoretical signification development of the new encryption methods and algorithms using bioinformatics with a higher degree of security and reliability in the encryption systems.

The thesis consists of the very valuable practical work as every proposed idea, method and algorithm has been developed implemented and tested.

Chapter 1, Motivation and challenges make an overview of recent research presenting the motivation and challenges of the author.

Chapter 2, Data Security and Cryptography presents key points of information security, communications and networks. The requirements are: confidentiality, authentication, nonrepudiation and integrity and the mechanism to meet all requirements is very complex.
Developing security mechanisms or algorithms as a must we need to consider potential attacks on those security features. Security mechanisms are divided into the one that are implemented in a specific protocol layer and the one that are not specific to any particular protocol layer. Security aspects come into role when is necessary to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity or integrity.

Chapter 3, Cryptography and DNA Steganography Principles describes the most important aspect of data communications- security which is a basic building block for data security. Encryption is the most important tool for data security in networks communication.
A cryptographic algorithm works in combination with a *key* to encrypt the plaintext. The security of encrypted data is dependent on the strength of the cryptographic algorithm and the secrecy of the key.
A cryptographic algorithm plus all possible keys and all protocols comprise a *cryptosystem*.
This chapter describes the DNA Steganography principles, Symmetric-Key encryption, algorithms used in encryption. The chapter describes the confidentiality and vulnerability using symmetric encryption. The chapter provides an overview from all that has gone before up to the present.
The chapter describes the Public-key algorithms which are based on mathematical functions rather than on substitution and permutation. Public-key cryptography is asymmetric, involving the use of two separate keys and the use of the keys has major consequences in the areas of data security and confidentiality.

Chapter 4, Java Language Security describes the Data Security using a Java Security Model. The chapter presents the Key Management: the way in which keys are stored, transmitted, and shared. The core Java API comes with the necessary classes to handle public and private keys and their certificates.

Chapter 5, DNA and Biology of the Cell living cells on Earth, store their hereditary information in DNA molecules, in the form of double-stranded, long paired polymer chains, formed always of four types of monomers A, T, C, G, arranged together in a long linear sequence that encodes the genetic information, similar to the sequence of 1s and 0s which encodes the digital information in a computer.
Bioinformatics and DNA Engineering describe the bioinformatic science which brings together the following areas of science: molecular biology, areas from mathematics, computer science, pattern recognition, complex systems, physics, graph theory and others.

Chapter 6, DNA Cryptography Model
With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's information infrastructure.

Biotechnological Methods as recombinant DNA have been developed for a wide class of operations on DNA and RNA strands. Bio Molecular Computation (BMC) makes use of biotechnological methods for doing computation and splicing operations allow for universal computation.

6.1 Architecture of a DNA Memory for DNA Cryptography Model
DNA memories have the potential to store vast amount of information with high density, and also have the potential to process the stored information through laboratory protocols matching context and content. This leads us to knowledge mining applications on a massively parallel scale, like DNA cryptography. The chapter describes the design for a DNA memory.

Chapter 7, Complexity of DNA Encryption System as a Subset of Java Cryptography Extension
7.1 Creating the DNA Security Provider with DNA Encryption
The security provider abstracts two ideas: engines and algorithms. Engine is used as a word for operation. *Engine classes* come with Java virtual machine as part of the core API. *Algorithm classes* are a set of classes that implement particular algorithms for particular engines. The goal of the security provider interface is to allow an easy mechanism where the specific algorithms and their implementations can be changed or substituted easily.
I developed cryptographic package provider, named DNAProvider as Java Cryptographic Extension. JCE offers this kind of support for developing cryptographic package provider, allowing me to extend the JCE by implementing faster and more secure cryptographic algorithms. I provided an independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB).
I got the Code Signing Certificate from Sun Microsystems for my DNAProvider which is available for 5 years, until 01/26/13, with the reference nr **679**.

7.3 The Security Class and the Security Manager
The Security class, maintains a list of the provider classes and consults each in turn to see which operations supports.
When the security package needs to perform an operation, it constructs a string representing that operation and asks the Security class for an object that can perform the operation with the given algorithm.
Some of the public methods of the Security class call the checkSecurityAccess( ) method of the security manager. This gives the security manager the opportunity to interfere before an untrusted class will affect the security policy of the virtual machine.
A program that wants to install my developed DNAProvider provider must have been granted the SecurityPermission named "insertProvider.DNA".

7.4 Steps to Implement and Integrate the DNAProvider
In order to implement the DNAProvider I performed the following steps: **Step 1:** Write The DNAProvider Service Implementation Code, **Step 2:** Give my Provider a Name, (DNAProvider), **Step 3:** Wrote the "Master Class," a subclass of Provider, **Step 4**: compile the Code **Step 5:** Prepare for Testing and Get a Code-Signing Certificate. In order, is necessary to mail all the hardcopy containing DNAProvider, (CSR) and contact information to was mailed to **Sun Microsystems, Inc. Santa Clara, CA 95054, U.S.A.**

Step 7: Run the Test Programs using the Provider Code Signing Certificate and JCE Root CA Certificate, **Step 8:** Document DNAProvider, **Step 9:** Make DNAProvider Software and Documentation Available to Clients.

7.5 DNA Encryption System as a Subset of Java Cryptography Extension
Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing me to extend the JCE by implementing faster and more secure cryptographic algorithms. By the same means I provided my independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB). I proposed to encode the medical records of an individual in DNA data strand flanked by unique primer sequences, which I obtain in the process of deriving a DNA secret key from human genome analysis.

Chapter 8, Deriving DNA Cryptographic Keys
I developed the cryptographic package provider, named DNAProvider as Java Cryptographic Extension (JCE), which allowed me to extend the JCE by implementing faster and more secure DNA cryptographic algorithm.
The DNA cryptographic algorithm developed by me encodes the medical records of an individual in DNA data strand flanked by unique primer sequences, which I obtain in the process of deriving the asymmetric DNA secret keys from human genome analysis. The development, implementation and testing of a DNA Encryption (DNAE) system is based on the Central Dogma of Molecular Biology (CDMB), where I derive asymmetric DNA Cryptographic Keys, for the public-key cryptography, based on evolutionary models. With the developed and implemented system pipeline evolutionary models, I extracted and align the DNA sequences of the same gene from related chosen species with respect to human DNA Sequences. The alignment in the evolutionary system pipeline is realized with ProbCons tool, which is a pair Hidden Markov Model based on progressive alignment algorithm that primarily differs from most typical approaches in its use of *maximum expected accuracy.*

8.1 Deriving Asymmetric DNA Cryptographic Keys
As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution. Because of the degeneracy of the genetic code leave the protein unchanged, called synonymous or silent changes. I aligned my extracted DNA Sequences with ProbCons tool. Fundamentally ProbCons is a pair Hidden Markov Model based progressive alignment algorithm that primarily differs from most typical approaches in its use of *maximum expected accuracy.* I derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

8.2 Unique Process System for Evolutionary Models
The molecular evolution model is a complex process, involving different influences mutational biases, interactions, heterogeneous recombination rates, population mixing patterns, temporal variations in population size, time-dependent selection

and frequency-dependent selection. Such models assign probabilities to multiple-alignment columns in terms of the substitution rates and lengths of the phylogenetic tree branches.

I computed the philogenitic tree (for my chosen species) and the branches length during evolution for my chosen species with respect to human (hg18), following the theorem of Golding and Felsenstein (1990), the Halpern and Bruno (1998), who have shown that mutation limit of the standard Kimura-Ohta theory, one can determine substitution rates in terms of the mutation rates and the equilibrium frequencies *w.* In particular, if *r* is the rate of substitution from a *base a,* to a *base b* at position *i*, µ is the rate of mutation from *a to b* and *w* is the equilibrium frequency of nucleotide i, at this position, [76].

8.3 SmartCipher Application Integrating DNAProvider with DNA Encryption (DNAE) System

In collaboration with Helios group I developed in Java Language a software application named SmartCipher and integrated my DNAProvider with DNA Encryption (DNAE) system I developed based on the Central Dogma of Molecular Biology (CDMB) in SmartCipher application.

The interface was developed using Java IDE Jigloo GUI Builder plugin for Eclipse, containing a CTextArea where the user can introduce the desired text and desired algorithm for encryption: Des, TripleDes, Blowfish and DNA encryption.

2. Data Security and Cryptography

2.1 Information security bases

Nowadays information systems involve more complexity because of their heterogeneity involving very big threats on such kind of networks which are widely spread open and interconnected. The security attacks and the technologies to exploit security attacks are growing continuously.

The importance of providing and maintaining the data and information security across networks is a major enterprise business activity resulting in a big demand and need to ensure and maintain information security.

Networks are based on a number of network level equipment and servers as:

- **Dynamic host configuration protocol (DHCP)**, server dynamically assigns an IP address.
- **Domain name system (DNS),** server translates a domain name (URL) into an IP address.
- **Network address translation (NAT**), performs translation between private and public addresses.
- **E-mail server** supports electronic mailing
- **Internet/Intranet/Extranet Web** servers
- **Access points (AP),** giving wireless equipment access to wired network.
- **Virtual LAN (VLAN)** which virtually separate flows over the same physical network, so that direct communications between equipment from different VLANs could be restricted and required to go through a router for filtering purposes
- **Network access server (NAS) / Broadband access server (BAS),** gateways between the switched phone network and an IP-based network
- **Intrusion detection system (IDS) / Intrusion prevention system (IPS)** used to detect intrusions based on known intrusion scenario signatures.

NIST Computer Security Handbook defines computer security as the protection afforded to an automated information system to preserve the key objectives: integrity, availability and confidentiality of information system resources (software, hardware, data and telecommunications), [99].

Cryptography gives us all of these services, linked with transmitted or stored data.

Considering GRID computing security where the heterogeneous resources are shared and located in different places belonging to different administrative domains over a heterogeneous network, additional security requirements must be satisfied compare to classical network security.

A GRID is a software toolbox and provides services for managing distributed software resources. Securing information in GRID computing encompasses verifying the integrity of the message against malicious modification, authenticating the source of a message and assuring the confidentiality of the message being sent.

The key objectives of computer security are three concepts known as CIA triad:

- Confidentiality with its two concepts: *data confidentiality* assuring that information is not available to unauthorized parties and privacy assuring an individual control of its one information.
- Integrity-assures that information is changed in authorized manner and assures authorized system manipulation.
- Availability assures the promptitude of the system and services to authorized parties.

Confidentiality implies the prevention to disclosure information by individuals and unauthorized systems, [71].

In information security, ***integrity*** implies the impossibility of data modification without the authorization and keeping the data unchanged. ***Authentication*** means the knowledge of the source, from where the data was received.

The information needs to be ***available*** when necessary. The assurance of availability implies the prevention of denial of service attacks.

Communication between GRID entities must be secure and confidentiality must be ensured for sensitive data, from communication stage, to potential storage stage. Problems of integrity should be detected in order to avoid treatment faults, availability is directly linked to performance and cost in GRID environment.

Cryptographic algorithms for confidentiality and authentication play a major importance role in nowadays information security, [49].

The authentication of entities leans on centralized or distributed approach. Distributed approach is based on defining the public private keys for each entity. To avoid spoofing attacks the public key is distributed in the form of electronic certificate and the authenticity is guaranteed by a certification authority (CA).

Certificates are used for signing and encrypting emails and using secure socket layer (SSL) for securing sessions with web servers.

Virtual Private Network (VPN) security leans on SSL and data security protocol, IP security (IPsec). IPsec protects IP packets exchanges with authentication of the origin, data encryption, integrity protection at the IP packet layer. VPNs secure interconnection between remote private networks by the use of VPN gateways positioned at the borders of the VPN tunnel. The IPsec or L2TP tunnel over IPsec is configured between this gateways replacing the border router with firewall or increasing border routers with IPsec capabilities. For private networks to remain protected from intrusions the traffic is filtered at the border of private network, [110].

A demilitarized zone (DMZ) is a restricted subnet, which is separated from the private, public networks allowing servers to be accessible from other areas and keeping them protected. Prior to deploy any security equipment in a network is to define all existing services in the close future and on of is the security levels: encryption, layer-3 and layer-4 filtering.

2.2 Security Services, Threats and Attacks

Defining a high-level of security architecture all the lines of defense must be introduced. The DNS servers and Internet servers must be located on a DMZ of the external firewall. In order to improve filtering level of sensitive servers (intranet WEB), additional proxies must be added. HTTP proxy for intranet Web must be installed in the DMZ in order to do user authentication and high control on HTTP data including format and content.

The external firewall must be configured so that HTTP traffic to Intranet Web is redirected to HTTP proxy for filtering, [42].

As a must in order to avoid direct communications between subnets of the internal network and protect servers from users, VLANs must be defined. For a higher security level protection wireless network need to be considered as a specific VLAN within the "internal" network. To design the appropriate security architecture to be protected against security attacks is a huge task as it depends on various parameters: network architectures, security constraints, size of the company, budget available, branch offices, management of remote users and branch offices.

Modern mobile devices, PDA computers and smart phone are portable and easily lost and stolen, they have connection interfaces to several types of wireless networks such as wireless local area network (WLAN), general packet radio service (GPRS), infrared data association (IrDA), and unfortunately only few such devices are presently equipped with anti-virus software or firewalls.

Encryption and authentication are therefore strongly recommended solutions in order to avoid loss of data and confidentiality, if a mobile device is lost or if is stolen, [66].

Malicious software does not only cause serious threats for the mobile device itself, it may also cause a threat for the network which the mobile device is connected to, [94].

Embedded security feature in Symbian OS are cryptographic module with implementations of symmetric algorithms (DES, 3DES, RC2, RC4, and RC5), asymmetric cryptographic algorithms (RSA, DSA, and DH), implementations of hash functions (MD5, SHA1, HMAC) and a pseudo-random number generator for cryptographic key generation and certificate management module, (according to Symbian Signed), [66].

There are two types of security attacks: ***active and passive attacks***. During an active attack a message or a file can be modified or system resources are altered, original data stream can be substituted with a false created data stream.

Passive attacks imply traffic analysis, transmission monitoring and unauthorized read of information transmitted over network. Passive attacks do not affect the resources of a system and are very difficult to detect. These kinds of attacks are possible to prevent by the means of encryption.

Security services must implement security polices which in order are implemented by security mechanisms. An *authentication security service* needs to assure that a communication is authentic, [12].

Authentication service must assure the receiver that the message is indeed from the claimed source sender and each entity is the one it claims to be. The authentication service as well, needs to assure that an attacker as a third party cannot masquerade as one of the legitimate party. Security services related to authentication as Data Origin Authentication and Peer entity authentication are defined in X.800.

Access control security service prevents the unauthorized use of resources, this service controls what entity can have access to resources and what are the conditions that this services will take place.

During *data confidentiality security service* the transmitted data is protected against passive attacks and traffic flow analysis. If the data sent by an authorized individual is received intact with no alteration and modification the *data integrity security service* is satisfied, [90].

Protection service against denial is a must; these requirements are generated by *nonrepudiation security service*. The resources need to be accessible upon request of an authorized system entity.

Availability security services protect a system to ensure the availability to an authorized entity and these services are defined by RFC 2828 and X.800. Access control on a mobile device is implemented using a combination of the following security services: authentication service, confidentiality service, nonrepudiation service and authorization, [23].

A PKI SIM card is a basic SIM with PKI functionality where a RSA coprocessor is added which performs public key based encryption and signing with private keys. The PKI SIM card contains space for storing private keys and certified public keys needed for digital signatures and encryption, [66].

The authentication and key management technique used in 3G (third generation/UMTS) networks is based on the same principles as in GSM networks. Typical network services transferring confidential data to and from a Symbian devices are E-commerce and electronic payments and run over HTTP and WAP connections. Connection security means: availability of data communication, mutual authentication of communicating partners, integrity of data communication and possibility of confidential data communication, intrusion prevention/detection and malware rejection, [66].

2.3 Security Mechanisms for Network Protection

The use of WLAN networks in enterprises nowadays is a popular method for providing connectivity. I will present the security mechanisms for protecting WLAN networks from security threats.

The main threats to WLAN networks are the radio waves since the radio waves broadcast, without respect to neither walls nor other limit. Denial of service (DoS) makes the network ineffective. It is easy to jam a radio network and network becomes unusable. By the use of rush access the network is overloaded with malicious connection request. Tools are able to detect this kind of traffic and help network administrator to identify and locate the origin, [19].

Intrusions threats are most common attacks where the intrusion is done via client station and protection is the same as for wired networks: the firewalls must be used. The most critical attack that aims to take the control of network resources of the enterprise is the network intrusion and in this case Wi-Fi dedicated Intrusion Detection Systems (IDS) are efficient against such attacks.

With falsification of access points the hacker fetches the traffic on the network and the security protection from such attacks is by detecting abnormal radio transmission in unexpected areas. Security protections can be applied to WLAN: network monitoring is a good defense to observe the network to be informed if something strange happens.
The intrusion detection system (IDS) is used against network intrusions. IDS correlates suspect events, tries to determine if they are due to an intrusion.

Traffic monitoring prevents against spoofing due to permanence observing of the Wi-Fi traffic in order to detect any inconsistent situations, [4].

Network engineering is another security mechanism for network protection. It is strongly recommended to deploy WLAN using switches instead hubs and to control the traffic between wired networks. WLAN dedicated switch manages radio, networking and security functions and access points are used only as emitters and receptors providing a better protection against attacks. The firewalls manage protections at addressing level by providing filters and log connections, managing access control list (ACL) which are used for access filtering and monitor the connections. The firewalls must be installed in a DMZ, VPN authentication with encryption mechanisms activated.

The use of VLAN must be done in order to split the network for the isolation of strategic data from the radio network. For this, VLAN must be deployed on a dedicated virtual LAN structures where network contains several VLANs and each associated to a WLAN subnet with own SSID. All VLANs must be connected on the WLAN switch.

Encryption is the security mechanism at the application level by its use if the information is intercepted is unusable. In this scope standard protocols like transport layer security (TLS) may be used. Authentication is done by a login password sequence and link between client and server is secured by TLS, authentication is done via a local authentication database.

MAC addresses filtering is a non-cryptographic security feature uses the unique link layer (MAC) address of the WLAN network card and identifies legitimates users.

One of security feature based on cryptography is wired equivalent privacy (WEP), defined in the initial IEEE 802.11 standard and provides authentication and encryption with 40-128 bit key length. The key should be changed in all nodes and in the access points, frequently and simultaneously, [34].

Because of WEP weakness, the IEEE designed a protocol named 802.11i, known as WPA 2 (Wi-Fi Protected Access 2). Temporal key integrity protocol (TKIP) is used for generating per-packet keys for the RC4 ciphering used. The key is called temporal because is changed frequently and is combined with the sender’s MAC address using the exclusive OR-operation. Resulting in the usage of different keys for upstream and downstream transmissions,

Two types of security mechanisms are known: first type is the one which are implemented in a certain protocol layer. Second type of the security mechanisms are not related to protocol layers or any security services.

Cryptography encrypts the data by the mean of using encryption security mechanism. *Encryption security mechanism* is an encryption algorithm which encrypts and decrypts the data, transforming it into unreadable format. The encryption mechanism depends on encryption keys being used (zero or more) and encryption algorithm. After the readable data is cryptographically transformed, digital signature is appended to it as a second security mechanism, [91].

Digital signature security mechanism proves the integrity of the data, the source of the data and protects the information send against forgery.

The access right to information and resources is realized thought the third security mechanism known as: *access control security mechanism*.

For preventing traffic analysis attempts the bits are inserted into the gaps of the data stream and this constitutes the *traffic padding security mechanism*.

To ensure the identity of an individual by the mean of information exchange *authentication security mechanism* comes into play, [99]

When a breach of security is suspected *routing control security mechanism* is able to generate the selection of physically secure routes for the data.

2.4 Security Model and Data Security in Information Systems

A security model represents a secure transfer of information across information channel (internet), between two principals: sender and receiver, by the use of communication protocols, (Fig. 1).

Data security model implies the protection of data against confidentiality and authentication threats coming from an opponent. Security related transformation is needed to satisfy these conditions of data protection during transfer through information channel. Encryption transforms the message in an unreadable format, by the opponent. The additional code is added to the secret information based on the content of the message and this way the identity of the sender is verified, [98].

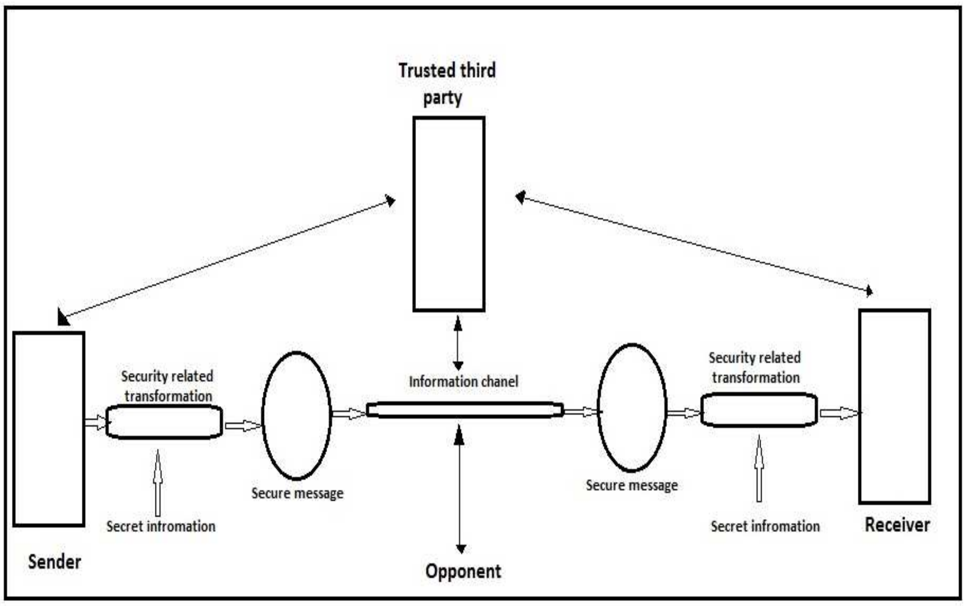


Fig. 1 Data security model

Implementing a security service we need a secure encryption algorithm to perform the security related transformation and methods for sending and sharing the secret messages.

As secret information traveling through information channel is vulnerable to information access threats and service threats it is necessary to make use of secure encryption algorithm and specify the protocol being used by the sender and receiver during secure communication.

Anytime a software working with cryptography deals with secrets this means that the respective software has to ensure that the secrets do not leak out.

Secure channel has two types of secret: keys and data. One important rule of writing secure software applications is to wipe the information as soon as we no longer need it.

In C we must take care of wiping information by ourselves. In object-oriented languages as C++ there is a destructor function for each object and the destructor can wipe the state. The main program must behave properly and destroy all objects no longer needed wiping the memory state.

In Java language situation is more complicated as all objects live on heap which is the garbage-collected and it means that finalization function is not called until garbage collector realize that the object is no longer in use, [99].
If we need to store large amounts of data the best solution is to encrypt a large block of data and store the ciphertext in the memory and the key needs to be stored in a way to avoid data retention. A best way is provided by Boojum, [26].

For Boojum, if m is the data we want to store we generate a random string R and store both: R and $h(R) \oplus m$, where h is a hash function. These two values must be stored in different memory location and not too close together. R needs to be changed regularly. At regular intervals say 2 seconds it is necessary to generate a a new random $R^{'}$ and update the memory to store $R \oplus R^{'}$ and $H(R \oplus R^{'}) \oplus m$, ensuring this way that each bit of memory is written with a sequence of random bits, [46]. The bits must be not adjacent on the RAM chip.
Modern computers have a smaller and faster memory called cache which keeps a copy of the most recently used data. CPU in order to access the data it checks the cache first. Cash keeps a copy of the data including secret data.

Information system known as Electronic Healthcare Record (EHCR) and Electronic Patient Record (EPR) permits to access a patient record from any place with internet connection, [9].

In Hospital Information System (HIS) security is a very complex issue related to legal and ethical dimensions, defined as security policy.
A Trusted Third Party (TTP) is an entity which facilitates interactions between two parties who both trust the third party.

Implementing security in Hospital Information System by using passwords for verification of claimed identity has disadvantages in distributed systems. The public key certificates are transferred to users and stored in a browser, [66].

There are two types of public key and attribute certificates: Clinicians and Patients.
One of TTP is a Certificate Authority (CA) defined in X.509 standard, which defines a framework for the authentication service provided by a director to all users.
X.509 defines the following authentication levels: in simple authentication a password is used for identity verification. Strong authentication is based on asymmetric cryptography and involves the use of a pair of keys public/private.

An example of using two-way strong authentication is given in CEN ENV 13729 standard, which uses strong authentication in health information systems (Fig. 2), using X509 standard.
TTP sites were established in 4 different locations in Europe:
1. Institute of Computer and Communication Systems - ICCS (Athens-Greece),
2. University Hospital Magdeburg - UHM (Magdeburg-Germany),
3. University of the Aegean – UoA (Samos-Greece)
4. University of Calabria - Uni-CAL (Calabria-Italy).

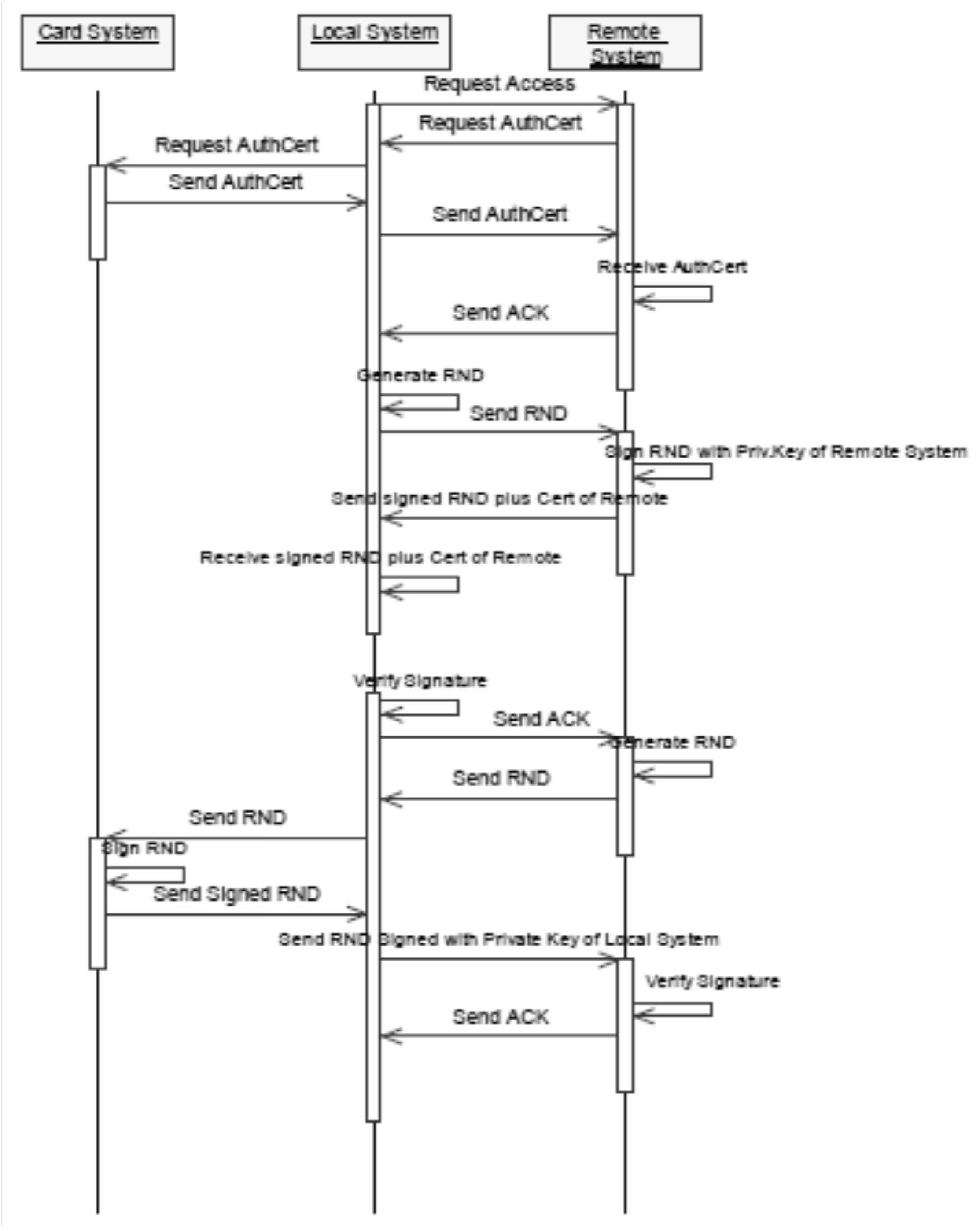


Fig. 2 Remote strong authentication according to CEN ENV 13729

The following TTP were responsible for defining TTP services as TTP components:
- Key generation instance
- Registration authority
- Naming authority
- Directory service authority

The functions performed by CA are: initialization, electronic registration, authentication, key generation, distribution, key personalization, certificate generation, certificate directory management, certificate revocation, CRL generation, maintenance, distribution storage and retrieval.

The aim of the project was to create a national infrastructure that would provide the security on the communication and application level of health information systems in the participating countries.

The architecture comprised the following as seen in Figure 2:
1. The use of Health Professional Cards (HPC), microprocessor cards used in the authentication process
2. Card reader services
3. Services of Trusted Third Party which manages certificates.

The following services have been provided:
1. One of service was to send a physician's report with relevant data to a central register.
2. Execution of pre specified and formed SQL queries relating to a patient's EHCR.
3. Statistical analyses by various criteria and by making a SQL query
4. Exchange of information, including HL7 messages, images.

The ***Card System***, (Fig.2) called Card Issuing System (CIS) and is an entity issuing microprocessor cards containing a private key and certificates as well.

The ***Local System*** performs the generation of a private public key pair and is called Local Central Key Generator (LKG), an entity that may be located locally with a Public Key Registration Authority (PKRA). PKRA identifies in a unique way the user requiring the service of the provision of a digital signature. Remote System is the system where the Certificates have to be stored in the Certificate Directory (CD).

Each user has a unique distinguished name assigned by naming authority and the user is identified by proving the possession of private key. In order to verify a user private key, the communication partner must possess a public key, available from the directory. A public key is given to the user from the trusted source represented by CA. CA uses its own public key in order to certify a user's public key and in this way produces a certificate. On a user's request, relevant information is verified, associated to a Distinguish Name (DN) and sent online to a certification entity, [10].

The Registration Authority (RA) uses the information provided by the Qualification Authentication Authority (QAA), or by Profession Authentication Authority (PAA).

Based on data obtained from the RA Certification Authority creates certificates.

1st: Certificates associating users distinguish name and the remaining relevant information to user's public key are referred to as public-key certificates.

A certificate offers the properties to the user having access to CAs public key can disclosure the public key on which the certificate was created and only CA can make a modification to a certificate. The certificate is obtained by creating users digital signature based on a set of information about the user as user public key unique name.

2nd: Certificates associating information about profession, qualification are attribute certificates.

The first service is provided by the CA and the second by the PCA.

Regarding the basic requirements of secure communication and secure cooperation in distributed systems based on networks as well Hospital Information System (HIS), security is a very complex issue and basic security services are required.

In information systems the security services required must provide:

- Availability, Audit, Accountability including Nonrepudiation and Access Control
- Identification
- Authentication
- Integrity
- Confidentiality

Generating good random numbers is vital and challenging for cryptography. The measure for randomness is called entropy and measures how uncertain one is about a value and does not measure how many bits are in a value, [106].

The most common definition for entropy is:

$$H(x) := -\sum_{x} P(X = x) \log_2 P(X = x) \qquad (2.1)$$

Where P(X=x) is the probability that variable X takes on value x. The primes a very important in cryptography as we can compute modulo prime:

To compute (a mod p) one need to find two integers such as: q and r that:

$$a = qp + r \text{ and } 0 \le r < p \qquad (2.2)$$

The RSA system is the mostly used public-key cryptosystem providing encryption and digital signature, based on the difficulty of factoring large numbers, [103].

Invented by Ronald Rivest, Adi Shamir, Leonard Adleman and published in 1978, instead of doing computations modulo a prime p as in DH systems RSA does computations modulo the composite number n based on Chinese Reminder Theorem(CRT), [102]. CRT is the product of different multiple primes with the additional software complexity and necessary conversions.

In Distributed Systems protecting human privacy rights as confidentiality are indisputable.

Information security in nowadays information systems: heterogeneous, widely spread and involving more complexity has a dramatic drawback regarding threats and vulnerabilities, [90].

Information security is more vulnerable to a wide range of threats and attacks, which appeared during the last few years and is continuously growing with an emergency in new and modern technologies for security of software applications.

3. Cryptography and DNA Steganography Principles

3.1 Cryptography and Cryptographic Systems

Cryptography is the science and art of encryption using mathematics for encryptions and decryption of the data.
Cryptography is a part of a security system and encounts wide range of fields such as higher algebra, computer security, quantum physics, economics, statistics, civil and criminal low, [103].

Kahn's book, "The Codebreakers" tells us that 4000 years ago the Egyptians were the first who made use of cryptography. Cryptography was used as a tool for protecting secrets by governments and federal agencies playing a crucial role in world wars during history, [67]. Cryptography is one set of mathematic techniques for transforming desired information to remain secret during transmission across insecure network and intangible for unwanted adversary.

Handbook of Applied Cryptography is an encyclopedia of cryptography, [81]. The theory of cryptography is covered in a good book by Goldreich, Foundation of Cryptography.

The measures of cryptography are: time and resources and ***the security of encrypted data depend on strength of the mathematic algorithm used in encryption and secrecy of the key***. Encryption has two forms: *symmetric-key encryption* and *public-key encryption*. In symmetric-key encryption the same key is used for encryption and decryption, [22].
In public-key encryption two different keys are used for encryption and decryption.

A cryptographic system includes a cryptographic algorithm, all keys and protocols and is characterized by following dimensions:

The first dimension is the type of operation used for encrypting the plaintext into ciphertext. Operation used in all encryption algorithms are based on substitution and transposition.

The second factor characterizing dimension of the cryptosystem is the method of processing the plaintext: as a stream cipher or as a block cipher. Stream cipher processes the input plaintext continuously by one element at a time compare to block cipher which takes from the input plaintext one block of its elements at a time.

The third dimension which is not the least one is in the number of the keys being used by the sender and receiver, making use of one key in the case of secret-key cryptography or two different keys in the case of public-key cryptography.

As long the secrecy of the data is be desired there will always be security attacks on it coming from intruders attempting to find out the secrecy of information transmitted along communication. Attackers in this case were given a name of *Cryptanalysts*; they use the science of cryptanalysis involving mathematical tools, analytical reasoning and not only, in the scope of breaking secure communication. Secret information transmitted during communication between parties is threatened and exposed to this kind of cryptanalytic attacks. The science of cryptology comprises cryptography and cryptanalysis, [7].

A cryptographic algorithm is said to be *computationally secure* if the cost of breaking the cipher text is higher than the value of the encrypted information and the time required for breakage exceeds the lifetime of the secret information itself.

A cryptographic algorithm is said to be *unconditionally secure* if for an attacker is impossible to decrypt the cipher text no matter how much time and cipher text she/he has available, [28].

The scheme of one-time pad proposed by Joseph Mauborgne comes as an improvement to Vernam cipher as it uses a random key long as a message itself and is not repeated in communication between parties. The key used in one-time-pad for encryption and decryption of the single message is discarded and not repeated as for every new message a new key is required.

3.2 Symmetric Ciphers and Symmetric-Key Encryption

Encryption is the original goal of cryptography, [103]. The encryption where the same key is used for encryption and decryption is called symmetric-key encryption. Symmetric-key encryption encrypts a message using a secret key and an encryption algorithm. The decryption is done using the same key and the decryption algorithm.

Symmetric ciphers uses two kind of techniques for mapping plaintext elements into ciphertext: *substitution* and *transposition*. Figure 3, represents the scheme of the symmetric-key encryption with its components: the plaintext serving as an input to the encryption algorithm with input secret key performs the encryption of the plaintext resulting in the ciphertext as an output. The decryption algorithm takes the ciphertext, the secret key and produces the plaintext what was send by the sender, [99].

For secure use of symmetric encryption there are two requirements: the sender and receiver must obtain the copies of the secret key in a secure mode and along with these a strong encryption algorithm is needed.
The sender inputs a *plaintext message*,

$$X = [X1, X2, \ldots, XM] \tag{3.1}$$

With *M* elements which are *letters* in a used alphabet. K is the generated key at the message source used for encryption:

$$K = [K1, K2, \ldots, Kj] \tag{3.2}$$

The encryption algorithm takes as an input the message *X* plus encryption key *K* and generates as the output the ciphertext:

$$Y = [Y1, Y2, \ldots, YN] \tag{3.3}$$

Ciphertext is obtained by encryption algorithm E as a function of the plaintext *X* , with the function determined by the value of the key *K*:

$$Y = \mathrm{E}(K, X) \tag{3.4}$$

Receiver who possesses his key is able to decrypt the message inverting the transformation:

$$X = \mathrm{D}(K, Y) \tag{3.5}$$

An attacker with knowledge of encryption and decryption algorithms who might intercept the cipher text but without access to the key and plaintext may attempt to obtain one component or both: key or the plaintext, [99].

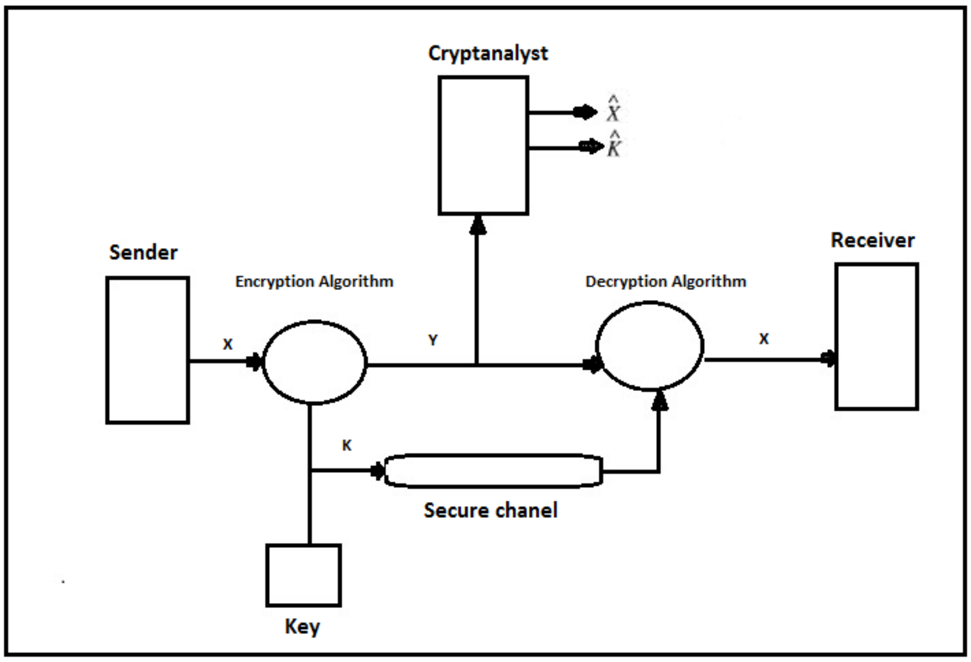


Fig. 3 Secret Key Encryption

Symmetric cryptosystem can be classified according to the used algorithm for encryption and decryption: *block ciphers* and *stream ciphers*.
Block ciphers operate on a group of bits of fixed dimension (blocks).The base transformation in this kind of ciphers are substitution and transposition which are iteratively repeated.

In order to encrypt a message with a length different from the length of the block the message is divided in chunks of the same length as the block, encrypted and concatenated resulting in the encrypted message.
Most of the messages can't be divided in blocks of the fixed length (32 and 128 bits), resulting in the shorter length of the last block. The padding method is used for solving this kind of issues. A fixed predefined number of bits are added to the plaintext resulting in the plaintext length equal as a multiple of block dimension.

The following encryption systems use algorithms with secret key: DES, IDEA, FEAL, LOKI and RC2.
An algorithm is considered secure if the cost of breaking the cipher text exceeds the value of the encrypted text and if the time required breaking the cipher exceeds the lifetime of the encrypted text, [107].
In this case the encryption scheme is called computationally secure. An encryption scheme is unconditionally secure if the encrypted information does not contain enough information to determine the correspondent plaintext.

DES (Data Encryption Standard) was developed by NBS (National Bureau of Standards) and NSA (National Security Agency) and published in 1977. DES encrypts the data in blocks of 64 bits in length, and the key used has 64 bits length. From 64 bits of the key only 56 are used accordingly the other 8 bits are used as parity bits, [98]. DES consists of 16 steps of processing called rounds,

which are producing the cipher text. The numbers of rounds are exponential proportional with the time necessary to find out the secret key in the case of a brute force attack, [6].
AES (Advanced Encryption Standard) is a symmetric block cipher and can process blocks of 128 bits in length, using the keys with the length of 128, 192 or 256 bits.

In the case of the *Stream ciphers* the encryption procedure operates symbol by symbol and encrypts one byte at a time. To protects the data against brute-force attacks, the key needs to be sufficiently long, at least 128 bits.
RC4 is a stream cipher designed in 1987 by Ron Rivest (letter R for RSA). The key size is variable, is a variable key-size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. RC4 is used in the following standards: SSL/TLS (Secure Sockets Layer/Transport Layer Security), [23].
The security of symmetric encryption relies on protection of the cryptographic key. The management of the cryptographic key is an important factor in data security as key generation, distribution and storing.
Traffic padding is an effective measure in security of symmetric encryption. During the traffic padding random bits are sent during a certain period when no encrypted data is transmitted, [56].

Key distribution involves usually the use of master keys and session keys. Master keys are infrequently used compare to session keys that are distributed for temporary use between individuals engaged in communication.
Two parties in communication making use of symmetric encryption must share the same key protected from others for their own use.

Cryptographic system strength is in the technique of key *distribution.* Considering Otto as a sender and Stefani the receiver, Otto can select the key and deliver the key physically to Stefani.

A third party individual can select the key and deliver the key to Otto and Stefani. If they used the key recently one of them can transmit a new key to the other, the new key is encrypted using the old key. If the connection between parties and the third parties is encrypted, the third party can transmit the key to Otto and Stefani using the encrypted link.
In the case of N hosts, the number of required keys is:

$$[N(N\ 1)]/2 \qquad (3.6)$$

If encryption is done at the application level, then a key is needed for every pair of users or processes that require communication.

3.3 Public Key Cryptography and Principles of Public-Key Cryptosystems

The development of public-key cryptography represents the evolution of cryptography. Public-key algorithms are based on mathematical functions and not on substitution and permutation. Public-key cryptography is asymmetric compare to symmetric cryptography, involving the use of two separate keys, [98].

Asymmetric algorithms are based on two keys: one used for encryption of the message and the other one used for decryption of the message. It is computationally impossible to determine the decryption key knowing only the cryptographic algorithm and encryption key.
During the process flow of public-key encryption an algorithm takes as an input a plaintext, a pair of selected keys, one key for use in encryption and the other key for use in decryption resulting in the cipher text as an output.

Each party generates a pair of keys and then places on of two keys in a public register the other key is kept secret. If Otto wants to send a message to Stefani encrypts the message with Stefanie's public key, (Fig.4).
When Stefani receives the cipher text she can decrypt it with the use of her private key.

Private keys are generated locally by each party engaged in communication and are never distributed as there is no need for it.
As long the user private key is well protected to remain secret the incoming communication is secure, [99].
Otto the sender as a source (Fig.4), produces a message in plaintext:

$$X = [X1, X2, \dots, XM,] \tag{3.7}$$

The M elements of X are letters in a used finite alphabet. The message is intended for Stefani as destination.
Otto generates a related pair of keys: a public key, *PUo*, and a private key, *PRo*. *PRo* is known only to Otto, whereas *PUo* is publicly available and accessible by Stefani. With the message X and the encryption key *PUo* as input, Stefani forms the ciphertext:

$$Y = [Y1, Y2, \dots, YN] \tag{3.8}$$

Resulting in:

$$Y = \mathrm{E}(PUo, X) \tag{3.9}$$

Stefani as the receiver, in possession of the matching private key, is able to invert the transformation:

$$X = \mathrm{D}(PRo, Y) \tag{3.10}$$

An adversary, observing Y and having access to *PUo* but not having access to *PRo* or X, must attempt to recover X and or *PRo*, the adversary does have knowledge of the encryption algorithm (E) and decryption algorithms (D).

Otto prepares a message to Stefani and encrypts the message using Stefani's private key before transmitting it. Using Otto's public key Stefani can decrypt the message

As the message was encrypted with Otto's private key it means that only Otto encrypted the sent message and the encrypted message serves as digital signature. It is impossible to modify and alter the encrypted message without Otto's private key, resulting in message source authentication and data integrity.
It is computationally easy for Otto to generate a pair of public/private keys PUo, PRo.

Otto as a sender knowing the public key and the message M for encryption generates the ciphertext:

$$C=E(PUo,M) \tag{3.11}$$

Stefani as a receiver using the private key can decrypt the message:

$$M=D(PRo,C)=D[Pro, E(PUo,M)] \tag{3.12}$$

If an intruder Rya knows somehow the Otto's public key PUo it is computationally impossible to determine Otto's private key. Public-key encryption is vulnerable to brute-force attack as no matter how long the key is.

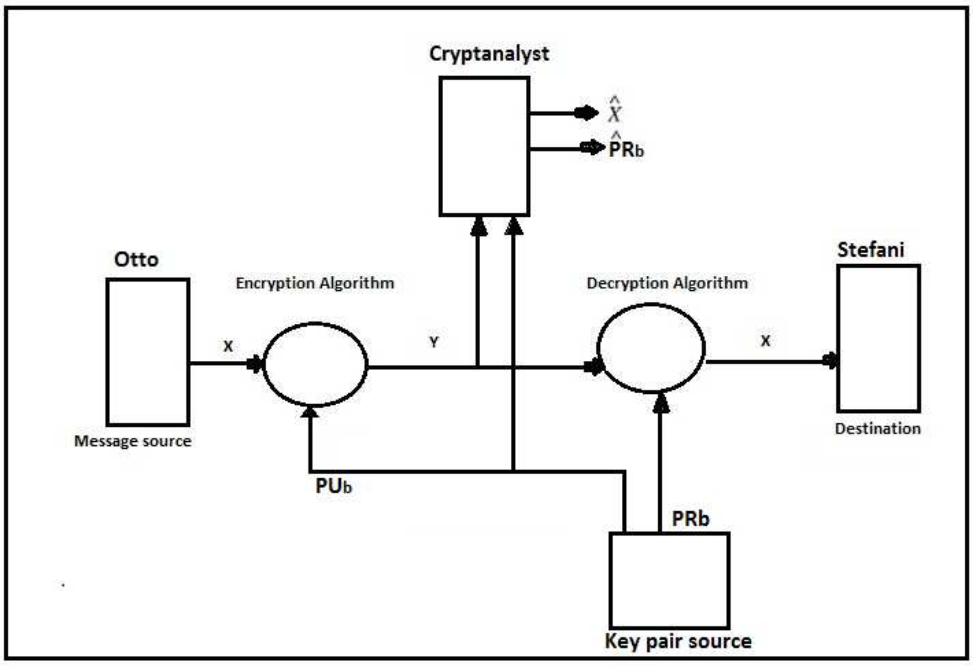


Fig. 4 Public-Key Encryption

In 1977 a new cryptographic algorithm was developed by Ron Rivest, Adi Shamir and Len Adleman and published in 1978. This algorithm is well known as RSA. This algorithm was proven by Paul Kocher to be vulnerable to timing attacks. In this kind of attacks an intruder can determine the private key keeping track of the necessary decryption time. Time complexity serves as a measure of efficiency for a certain algorithm, [102].

In public-key encryption the public keys are public and distributed by public announcement, public-key certificates, public available directory and public-key authority. The public-key authority maintains public keys, names and entries for each participant who registered a public-key with the authority. The registered key can be replaced by the participant at any time.

Otto as a sender sends a request to the public-key authority, with a times tamped message, requesting Stefani's public key as a receiver, [57].
The public-key authority sends to Otto an encrypted message with authority private key.

Otto can decrypt the message using authority public key and is assured that the message is indeed from the authority to which he sent the request.
The plain message from the authority contains Stefani's public key PUs and Otto can use to encrypt a message for Stefani. The message contains as well the original timestamp that he can see that the message is not an old message.

After an encrypted message is sent to Stefani from Otto she can retrieve from the public-key authority in the same manner Otto's public key for further use in communication between them.

Public-key message authentication codes are represented by the digital signature in author's case: Otto uses a key generator algorithm to generate a key pair(Sotto,Potto) and publishes his public key Potto. In order to send a signed message *m* to Stefani he computes a signature and sends the m and s to Stefani:

$$S := \sigma(Sotto, m) \tag{3.13}$$

Stefani in his turn uses a verification algorithm that uses her public key to verify the signature:

$$\upsilon(Potto, m, s) \tag{3.14}$$

She needs to have Otto’s public-key to verify that the message indeed came from Otto. In real life digital signature has a number of limitation: then main problems is that Otto’s computer computes the digital signature and therefore is no proof that Otto approves the message or even saw it on his screen.

Having a central authority called the Certificate Authority (CA) each user can take his public key to CA and identify him to the CA. CA signs the user public key using a digital signature, [103].

Public-key certificates serve as an alternative approach in key management different than the one including public-key authority. A certificate contains a public-key with an identifier of the key owner and this block is signed by the trusted third party, what can be a government or certificate authority.
User presents his own public key to the authority, obtains the certificate and publishes it. Anyone who needs the user public key can obtain user’s certificate and verify that is valid by the attached trusted signature from the third trusted party.
For Otto the participant, the authority will provide a certificate in the following form:

$$CA = \mathrm{E}(PRauth, [T||IDO||PUo]) \tag{3.15}$$

PRauth is the authority private key and *T* is a timestamp. Otto can give this certificate to any other participant in communication who can read and verify Otto’s certificate in the following manner:

$$\mathrm{D}(Puauth, CA) = \mathrm{D}(Puauth, \mathrm{E}(Pr\mathrm{auth}, [T||IDO||Puo])) = (T||IDA||Puo) \tag{3.16}$$

PUauth is the authority public key, T is the timestamp. IDO is Otto’s identifier and PUo Otto’s public key. For formatting public key certificates the X.509 standard is used along with X.509 certificates.

When Otto wants to communicate with Stefani he needs to generate his pair of keys {PUo, PRo} and transmit a message to Stefani with his public key PUo and his identifier IDO. Stefani will generate secret key Ks and transmit this key to Otto, encrypted with Otto’s public key PUo. Otto computes D(PRo, E(PUo, Ks)) and recovers the secret. Because only Otto can decrypt the message, only he and Stefani will know the Ks. Otto then discards his public and private key, PUo and PRo and Stefani discards PUo and they can secure communicate using encryption, [79].

This protocol is simple but secure against eavesdropping same time is insecure against man-in-the-middle attack, when an intruder can intercept the message and substitute it and compromise the communication between them.

The secure communication between parties is exposed to different attacks: traffic analysis, content modification and sequence modification, masquerade, timing modification, source repudiation and destination repudiation. To verify that the received message is indeed from the sender message authentication is used.

NIST chose to standardize two called CCM and GCM (block ciphers modes, which provide both encryption and authentication.
A hash function maps an input m to an output of fix size h(m), 128-1024 bits long.

One important requirement for a hash function is that it must be one-way function: given m, compute h(m), but given a value x it must be impossible to find an m that:

$$h(m)=x \tag{3.17}$$

A hash function must be collision resistant such that for two different inputs m1 and m2 the output h(m1) and h(m2) must be different.
The ideal hash function behaves like a random mapping from all possible input values to set of all possible output values, [103].
If h is an iterative hash function considering b denotes the block length of the compression function, then I define the hash function hd by:

$$hd(m) := h(h(O^{b} || m)) \tag{3.18}$$

and has a claimed security level of:

$$\min(k, n/2) \tag{3.19}$$

Where k is the security level of hand n is the size of the hash result, [103].

3.4 DNA Steganography

Steganography is a technique that hides and conceals the message. The "Junior Nobel Price" was awarded in 2000 to a young American student of Romanian origin, Viviana Risca, for DNA Steganography. Researches try to consider the Human genetic code in cryptography.

The microdot is a means of concealing messages (steganography). The technique was developed by Professor Zapp. A microdot was formed by greatly reduced photograph of a type written page that was pasted over a full stop in an innocuous letter, [20]. A DNA encoded message is first camouflaged within the enormous complexity of human genomic DNA and then further concealed by confining this sample to a microdot. A prototypical 'secret message' DNA strand contains an encoded message flanked by polymerase chain reaction (PCR) primer sequences. Encryption is not of primary importance in steganography, a simple substitution cipher was used by Viviana Risca to encode characters in DNA triplets. Because the human genome contains about 32109 nucleotide provides a very complex background for concealing DNA secret-message. Confining such a sample to a microdot might then allow even the medium containing the message to be concealed from an adversary. The intended recipient, knowing the secret-message DNA, the PCR primer sequences and the encryption key, could amplify the DNA message, decode and read the message, [20].

Even if an adversary somehow detected such a microdot, it would be still almost impossible to read the message without knowing the specific primer sequences.

4. Java Language Security

Many language features within the Java programming language contributed to make Java a secure programming language. *Primarily* is that the Java programming language does not let to perform *arithmetic pointer*. This is a very important language feature and contributes to the Java language's safety. Arithmetic pointer leads to access inappropriate memory areas and this can lead to runtime crashes, [33].

Another feature of the Java language that contributes to Java safe and robust code is automatic *garbage collection*. Garbage collection is the runtime environment's ability to automatically release the memory no longer referenced. The dynamic memory requests are allocated from the heap and the garbage collector only needs to monitor references to places within it. When an area of memory no longer has a references to it, the garbage collector releases the memory, placing the free memory block into the free storage pool. The memory will be reallocated by another part of the program.

The Java language has a future of a *stack* for memory allocation, [16]. Within a program can be multiple stacks one for each executing thread but only the memory allocated on the stack is for local variables of a method. When the method ends, the variable space is relinquished. With garbage collector future a programmer doesn't need to determine if it is safe to release memory as the garbage collector will release the memory when is safe and when is no longer referenced.

Another security feature of the Java language is the uses of packages that provide namespace encapsulation, allowing downloaded code to be distinguished easily from local code. In Java Language when a class is referenced the system looks first in the local namespace and second in the namespace of the referencing class. This guarantees that a local class cannot accidentally reference a downloaded class. Variables access in Java is realized by name and not Like in C/C++ by a fix offset, [44].

An important future in Java is that Java library contains a definition for a File object containing a public *method* for reading callable by anyone, plus *a low-level private method* for reading only callable by the class methods. The public read call first performs security checks and then calls, the private read.
The Java language ensures that non-trusted code is able safely manipulate a File object, providing only access to public methods. The access control facilities allow programmers to write libraries guaranteed by the language to be safe, by correctly specifying the library's access controls, [21].

Another security feature of the Java language is the ability to declare classes or methods as *final* preventing a malicious programmer from sub-classing a critical library class or overriding some methods of a class providing a guarantee that certain parts of an object's behavior have not been modified. Java is a type-safe language with precise respected rules.

Java Security Manager: is one the most important aspect of Java technology security. *The security manager* is the interface between the core API and the operating system. It has the responsibility for allowing or preventing access to system resources. A running JVM contain at most one Security Manager, which is a class in the *java.Lang* package.

The Security Manager class contains methods intended to be called to check specific types of actions. Once a manager is installed, it cannot be replaced and once a program has set the security manager, a *Security Exception* will be thrown if another attempt is made. The Security Manager allows the establishment of a security policy that we can trust, or restrict the operations of a Java program, preventing the loaded applets from reading or writing files from the client file system, allowing us to create network connections back to the originating host of the applet, [70].

Some other restricted operations include that classes loaded over the network cannot load libraries or native methods so that native methods cannot be part of the classes loaded over the network.

The Security Manager provides a powerful mechanism, which allows access to resources. The Security Manager methods that check access are passed arguments, necessary to implement conditional access policies and have the ability to check the execution stack and determine if the code has been called by local or downloaded code. In a Java program a lot of resources can be used the consideration of adequate methods is necessary in order to control each resource, [72].

As main resources we have:

1. File system, where the access to the file system is well protected with specific Security Manager, checks for read and writes for a given file. The access control can be easily implemented.

2. Network, where the access to the file system is well protected with specific Security Manager, checks on the methods necessary for both accepting and creating of sockets, as well as protection of calls to other network related methods.

3. Random memory, the protection of memory is done by language specification itself. There is protection against access to already allocated memory, but no protection against an applet allocating all of the current memory available.

4. O*utput devices*, as protection on may consider that any applet window can be forced to have a special marking noting that is unsafe. Additionally an applet cannot directly access devices, instead it must use the mechanisms provided by the Java libraries.

5. I*nput devices*, as protection on may consider that an applet can only access the keystrokes or mouse clicks of the user when the applet’s window has been selected. Currently other input devices are not supported; the access to any other device (camera, microphone, etc.) is realized with methods from dedicated Java libraries.

6. P*rocess control*, here the threads and thread groups are checked by the Security Manager and it is possible to explicit establish priorities.

7. Environment, as protection the access to environment variables is protected by Security Manager, checks and the language access control mechanism for the classes.

8. System calls, the Security Manager checks any attempted system calls, including attempts to exit, [60].

The access controller allows (or prevents) most access from the core API to the operating system, based upon policies set by the end user or system administrator..

The security package allows us to add security features to our own application as well as providing the basis by which Java classes may be signed.

4.1 Java Application Security

The security provider interface the means by which different security implementations may be plugged into the security package, Message digests, Keys and certificates, Digital signatures, Encryption through JCE and JSSE and the Authentication through JAAS, [92]. *The key database* is a set of keys used by the security infrastructure to create or verify digital signatures.

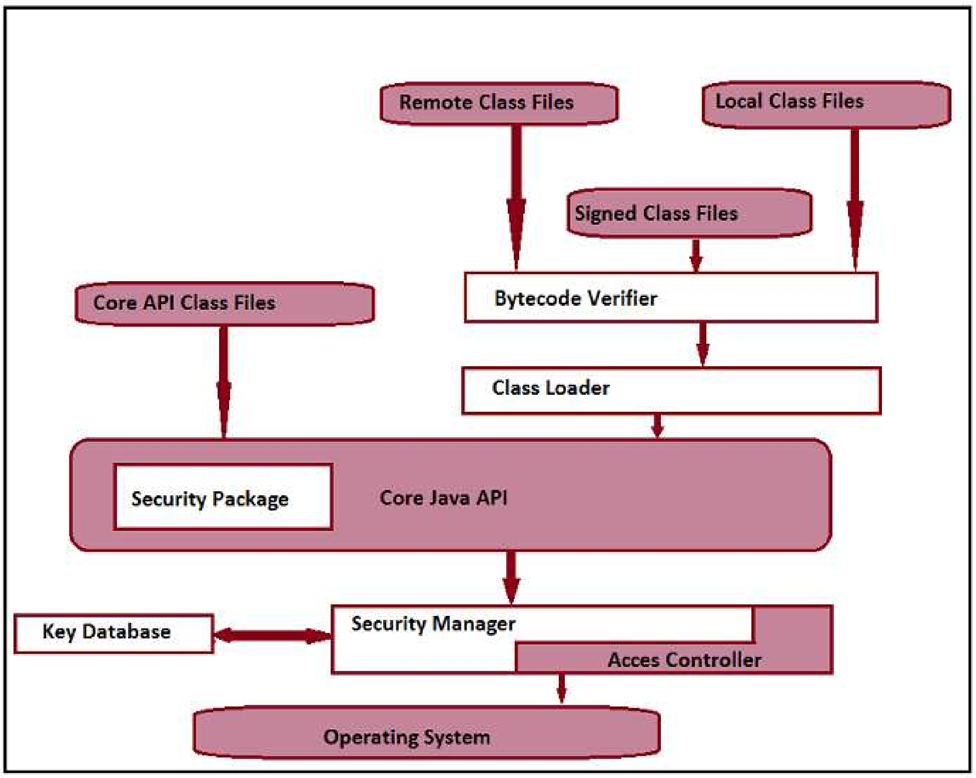


Fig. 5 Java Application Security, (Hodorogea, 2011)

The sandbox is composed of five elements. Permission is a specific action that code is allowed to perform and is composed of three elements: the *type* of the permission, permissions *name*, and permissions *actions*. The type of the permission is required; it is the name of a particular Java class that implements the permission. Even no programming is involved in administering the default sandbox; administrators must know the Java class name of different permissions in order to allow code to perform those operations, [100].

The actions of permission vary based on the type of the permission, some permissions have no action. The action specifies what can be done to the target; file permission may specify if a file can be read, written, deleted.

Code sources are the location from which a class is loaded together with information about who signed the class. The location is specified as a URL and follows Java practice: code can be loaded from the file system through a file based URL or from the network via a network based URL.

A protection domain is the basic concept of the default sandbox and is an association of permissions with a particular code source.

Policy files are the administrative element that controls the sandbox, [97].

A policy file contains one or more entries which define a protection domain. Policy files are simple files that can be created and modified with a text editor, and the JRE comes with a policy tool that allows them to be administered as well. Programs vary depending of the defined policy files. There are two policy files in use: a global policy file that all instances of the virtual machine use and a user specific policy file.

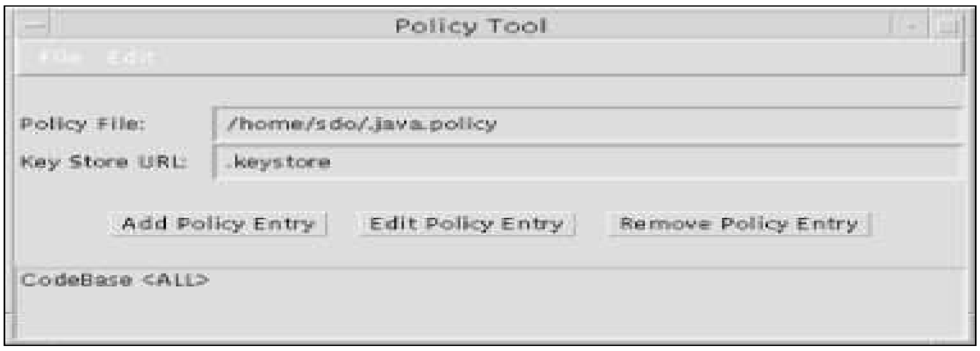
In Java, signed code depends on public key certificates which are held in a location called the ***key store.***

When a Java program attempts to perform an operation, the permissions for all active classes are consulted. Classes that make up the core Java API are always given permission to perform an action. Other classes, including those on the *class path* we must explicitly give a permission to perform sensitive operations. These permissions are listed in different policy files with the code source to which they apply, [70].

The end users system administrators must define the parameters of the sandbox by administering these policy files. Before running the signed code we must obtain the public key certificate of the signing entity and install that certificate into key store. The key store by default is held in a file called *.key store* in the user's home directory. The signer field must match the alias listed in the key store, [50].

Java virtual machines can use any number of policy files, but there are two numbers used by default. There is a global policy file named *$JREHOME/lib/security/java. policy* that is used by all instances of a virtual machine on a host. There is a specific policy file called *.java. policy* that can exist in user's home directory.

Policy files are text files and we can administer them with policy tool, or we can edit them by hand. Policy files are also used with JAAS, in which case their syntax changes a little bit, [35]. The policy tool allows us to manage entries in a *java. policy* file.



Policy File: /home/sdo/.java.policy
Key Store URL: .keystore

CodeBase <ALL>

Fig. 6 Policy Tool

The policy tool is a graphical tool. It takes one command line argument, the file filename, which allows us to specify the name of the initial policy file to edit. This defaults to *$HOME/.java. policy*, if the file doesn't exist no file is loaded by default. Editing Policy Entry button, we get a window with a list of all permissions associated with the given codebase, with the provided opportunity to add or remove individual permissions. Advanced applications are allowed to grant additional permissions to code that they load, and standard Java class loaders grant some additional permission to every class that they load, [57].

Some parameters of the default sandbox are controlled by entries in the *java. security* file. This file (*$JREHOME/lib/security/java.security*) can be edited by system administrators. In terms of the default sandbox, here are the important entries:

```
policy.expandProperties=true
policy.allowSystemProperty=true
policy.url.1=file:${java.home}/lib/security/java.policy
policy.url.2=file:${user.home}/.java.policy
```

The default sandbox is set up to be administered through a series of policy files, which contain sets of explicit permissions associated with code and this association depends on where the code was loaded from and who signed the code.

4.2 Java and Data Security

In a Java program every entity: object reference, primitive data element has an access level associated with it. *Private level:* the entity can only be accessed by code that is contained within the class that defines the entity. The *protected level:* the entity can be accessed by code that is contained within the class, by classes in the same package as the defining class. The *public level*: the entity can be accessed by code in any class, [50].
Default level: the entity can be accessed by code that is contained within the class that defines the entity.

Byte code verification helps to prevent malicious attacks from violating rules of the Java language. The security mechanisms at this level establish a set of rules creating an environment where an object's view of memory is known and defined. This way a developer can ensure that items in memory cannot be accidentally and intentionally read, corrupted and misused.

Security manager determines if a particular operation should be permitted or denied. The role of the security manager is in defining the security policy under which a particular program will operate.
Security manager and is responsible for determining most of the parameters of the Java sandbox. Security manager also determines if particular operations should be permitted or rejected.

If a Java program wants to open a file, the security manager decides if that operation should be permitted. If a Java program wants to connect to a particular machine on the network, it must first ask permission of the security manager. If a Java program wants to alter the state of some threads, the security manager intervenes if an operation is considered dangerous, [57].
To be used the security manager must be explicitly installed, by running a Java application and specifying the *Djava.security.manager.* The security manager is installed *programmatically* by the applet viewer and the Java Plug-in.

By default Java applications have no security manager and by default Java Applets have a very strict security manager. The security manager is a partnership

between the Java API and the implementer of a specific Java application or of a specific Java enabled browser, [64]. The class in the Java API called *java.lang.SecurityManager,* provides the interface that the Java API uses to check whether particular operations are permitted. It is possible to extend the SecurityManager class by providing a different implementation of the sandbox. The Java Plug-in and applet viewer use a modified implementation installed before they load an applet.
The methods the security manager uses to track file access:
1. public void checkRead(File Descriptor fd)
2. public void checkRead(String file)
3. public void checkRead(String file, Object context)-Check whether the program is allowed to read the given file. The first of these methods succeeds if the current protection domain has been granted the runtime permission with the name, *readFileDescripto*r.
-public void check Write(File Descriptor fd)
-public void check Write(String file: -Check if the program is allowed to write the given file. The first of these methods succeeds if the current protection domain has been granted the runtime permission with the name.
-public void check Delete(String file): Check whether the program is allowed to delete the given file. This succeeds if the current protection domain has file permission with a name that matches the given file and an action of delete, [70].

The security manager uses the following methods to check network access:
-public void check Connect(String host, int port)
-public void checkConnect(String host, int port, Object context): Check if the program can open a client socket to the given port on the given host. This succeeds if the current protection domain has a socket permission with a name that matches the host and port and an action of connect.
-public void checkListen(int port): Check if the program can create a server socket that is listening on the given port. The protection domain must have a socket permission where the host is local host, the port range includes the given port, and the action is listen.
-public void check Accept(String host, int port): Check if the program can accept (on an existing server socket) a client connection that originated from the given host and port. The protection domain must have a socket permission where the host and port match the parameters to this method and an action of accept.
-public void check Multicast(InetAddress addr)
-public void check Multicast(InetAddress addr, byte ttl): Check if the program can create a multicast socket at the given multicast address
-public void checkSetFactory( ). Check if the program can change the default socket implementation. When
The checkSetFactory( )method of the security manager class is responsible for arbitrating the use of low level aspects of Java's network classes. There are a number of methods in the *SecurityManager* class that protect the integrity of the Java virtual machine and the security manager, [70].
- *public void checkCreateClassLoader( ), a*s a result, the class loader takes on an important role since the security manager must ask the class loader where a particular class came from.
-public void check Exec(String cmd), this method is used to prevent execution of arbitrary system commands by untrusted classes.
-public void checkExit(int status), this method prevents an untrusted class from shutting down the virtual machine.

-public void checkPermission(Permission p)
*-public void checkPermission(Permission p, Object context), c*heck to see if the current thread has the given permission.

Java depends on threads for its execution and the security manager protects threads with the following methods:
*-public void checkAccess(Thread g), c*heck if the program is allowed to change the state of the given thread.
-public void checkAccess(Thread Group g), check if the program is allowed to change the state of the given thread group (and the threads that it holds). This call succeeds if the current protection domain has a runtime permission with the name *modifyThreadGroup*.
*-public ThreadGroup getThreadGroup( ),*supply a default thread group for newly created threads to belong to.

There are a number of methods in the security manager that protect Java's security, [101].
-public void checkSecurityAccess(String action), this package implements a higher order notion of security, including digital signatures, message digests, public and private keys. The security package depends on this method in the security manager and evaluates which classes can perform security related operations.
-public void checkPackageAccess(String pkg)
-public void checkPackageDefinition(String pkg)
These methods are used together with a class loader, when it needs to load a class with a particular package name, it asks the security manager if it is allowed to do so by calling the following method: *checkPackageAccess( )*.

This permits the security manager to make sure that the untrusted class is not trying to use application specific classes that it should not know about. Security manager arbitrates access to operating system features as files, network sockets, printers.

The purpose of the security manager is to grant access to a class based on the amount of trust the user has in the class and is used to enforce a specific policy and the issues involved when defining such a policy, [63].

Java Access Controller: the access controller is built based on four concepts:
1. Code sources: an encapsulation of the location from which certain Java classes were obtained.
2. Permissions: an encapsulation of a request to perform a particular operation.
3. Policies: an encapsulation of all the specific permissions that should be granted to specific code sources.
4. Protection domains: an encapsulation of a particular code source and the permissions granted to that code source.

A code source is an object that reflects the URL from which a class was loaded and the keys used to sign that class. Class loaders are responsible for creating and manipulating code source objects.
-public CodeSource(URL url, Certificate cers[]): creates a code source object for code that has been loaded from the specified URL, optional array of certificates is the array of public keys, that signed the code loaded from the URL.
-public boolean equals(Object o): two code source objects are considered equal if they were loaded from the same URL.
-public final URL getLocation( ): returns the URL that was passed to the constructor of this object.

-public final Certificate[] getCertificates( ): returns a copy of the array of certificates used to construct code source object and original certificates are not returned so that they cannot be modified maliciously.
-public boolean implies(CodeSource cs): determines if the code source implies the parameter according to the rules of the Permission class.

The Permission class is an abstract class that represents a particular operation and associated with a class a permission object represents an actual permission that has been granted to that class. An instance of the Permission class represents a specific permission. Permission is an abstract class that contains these public methods, [68]:
-public Permission(String name), construct a permission object that represents the desired permission.
-public abstract boolean equals(Object o), subclasses of the Permission class are required to implement their own test for equality.
-public final String getName( ), returns the name that was used to construct this permission.
-public abstract boolean implies(Permission p): responsible for determining whether a class that is granted one permission is granted another
-public PermissionCollection newPermissionCollection(): returns a permission collection suitable for holding instances of this type of permission.
-public void checkGuard(Object o): calls the security manager to see if the permission has been granted, generating a SecurityException if the permission has not been granted.

Implementing our permission we need to provide a class with concrete implementations of these abstract methods.
A policy class is constructed as follows:
1. public Policy( ), create a policy class.
2. public abstract Permissions getPermissions(CodeSource cs)
3. public abstract void refresh( ), refresh the policy object..

A protection domain is a grouping of a code source and permissions. A protection domain represents all the permissions that are granted to a particular code source.
-public ProtectionDomain(CodeSource cs, PermissionCollection p), construct a protection domain based on the given code source and set of permissions.

The notion of permissions and the access controller can be encapsulated into a single object, a guarded object, which is implemented by the GuardedObject class (java.security.GuardedObject).
There are two methods in the GuardedObject class:
-public GuardedObject(Object o, Guard g), create a guarded object and the given object is embedded within the guarded object and the access to the embedded object will not be granted untill the guard allows it.
-public Object getObject( ), returns the embedded object.

The checkGuard() method of the guard is first called; if the guard prohibits access to the embedded object, an AccessControlExceptionwill be thrown. The guard can be any class that implements the Guard interface java.security.Guard.
The interface has a single method, *public void checkGuard(Object o),* if access to the given object is not granted, this method throws an AccessControlException.

The access controller is a very powerful security feature of the Java platform as it protects most of the vital resources on a user's machine, allows users to customize the security policy of a particular application, by modifying entries in *java.policy* file. The access controller is able to control access to a set of system resources: files, sockets and other resources. We can create permission classes that the access controller can use to grant or deny access to any wanted resource.

Java Class Loaders are the mechanism by which files or other sources containing Java bytecodes are read into the Java virtual machine and then converted into class definitions.

There are three areas in which the class loader operates with the security model:
- *First area*: the class loader cooperates with the virtual machine to define namespaces, which protect the integrity of the security features of the Java language.
-*Second area*: class loader calls the security manager when appropriate and ensures that code has the right permissions to access or define classes.
-*Third area*: class loader sets up the mapping of permissions to class objects, the protection domain of each class, that the access controller knows which classes have which permissions.

When the virtual machine needs access to a particular class asks the appropriate class loader. Class loaders are organized into a tree hierarch. The class loader is also called the primordial class loader as it is used to load classes from the core Java API.

The system class loader has at least one child; the URL class loader, used to load classes from the *classpath*. Class loaders are responsible for asking their parent to load a class and the basic class that defines a class loader is the ClassLoader class (java.lang.ClassLoader).
-*public abstract class ClassLoader,* turn a series of Java bytecodes into a class definition and does not define how the bytecodes are obtained but provides all functionality needed to create the class definition.
-*public class SecureClassLoader extends ClassLoader,* turn a series of Java byte codes into a class definition and adds secure functionality to the ClassLoader class
-*public class URLClassLoader extends SecureClassLoader,* load classes securely by obtaining the bytecodes from a set of given URLs.

The class loading mechanism is integral to Java's security features and is considered in the relationship between the class loader, the access controller and the security manager. The class loader enforces the namespace separation between classes that are loaded from different sites, when these different sites are untrusted, helping this way to enforce the security mechanisms of the Java language. For sites that need a more flexible security policy we need a custom class loader which allows the security policy to be modified as classes are defined, [70].

Because of the access control futures to resources considering all presented previously the conclusion is that Java can response in an effective mode to integrity attacks, availability attacks, disclosure attacks and annoyance attacks. The analysis show that Java is effective at preventing the more dangerous attacks but the problem is that denial of service attacks is still difficult to prevent entirely.

4.3 Java Cryptography Architecture (JCA)

The JCA was designed to meet the principles of implementation independence and uses architecture based on provider that implement one or more cryptographic services. Such as digital signature algorithms, message digest algorithms, and key conversion services. A program may request a particular type of object: Signature object which implements a particular service: Digital Signature

Algorithm and get an implementation from one of the installed providers, [32].

Algorithm independence is realized by defining types of cryptographic "engines" (services), and by defining classes that provide the functionality of these cryptographic engines.
The classes are called *engine classes*. As an example can serve the *MessageDigest signature, KeyFactory and KeyPairGenerator classes.*

Sun's version of the Java runtime environment comes standard with a default provider, named SUN. The SUN provider package includes:
1. An implementation of the DSA, described in NIST FIPS 186.
2. An implementation of the MD5 (RFC 1321) and SHA-1 (NIST FIPS 180-1) message digest algorithms.
3. A DSA key pair generator for generating a pair of public and private keys suitable for the DSA algorithm.
4. A DSA algorithm parameter generator.
5. A DSA algorithm parameter manager.
6. A DSA key factory
7. An implementation of pseudo-random number generation algorithm "SHA1PRNG”
8. A certificate path builder for PKIX,

The following main ***engine classes are defined in Java 2 SDK***:
-*KeyPairGenerator:* generates a pair of public and private keys for a specified algorithm. -*Signature*: signs the data and verifies the digital signatures.
-*MessageDigest*: calculates the message digest of specific data.
-*CertificateFactory*: create public key certificates and Certificate Revocation Lists (CRLs).
-*AlgorithmParameters*: manage the parameters for a particular algorithm.
-*SecureRandom*: generates random or pseudo-random numbers.
-*KeyFactory*: converts opaque cryptographic keys of type Key into key specifications and vice versa.
-*KeyStore*: creates and manage a key store which is a database of keys. Private keys in a key store have a certificate chain associated with them, which authenticates the corresponding public key. A key store also contains certificates from trusted entities.
AlgorithmParameterGenerator: is used to generate a set of parameters suitable for a specified algorithm

The Java Cryptography Extension (JCE) provides a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. Support for encryption includes symmetric, asymmetric, block, and stream ciphers, [60].

JCE is based on the design principles as in the JCA the implementation independence, algorithm independence and uses the same “provider" architecture.
Providers signed by a trusted entity can be plugged into the JCE framework, and new algorithms can be added seamlessly, [17].
The Java 2 SDK, version 1.4 comes standard with a JCE provider named "SunJCE", pre-installed and registered and supplies the following cryptographic services:
1. An implementation of the DES, Triple DES, Blowfish encryption algorithms in the Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Propagating Cipher Block Chaining (PCBC) modes.
2. Key generators for generating keys for the DES, Triple DES, Blowfish, HMAC-MD5, and HMAC-SHA1 algorithms.

3. An implementation of the MD5 with DES-CBC password-based encryption (PBE) algorithm defined in PKCS #5.
4. "Secret-key factories" providing bi-directional conversions between opaque DES, Triple DES and PBE key objects and transparent representations of their underlying key material.
5. An implementation of the Diffie-Hellman key agreement algorithm between two or more parties.
6. A Diffie-Hellman key pair generator for generating a pair of public and private values suitable for the Diffie-Hellman algorithm.
7. A Diffie-Hellman algorithm parameter generator.
8. A Diffie-Hellman "key factory" providing bi-directional conversions between opaque Diffie-Hellman key objects and transparent representations of their underlying key material, [53].
9. Algorithm parameter managers for Diffie-Hellman, DES, Triple DES, Blowfish, and PBE (Password Based Encryption) parameters.
10. An implementation of the HMAC-MD5 and HMAC-SHA1 keyed-hashing algorithms defined in RFC 2104.
11. An implementation of the padding scheme described in PKCS #5.
12. A keystore implementation for the proprietary keystore type named "JCEKS".
A cryptographic service offered by JCE has the following properties:

-is associated to an algorithm
-will implement the two basic functions, digest and sign
-will generate keys and parameters for the algorithms
-will generate certificates and databases for the keys (*keystore*).

For the cryptology process, Java offers the *keytool* and *jarsigner* to generate keys and certificates with the possibility to sign jar files. JCE also offers the possibility to make conversions between different key types.

The Java Secure Socket Extension (JSSE): Someone who is not the intended recipient can easily access data that travels across a network. It is important to ensure the data has not been modified, either intentionally or unintentionally, during transport. The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols were designed to protect the privacy and integrity of data while it is transferred across a network.

The Java Secure Socket Extension (JSSE) *enables secure Internet communications* and provides a framework and an implementation for a Java version of the SSL and TLS protocols. JSSE includes functionality for authentication, message integrity, data encryption, server, and client authentication. Using JSSE, developers can provide for the secure passage of data between a client and a server running any application protocol, such as Hypertext Transfer Protocol (HTTP), Telnet, or FTP, over TCP/IP.
The JSSE standard API covers:
1. Secure (SSL) sockets and server sockets.
2. Factories for creating sockets, SSL sockets, server sockets and SSL server sockets.
3 One class representing a secure socket context and acts as a factory for secure socket factories.
4. Keys and factories for creating them, trust manager interfaces
5. A class for secure HTTP URL connections. The specific classes and interfaces used are:
-*SocketFactory* and *SSLSocketFactory*
-*SSLSocket* and *SSLServerSocket*

-*SSLSesion* interface
-*HttpsURLConection*
-*SSLContext*
-*TrustManager*.

Certificates are used to authenticate public keys. When public keys are transmitted electronically, they are often embedded within certificates. The core Java API comes with classes to handle public/private keys and their certificates. The classes necessary to handle secret keys come only with JCE. Keys and certificates are associated with persons, organization. The way in which keys are stored, transmitted, shared is a very important aspect in the security package.

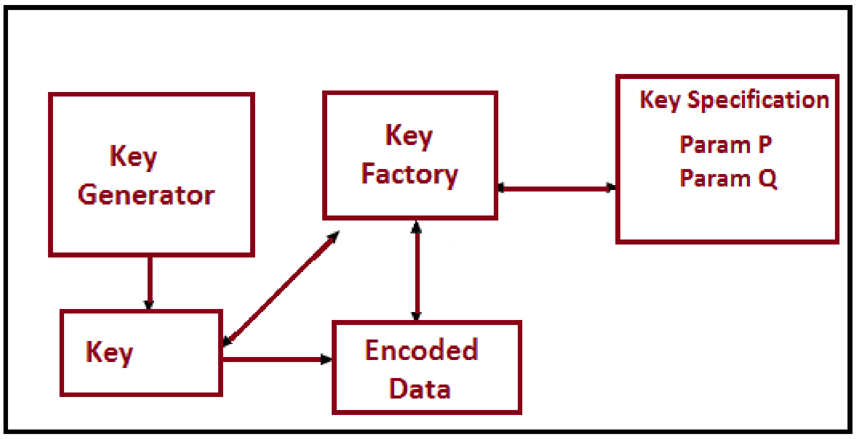


Fig. 7 Interaction of key classes in Java, [59]

A generator class creates keys from scratch and the generator can produce one or more keys, (Fig. 7).
Symmetric keys are generated by *the KeyGenerator class,* while asymmetric key pairs are generated by *the KeyPairGenerator* class.

The *KeyFactory* class translates between key objects and their external representations, which may be a byte array or a key specification. In addition to the engine classes there are several classes and interfaces that represent the key objects and the key specifications.

The important operations that developers need are:
1. The ability to create new keys from scratch using the key pair generator
2. The ability to export a key as a parameter specification or as a set of bytes plus the corresponding ability to import that data in order to create a key.

The *concept of a key* is modeled by the ***key interface*** *(java.security.Key):*
-public interface Key extends Serializable: models the concept of a single key, as keys must be transferred to and from various entities, all keys must be serializable.
-public String getAlgorithm( ): returns a string describing the algorithm used to generate this key

Keys tell us the format they use for encoding their output with this method:
-public String getFormat( ): returns a string describing the format of the encoding the key supports.
The encoded data of the key is produced by this method:

-public byte[] getEncoded( ): returns the bytes that make up the particular key in the encoding format the key support.

Asymmetric keys come in pairs, hence the core Java API contains these two additional interfaces:

-public interface PublicKey extends Key -public interface PrivateKey extends Key: contains no additional methods, used simply for type convenience.

The security providers that come with Sun's implementation of Java provide two types of asymmetric keys: DSA and RSA.

JCE provides an additional type of asymmetric key: Diffie−Hellman and each of these have their own interface that allows us to determine the information about the key.

This is the existing interfaces for particular RSA keys within the *java.security.interfaces package*:

-public interface RSAPrivateKeyCrt extends RSAPrivateKey

-public interface RSAPrivateKey extends PrivateKey

-public interface RSAPublicKey extends PublicKey

These interfaces define keys for use in RSA algorithms, allows retrieving the parameters used to create the RSA key. RSA public key interface has methods to return its modulus and public exponent, while the private key has methods to return its modulus and private exponent.

Diffie−Hellman keys are used in secret key agreement Diffie−Hellman and the key interfaces are defined in the *javax.crypto.interfaces* package, [54]:

-public interface DHKey

-public interface DHPublicKey extends DHKey, PublicKey

-public interface DHPrivateKey extends DHKey, PrivateKey

This set of interfaces defines keys suitable for use in Diffie−Hellman algorithms.

Diffie−Hellman parameters are encoded into the *javax.crypto.spec.DHParameterSpec class.*

Symmetric keys are used only within JCE and they are defined by the *SecretKey interface (javax.crypto.SecretKey*):

-public interface SecretKey extends Key: identify a class as being a symmetric key.

Secret keys have no specific identifying information, interface is empty and is used only for type identification.

Java's security API provides two standard engines to generate keys: an engine to generate a pair of asymmetric keys and one to generate a secret key, [70].

Generation of public and private keys is provided by the *KeyPairGenerator* class (java.security.KeyPairGenerator):

*-public abstract class KeyPairGenerator extends KeyPairGeneratorSpi, g*enerates information regarding public/private key pairs.

KeyPairGenerator class has no implementation in the core API, retrieves instances of the *KeyPairGenerator class* via methods:

-public static KeyPairGenerator getInstance(String algorithm)

-public static KeyPairGenerator getInstance(String algorithm, String provider):

The first format of the method searches all available providers and the second method searches only the named provider throwing a *NoSuchProviderException* if the provider has not been loaded. The methods search the providers registered with the *security provider interface for a key pair generator* that supports the named algorithm. The JSSE security provider has its own implementation of RSA keys.

Having the key pair generator we can invoke the methods:

-public String getAlgorithm( ): the method returns the name of the algorithm that is implemented by this key pair generator

-public void initialize(int strength)
-public abstract void initialize(int strength, SecureRandom random): initializes the key pair generator to generate given strength keys. Strength typically means the number of bits that are used as input to the engine to calculate the key pair and if an invalid number is passed for strength, an *InvalidParameterException* will be thrown.
Key pairs require a random number generator and if desired we can specify a particular random number generator else a default random number generator is used; an instance of the *SecureRandom class*
-public void initialize(AlgorithmParameterSpec params)
-public void initialize(AlgorithmParameterSpec params, SecureRandom random: initializes the key pair generator and uses the given parameter specification.
-public abstract KeyPair generateKeyPair( )
-public final KeyPair genKeyPair( ): the method generates a key pair using the initialization parameters previously specified. A *KeyPairGenerator object* can repeatedly generate key pairs by calling one of these methods and every new call generates a new key pair. The *genKeyPair( ) method* simply calls the *generateKeyPair( ) method.*

Using these methods we can generate a pair of keys as follows:
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DSA");
kpg.initialize(512);
KeyPair kp = kpg.generateKeyPair( );
If I want to implement my own key pair generator using a new algorithm and a new implementation of an algorithm, I must create a subclass of the *KeyPairGenerator class*.
The *KeyPairGenerator class* already extends the *KeyPairGeneratorSpi* class; in this case I don't have to extend the SPI class directly.
There are two abstract public methods of the key pair generator SPI:
the *initialize( ) method* and the *generateKeyPair( ) method*.
The *KeyGenerator class (javax.crypto.KeyGenerator)* is used to generate secret keys. It is similar to the *KeyPairGenerator class* but it generates instances of secret keys instead of pairs of public, private keys:
- *public class KeyGenerator:* generates instances of secret keys for use by a symmetric encryption algorithm. The KeyGenerator class is an engine within JCE and has the cryptographic engine plus a complementary SPI and a set of public methods that are used to operate upon it. Its implementation must be registered with the security provider, [57]

KeyGenerator class doesn't have any public constructors. An *instance of a KeyGenerator* is obtained by calling one of the following methods:
- *public static final KeyGenerator getInstance(String algorithm, String provider):* Return an object capable of generating secret keys that correspond to the given algorithm,
-public static final KeyGenerator getInstance(String algorithm):
NoSuchAlgorithmException is thrown if the named provider cannot be found.
When an object was obtained the generator must be initialized by calling one of these methods:
-public final void init(AlgorithmParameterSpec aps, SecureRandom sr),
-public final void init(int strength), public final void init(SecureRandom sr),
-public final void init(AlgorithmParameterSpec aps),
-public final void init(int strength, SecureRandom sr)

A key generator does not have to be initialized explicitly but it is initialized internally with a default instance of the *SecureRandom class*.

A secret key can be generated by calling the method:

-public final SecretKey generateKey( ): the method generates a secret key.

A generator can produce multiple keys by repeatedly calling this method.

-public final String getAlgorithm( ): the method returns the string representing the name of the algorithm this generator supports.

Another way to generate keys is through the use of **key factories** that are most often used to import and export keys. The factory translates between an external format of the key,easily transportable and the internal implementation of the key.

There are two external representations by which a key may be transmitted: by its encoded format or by the parameters used to generate the key
Any of these representations can be encapsulated in a key specification used to interact with the *KeyFactory class (java.security.KeyFactory)* and the *SecretKeyFactory class (javax.crypto.SecretKeyFactory).*

The key factory depends on a service provider interface class: the *KeyFactorySpi class (java.security.KeyFactorySpi):*

-public abstract class KeyFactorySpi: provides the set of methods necessary to implement a key factory that is capable of importing and exporting keys in a particular format.

The KeyFactorySpi class contains the following methods and since each of these methods is abstract my class must provide an implementation of all of them:

-protected abstract PublicKey engineGeneratePublic(KeySpec ks)

-protected abstract PrivateKey engineGeneratePrivate(KeySpec ks): the method generate s the public or private key and depending on the key specification means either decoding the data of the key or regenerating the key based on specific parameters to the key algorithm. If the key cannot be generated, an InvalidKeyException is thrown.

-protected abstract KeySpec engineGetKeySpec(Key key, Class keySpec): export the key depending on the key class specification, this means either encoding the data or saving the parameters that were used to generate the key. If the specification cannot be created, an *InvalidKeySpecException* is thrown.

-protected Key engineTranslateKey(Key key): performs the actual translation of the key, translating the key to its specification and back and if the key cannot be translated, an *InvalidKeyException* is thrown, [70].

The second engine is the *SecretKeyFactory class, (javax.crypto.SecretKeyFactory),* this class can convert from algorithmic or encoded key specifications to actual key objects and can translate key objects from one implementation to another and the *SecretKeyFactory class* can operate only on secret keys.

-public class SecretKeyFactory: provides an engine that can translate between secret key specifications and secret key objects allowing for secret keys to be imported and exported in a neutral format.

In addition the *SecretKeyFactory class* contains the following method: *public final SecretKey generateSecret(KeySpec ks)* used to generate the secret key according to the given specification and if the specification is invalid, an InvalidKeySpecException is thrown.

The SecretKeyFactory class is an engine class and if desired we can implement a secret key factory by subclassing the *SecretKeyFactorySpi class*; (*javax.crypto.SecretKeyFactorySpi),* because it is a *JCE engine*, the constructor of the secret key factory engine must invoke the *verifyForJCE( ) method* my sample provider.

The problem of **key management** in Java is a hard problem to solve as there is no universally accepted approach to key management and all key management techniques remain very much works in progress.

Keys are very important because they allow us to perform a number of cryptographic operations: digital signatures and to encrypted data streams, [50].
The *key management system* is built around the notion of a **keystore**. Key stores are created and manipulated though an administrative tool, ***keytool*** and there is a Java API that allows us to use key stores programmatically.

The keystore is the file that holds the set of keys and certificates and the file is called *keystore. Keystore is* held in the user's home directory; *$HOME* on Unix Systems, *C:\WINDOWS* on Microsoft Windows Systems

A key management tool allows us to specify the location of the file. The key management API allows us to use any arbitrary input stream.

An alias is a keystore specific name for an entity that has a key or certificate in the keystore.
DN (distinguished name) for an entity in the keystore is a subset of its full X.500 name.
A keystore may hold two types of entries:

1st: **key entry** which holds either an asymmetric key pair or a single secret key and if the entry holds a key pair it can store a chain of certificates and the first certificate always contains the public key of the entity.

2nd: **certificate entry** contains only a public key certificate and there is no private key associated with this entry and the certificate entries hold a single certificate which is self-signed.

Sun's implementation of Java comes with a set of trusted certificates from known certificate authorities and the certificates are held in *$JREHOME/lib/security/cacerts*, which is a keystore, holding certificate entries, each of which is the root certificate of a CA.

5. DNA and Biology of the Cell

Research considers the use of the Human genome in cryptography. The basic idea using human genome in encryption techniques is the exploitation of DNA cryptographic strength, strong capabilities and its parallelism in order to enforce other conventional cryptographic algorithms.
DNA has the potential to be a dense information system storage as massive amounts of the data can be stored in a high field compound on the molecular level.

A single gram of DNA is capable to store the same amount of information that could be stored on trillion of CDs, [39].
DNA computing offers massive parallelism and with enough DNA very big problems can be solved using parallelism search.

There has been growing interest in using DNA to store information, one of the principal attractions being the very high storage densities achievable with it .

The durability of DNA would make it particularly useful for preserving archival material over extensive periods of time.

In 2000, the Junior Nobel Prize was awarded to Romanian-American student, Viviana Risca with the coordination of Professor Bancroft, for her work in DNA steganography, [20].

A DNA-encoded message is first camouflaged within the enormous complexity of human genomic DNA, and then further concealed by confining this sample to a microdot. A prototypical 'secret message' DNA strand contains an encoded message flanked by polymerase chain reaction (PCR) primer sequences.

Denatured human DNA provides a very complex background for concealing a secret-message. Risca, knowing both the secret-message DNA, PCR primer sequences and the encryption key could readily amplify the DNA and then proposed a mechanism to read and decode the message.

Potential limitations of the DNA Steganography method shows that certain DNA steganography systems can be *broken*, with some assumptions on information theoretic entropy of plaintext messages, DNA-based, molecular cryptography systems where the plaintext message data is encoded in DNA strands by use of short oligonucleotide sequences and based on one-time-pads that are in principle unbreakable but practical applications of this cryptographic systems based on one-time-pads are limited in conventional electronic media, by the size of the one-time-pad, [39].

The famous DNA one-time-pad encryption schemes involve the use of a *substitution method* using libraries of distinct pads, each of which defines a specific, randomly generated, pair-wise mapping and an *XOR scheme* utilizing molecular computation and indexed, random key string first developed by Ashish Gehani, Thomas H. LaBean and John H. Reif.

Cells store hereditary information in double-stranded molecules of DNA which is long unbranched paired polymer chains, formed of four types of monomers A, T, C, G. The monomers are linked together in a long linear sequence that encodes the genetic information, similar to how the sequence of 1s and 0s encodes the information in a computer file, [13].

Using chemical methods, scientists can read out the sequence of monomers in any DNA molecule for millions of nucleotides long and decipher the contained hereditary information for the given organism. A monomer in a single DNA strand composed from nucleotides. A nucleotide consists of two parts: a sugar (deoxyribose) and a phosphate group attached to it, and a *base*, which is: adenine (A), guanine (G), cytosine (C) or thymine (T). Each sugar is linked to the next via the phosphate group, creating a polymer chain composed of a repetitive sugar-phosphate backbone with a series of bases protruding from it, [3]. The DNA polymer is extended by adding monomers at one end.

For a single strand the adding can be in any order, because each one links to the next in the equal way, through the part of the molecule that is the same for all of them. In the living cell DNA is not synthesized as a free strand but on a template formed by a preexisting DNA strand. The bases protruding from the existing strand bind to bases of the strand being synthesized, according to a strict rule defined by the complementary structures of the bases: A binds to T and C binds to G, [3] This base-pairing holds monomers in place and controls the selection of which one of the monomers shall be added next to the strand. In this way, a double-stranded structure is created and consists of two complementary sequences of As, Cs, Ts, and Gs. The two strands twist around each other and form a double helix.

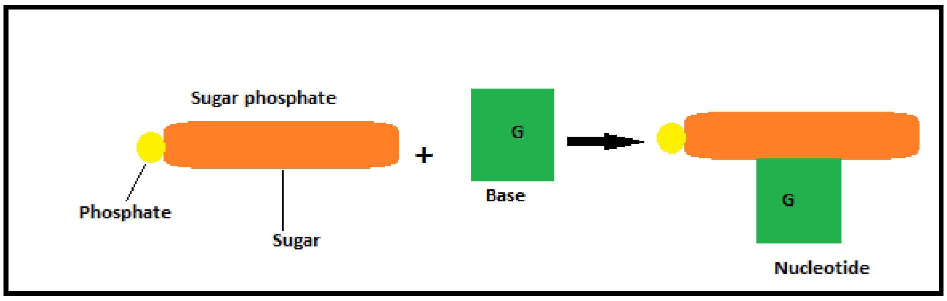


Fig. 8 DNA and its building blocks

DNA is made from subunits, called nucleotides and each nucleotide consists of a sugar-phosphate molecule with nitrogen containing side group attached to it, or base. The bases are of four types (adenine, guanine, cytosine, and thymine) and correspond to four nucleotides labeled A, G, C, and T.

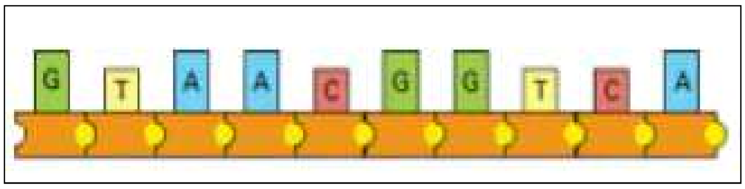


Fig. 9 DNA Strand, [3]

A single strand of DNA consists of nucleotides joined together by sugar-phosphate linkages. The sugar-phosphate units are asymmetric and give the backbone of the strand a definite directionality and polarity. The directionality leads the molecular processes by which the information in DNA is interpreted and copied in cells, from left to right.

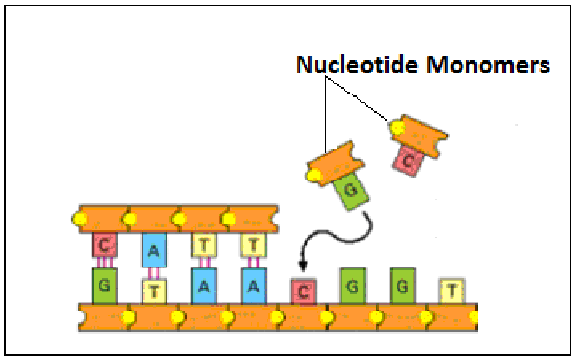


Fig. 10 Templated polymerization of the new strand, [3]

Through polymerization, the sequence of nucleotides in a DNA strand controls the sequence in which nucleotides are joined together in a new DNA strand; T in one strand pairs with A in the other, and G in one strand with C in the other, [3]. The new strand has a nucleotide sequence complementary to that of the old strand the backbone has the opposite directionality.

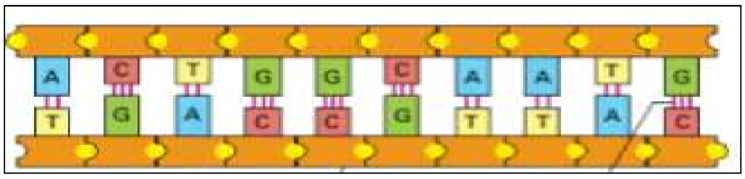


Fig. 11 Double stranded DNA, [3]

A DNA molecule consists of two complementary strands. The nucleotides within each strand are linked by covalent chemical bonds. The complementary nucleotides on opposite strands are linked together by hydrogen bonds.

The two strands twist around each forming a double helix in a robust structure which accommodates the sequences of nucleotides without altering its basic structure.

Genetic information is read out and used through two processes.

During the first process of *transcription* the segments of the DNA sequence guide the synthesis of RNA molecules, [13].

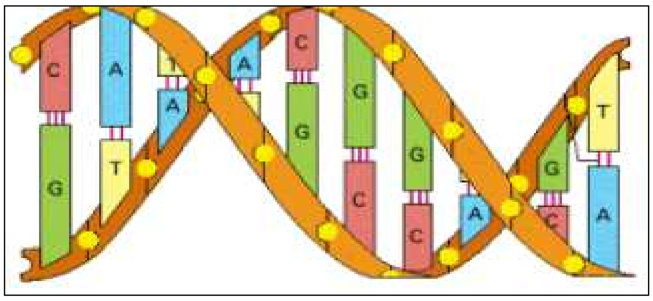


Fig. 12 DNA double helix, [3]

During *translation*, the RNA molecules guide the synthesis of protein molecules. Every cell contains a fixed set of DNA molecules storing the genetic information. A given segment of DNA guides the synthesis of identical RNA transcripts, which are copies of the information stored in the archive.

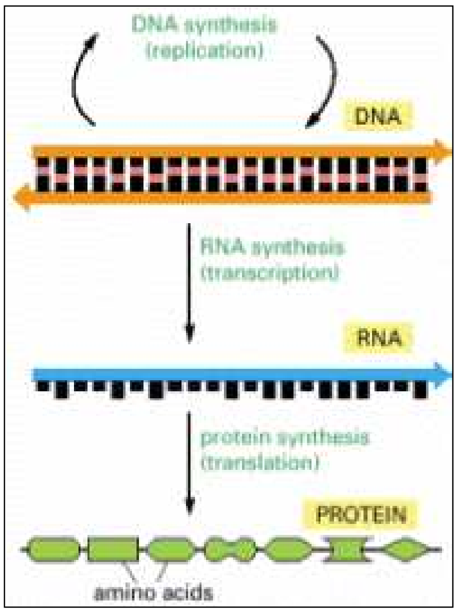


Fig. 13 From DNA to Protein, [3]

Many different sets of RNA molecules can be made by transcribing selected parts of a long DNA sequence, allowing each cell to use its information store differently.

5.1 Chromosomal DNA

Genetic information for every organism is written the code of DNA sequences. The DNA sequence can be obtained by biochemical techniques we can characterize, and compare any set of DNA sequences with reference to a given DNA sequences.

Cells have the ability to store, retrieve, and translate genetic information passing it to its daughter cells in the process of cell division and reproduction. The genetic information it is stored in genes of the given cell. Genetic information consists of instructions for making macromolecules called protein which perform most cellular functions starting with cellular structure. Proteins form enzymes which catalyze the chemical reactions, regulate gene expression, enabling cells to move.

The answer to how proteins are specified by instructions in the DNA was determined in 1953 by James Watson and Francis Crick that had recognized that genes are carried in chromosomes. Chromosomes are made of DNA and proteins. Chromosomes pack the DNA molecules which fit inside cells. Genes in there turn are important segments of DNA present in chromosomes. The cells maintain, replicate and improve sometime the genetic information carried in its DNA, [3].

5.2 DNA molecule structure

In the 1953 by a technique to determine the three dimensional structure of a molecule, called x-ray diffraction analysis was established that DNA is composed of two strands of the polymer chains composed of four types of nucleotide subunits wound into helix where h*ydrogen bonds* between the base of the nucleotides hold the DNA chains together.

Nucleotides are made from carbon sugar attached to phosphate groups and a nitrogen-base either *adenine (A), cytosine (C), guanine (G),* or *thymine (T)*., [13]. The nucleotides are linked together in a chain through the sugars and phosphates forming a backbone, analogous to a necklace with four types of beads.

The chemical polarity of the DNA strand is given by the way in which nucleotide subunits are linked together.

The polarity of the DNA strand is indicated by referring to one end as the *3 end* and the other as the *5 end.*

DNA **the double helix** is formed from two polynucleotide chains, held together by hydrogen bonds between bases on the strands, the bases are located inside double helix, and sugar-phosphate backbones are located outside the helix.
Always a purine which is two ring base is paired with a pyrimidine with one complete turn every ten base pairs.

The two strands of the helix are antiparallel and the polarity of one strand is oriented opposite to that of the other strand.

Each strand of a DNA molecule contains a sequence of nucleotides complementary to the nucleotide sequence on the other strand.
Genes contain instructions and information which are copied and transmitted to daughter cell each time a cell divides.

The information in DNA has a certain order of sequences composed from nucleotides situated along the strands. Bases are considered as letters in DNA alphabet that transmit biological messages in the chemical structure of the DNA helix.

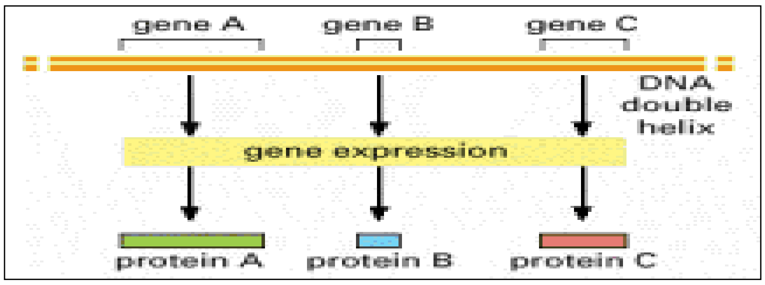


Fig. 14 Relationship between DNA and Proteins, [3]

Every organism is different from one another as the DNA molecules have different sequence of nucleotides sequences giving mean to different biological messages.

The complete set of information contained in DNA is called Genome. A human cell genome contains 2 meters of DNA and carries instructions for 30,000 proteins.

5.3 Chromatin the complex of DNA and protein

The DNA in the nucleus is divided between a set of different chromosomes and human genome is distributed over 24 chromosomes.
A chromosome consists of a single very long linear DNA molecule associated with proteins folding the DNA thread into a compact structure. The complex of DNA and protein is called *Chromatin.* Human cell contains 46 chromosomes arranged in 22 pairs the two copies of chromosomes are inherited from the mother and from the father and are called homologous chromosomes. Chromosomes carry genes or segments or DNA with instructions to make protein. The human chromosome 22 is the smallest human chromosome, completed in 1999. The DNA is made of so called coding sequences called exons and so called noncoding sequences introns in addition each gene is associated with regulatory DNA sequences, responsible for gene expression, [13].

In the process of cell division and replication occurring in a order of stages, (known as cell cycle) the chromosomes are replicated, extended and distributed after separation to its daughter nuclei. After replication to daughter cells they remain attached to one another and with proceeding cell cycle producing mitotic chromosomes.

A so called protein complex which forms the centromere attaches duplicated chromosomes to mitotic spindle pulling them apart this way.
The ends of chromosome DNA sequences form the telomeres and there function is to allow chromosomes to replicate and protect the cell from wrong recognition by the cell as a broken DNA molecule in need of repair.

Nucleosomes are the basic unit for the chromosome structure. Chromosome complex known as chromatin is divided in histones protein and nonhistone.

5.4 Transcription and translation of the DNA

Cells express the genetic information in their genes through the process of transcription and translation. During the transcription process RNA is made. The process starts by opening a portion of DNA helix exposing bases on each DNA, one strand of DNA acts as a template for the synthesis of the RNA molecule.

In the first step a cell reads its genetic instruction and copies a DNA portion into an RNA nucleoide sequence. RNA is made of four subunits called ribonucleotides and the four bases: adenine, guanine, cytosine and uracil.
RNA is single stranded chain. The process by which RNA is made in the cell is called transcription, (Fig.15). Transcription is performed by enzymes called RNA polymerases.

RNA polymerase moves along DNA and unwinds the DNA helix, extending this way a growing RNA chain by a nucleotide at a time.

The RNA molecules copied from genes are called messenger RNA(mRNA). During the process of the RNA splicing the intron sequences are removed one by one and exons are joined.

The process of catalyzez pre-mRNA splicing is very complex, consisting of over 50 proteins and many ATP molecules per splicing event. RNA splicing is performed by RNA molecules which recognize intron-exon borders. RNA molecules are known as small nuclear RNA (snRNAs) and there are five known as U1, U2, U4, U5, U6.

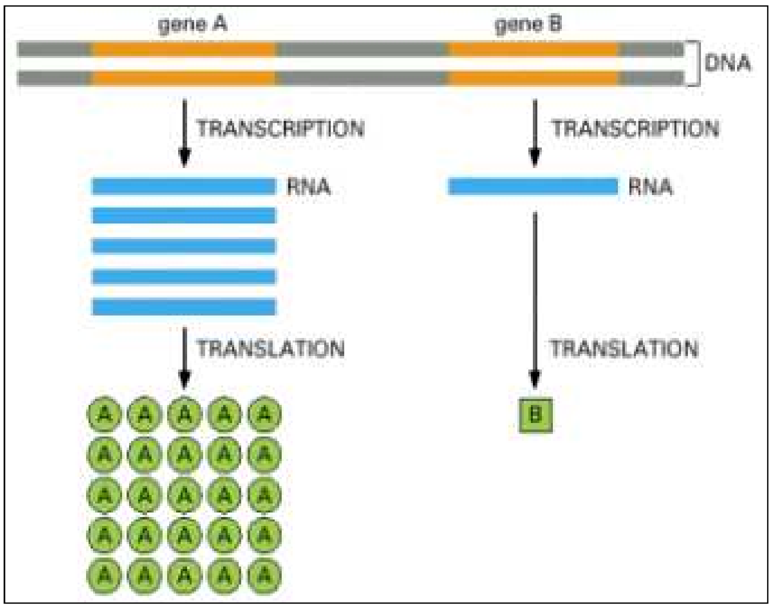


Fig. 15 Expression of Genes [3]

The mRNA is transported from nucleus to cytoplasm and translated into protein. The mRNA is translated into amino acid sequence of a protein by rules of genetic code, [13].

The mRNA is read in group of three, there are 64 possible combinations of three nucleotides but only 20 different amino acids are found in proteins.
The translation of mRNA into protein depends of adaptor molecules known as tRNA (transfer RNA). Specific enzymes couple amino acids to the appropriate tRNA molecule.

In Ribosome's is performed the protein synthesis. Ribosomes are made from different proteins and rRNAs molecules. Ribosomes are responsible for the correct reading frame and accuracy in the protein synthesis. Using the tRNAs the mRNA nucleotide sequence is translated into amino acid sequence. When a stop codon is reached ribosome releases a finished protein. Stop codons mark the end of translation. The end is signaled by the presence of one stop codon: UAA, UAG and UGA. Abnormally folded proteins can cause human diseases.

5.5 DNA Engineering and Bioinformatics

Progress in DNA engineering and Bioinformatics was driven by advances in technology. New methods for DNA engineering allowed scientists to study information's and details about genes, proteins, macromolecules and cells.

The scope of DNA Engineering is to understand what takes place inside the cell influenced by the environment and by interaction with its neighbors. By DNA engineering we want to know which genes are on, active proteins, there location and the pathway to which they belong.

The scope of DNA engineering and bioinformatics is to understand and predict how genes and proteins operate. By DNA engineering methods different cells are separated from tissues and grown apart, disrupting them in isolating there macromolecules and organelles in pure form by recombinant DNA technology.

To study the function of molecules a biochemical analysis are needed. Large cells are separated from small cells by centrifugation or antibody cell-separation technique with fluorescent dye labeling specific cells separating them in an electronic fluorescence-activated cell sorter machine. In the machine cells pass through a laser beam and there fluorescence is measured.

5.5.1 Engineering DNA, RNA and Proteins

The ability to manipulate DNA had a very big impact on cell biology giving scientists the possibility to study cells, their macromolecules in previously impossible ways by the following techniques: DNA cloning, Nucleic acid hybridization, Rapid sequencing of nucleotides in purified DNA, Cleavage of DNA and simultaneous monitoring of expression level of genes in the cell.

To obtain information about a cell biologists dissociate cells from tissues and separate them. The new obtained homogeneous population of cells can be analyzed directly or increasing the population by allowing cells to proliferate as a culture.

To isolate cells from a tissue the extracellular matrix that holds the cells together needs to be disrupted in a way and then the tissue sample are treated with proteolytic enzymes named trypsin and collagenase.

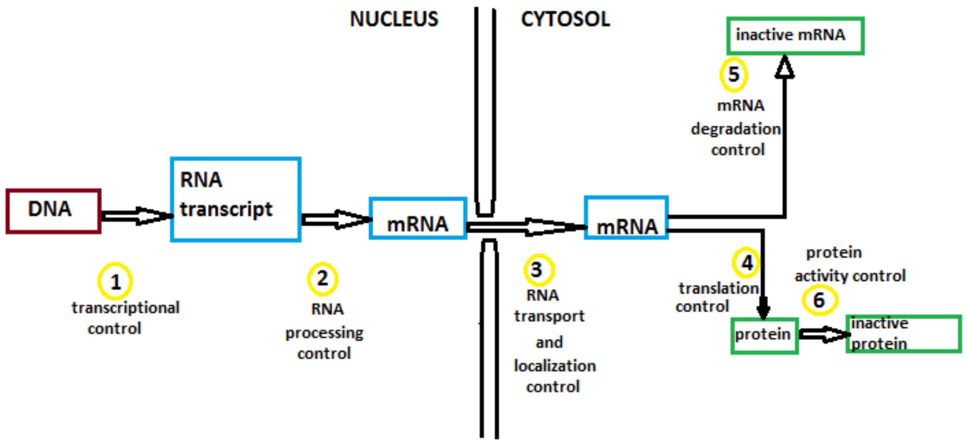


Fig. 16 Steps at which a gene can be controlled

The enzymes digest proteins in the extracellular matrix. It is also a need of using agents as ethylene, tetracetic acid that bind cell-cell adhesion. The most sophisticated cell-separation technique is using an antibody coupled to a fluorescent dye to label specific cells, separating labeled cells from the unlabeled ones in an electronic *fluorescence-activated cell sorter*.

5.5.2 DNA amplification by PCR technique

A DNA is amplified and used to construct two synthetic DNA oligonucleotides complementary to the sequence on one strand of the DNA double.
DNA oligonucleotides determine the segment of the DNA that is amplified and serve as primers for DNA synthesis performed by a DNA polymerase.
Every PCR reaction cycle begins with a heat treatment of the two strands.
Second step in PCR reaction cycle is the cooling of the DNA in the presence of two primer DNA oligonucleotides and the primers hybridize to complementary sequences in given DNA strands.
During the third step DNA is synthesized: the mixture is incubated with DNA polymerase and the four deoxyribonucleoside triphosphates
The cycle is then begun again and the procedure is performed over and over again and the newly synthesized fragments serve as templates

Chromosomal DNA is first purified from cells to obtain a genomic clone using PCR and the primers that flank the DNA sequence to be cloned are added. Only the DNA between the primers is amplified these way by PCR technique we obtain a stretch of chromosomal DNA in a pure form.

To obtain a cDNA clone of a gene in the first step using PCR, mRNA is purified from cells and then the first primer is added to the population of mRNAs. In the second step to make a complementary DNA strand reverse transcriptase is used and the second primer is added and the one strand DNA molecule is amplified through many cycles of PCR, [3].

5.5.3 DNA Microarray

To prepare the microarray a robot spots DNA fragments corresponding to a gene are onto a slide. In Figure 17 from two different cell samples mRNA is collected then the used samples are converted to cDNA, labeled with flurochrome of two different colors one green one red and then mixed and hybridized to the microarray.
During the process of incubation is necessary to wash the array and scan the fluorescence. This microarray example represents the expression of 110 genes, [3].

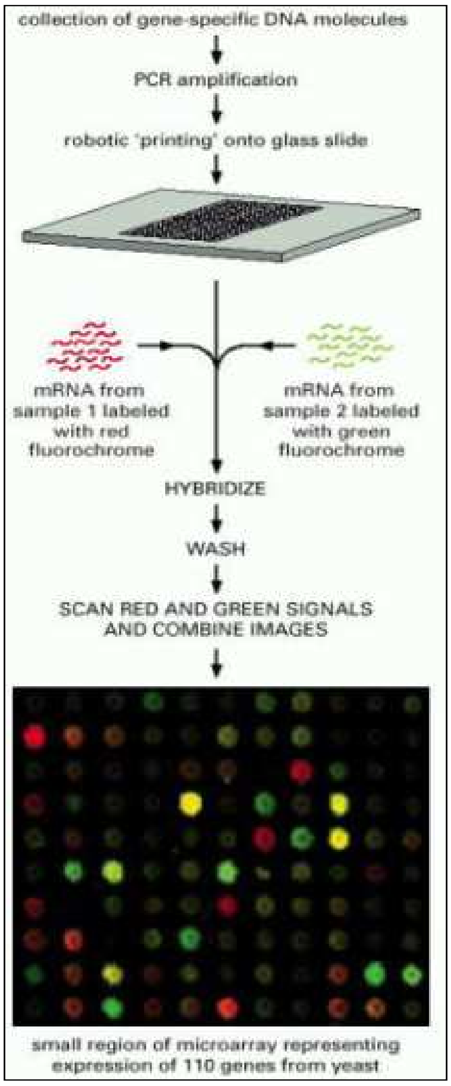

collection of gene-specific DNA molecules
PCR amplification
robotic 'printing' onto glass slide
mRNA from sample 1 labeled with red fluorochrome
mRNA from sample 2 labeled with green fluorochrome
HYBRIDIZE
WASH
SCAN RED AND GREEN SIGNALS AND COMBINE IMAGES
small region of microarray representing expression of 110 genes from yeast

Fig. 17 DNA microarrays expression of thousands of genes, [11]

We can see that gene in first sample is expressed at a higher level than the corresponding gene in sample second sample set color. The expression of the gene is higher and is indicated by spots.

The third sample color set shows the genes which are expressed at equal levels. Dark spots indicate that there is no expression in the sample of the gene located at that position in the array, (Fig. 17).

Bioinformatics contains areas from biology, chemistry, physics, and mathematics, computer science. This fusion offers benefits by increasing biological knowledge for better human health.

Genes from the same cluster may be involved in the same biological processes or pathways.

For cluster analysis we obtain from cell samples microarray data exposed to different conditions. Genes that show changes in expression pattern are grouped together.

Figure 18 shows about 8600 genes which have been grouped in clusters and analyzed on the DNA microarray and as a result over 300 showed variations in expression patterns.

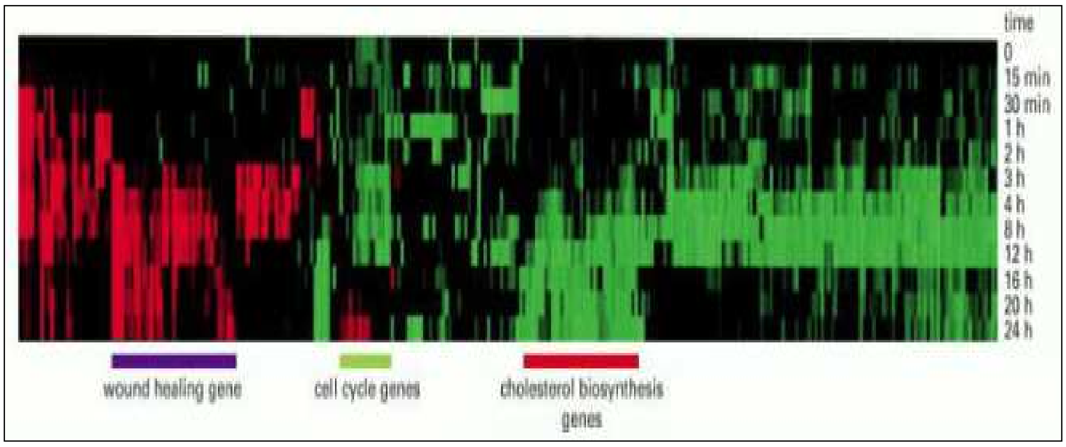


Fig. 18 DNA microarray results, [11]

Red color set indicates an increase in expression and green color set indicates a decrease in expression, genes involved in healing are turned on in response to serum, while genes involved in regulating cell cycle progression are shut down.

5.5.4 Bioinformatics and Biology

Organisms are an arrangement of biological interacting structures made from smaller building blocks (cells), cells are made from smaller blocks as proteins and nucleic acids. The major aim of biology is to identify and characterize these structures. An important part of biology is search for ordered systems which make possible to establish general rules.

The discovery of DNA structure, understanding of RNA have been important milestones of modern molecular biology allowing an understanding of central cellular mechanisms. RNA molecules and proteins are involved at all steps of biological processes, coded in DNA sequences, in universal manner.

At the lowest level, a genome is composed from long string of nucleotides such as a very long text made of four letters. DNA is translated into RNA that in turn is translated into proteins which contains motifs with different functions. The genome contains also genes (enhancer, promoter, introns, exons), [13].

Identification of biological structures at a molecular level and function characterization are the aims of bioinformatics and molecular biology. To understand the mechanisms involved in functions is necessary to understand if this function is performed by a protein, regulatory sequence or RNA, to understand what portions of the proteins are involved in a certain operations.

In the past the only way available to obtain this information was to use wet lab techniques, such as sequencing, cloning and genetic analysis. The cost involved in wet lab techniques is very high considering time and money.

Bioinformatics uses dry lab techniques and starts from a phenotype, a sequence and then bioinformaticians gather information by comparing the sequence to other sequences. Bioinformatics is based on a series of comparisons and predictions. Sequence conservation is correlated with function conservation.

Darwinian laws of evolution are the assumption that biological systems evolved from the same origin and basic building blocks, later adapting them to their environmental constraints.

Cycles of mutations and selections are considered to constitute evolution by this are considered that new functions have been created, reusing existing machinery, [89].

The existing functions evolved to adapt to the environment. Two sequences responsible for the same functions may be different it depends how long the sequences diverged.

Cell elements exist only as 3D arrangement of atoms and only because the molecules 3D structure it has mechanical and chemical properties.

Molecular biology and bioinformatics aim is to study the relation between structure and function. Different amino acid sequences can code for almost identical 3D folds. This made us to understand why proteins with different sequences have similar function and structure, [62].

Bioinformatics aims as well are the representation, organization manipulation and use of information in digital form such as representation, storage, distribution of data, design of databases, creation of tools to query databases, development of interfaces and analytical tools to study biological data.

5.5.5 DNA, RNA, Proteins and Sequence Analysis

All living organisms consist of **cells** with the same set of chromosomes. C**hromosomes** are strings of DNA consisting of **genes** (Fig.19). A gene encodes a protein (**trait)**. The settings for a trait are called **alleles**. Each gene has its own position in the chromosome, [13]. This position of the gene in chromosome is called **locus**. The **genome** is the complete set of genetic material.

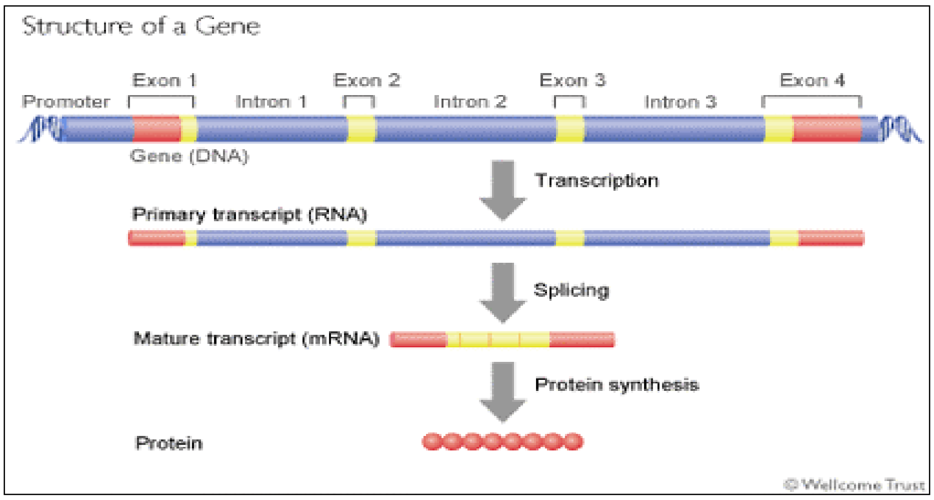


Fig. 19 Structure of a Gene [EBI, Welcome Trust]

RNA contains the sugar *ribose* as opposed to *deoxyribose* in DNA and three bases are the same: *guanine* (G), *adenine* (A) and *cytosine* (C). The fourth base for DNA is *thymine* (T) and in RNA, *uracil* (U). RNA copies of DNA are made by *transcription process* followed by translation process resulting in the protein synthesis.

5.5.6 Central Dogma of Molecular Biology

Central Dogma of Molecular Biology involves two important process, transcription and translation.

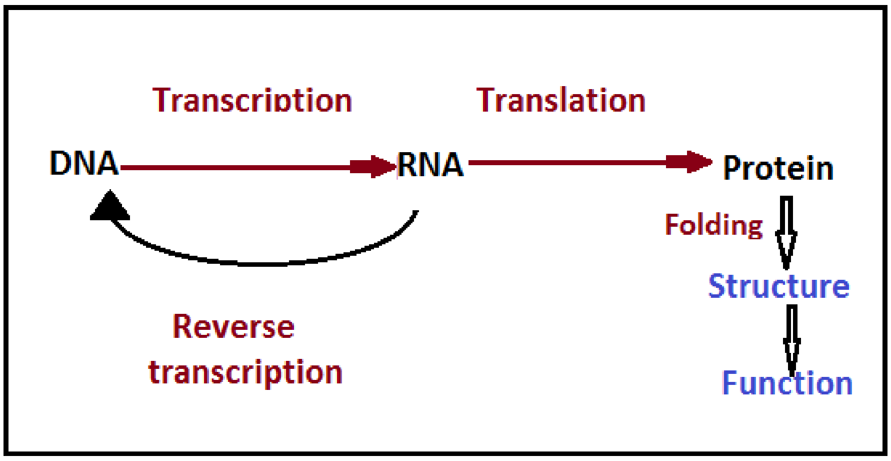


Fig. 20 Central Dogma of Molecular Biology, [59]

During the process of *Transcription* the DNA information is converted to RNA, followed by translation process where RNA is converted to Protein. Three nucleotide bases, called *triplets (codons*), code for individual amino acid, there are 20 amino acids, [59].

Ribosome uses the RNA *transcript* and translates the order of successive codons into the corresponding order of amino acids. The termination of polypeptide chain elongation is introduced to ribosome machinery by *stop codons*: UAA, UAG and UGA.

In RNA the start signal for translation serves the codon for the amino acid called *methionine* AUG.

The process of section between the *start* and *stop* codons is called an *open reading frame* and *correct* open reading frame for a certain region of DNA is the region which has the longest distance between any start codons and stop codons.

5.5.7 Sequence Alignment

When a new sequence is available we might want to search the database and find out if relatives of this sequence have been reported. If yes we may wish to determine experimental data acquired to the new sequence and the solution is to compare the given sequences to sequences contained in the database, finding similarities.

Two popular tools to perform database similarity searches are FASTA and BLAST. Sequence comparison can be complex as by combining experimental data from the databases we may want to know a specific motif for a protein.

Finding alternate motifs we might want to build a classification and establish functional differences between the binding motifs. This idea was developed in the Prosite database. New sequences may be scanned for the known motifs contained and such strategies even they seem simple yet they present very complex difficulties needed to be overcome.

When comparing two sequences or more it depends on the aim and scope what kind of comparation to perform. If we need to do detailed analysis, sequence alignments is one of the most powerful solutions.

If we have only two sequences involved we can do the pairwise alignments and we can extend it also to a larger number of sequences performing multiple sequence alignments. It depends very much what kind of biological information we want to extract from a certain alignment.

In some cases the criteria that allow evaluation of the quality of an alignment takes the form of a mathematical objective function and is associated with a value to an alignment, [84].

A very important aspect in sequence alignment is to build the best scoring alignment according to the quality criterion. And this is not an easy task in most cases as many of the problems in bioinformatics and more specifically in sequence alignment are NP complete, meaning that that the number of potential solutions rises exponentially with the number of sequences and their length. It means that the solution cannot be found in polynomial time and space.

The solution to overcome these limitations requires the development of new powerful algorithms for sequence alignments. Performing a sequence alignment we can determine reflected relationship between two sequences.

If we aim to determine the relationship between two structures we will perform the alignment of two residues as they are equivalent in the 3D structures.

If we want to determine the phylogenetic relationships, two residues will be aligned when they originate from the same residue in the common ancestor.

If an analyzed sequence lacks one residue is inserted a gap in its place at the corresponding position.
Gaps take the form of strings of nulls and in evolution context, a null means that a residue was deleted or inserted in a sequence while the sequences were diverging from their common ancestor, [74]

There are two types of alignments: global alignment and local alignment.
In a local alignment, the aligned portions are those which are homologous, meaning that they diverged from the common ancestor and the rest of the sequence is ignored.

In a global sequence alignment the whole sequences are aligned.
The scope of these alignments is different. Local alignments scope is destined when the sequences analyzed are related and share a few domains. Global alignments are used to analyze sequences that are homologous.

The most used application of sequence alignment is the database searching with the aim to find for a given query sequence related sequences from the database.
Sequence alignments involve structure prediction, identification of new motifs or domains and for these applications pairwise alignments are of limited use.
A way to simultaneously combine the information contained in several sequences is needed and serves as a main motivation for building multiple sequence alignments, [25].

We also perform sequence alignment with a scope to localize *highly conserved area to design efficient PCR* primers needed to clone new members of a family.
The given examples reflect the importance of multiple sequence alignments in the domain of sequence analysis.

A scoring function associates a score to sequence alignment, the better the score is it means the more biologically accurate the sequence alignment and is said to be optimal, whether it is biologically relevant or not.
An optimal alignment always exists. Powerful statistical tools have been developed for this purpose.

As presented the problem of aligning two sequences is NP complete and it can be solved with dynamic programming technique. Because of this NP completeness most of the algorithms developed for sequence alignments are heuristics and they do not guarantee a mathematically optimal solution, but rather a good approximation.

5.5.8 DNA Gene Databases

Databases important resources for bioinformatics and are essential to support genomics research in the lab. Every published DNA, RNA or protein sequence is deposited in a gene database.
The databases can be separated into three categories:

DNA databases: There are three main DNA databases freely accessible. They are:
1) EMBL (Europe) http://www.ebi.ac.uk/embl/
2) Genbank (USA) http://www.ncbi.nlm.nih.gov/genbank
3) DNA Data Bank of Japan DDJB (Japan) http://www.ddbj.nig.ac.jp/
These three gene databases exchange their data daily and are identical in content.

Each database consists of number of entries and each entry consists of a single sequence with an annotation that gives information regarding function, identity and history.

Special bioinformatics databases: Not all biologically relevant databases consist of sequences and annotation.

1) PDB: a database of protein 3D structures (http://www.rcsb.org/pdb)
2) TIGR: a database of complete genomes (http://www.tigr.org/)
3) Ensembl: the database containing the human genome (http://www.ensembl.org)
4) Prosite: a database of protein families. (http://ca.expasy.org/prosite/)
5) RFAM: database of non-coding RNA http://www.sanger.ac.uk/Software/Rfam/
6) REBASE: restriction enzyme database:
http://rebase.neb.com/rebase/rebase.html

There are also databases of scientific literature: journal abstracts, taxonomy, protein structures, protein families and metabolic pathways very useful when looking for very specific information about a DNA sequence.

6. DNA Cryptography Model

With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's information infrastructure.

Biotechnological Methods as recombinant DNA have been developed for a wide class of operations on DNA and RNA strands. Bio Molecular Computation (BMC) makes use of biotechnological methods for doing computation and splicing operations allow for universal computation.

The first applications of DNA-based cryptography systems using biotechnologies techniques included: methods for 2D data input and output by use of chip-based DNA micro-array technology and transformation between conventional binary storage media via (photo-sensitive and/or photo emitting) DNA chip arrays
Lately DNA Cryptosystem using substitution and biotechnologies have been developed: *Substitution* one-time-pad encryption: is a substitution method using libraries of distinct pads, each of which defines a specific, randomly generated, pair-wise mapping. The decryption is done by similar methods. The *Input is a* plaintext binary message of length n, partitioned into plaintext words of fixed length, [39].

Substitution One-time-pad, a table randomly mapping all possible strings of plaintext words into cipher words of fixed length, such that there is a unique reverse mapping and the e*ncryption is done by* substituting each i-th block of the plaintext with the cipher word given by the table, and is decrypted by reversing these substitutions. Using long DNA pads containing many segments, each segment contains a cipher word followed by a plaintext word and the cipher word, acts as a hybridization site for binding of a primer. Cipher word is appended with a plaintext word to produce word-pairs. The word-pair DNA strands are used as a lookup table in conversion of plaintext into cipher text, [57].

*One-time-pad DNA Sequence with l*ength n, contains d = n/(L1+ L2+ L3) copies of repeating unit *Repeating unit* made up of:
1. Bi = a cipher word of length L1 = c1log n
2. Ci = a plaintext word length L2= c2log n

Each sequence pair uniquely associates a plaintext word with a cipher word and the Polymerase acts as a "stopper" sequence of length L3 = c3.
To generate a set of oligonucleotides corresponding to the plaintext/cipher and word-pair strands, ~Bi used as polymerase primer and *extended* with polymerase by specific attachment of plaintext word Ci. The *Stopper sequence* prohibits extension of growing
DNA strand beyond boundary of paired plaintext word.

Methods for Construction of DNA one-time pads are based on the biotechnologies rather than bioinformatics and present difficult to achieve both full coverage and yet still avoiding possible conflicts by repetition of plaintext and cipher words. These methods make use of DNA chip technology for random assembly of one-time pads. The advantages are that are currently commercially available (Affymetrix) chemical methods for construction of custom variants are well developed.

Other method also based on biotechnologies is so called method *DNA chip Method* for Construction of DNA one-time pads where is used an array of

immobilized DNA strands and multiple copies of a single sequence are grouped together in a microscopic pixel which is optically addressable. Using the technology for synthesis of distinct DNA sequences at each (optically addressable) site of the array and combinatorial synthesis conducted in parallel at thousands of locations, prepared of oligonucleotides of length L, the 4L sequences are synthesized in 4n chemical reactions.
As an Example: 65,000 sequences of length 8 use 32 synthesis cycles and 1.67x107 sequences of length 10 use 48 cycles. The construction of DNA One-time pads based on biotechnologies was first developed by the pioneer in this field, Adleman [39].

XOR One-time-pad (Vernam Cipher) Cryptosystem based on biotechnologies
One-time-pad: S is a sequence of independently distributed random bits
M is a plaintext binary message of n bits resulting in the following cipher text,
Ci = Mi XOR Si for = 1,…,n.
*Decrypted bits, u*se commutative property of XOR Ci XOR resulting in:
Si = (Mi XOR Si) XOR Si= Mi XOR (Si XOR Si)= Mi.

DNA Implementation of XOR One-time-pad Cryptosystem:
The *plaintext messages is* one test tube of short DNA strands
The *encrypted message is* another test tube of different short DNA strands

Encryption by XOR One-time-pad maps these in a random and reversible way such as plaintext is converted to cipher strands and plaintext strands are removed. For the *efficient* DNA encoding Adleman proposed to use *modular base 4 as* DNA has four nucleotides. Encryption constitutes the addition of one-time-pad elements modulo 4 and decryption is the subtract one-time-pad elements modulo.
Details of DNA Implementation of XOR One-time-pad Cryptosystem based on biotechnologies:

Each plaintext message has appended a unique *prefix index tag* of length L indexing it. Each of one-time-pad DNA sequence has appended unique *prefix index tag* of same length L, forming *complements* of plaintext message tags. Using recombinant DNA bio techniques such as annealing and ligation in order to *concatenate into a single DNA strand* each corresponding pair of a plaintext message and a one-time-pad sequence resulting in *enciphered by bit-wise XOR computation and* fragments of the plaintext are converted to cipher strands using the one-time-pad DNA sequences, and plaintext strands are removed.
The reverse decryption is similar using commutative property of bit-wise XOR operation.

*BMC Methods to effect bit-wise XOR on Vectors. This method c*an adapt BMC methods for *binary addition* and similar to bit-wise XOR computation can *disable carry-sums* logic to do XOR. *BMC techniques for Integer Addition were implemented by* Bancroft in 1996 first BMC addition operations (on single bits) permit chaining on n bits.

Addition by *Self Assembly* of DNA tiles was exploited by, Reif in 1997 and LaBean in 1999. *XOR by Self Assembly of DNA tiles [39].*
*XOR by Self Assembly of DNA tiles includes that f*or each bit Mi of the message, construct sequence ai that represents the *ith* bit.
Scaffold strands for binary inputs to the XOR are the usage of linkers to assemble the message M's n bits into scaffold strand sequence *a1, a 2 … a n.*

The One-time-pad is further portion scaffold strand *a' 1a' 2… a'n* and is created from random inputs add output tiles, the annealing give self-assembly of the tiling.
The next step: adding ligase yields to the reporter strand:

R = a 1 a 2 … a n.a' 1 a' 2… a'n.b 1 b 2 … b n, where *b i = a i XOR a'i, for i = 1,…,n.*
In the next step the reporter strand is extracted by biotechnique of melting away the tiles, smaller sequences, and purifying it, contains concatenation of input message, encryption key, ciphertext.
Before the final last step using a marker sequence the ciphertext can be excised and separated based on its length being half that of remaining sequence. In the last step ciphertext is stored in a compact form.

These increasing importance of information security and the protection of human privacy rights as Confidentiality lead me to develop new security solutions based on modern technologies: Bioinformatics and Biotechnology.

In this work I present a technical process for protecting data assets such as personal medical information using Bioinformatics and a DNA cryptography technique based on bioinformatics rather than biotechnologies in this bioinformatics technique a person's own blood mineral levels serve as a seed for selecting, transmitting, and recovering his sensitive personal data.

As we know that the management of security keys remains a challenge, I also developed a bioinformatic mechanism to generate encrypt-decrypt keys by taking into consideration specifics of the cryptography method and the individual's DNA genome analysis described in the 8th chapter of these thesis: DNA Cryptographic Keys Based on Evolutionary Models

My work was based on the complexity of developing, as a subset of JCE, an unconditionally secure DNAE System as part of my security provider, named DNAProvider, [60]

A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory. When an attacker is theoretically able to intrude, but it is computationally infeasible with available resources, the cryptographic system is said to be *conditionally secure.* The mathematics in such systems is based on computational complexity theory. To design a secure cryptographic system is a very challenging, [27].

A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output. If an algorithm's behavior is completely determined by the input, it is called *deterministic,* and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*.

A distributed algorithm in which two or more entities take part is defined as a protocol including a set of communicational and computational steps. Each communicational step requires data to be transferred from one side to the other and each computational step may occur only on one side of the protocol.

The goal of every cryptographer is to reduce the probability of a successful attack against the security of an encryption system – to zero. Probability theory provides the answer for this goal. MY work was based on the complexity of developing an unconditionally-secure DNA Encryption System as part of DNA Provider.

Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing me to extend the JCE by implementing faster or more secure cryptographic algorithms. By the same means I provide my independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB), [59].

6.1 Architecture of DNA Memory for DNA Cryptography Model

DNA memories have the potential to store vast amount of information with high density, and are able to process the stored information through protocols matching the context, [39].

DNA Cryptography model application requires a massive parallel computation to solve computational problem through massive parallelism.

Current technology is not capable of control biomolecules required complex computations. DNA memory has the potential to store massive amount of information.

The DNA memory should be a database of information stored in the sequences of DNA molecules.

DNA computer can do computations on the stored data by manipulating the contents of the test tube.

Four steps need investigation:

1. To map records onto DNA sequences coding strands.
2. To design methods for retrieving and manipulation of information stored in DNA to build a DNA memory architecture
3. To retrieve information we need to quire the memory to match the data with stored records. For this we need to design protocols to process in vitro the queried strands.
4. DNA memory output strands needs to be read and converted back into a readable format.

The goal for the DNA memory is to store the data, recall the data, and manipulate data content in order to make reasoned inferences about relationships among the data.

Many data processing and mining tasks involve error prone data, and thus, a modicum of machine intelligence is an advantage to recognize relevant relationships and matches.

Reaton and Chen proposed a DNA memory architecture modeled upon the following context and concepts, [17]:

Definition 1: *A formal context* $K := (O, A, I)$ *) consists of a set of objects O and attributes A, where* $I \subseteq O \times A$ *(denoted oI which means o has attribute a), [17].*

The sequence for the object becomes a label for a molecular record composed of attribute sequences.

Definition 2: *A formal concept of the context (O, A, I) is a pair (B, C) where* $B \subseteq O, C \subseteq A, C = B^{'} := \{a \in A \mid oIa \forall o \in B\}$ and $B = C^{'} := \{o \in O \mid oIa \forall a \in C\}$ *, [17].*

The conceptual space needs to be created in DNA and explored *in vitro* operations. A DNA memory is represented by a collection of DNA words *M*. The DNA words can be divided in subsets: O-representing objects and A representing attributes.

DNA words defined by union and intersection are a complete lattice, if an operation correspond to set complement (*S_* = *M* \ *S*) then 1 is represented by set of all sequences and 0 is represented by the empty set.

Set union corresponds to logical OR and involves to mix two set of molecules. Set intersection corresponds to logical AND, and the implementation of this set involves several key components.

Set complement corresponds to logical NOT and the set to be complemented *S* is separated from the set of sequences *M* to form *S1* = *M* \ *S*. Set intersection and complement could be realized based on magnetic bead extraction.

6.2 Mapping Data onto DNA Sequences

In order to synthesizing DNA strands cost effective a good approach is to use cloning technology. Cost of synthesis needs to be reduced by starting with random sequences. These random sequences would have known primer sequences on each end, similar to the selection protocol for noncross hybridizing oligonucleotides, [17]. This way is possible to amplify the starting DNA material, and then isolate a sequence from which we can form a number of records.

Internal sequences can be inserted in a plasmid and the primer sequences will have a restriction site. Then we can extract the sequences and build a library of coding strands mapped to our information.

Sequences representing attributes and objects we map to different colonies, ligated together to form complete records and producing in vitro molecules corresponding to attributes in objects permutations and combinations, (Fig.21).

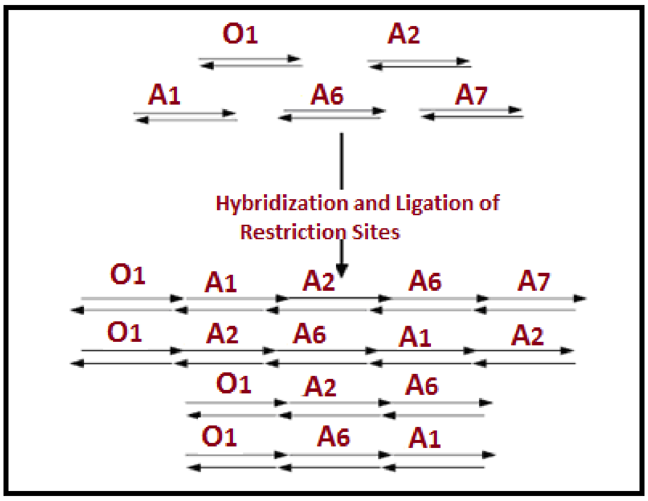


Fig. 21 Sequences representing attributes and objects

Ideally, DNA molecules corresponding to the rows of the Table would be created, with DNA words for labeling objects ligated to words representing attributes, [39].The objects will correspond to a record identifier, and the attributes will correspond to the terms.

By hybridizing input words it is possible to produce every possible combination and permutation of words representing the attributes of the object.

An individual coding strands representing a record needs to be mixed in a test tube and their primers or restriction sites need to be allowed to hybridize to perform ligation.

The design is similar to how Adleman formed all possible paths in a graph to find the Hamiltonian path. Then we shall search the memory for matching queries to the closest object forming a molecular representation of the query composed of permutations and combinations of complements of the query terms, used to separate objects with the desired attributes,[39].

In Figure 21, a query of (*A*6 ∩*A*7) extracts all attributes that have corresponded objects, forming a molecular representation of the formal concept *{{O*1,*O*2*}, {A*6,*A*7*}},* [17].

Shared attributes mean that molecules share sequences that can be used through affinity separation to represent similar content. Molecular representation of an object, attributes occurs in the same context as their sequences are common to a given molecule. Content and context is translated in the DNA memory following the idea that sequences representing different attributes occur in the context of the same molecule.

These capabilities were achieved *in vitro* with the advantages of DNA massive parallelism. Output was formed by attaching the cloned; coding sequences to an array and each spot from an array represented either an object or an attribute.
The reading was performed directly from sensing fluorescent tags attached to memory strands as probes.

The DNA Computing Model based on molecular Computation for DNA Information Storage and Retrieval is needed in order to use DNA cryptography at a large scale with capabilities to store vast amount of information with high density, like DNA public and private keys used for encryption and decryption in DNA cryptography model.

Realization of this architecture was implemented in vitro by Deaton and Chen, using DNA molecules. They implemented an algorithm that automatically store and retrieves the data from the model architecture using parallel molecular operations, [17].

DNA cryptography is a cryptographic field with research of DNA computing. DNA is used as information carrier and biological technology is used as implementation tool.

7. Complexity of DNA Encryption System as a Subset of Java Cryptography Extension

7.1 Creating the DNA Security Provider with DNA Encryption

The goal of the security provider interface is to allow a means whereby specific algorithm implementations can be substituted for the default provider, SUN JCE. JCE was developed as an extension package which includes implementation for cryptographic services. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. Thus, JCE offers support for developing alternative cryptographic package providers, (Fig.22).

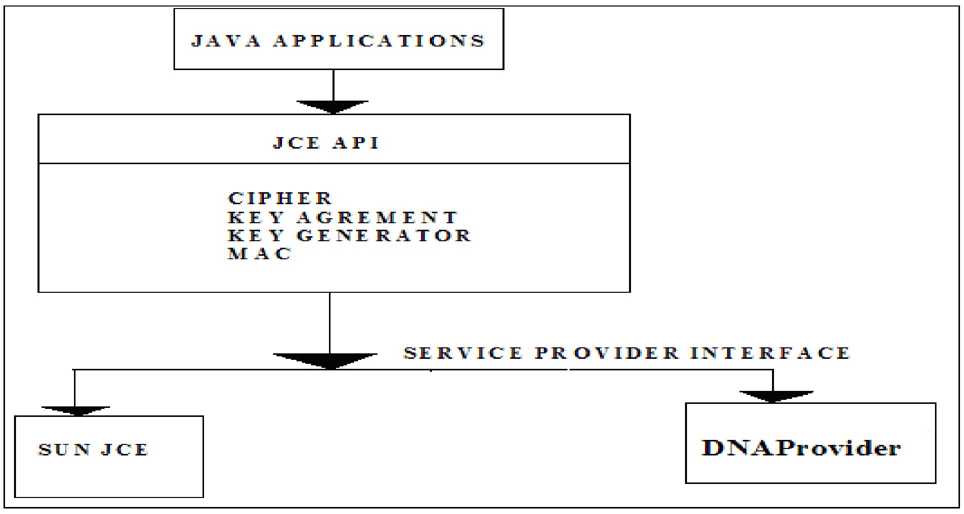


Fig. 22 Java Cryptography Extensions architectural model with unconditional secure DNA Encryption as part of our security provider (DNAProvider), [60]

This support allows us to provide our independent implementation of DNAE System, based on the CDMB (Central Dogma of Molecular Biology), [60].

The application code calls the appropriate JCE API classes. The JCE API classes invoke the classes in a provider that implements the interface classes, JCE SPI. The JCE SPI classes, in turn, invoke the requested functionality of the DNAProvider.

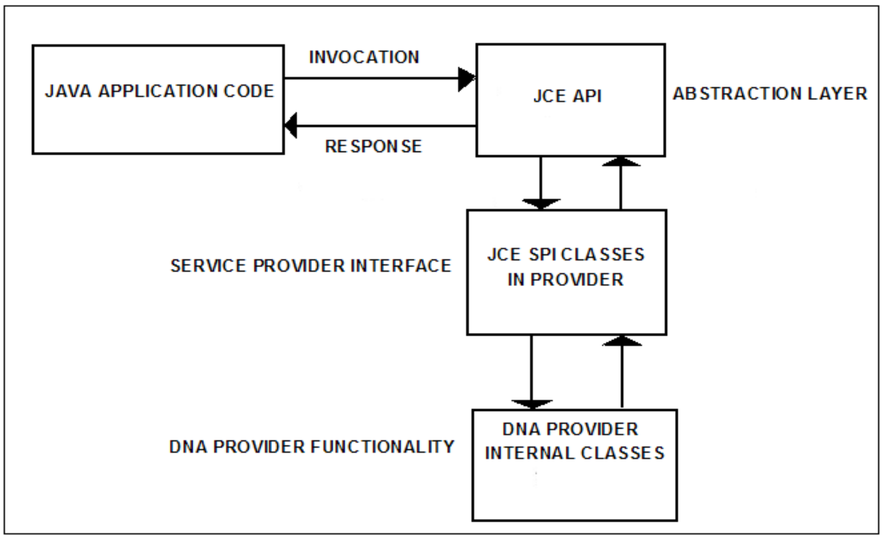

JAVA APPLICATION CODE
INVOCATION
RESPONSE
JCE API
ABSTRACTION LAYER
SERVICE PROVIDER INTERFACE
JCE SPI CLASSES IN PROVIDER
DNA PROVIDER FUNCTIONALITY
DNA PROVIDER INTERNAL CLASSES

Fig. 23 Invocation of DNAProvider for providing requested functionality, [57]

When the Java Virtual Machine starts execution, it examines the user's properties to determine which security providers should be used. The user's properties are located in the file *java.security,* in which each provider is also enumerated. If users prefer to use DNAProvider as an additional security provider they can edit this file and add the DNAProvider. When the Security Class is asked to provide a particular engine and algorithm, it searches the listed providers for the first that can supply the desired operation.

7.2 The Architecture of Security Providers

The security provider abstracts two ideas: engines and algorithms. An Engine Class defines an abstract cryptographic service, without its concrete implementation. JCE 1.2.2 is provided as an optional package and adds engine classes such as: Cipher, KeyAgreement, KeyGenerator, MAC, and SecretKeyFactory. The application interfaces given by an engine class are implemented and referred to as the *Service Provider Interface,* [70].

The goal of the security provider interface is to allow an easy mechanism where the specific algorithms and their implementations can be easily changed or substituted. The architecture including all of this contains:
Engine classes, these classes come with the Java virtual machine as part of the core API.
Algorithm classes, at the basic level, there is a set of classes that implement particular algorithms for particular engines.

A default set of these classes is provided by the supplier of the Java platform. Other third-party organizations or individual can supply additional sets of algorithm classes. These classes may implement one or more algorithms for one or more engines. A single algorithm class provides a particular algorithm for a particular engine.

The Provider class, each set of algorithm classes from a particular vendor is managed by an instance of the class Provider. A provider knows how to map particular algorithms to the actual class that implements the operation.

The Security class, maintains a list of the provider classes and consults each in turn to see which operations it supports.

When the security package needs to perform an operation, it constructs a string representing that operation and asks the Security class for an object that can perform the operation with the given algorithm, [35].

For example, the idea of generating a message digest is represented by a particular engine.

Seventeen cryptographic engines are supported by Sun's security providers; there are implementations of at least one algorithm of each engine in one of Sun's providers. The engines and the algorithms implemented by Sun are listed in Table 1. SunJCE is the provider that comes with the Java Cryptography Extension, and SunJSSE is the provider that comes with the Java Secure Sockets Extension.

When the Java virtual machine begins execution, it is responsible for consulting the user's properties in order to determine which security providers should be in place. These properties must be located in the file *$JREHOME/lib/security/java.security*. The file contains these lines (among others):
security.provider.1=sun.security.provider.Sun
security.provider.2=com.sun.rsajca.Provider

These lines tell us that there are at least two provider classes that should be consulted. The first class to be consulted is an instance of the sun.security.provider.Sun class and the second class is an instance of the com.sun.rsajca.Provider class.

Each provider given in this file must be numbered, starting with 1.
If I want to use additional provider I need to add these lines to the *java.security* file.
security.provider.3=com.sun.crypto.provider.SunJCE
security.provider.4=com.sun.net.ssl.internal.ssl.Provider

The order of these properties is significant; when the Security class is asked to provide a particular engine and algorithm, it searches the listed providers in order to find the first one that can supply the desired operation. All engine classes use the security class to supply objects. When the message digest engine is asked to provide an object capable of generating SHA message digests, the engine will ask the Security class which provider to use. The number that follows the security.provider string indicates the order in which providers will be searched for particular implementations.

In the core Java API, the Provider class is abstract and there are no classes in the core Java API that extend the Provider class. The default provider class that comes with Sun's implementation of Java is the class Sun in the sun.security.provider package.

The Provider class contains a number of useful methods:
-*public String getName( ),* return the name of the provider.
-*public double getVersion( ),* return the version number of the provider.
-*public String getInfo( ),* return the info string of the provider.
-*public String toString( ),* return the string specifying the provider, provider's name concatenated with the provider's version number.

Going to provide my own set of classes to perform security operations, I must extend the Provider class and register that class with the security infrastructure. Provider class is abstract, none of its methods are abstract, I need do is subclass the Provider class and provide an appropriate constructor. The subclass must provide a constructor since there is no default constructor within the Provider class, [60]. The only constructor available is: *protected Provider(String name, double version, String info)*

The basic implementation of a DNAProvider security provider is:

```
public class DNAProvider extends Provider {
        public DNAProvider( ) {
                super("DNAProvider", 1.0, "DNA Security Provider
                     v1.0");
        }
}
```

Here I am defining the skeleton of a DNAProvider that is going to provide certain facilities based on CDMB.

Properties Included by DNAProvider and Corresponding Class are listed in Table 1.

Table 1. Properties and Corresponding Class

KeyGenerator.XOR	tanyasec.referate.ex01.XORKeyGenerator
KeyPairGenerator.DNA	tanyasec.referate.ex01.DNAKeyPairGenerator
KeyFactory.DNA	tanyasec.referate.ex01.DNAKeyFactory
MessageDigest.DNA	tanyasec.referate.ex02.DNAMessageDigest
Signature.DNAwithSHA	tanyasec.referate.ex03.DNASignature
Cipher.XOR	tanyasec.referate.ex04.XORCipher
KeyManagerFactory.DNA	tanyasec.referate.ex05.SSLKeyManagerFactory

In order to make the associations from this table, then the DNAProvider needs to be as is described.

There are a number of other methods in the Security class that provide basic information about the configuration of the security provider:

-*public static void removeProvider(String name):* Remove the named provider from the list of provider classes. The remaining providers move up in the array of providers if necessary.

-*public static Provider[] getProviders( ):* Return a copy of the array of providers on which the Security class operates.

-*public static Provider getProvider(String name):* Return the provider with the given name. If the named provider is not in the list held by the Security class, this method returns null.

-*public static String getProperty(String key):* Get the property of the Security class with the associated key. When the addProvider( ), insertProviderAt( ), and removeProvider( ) methods are called, the order of the providers changes. These changes are not reflected in the internal property list. The *java.security* file has a

number of other properties within it; these other properties may also be retrieved with the folowing method.
-public static void setProperty(String property, String value): Method sets the given property to the given value.
-public static String getAlgorithmProperty(String algName, String propName): This method searches all the providers for a property in the form Alg.propName.algName and returns the first match it finds. Example 2 lists all cryptographic providers installed on a machine. After running this program with the default security providers, we get the following output, [57].

```
Compiling 1 source file to C:\Documents and
Settings\Administrator\tatiana\build\classes
compile-single:
run-single:
Provider[0]:: SUN 1.5
Provider[1]:: SunRsaSign 1.5
Provider[2]:: SunJSSE 1.5
Provider[3]:: SunJCE 1.5
Provider[4]:: SunJGSS 1.0
Provider[5]:: SunSASL 1.5
```

7.3 The Security Class and the Security Manager

Some of the public methods of the Security class call the checkSecurityAccess( ) method of the security manager. This gives the security manager the opportunity to intervene before an untrusted class affects the security policy of the virtual machine.

If a program that wants to install the DNAProvider security provider must have been granted the SecurityPermission named "insertProvider.DNA". The methods of the security class that require security permission and the names of the permission they require are listed in Table 2.

Table 2 Security Checks of the Security Class

Method	**Parameter**
insertProviderAt( )	insertProvider. + provider.getName( )
removeProvider( )	removeProvider. + provider.getName( )
getProperty( )	getProperty. + key
setProperty( )	setProperty. + key

7.4 Steps to Implement and Integrate the DNAProvider

Step 1: Write the DNAProvider Service Implementation Code
Step 2: Give my Provider a Name, (DNAProvider)
Step 3: Write my "Master Class," a subclass of Provider
Step 4: Compile the Code
Step 5: Prepare for Testing and Get a Code-Signing Certificate

In order, I mailed all the hardcopy containing DNAProvider (CSR) and contact information to:
Sun Microsystems, Inc.
International Trade Services/Export Compliance
Attn: Encryption Export
4120 Network Circle MS: USCA12-204
Santa Clara, CA 95054
U.S.A.
DNAProvider Certificate Signing Request (CSR) from Sun Microsystems, Inc.
Step 6: Run the Test Programs using the Provider Code Signing Certificate and JCE Root CA Certificate
Step 7: Document DNAProvider
Step 8: Make DNAProvider Software and Documentation available to Clients, [58].

7.5 DNA Encryption System as a Subset of Java Cryptography Extension

Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing us to extend the JCE by implementing faster or more secure cryptographic algorithms.
By the same means I provide my independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB). In this work I present a technical process for protecting data assets such as personal medical information using a DNA cryptography technique in which a person's own blood mineral levels serve as a seed for selecting, transmitting, and recovering his sensitive personal data, [58].

As we know that the management of security keys remains a challenge, I also developed a mechanism to generate encrypt-decrypt keys by taking into consideration specifics of the cryptography method and the individual's blood analysis. The intention was to use the DNA Provider with the DNAE System in medical applications to ensure security of medical information

Research considers the use of the Human genome in cryptography. In 2000, the Junior Nobel Prize was awarded to Romanian-American student, Viviana Risca, for her work in DNA steganography. A DNA-encoded message is first camouflaged within the enormous complexity of human genomic DNA, and then further concealed by confining this sample to a microdot.

A prototypical ‘secret message’ DNA strand contains an encoded message flanked by polymerase chain reaction (PCR) primer sequences. Denatured human DNA provides a very complex background for concealing a secret-message. Risca, knowing both the secret-message DNA, PCR primer sequences and the encryption key could readily amplify the DNA and then proposed a mechanism to read and decode the message, [20].

I proposed to encode the medical records of an individual in DNA data strand flanked by unique primer sequences, which I obtain in the process of deriving a DNA secret key from blood analysis, [57]. The specific mineral levels and their deviation from normal values are considered as a first step. I then mix the message-encoded DNA strand among other decoy DNA strands that will together be sent to a receiver through a public channel. Then I mix the message-encoded DNA strand among other decoy DNA strands that will together be sent to a receiver through a public channel.

A DNA segment that constitutes a gene is read, starting from the promoter (starting position) of the DNA segment.

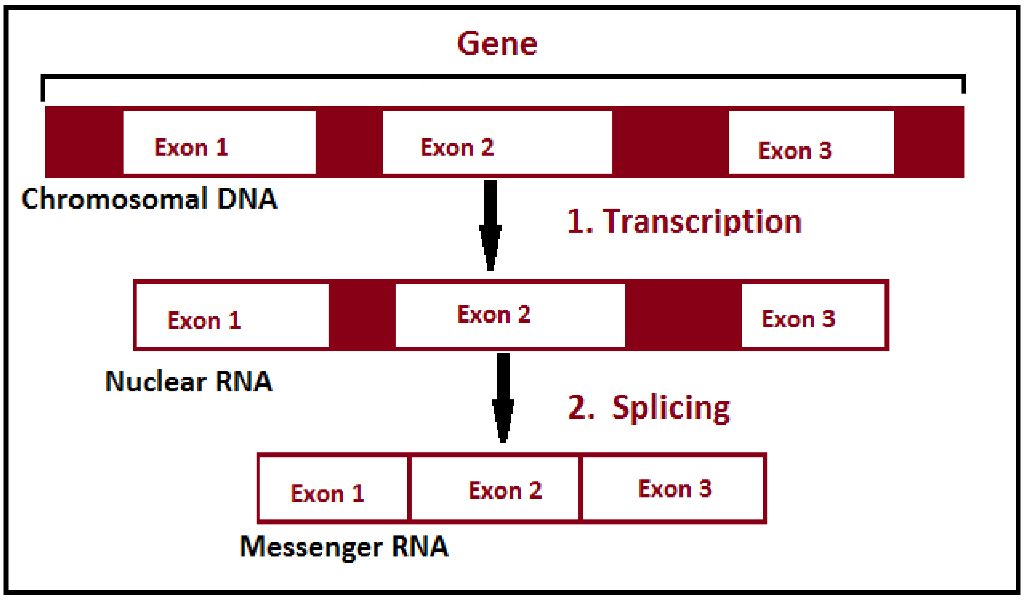


Fig. 24 Central Dogma of Molecular Biology

The non-coding areas (introns) are removed according to certain tags. The remaining coding areas (extrons) are rejoined and capped, (Fig.24). Then the sequence is transcribed into a single stranded sequence of mRNA (messenger RNA). The mRNA moves from the nucleus into the cytoplasm. In chromosomes, DNA acts as a template for the synthesis of RNA in a process called transcription. During RNA Synthesis and Processing in the transcription and splicing steps, introns are excised and extrons are retained to form mRNA, which will perform the translation work.

In the translation process, codons are translated into the amino acids according to the genetic code. The DNA form of information is scanned by a hypothetical operator, Stefani, to find the locations of the introns, which she then records. She cuts out the introns according to the specified pattern so that the DNA form of data is translated into the mRNA form. The mRNA form then translates into the protein form of data according to the genetic code table (64 codons to 20 amino acids).

7.5.1 The DNA Encryption Protocol

Adleman began the new field of bio-molecular computing research. His idea was to use DNA biochemistry for solving problems that are impossible to solve by conventional computers, or that require an enormous number of computation steps. The DNAE technique simulates the CDMB steps: transcription, splicing, and translation process. The time complexity of an attack on a message of length n, is

O(2^n^). DNA computing takes advantages of combinatorial properties of DNA for massively-parallel computation, [39].
Introducing DNA cryptography into the common PKI scenario, it is possible to follow the pattern of PKI, while also exploiting the inherent massively-parallel computing properties of DNA bonding to perform the encryption and decryption of the public and private keys. The resulting encryption algorithm used in the transaction is much more complex than the one used by conventional encryption methods.

To put this into the common description of secure data transmission and reception with respect to DNA cryptography, let me say Stefani is the sender, and Otto, the receiver. Stefani provides Otto her public key which will comprise someone's unique blood analysis. The Public Key (PK) encryption technique splits the key into a public key for encryption and a secret key for decryption. As an example: Otto generates a pair of keys and publishes his public key, while only he knows his secret key. Thus, anyone can use Otto's public key to send him an encrypted message, but only Otto knows the secret key to decrypt it, [59].

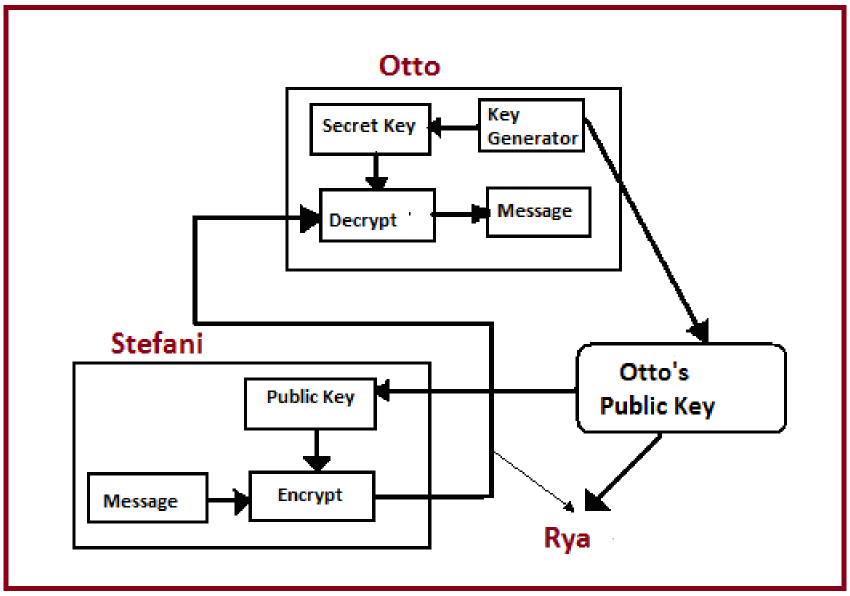


Fig. 25 DNA Public Key Encryption, [59]

A secret DNA data strand contains three parts: a secret DNA data strand in the middle, and unique primer sequences on each side S1. Stefani uses the technique of deriving DNA private key from blood analysis, (Fig. 25)
In this process, Stefani uses a program that associates a specific mineral to the nucleotide sequence based on someone medical results, which will constitute the unique primer sequences S1.

Using an information conversion program, Stefani encodes the medical records in a DNA data strand flanked by unique primer sequences S1 and mixes it among other decoy DNA strands.

According to the CDMB, during the process of transcription, Stefani removes the introns from the data-encoded DNA, resulting in encryption key 1, E1 (starting and pattern codes of introns). Thus, E1 => C1 = E1(P), where P is plain-text and C is the cipher-text. Stefani translates the resulting spliced form of the data from which she derives Encryption key 2, E2 (codon-amino acid mapping). E2 => C = E2(C1) obtains the data-encoded protein after the translation process. Stefani sends Otto the keys E1 and E2 through a public channel.

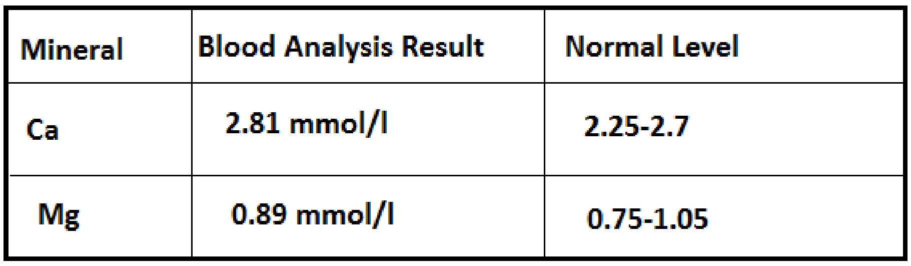
Then she sends Otto the encoded protein form of the data through a public channel. Otto uses the key E2 to recover the mRNA form of the data from the protein form of the data. Decryption key, D1 = E2 => P1=D1(C). Otto recovers the DNA form of the data in the reverse order that Stefani encrypted it. Decryption key, D2 = E1 => P = D2(P1). Otto identifies the secret data-carrying DNA strand using the program that associates the nucleotide sequence based on someone's blood mineral analysis.

He obtains the unique primer sequences S1 that mark the beginning and end of the secret data DNA strand hidden among the decoy strands. In this last step, Otto uses the information conversion program and reads the medical record of the individual.

7.5.2 The Technique of Deriving DNA Cryptographic Keys Sequences from Blood Analysis

As blood analysis results are specific for each person (Table 3), it is possible to associate a mineral such as Calcium, Magnesium, etc., from the medical result, to a nucleotide sequence based on its concentration level. This nucleotide sequence based on the medical results of the specific person will constitute the unique primer sequences, [57].

Table 3, Example Blood Mineral Analysis



Mineral	Blood Analysis Result	Normal Level
Ca	2.81 mmol/l	2.25-2.7
Mg	0.89 mmol/l	0.75-1.05

Thus a person's data-carrying DNA strand will be flanked by primer sequences unique to that individual. And the association could be made in such way that from every most recent blood analysis results, a new primer sequence will be generated.

After generating the unique primer sequence, an n-base primer will result. As we know that the management of private keys remains a challenge, we will use each unique blood analysis as the basis for a secret key generation mechanism.

It is estimated that in the next 3-4 years the DNA sequencing of every individual will be possible, giving us the possibility to derive the public/private keys used in Java KeyStore with respect to each individual blood analysis results using the same model of "Deriving DNA Cryptographic Keys based on evolutionary model mathematical functions" using the desired length of bases from the first column and unique blood analysis results for each person.

Resulting in new set of desired public/private *DNA Cryptographic Keys for future use in DNA Cryptography.* The medical results will be of no use to an unauthorized person, and for an intruder, it would prove extremely difficult to read and detect the DNA strand that contains someone's medical history, without knowing the specific unique primer sequences of the specific person.

Step 1. Stefani (the sender) provides Otto (the receiver) her public key which will constitute each unique blood analysis of the specific person

Step 2. A secret DNA data strand contains 3 parts: Secret DNA data strand in the middle and unique primer sequences on each side S1.

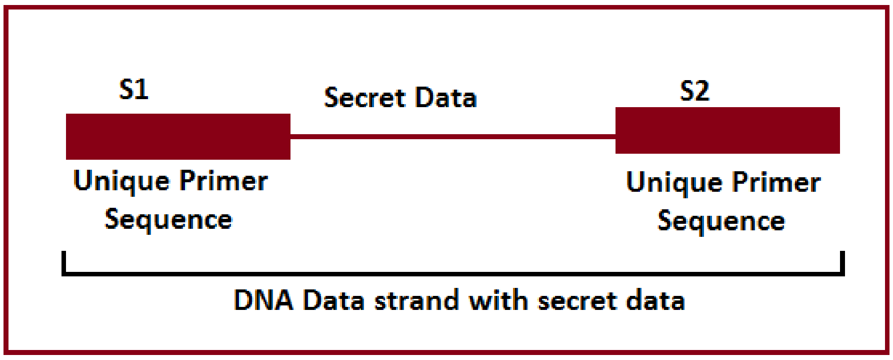


Fig. 26 DNA Data Strand

Step 3. Stefani uses the technique of deriving DNA private key from blood analysis.

3.1 In this process Stefani uses a program which associates to a specific mineral such as Calcium from Table 4, the nucleotide sequence based on the medical results of a specific person, which will constitute the *unique primer sequences* S1.

The Technique of Deriving DNA Private Keys from Blood Analysis*:*
Starting from the idea that the DNA alphabet having 4 letters corresponding to the *four nucleotides*, A, C, G, T. There are 64 possible triplet sequences or codons (4x4x4).

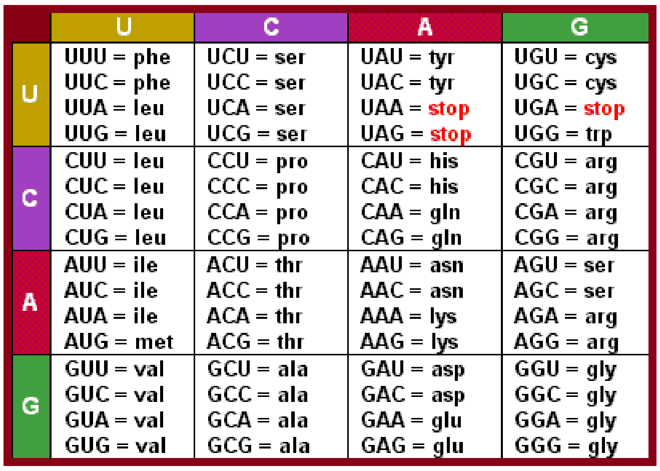
A computer program generates a *nucleotide sequence* S, of *length* **L,** according to the Genetic Code.

I choose a *number* **n** (dependent of the specific mineral value from the particular individual's blood analysis), where **n** represents the number of codons. By a random combination of codons results :

$$L=3*n \tag{7.1}$$

(Ex. L=3*5=15)

3.2 Then I associate a specific *mineral* **M**, with the corresponding *nucleotide sequence* **S**.



	U	C	A	G
U	UUU = phe	UCU = ser	UAU = tyr	UGU = cys
	UUC = phe	UCC = ser	UAC = tyr	UGC = cys
	UUA = leu	UCA = ser	UAA = stop	UGA = stop
	UUG = leu	UCG = ser	UAG = stop	UGG = trp
C	CUU = leu	CCU = pro	CAU = his	CGU = arg
	CUC = leu	CCC = pro	CAC = his	CGC = arg
	CUA = leu	CCA = pro	CAA = gln	CGA = arg
	CUG = leu	CCG = pro	CAG = gln	CGG = arg
A	AUU = ile	ACU = thr	AAU = asn	AGU = ser
	AUC = ile	ACC = thr	AAC = asn	AGC = ser
	AUA = ile	ACA = thr	AAA = lys	AGA = arg
	AUG = met	ACG = thr	AAG = lys	AGG = arg
G	GUU = val	GCU = ala	GAU = asp	GGU = gly
	GUC = val	GCC = ala	GAC = asp	GGC = gly
	GUA = val	GCA = ala	GAA = glu	GGA = gly
	GUG = val	GCG = ala	GAG = glu	GGG = gly

Fig. 27 Genetic Code

3.3. Then I associate a specific *mineral*, **M** (like calcium), based on its *concentration level* **CL (Ex. 2.81mmol/l)** , with the *new nucleotide sequence S***1**
I derive **S1** from *nucleotide sequence* **S**, based on unique **CL** with *value* **V**. This value represents a unique concentration level of a certain M.

$$\mathbf{V= x.y1y2, \quad (Ex.\ 2.81)} \tag{7.2}$$

Where **x.y1y2,** is the number that represent the value V.

$$\mathbf{\Rightarrow CL=x.y1y2} \tag{7.3}$$

The resultant new ***nucleotide sequence S1*** will have the ***length L1***, **S1=L1**

$$\mathbf{L1=x*S+ (L-(y1+y2))} \tag{7.4}$$

Ex. **L1=2*GCAAGAGATAATTGT+(15-(8+1))=>**
L1=GCAAGAGATAATTGTGCAAGAGATAATTGT + (6 nucleotide)=>
L1=GCAAGAGATAATTGTGCAAGAGATAATTGT +GCAAGA=>
S1=GCAAGAGATAATTGTGCAAGAGATAATTGTGCAAGA

$$\mathbf{\Rightarrow \quad S1=x*S+ (L-(y1+y2))} \tag{7.5}$$

S1 will constitute the ***unique primer sequence*** (private key).
Step 4: According to CDMB, using an information conversion program Stefani encodes the medical records of an individual in DNA data strand flanked by unique primer sequences S1 and mixes it among other decoy DNA strands.
4.1: According to CDMB during the process of transcription Stefani cuts out the introns from the data-encoded DNA, resulting in *Encryption key 1*. **E1**= starting and pattern codes of introns, **=>C1=E1(P),** where **P** is *plaintext* and **C** is the *ciphertext*.
4.2: Stefani translates the resulted spliced form of the data
E2= the codon amino acids mapping => **C=E2(C1).**
4.3: Stefani obtains the data-encoded protein after the translation process.
Step 5: Stefani sends to Otto through a public channel the encoded form of the data.
Step 6: Otto then obtains the unique primer sequences that mark the beginning and the end of secret data DNA strand hidden among the decoy strands.

Comments: *Otto will use programs that perform the reverse processes as those performed by Stefani: simulating the transcription, splicing, and translation per the Central Dogma of Molecular Biology (CDMB).*
Step 7 Otto uses the key E2 to recover the mRNA form of the data from protein form of the data. *Decryption key* ***D1****=E2 => P1=D1(C).*
Step 8: Otto recovers the DNA form of the data in the reverse order as Stefani encrypted it. *Decryption key* ***D2****=E1=> P=D2(P1).*
Step 9: In this last step, Otto uses the information conversion program and reads the medical record of the individual, (Hodorogea, Vaida, 2008).
In order to use this data for the decryption algorithm Otto needs to know the correct start and end coordinates related to frames for each chromosome used in DNA encryption.

Nowadays, the software applications dedicated to different domains must respect many security elements. The DNA cryptography is used to implement security facilities for software applications, including the use of blood analyses to generate encryption keys. I provide my independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB).

8. Deriving DNA Cryptographic Keys

8.1 Deriving Asymmetric DNA Cryptographic Keys

I came with the novel idea for which I developed an algorithm, implemented and tested: I used Bioinformatics and the mathematics of Evolutionary Models based on mathematics of the Probability Theory for deriving the asymmetric cryptographic keys for use in public-key cryptography called, DNA Cryptographic Keys for use in the DNA Encryption (DNAE), based on the Central Dogma of Molecular Biology (CDMB).

The derivation of asymmetric cryptographic keys is based on the mathematical functions of probability theory and the complexity to compute the key for a cryptanalyst is NP complete.

The complexity of breaking the algorithm with brute force attack is proportional with the quantity of possible keys depending exponential of the key length, [27].

In my case the length of the key is *n* and the complexity of breaking by brute force attack is: $O(2^n)$

My algorithm implementation steps:

- **1st Step of implementation:**
  I came with the novel idea to develop an algorithm, for extracting all DNA coding sequences for all genes from Human Genome

- **2nd Step of implementation:**
  I developed, implemented and tested the algorithm proposed in first step I extracted all DNA coding sequences for all genes from Human Genome.
  I came with the novel idea to develop an algorithm which I implemented and tested: I extracted all DNA coding sequences with the same function ***(orthologus)***, during evolution as in Human Genome, from genomes of six more species.
  For the exemplification in the thesis I used only the genomes of two more species: ***Taurus Genome***, and ***Dog Genome*** labeled in genomics as ***bosTau*** (Taurus) and ***Can***(Dog).
  Orhtologus coding regions have the same function in all related species (Human, Taurus, Dog), during evolution and selection process.

- **3rd Step of implementation:**
  Multiple alignments provide a way to compute evolutionary distances needed to compute the philogenetic tree.
  I developed and implemented a software pipeline for multiple alignments for the DNA coding regions of the same gene from two more chosen genomes with respect to Extracted Human DNA Coding Sequences.
  In order to perform multiple alignment I need the most accurate tool for the multiple sequence alignment ProbCons is the best tool to perform the multiple alignments with the best accuracy but to make use of ProbCons I

need to calculate and compute best parameter set of the data I will use in the multiple sequences alignments.
The parameter set used by ProbCons for multiple sequences alignments must be calculated based on the extracted orthologus DNA coding sequences in chosen genomes: Human, Dog and Taurus, [60].

- **4th Step of implementation:** *Calculating the best parameter set for performing multiple alignments.*
  I implemented a software tool to train ProbCons tool in order to obtain the best parameter set which I use in the 3rd step for performing multiple alignments.
  The best alignment for coding regions (DNA Cryptographic Keys Sequences), of the same gene, from related chosen species, with respect to Human DNA can be realized only with a trained best parameter set of ProbCons tool. The software uses my obtained coding sequences from 1st and 2nd step of implementation.
  ProbCons tool, a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy.

- **5th Step of implementation**
  I developed and implemented software to calculate the Evolutionary Model Philogenetic tree related to my chosen species and the branches length during evolution for every chosen species, [60].
  Golding and Felsenstein (1990), Halpern and Bruno, (1998) have shown that mutation limit of the standard Kimura-Ohta Theory, one can uniquely determine substitution rates in terms of the mutation rates and the equilibrium frequencies *w.* In particular, if *r* is the rate of substitution from a base a to a base b at position *i*, μ is the rate of mutation from a to b and *w* is the equilibrium frequency of nucleotide i, at this position, [25], [48], [83].

- **6th Step of implementation**
  I developed and implemented a software pipeline by which I computed the asymmetric ***DNA Public/Private Keys.*** Public-key algorithms are based on mathematical functions, rather than on substitution and permutation and involve the use of two separate keys in contrast to symmetric encryption.
  In Table 3, Second Colum (C2) model represents the computed *DNA Public Keys*, with respect to Colum C1, assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, (Table 3) and represents the public keys.

- **7th Step of implementation**
  I developed and implemented a software pipeline by which I computed the ***DNA Private Keys.*** The third Colum (C3) assumes that at a given position, the substitution rates are altered during due to specific selection preferences for a certain base. The last Colum is the ratio C3/C2 and represents the private key, (Table 3).

Description of the 1st Step of implementation

I proposed a novel idea and I developed an algorithm where I use the mathematics of Evolutionary Models Based on probability theory to implement an algorithm for Deriving the DNA Cryptographic Keys for the use in public key encryption system.

Mathematical Metrics for the Encryption System:

Modern Cryptography is based on fundamental results of mathematics such as probability theory, information theory, theory of numbers and complexity of algorithms.

An encryption system in modern cryptography is defined as a structure of five elements (8.7)*(P, C, K, E, D)*, [29].

Where P is the space of the text in readable format:

$$P = \{pt / pt \in T^*\} \text{ and } T=\{0,1\} \tag{8.1}$$

K is the space of encryption keys $k \in K$

The encryption function dependent of keys and encryption algorithm:

$$E_k : P \to C, E_k = \{e_k / e_k(pt) = c\,\mathrm{t}\} \tag{8.2}$$

The decryption function dependent of keys and decryption algorithm D

$$D_k : C \to P, D_k = \{d_k / d_k(e_k(pt)) = t \forall t \in P\} \tag{8.3}$$

The space of encrypted messages:

$$C = \{ct / \exists k \in K, a \in P, ct = E_k(a)\} \tag{8.4}$$

Shanon defined exactly the mathematical model for the Security of the Encryption System.

Cryptanalyst's work consists in determining the key K, the text T, or both at the same time.

1. **The entropy** of the encryption is a measure of the keys space equal to, [108]:

$$H(k) = \log_2 K \tag{8.5}$$

For a message of the length *n,* the quantity of the different keys which decipher the encrypted text in the texts language is determined with the equation, [108]:

$$2^{H(k)-nD} - 1 \tag{8.6}$$

Shannon defined the unicity distance *U*, which is called unicity point:

$$U = \frac{H(k)}{D} \tag{8.7}$$

2. The unicity distance is a probabilistic metrics allowing evaluating the minimal quantity of encrypted text for which decryption through brute force attack exists probably only one decryption method. As greater the unicity distance better is the encryption system, [106].

3. **The theory of complexity** ensures the analyses methodology of computing complexity for different encryption algorithms.

The complexity of an algorithm is determined by the computing power needed for algorithm execution characterized by two parameters:

T –temporal complexity

S –space complexity

Both parameters are presented as a function of variable *n*, where *n* is the measure of input data. Existing algorithm has a polynomial complexity:

$$O(t^{f(n)}) \tag{8.8}$$

Where t is a constant greater than 1 and *f(n)* is a polynomial function of variable n.

The exponential algorithms which have this complexity where c is a constant and f(n) grows faster than the constant and slower than a linear function are called suprapolinominal algorithms.

If the cryptanalysis algorithm for breaking an encryption algorithm has a suprapolinomial complexity the encryption algorithm is ideal, [29].

It is impossible to demonstrate if it has not been founded a cryptanalysis algorithm with polynomial complexity.

The complexity of breaking an algorithm with brute force attack is proportional with the quantity of possible keys depending exponential of the key length.

In my case the length of the key is n and the complexity of breaking by brute force attack is:

The derivation of cryptographic keys is based on the mathematical function of probability theory and the complexity to compute the key is NP complete.

Mathematical metrics for molecular evolution

A DNA sequence is a representative of its species and the branch length corresponds to the evolutionary distance in time, [96].

Selective pressure is the differences between sequences. Thomas Hunt Morgan defined genes as a genetic unit encoding a ***trait***.

The ***locus*** is the position on the chromosome where a particular trait is encoded

I describe the ***population at a time t by the frequencies***:

$$P(A,t) \text{ and } P(a,t)=1-P(A,t) \tag{8.10}$$

To calculate *P(A,t+1)given P(A,t) where:*

$$P(A,t+1)=G \tag{8.11}$$

G represents the change (Reproduction, mutation, etc)

The selection after t generation is:

$$P(A,t)=\frac{F(A)^t P(A,0)}{F(A)^t P(A,0)+F(a)^t P(a,0)}=\frac{(\frac{F(A)}{F(a)})^t P(A,0)}{\left(\frac{F(A)}{F(a)}\right)^t P(A,0)+P(a,0)} \tag{8.12}$$

Fisher's Theorem of Selection

For different genotypes g with frequency *P(g,t)* in the population time t:

f(g) is the fitness of genotype *g* and the average fitness at a time *t* is given by

$$P(g,t+1)=\frac{f(g)P(g,t)}{\sum_{g'} f(g')P(g',t)} \tag{8.13}$$

The fitness at a time (t+1) is

$$\langle f \rangle_{t+1} = \sum_g f(g)P(g,t+1) = \frac{\sum_g [f(g)]^2 \, P(g,t)}{\sum_{g'} \left[f(g') \right] \, P(g',t)} = \frac{\langle f^2 \rangle_t}{\langle f \rangle_t} \tag{8.14}$$

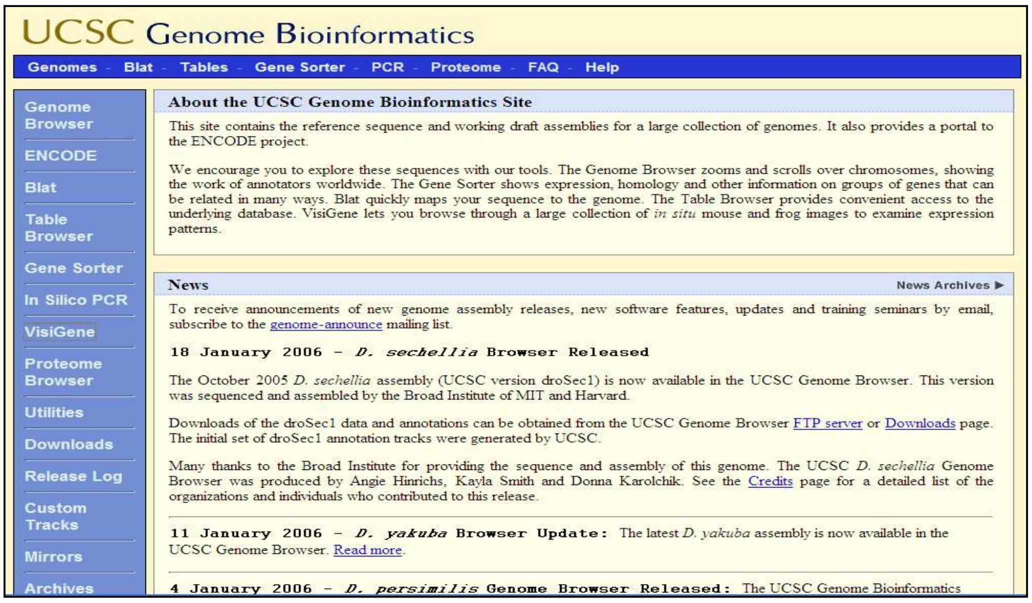
Theorem:

$$\langle f \rangle_{t+1} - \langle f \rangle_t = \frac{\langle f^2 \rangle_t - \langle f \rangle_t^2}{\langle f \rangle_t} = \frac{\mathrm{var}_t(f)}{\langle f \rangle} \tag{8.15}$$

To extract from the Human Genome all Coding DNA Sequences it is assumed that the many loci in the chromosomes of an organism code for a particular trait and all are under selection.

For each locus is a type of highest fitness that population has. There are also multiple alleles present in the population (***polymorphism)*** at almost every locus. The Polymorphism is maintained by selection.

One hundred of amino acids correspond to 300 base pairs of DNA and there are 4 billion base pairs in the DNA of a mammalian genome making the process of extracting from public available database all DNA coding sequences for all genes from Human genome a very complex and difficult one as the data are limited and available only in the .txt format, (Fig.28).



About the UCSC Genome Bioinformatics Site

This site contains the reference sequence and working draft assemblies for a large collection of genomes. It also provides a portal to the ENCODE project.

We encourage you to explore these sequences with our tools. The Genome Browser zooms and scrolls over chromosomes, showing the work of annotators worldwide. The Gene Sorter shows expression, homology and other information on groups of genes that can be related in many ways. Blat quickly maps your sequence to the genome. The Table Browser provides convenient access to the underlying database. VisiGene lets you browse through a large collection of *in situ* mouse and frog images to examine expression patterns.

News

To receive announcements of new genome assembly releases, new software features, updates and training seminars by email, subscribe to the genome-announce mailing list.

18 January 2006 - *D. sechellia* Browser Released

The October 2005 *D. sechellia* assembly (UCSC version droSec1) is now available in the UCSC Genome Browser. This version was sequenced and assembled by the Broad Institute of MIT and Harvard.

Downloads of the droSec1 data and annotations can be obtained from the UCSC Genome Browser FTP server or Downloads page. The initial set of droSec1 annotation tracks were generated by UCSC.

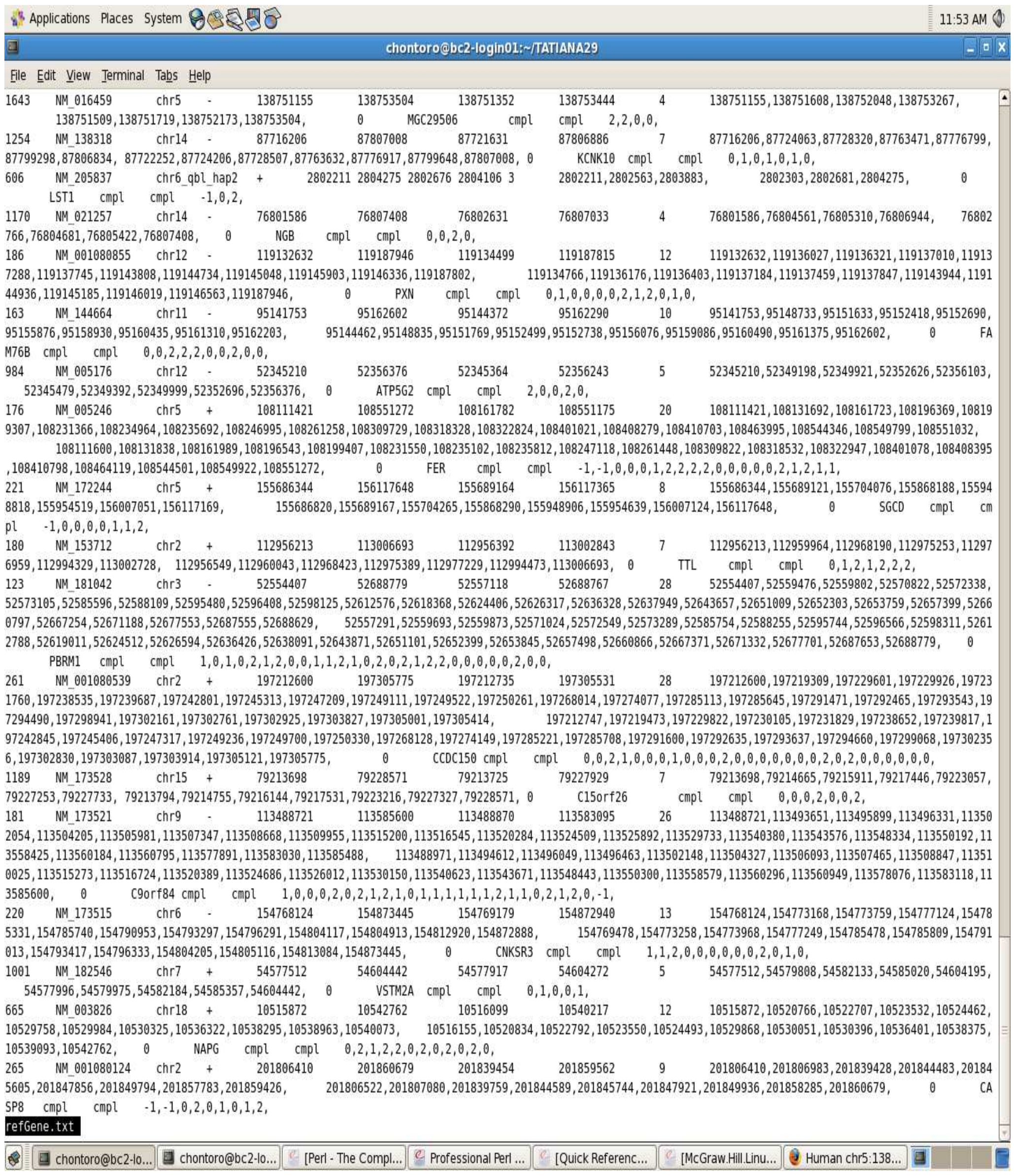
Many thanks to the Broad Institute for providing the sequence and assembly of this genome. The UCSC *D. sechellia* Genome Browser was produced by Angie Hinrichs, Kayla Smith and Donna Karolchik. See the Credits page for a detailed list of the organizations and individuals who contributed to this release.

11 January 2006 - *D. yakuba* Browser Update: The latest *D. yakuba* assembly is now available in the UCSC Genome Browser. Read more.

4 January 2006 - *D. persimilis* Genome Browser Released: The UCSC Genome Bioinformatics

Fig. 28 UCSCC Genome Browser

In my case the only available data for HumanGenome (hg18) is *refGene.txt*, which only contains the framesets for human chromosomes, the total number of coding sequences in the given chromosome with the position where the coding regions (exons) start and the position where they end, plus the correspondent gene where the given chromosome is located plus the location of the direction of the fourth DNA Strand, positive or negative twisted DNA, (Fig. 29).



```
1643    NM_016459       chr5    -       138751155       138753504       138751352       138753444       4       138751155,138751608,138752048,138753267,
        138751509,138751719,138752173,138753504,        0       MGC29506        cmpl    cmpl    2,2,0,0,
1254    NM_138318       chr14   -       87716206        87807008        87721631        87806886        7       87716206,87724063,87728320,87763471,87776799,
87799298,87806834, 87722252,87724206,87728507,87763632,87776917,87799648,87807008, 0       KCNK10  cmpl    cmpl    0,1,0,1,0,1,0,
606     NM_205837       chr6_qbl_hap2   +       2802211 2804275 2802676 2804106 3       2802211,2802563,2803883,        2802303,2802681,2804275,        0
       LST1    cmpl    cmpl    -1,0,2,
1170    NM_021257       chr14   -       76801586        76807408        76802631        76807033        4       76801586,76804561,76805310,76806944,    76802
766,76804681,76805422,76807408,    0       NGB     cmpl    cmpl    0,0,2,0,
186     NM_001080855    chr12   -       119132632       119187946       119134499       119187815       12      119132632,119136027,119136321,119137010,11913
7288,119137745,119143808,119144734,119145048,119145903,119146336,119187802,        119134766,119136176,119136403,119137184,119137459,119137847,119143944,1191
44936,119145185,119146019,119146563,119187946,        0       PXN     cmpl    cmpl    0,1,0,0,0,0,2,1,2,0,1,0,
163     NM_144664       chr11   -       95141753        95162602        95144372        95162290        10      95141753,95148733,95151633,95152418,95152690,
95155876,95158930,95160435,95161310,95162203,      95144462,95148835,95151769,95152499,95152738,95156076,95159086,95160490,95161375,95162602,      0       FA
M76B  cmpl    cmpl    0,0,2,2,2,0,0,2,0,0,
984     NM_005176       chr12   -       52345210        52356376        52345364        52356243        5       52345210,52349198,52349921,52352626,52356103,
   52345479,52349392,52349999,52352696,52356376,   0       ATP5G2  cmpl    cmpl    2,0,0,2,0,
176     NM_005246       chr5    +       108111421       108551272       108161782       108551175       20      108111421,108131692,108161723,108196369,10819
9307,108231366,108234964,108235692,108246995,108261258,108309729,108318328,108322824,108401021,108408279,108410703,108463995,108544346,108549799,108551032,
        108111600,108131838,108161989,108196543,108199407,108231550,108235102,108235812,108247118,108261448,108309822,108318532,108322947,108401078,108408395
,108410798,108464119,108544501,108549922,108551272,        0       FER     cmpl    cmpl    -1,-1,0,0,0,1,2,2,2,2,0,0,0,0,0,2,1,2,1,1,
221     NM_172244       chr5    +       155686344       156117648       155689164       156117365       8       155686344,155689121,155704076,155868188,15594
8818,155954519,156007051,156117169,        155686820,155689167,155704265,155868290,155948906,155954639,156007124,156117648,        0       SGCD    cmpl    cm
pl    -1,0,0,0,0,1,1,2,
180     NM_153712       chr2    +       112956213       113006693       112956392       113002843       7       112956213,112959964,112968190,112975253,11297
6959,112994329,113002728,  112956549,112960043,112968423,112975389,112977229,112994473,113006693,  0       TTL     cmpl    cmpl    0,1,2,1,2,2,2,
123     NM_181042       chr3    -       52554407        52688779        52557118        52688767        28      52554407,52559476,52559802,52570822,52572338,
52573105,52585596,52588109,52595480,52596408,52598125,52612576,52618368,52624406,52626317,52636328,52637949,52643657,52651009,52652303,52653759,52657399,5266
0797,52667254,52671188,52677553,52687555,52688629,    52557291,52559693,52559873,52571024,52572549,52573289,52585754,52588255,52595744,52596566,52598311,5261
2788,52619011,52624512,52626594,52636426,52638091,52643871,52651101,52652399,52653845,52657498,52660866,52667371,52671332,52677701,52687653,52688779,    0
       PBRM1   cmpl    cmpl    1,0,1,0,2,1,2,0,0,1,1,2,1,0,2,0,2,1,2,2,0,0,0,0,0,2,0,0,
261     NM_001080539    chr2    +       197212600       197305775       197212735       197305531       28      197212600,197219309,197229601,197229926,19723
1760,197238535,197239687,197242801,197245313,197247209,197249111,197249522,197250261,197268014,197274077,197285113,197285645,197291471,197292465,197293543,19
7294490,197298941,197302161,197302761,197302925,197303827,197305001,197305414,        197212747,197219473,197229822,197230105,197231829,197238652,197239817,1
97242845,197245406,197247317,197249236,197249700,197250330,197268128,197274149,197285221,197285708,197291600,197292635,197293637,197294660,197299068,19730235
6,197302830,197303087,197303914,197305121,197305775,        0       CCDC150 cmpl    cmpl    0,0,2,1,0,0,0,1,0,0,0,2,0,0,0,0,0,0,0,2,0,2,0,0,0,0,0,0,
1189    NM_173528       chr15   +       79213698        79228571        79213725        79227929        7       79213698,79214665,79215911,79217446,79223057,
79227253,79227733, 79213794,79214755,79216144,79217531,79223216,79227327,79228571, 0       C15orf26        cmpl    cmpl    0,0,0,2,0,0,2,
181     NM_173521       chr9    -       113488721       113585600       113488870       113583095       26      113488721,113493651,113495899,113496331,11350
2054,113504205,113505981,113507347,113508668,113509955,113515200,113516545,113520284,113524509,113525892,113529733,113540380,113543576,113548334,113550192,11
3558425,113560184,113560795,113577891,113583030,113585488,    113488971,113494612,113496049,113496463,113502148,113504327,113506093,113507465,113508847,11351
0025,113515273,113516724,113520389,113524686,113526012,113530150,113540623,113543671,113548443,113550300,113558579,113560296,113560949,113578076,113583118,11
3585600,    0       C9orf84 cmpl    cmpl    1,0,0,0,2,0,2,1,2,1,0,1,1,1,1,1,1,2,1,1,0,2,1,2,0,-1,
220     NM_173515       chr6    -       154768124       154873445       154769179       154872940       13      154768124,154773168,154773759,154777124,15478
5331,154785740,154790953,154793297,154796291,154804117,154804913,154812920,154872888,      154769478,154773258,154773968,154777249,154785478,154785809,154791
013,154793417,154796333,154804205,154805116,154813084,154873445,      0       CNKSR3  cmpl    cmpl    1,1,2,0,0,0,0,0,0,2,0,1,0,
1001    NM_182546       chr7    +       54577512        54604442        54577917        54604272        5       54577512,54579808,54582133,54585020,54604195,
   54577996,54579975,54582184,54585357,54604442,   0       VSTM2A  cmpl    cmpl    0,1,0,0,1,
665     NM_003826       chr18   +       10515872        10542762        10516099        10540217        12      10515872,10520766,10522707,10523532,10524462,
10529758,10529984,10530325,10536322,10538295,10538963,10540073,    10516155,10520834,10522792,10523550,10524493,10529868,10530051,10530396,10536401,10538375,
10539093,10542762,    0       NAPG    cmpl    cmpl    0,2,1,2,2,0,2,0,2,0,2,0,
265     NM_001080124    chr2    +       201806410       201860679       201839454       201859562       9       201806410,201806983,201839428,201844483,20184
5605,201847856,201849794,201857783,201859426,      201806522,201807080,201839759,201844589,201845744,201847921,201849936,201858285,201860679,      0       CA
SP8   cmpl    cmpl    -1,-1,0,2,0,1,0,1,2,
refGene.txt
```

Fig. 29 Full set of human chromosomes with correspondent numbers of coding regions

A big constraint and problem is that in the given .txt file the start and end coordinates of coding regions are not given correct with respect to frames of DNA and based only on this coordinates it is impossible to extract the DNA coding sequences for all human genome coding regions, making the process of extracting (from public available database) all DNA coding sequences for all genes from Human genome a difficult and complex process.

As an example I present the first top gene from Fig.29 labeled **gene NM_016459. The gene is located in chromosome Nr. 5(chr 5) and contains 4 exons** (coding regions), given to me by the number 4 which I colored in red, the blue numbers indicate to me the approximate coordinate positions where exons start and the green numbers indicates me the approximate coordinates positions where exons end. This means that the start and ending positions are not correlated to frames of reading the DNA. The genetic code reads DNA sequences in groups of three base pairs, which means that a double-stranded DNA molecule can read in any of six possible reading frames, three in the forward direction and three in the reverse.

The pink minus sign shows the location of given exons on the fourth DNA twisted strand.

Ex:
1643 **NM_016459 chr5 - 138751155 138753504 138751352 138753444 4 138751155, 138751608, 138752048, 138753267**

To extract the correct DNA coding sequences from Human Genome I needed to develop and implement a new algorithm to determine the correct start and end coordinates correlated to DNA reading frames.

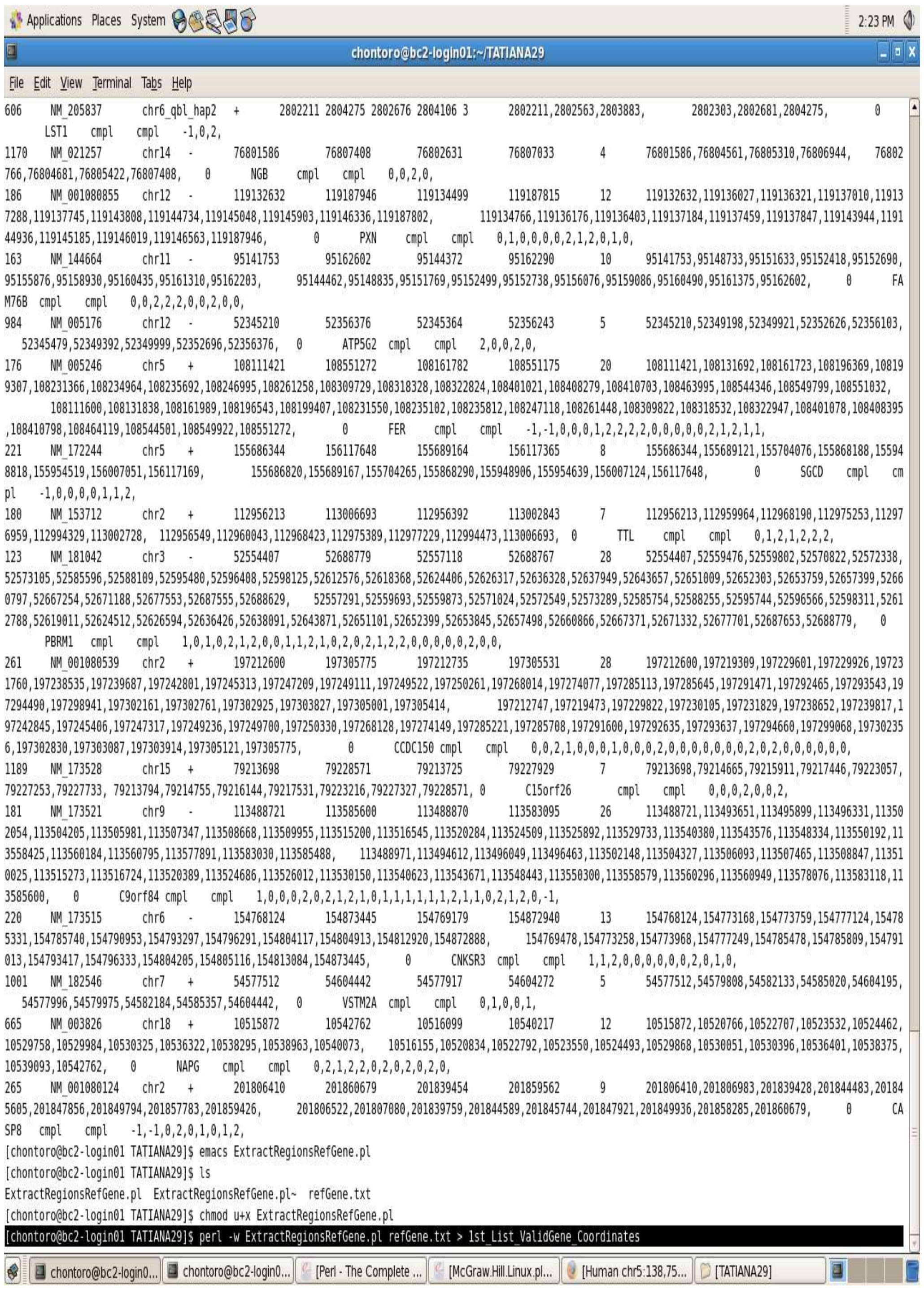
- I developed an algorithm to extract the correct DNA coding sequences start and end coordinates correlated to DNA reading frames.

-For this I implemented the software named: **ExtractRegionsRefGene.pl,** (Fig.30),

The software extracted in a flat file named *1st_List_ValidGene_Coordinates,* (Fig.32) the correct DNA coding sequences start and end coordinates correlated to DNA reading frames for Human genome in the following format:

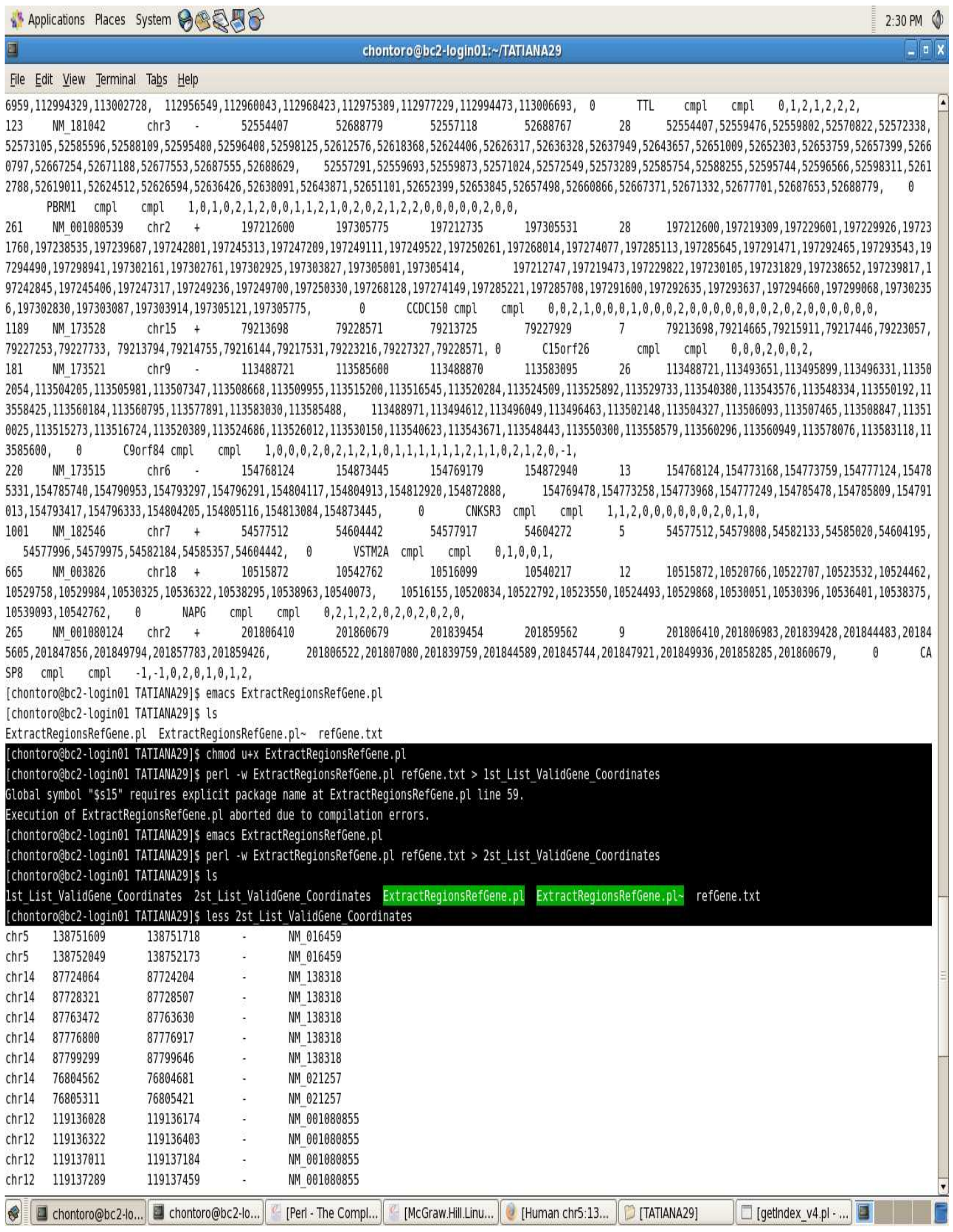
Chromosome number, Start Coordinate, End Coordinate, Strand, Gene Label (fig.32)

Ex: Chr5 87724064 87724204 - NM_130318



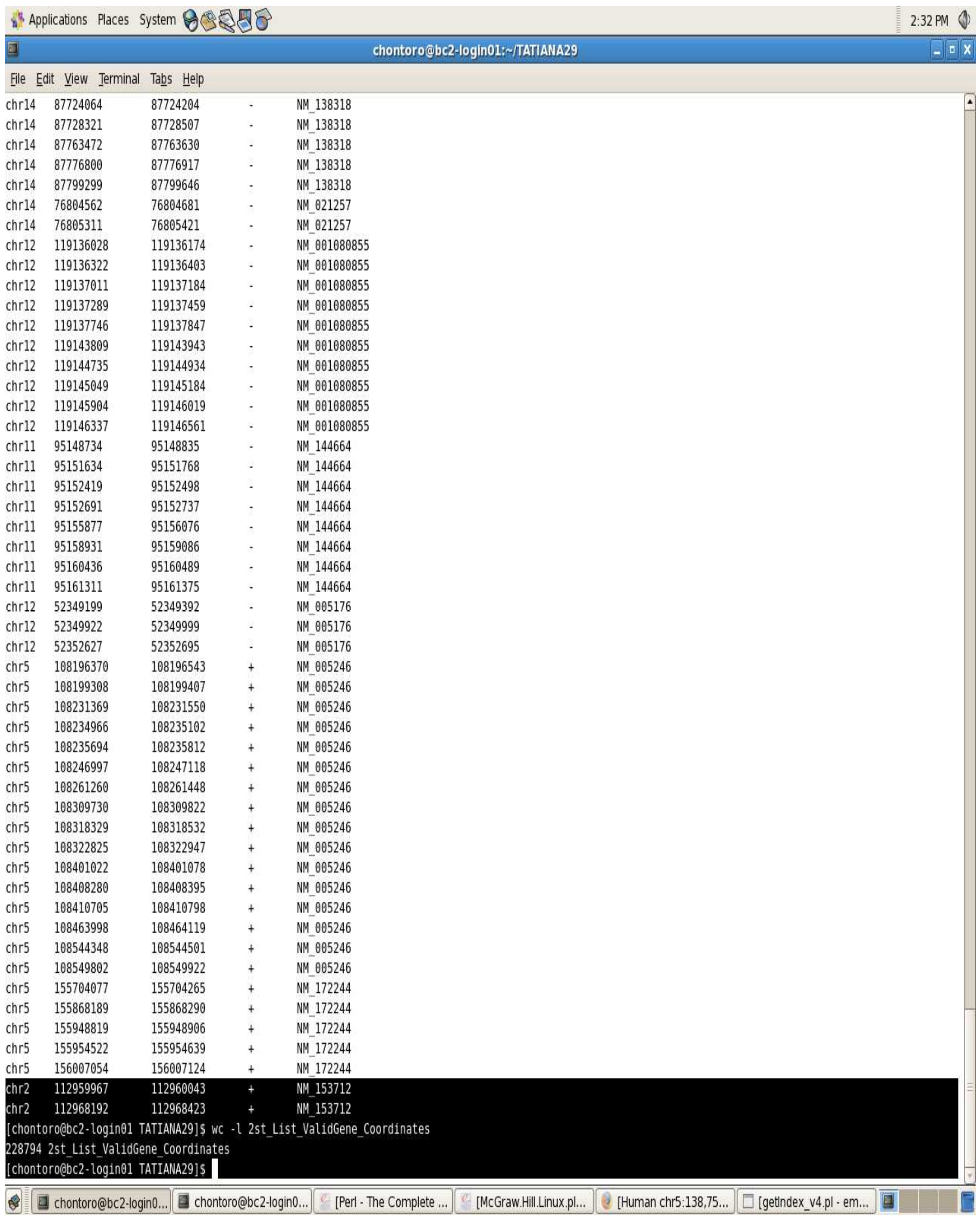
```
606     NM_205837       chr6_qbl_hap2   +       2802211 2804275 2802676 2804106 3       2802211,2802563,2803883,        2802303,2802681,2804275,        0
        LST1    cmpl    cmpl    -1,0,2,
1170    NM_021257       chr14   -       76801586        76807408        76802631        76807033        4       76801586,76804561,76805310,76806944,    76802
766,76804681,76805422,76807408,    0       NGB     cmpl    cmpl    0,0,2,0,
186     NM_001080855    chr12   -       119132632       119187946       119134499       119187815       12      119132632,119136027,119136321,119137010,11913
7288,119137745,119143808,119144734,119145048,119145903,119146336,119187802,        119134766,119136176,119136403,119137184,119137459,119137847,119143944,1191
44936,119145185,119146019,119146563,119187946,        0       PXN     cmpl    cmpl    0,1,0,0,0,0,2,1,2,0,1,0,
163     NM_144664       chr11   -       95141753        95162602        95144372        95162290        10      95141753,95148733,95151633,95152418,95152690,
95155876,95158930,95160435,95161310,95162203,      95144462,95148835,95151769,95152499,95152738,95156076,95159086,95160490,95161375,95162602,      0       FA
M76B  cmpl    cmpl    0,0,2,2,2,0,0,2,0,0,
984     NM_005176       chr12   -       52345210        52356376        52345364        52356243        5       52345210,52349198,52349921,52352626,52356103,
   52345479,52349392,52349999,52352696,52356376,   0       ATP5G2  cmpl    cmpl    2,0,0,2,0,
176     NM_005246       chr5    +       108111421       108551272       108161782       108551175       20      108111421,108131692,108161723,108196369,10819
9307,108231366,108234964,108235692,108246995,108261258,108309729,108318328,108322824,108401021,108408279,108410703,108463995,108544346,108549799,108551032,
        108111600,108131838,108161989,108196543,108199407,108231550,108235102,108235812,108247118,108261448,108309822,108318532,108322947,108401078,108408395
,108410798,108464119,108544501,108549922,108551272,        0       FER     cmpl    cmpl    -1,-1,0,0,0,1,2,2,2,2,0,0,0,0,0,2,1,2,1,1,
221     NM_172244       chr5    +       155686344       156117648       155689164       156117365       8       155686344,155689121,155704076,155868188,15594
8818,155954519,156007051,156117169,        155686820,155689167,155704265,155868290,155948906,155954639,156007124,156117648,        0       SGCD    cmpl    cm
pl    -1,0,0,0,0,1,1,2,
180     NM_153712       chr2    +       112956213       113006693       112956392       113002843       7       112956213,112959964,112968190,112975253,11297
6959,112994329,113002728,  112956549,112960043,112968423,112975389,112977229,112994473,113006693,  0       TTL     cmpl    cmpl    0,1,2,1,2,2,2,
123     NM_181042       chr3    -       52554407        52688779        52557118        52688767        28      52554407,52559476,52559802,52570822,52572338,
52573105,52585596,52588109,52595480,52596408,52598125,52612576,52618368,52624406,52626317,52636328,52637949,52643657,52651009,52652303,52653759,52657399,5266
0797,52667254,52671188,52677553,52687555,52688629,    52557291,52559693,52559873,52571024,52572549,52573289,52585754,52588255,52595744,52596566,52598311,5261
2788,52619011,52624512,52626594,52636426,52638091,52643871,52651101,52652399,52653845,52657498,52660866,52667371,52671332,52677701,52687653,52688779,    0
        PBRM1   cmpl    cmpl    1,0,1,0,2,1,2,0,0,1,1,2,1,0,2,0,2,1,2,2,0,0,0,0,0,2,0,0,
261     NM_001080539    chr2    +       197212600       197305775       197212735       197305531       28      197212600,197219309,197229601,197229926,19723
1760,197238535,197239687,197242801,197245313,197247209,197249111,197249522,197250261,197268014,197274077,197285113,197285645,197291471,197292465,197293543,19
7294490,197298941,197302161,197302761,197302925,197303827,197305001,197305414,        197212747,197219473,197229822,197230105,197231829,197238652,197239817,1
97242845,197245406,197247317,197249236,197249700,197250330,197268128,197274149,197285221,197285708,197291600,197292635,197293637,197294660,197299068,19730235
6,197302830,197303087,197303914,197305121,197305775,        0       CCDC150 cmpl    cmpl    0,0,2,1,0,0,0,1,0,0,0,2,0,0,0,0,0,0,0,2,0,2,0,0,0,0,0,0,
1189    NM_173528       chr15   +       79213698        79228571        79213725        79227929        7       79213698,79214665,79215911,79217446,79223057,
79227253,79227733, 79213794,79214755,79216144,79217531,79223216,79227327,79228571, 0       C15orf26        cmpl    cmpl    0,0,0,2,0,0,2,
181     NM_173521       chr9    -       113488721       113585600       113488870       113583095       26      113488721,113493651,113495899,113496331,11350
2054,113504205,113505981,113507347,113508668,113509955,113515200,113516545,113520284,113524509,113525892,113529733,113540380,113543576,113548334,113550192,11
3558425,113560184,113560795,113577891,113583030,113585488,    113488971,113494612,113496049,113496463,113502148,113504327,113506093,113507465,113508847,11351
0025,113515273,113516724,113520389,113524686,113526012,113530150,113540623,113543671,113548443,113550300,113558579,113560296,113560949,113578076,113583118,11
3585600,    0       C9orf84 cmpl    cmpl    1,0,0,0,2,0,2,1,2,1,0,1,1,1,1,1,1,2,1,1,0,2,1,2,0,-1,
220     NM_173515       chr6    -       154768124       154873445       154769179       154872940       13      154768124,154773168,154773759,154777124,15478
5331,154785740,154790953,154793297,154796291,154804117,154804913,154812920,154872888,      154769478,154773258,154773968,154777249,154785478,154785809,154791
013,154793417,154796333,154804205,154805116,154813084,154873445,      0       CNKSR3  cmpl    cmpl    1,1,2,0,0,0,0,0,0,2,0,1,0,
1001    NM_182546       chr7    +       54577512        54604442        54577917        54604272        5       54577512,54579808,54582133,54585020,54604195,
   54577996,54579975,54582184,54585357,54604442,   0       VSTM2A  cmpl    cmpl    0,1,0,0,1,
665     NM_003826       chr18   +       10515872        10542762        10516099        10540217        12      10515872,10520766,10522707,10523532,10524462,
10529758,10529984,10530325,10536322,10538295,10538963,10540073,    10516155,10520834,10522792,10523550,10524493,10529868,10530051,10530396,10536401,10538375,
10539093,10542762,    0       NAPG    cmpl    cmpl    0,2,1,2,2,0,2,0,2,0,2,0,
265     NM_001080124    chr2    +       201806410       201860679       201839454       201859562       9       201806410,201806983,201839428,201844483,20184
5605,201847856,201849794,201857783,201859426,      201806522,201807080,201839759,201844589,201845744,201847921,201849936,201858285,201860679,      0       CA
SP8   cmpl    cmpl    -1,-1,0,2,0,1,0,1,2,
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ ls
ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ chmod u+x ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ perl -w ExtractRegionsRefGene.pl refGene.txt > 1st_List_ValidGene_Coordinates
```

Fig. 30 Creating the list with valid coordinates



```
6959,112994329,113002728,  112956549,112960043,112968423,112975389,112977229,112994473,113006693,  0       TTL     cmpl    cmpl    0,1,2,1,2,2,2,
123     NM_181042       chr3    -       52554407        52688779        52557118        52688767        28      52554407,52559476,52559802,52570822,52572338,
52573105,52585596,52588109,52595480,52596408,52598125,52612576,52618368,52624406,52626317,52636328,52637949,52643657,52651009,52652303,52653759,52657399,5266
0797,52667254,52671188,52677553,52687555,52688629,    52557291,52559693,52559873,52571024,52572549,52573289,52585754,52588255,52595744,52596566,52598311,5261
2788,52619011,52624512,52626594,52636426,52638091,52643871,52651101,52652399,52653845,52657498,52660866,52667371,52671332,52677701,52687653,52688779,    0
       PBRM1   cmpl    cmpl    1,0,1,0,2,1,2,0,0,1,1,2,1,0,2,0,2,1,2,2,0,0,0,0,0,2,0,0,
261     NM_001080539    chr2    +       197212600       197305775       197212735       197305531       28      197212600,197219309,197229601,197229926,19723
1760,197238535,197239687,197242801,197245313,197247209,197249111,197249522,197250261,197268014,197274077,197285113,197285645,197291471,197292465,197293543,19
7294490,197298941,197302161,197302761,197302925,197303827,197305001,197305414,        197212747,197219473,197229822,197230105,197231829,197238652,197239817,1
97242845,197245406,197247317,197249236,197249700,197250330,197268128,197274149,197285221,197285708,197291600,197292635,197293637,197294660,197299068,19730235
6,197302830,197303087,197303914,197305121,197305775,        0       CCDC150 cmpl    cmpl    0,0,2,1,0,0,0,1,0,0,0,2,0,0,0,0,0,0,0,2,0,2,0,0,0,0,0,0,
1189    NM_173528       chr15   +       79213698        79228571        79213725        79227929        7       79213698,79214665,79215911,79217446,79223057,
79227253,79227733, 79213794,79214755,79216144,79217531,79223216,79227327,79228571, 0       C15orf26        cmpl    cmpl    0,0,0,2,0,0,2,
181     NM_173521       chr9    -       113488721       113585600       113488870       113583095       26      113488721,113493651,113495899,113496331,11350
2054,113504205,113505981,113507347,113508668,113509955,113515200,113516545,113520284,113524509,113525892,113529733,113540380,113543576,113548334,113550192,11
3558425,113560184,113560795,113577891,113583030,113585488,    113488971,113494612,113496049,113496463,113502148,113504327,113506093,113507465,113508847,11351
0025,113515273,113516724,113520389,113524686,113526012,113530150,113540623,113543671,113548443,113550300,113558579,113560296,113560949,113578076,113583118,11
3585600,    0       C9orf84 cmpl    cmpl    1,0,0,0,2,0,2,1,2,1,0,1,1,1,1,1,1,2,1,1,0,2,1,2,0,-1,
220     NM_173515       chr6    -       154768124       154873445       154769179       154872940       13      154768124,154773168,154773759,154777124,15478
5331,154785740,154790953,154793297,154796291,154804117,154804913,154812920,154872888,      154769478,154773258,154773968,154777249,154785478,154785809,154791
013,154793417,154796333,154804205,154805116,154813084,154873445,      0       CNKSR3  cmpl    cmpl    1,1,2,0,0,0,0,0,0,2,0,1,0,
1001    NM_182546       chr7    +       54577512        54604442        54577917        54604272        5       54577512,54579808,54582133,54585020,54604195,
   54577996,54579975,54582184,54585357,54604442,   0       VSTM2A  cmpl    cmpl    0,1,0,0,1,
665     NM_003826       chr18   +       10515872        10542762        10516099        10540217        12      10515872,10520766,10522707,10523532,10524462,
10529758,10529984,10530325,10536322,10538295,10538963,10540073,    10516155,10520834,10522792,10523550,10524493,10529868,10530051,10530396,10536401,10538375,
10539093,10542762,    0       NAPG    cmpl    cmpl    0,2,1,2,2,0,2,0,2,0,2,0,
265     NM_001080124    chr2    +       201806410       201860679       201839454       201859562       9       201806410,201806983,201839428,201844483,20184
5605,201847856,201849794,201857783,201859426,      201806522,201807080,201839759,201844589,201845744,201847921,201849936,201858285,201860679,      0       CA
SP8   cmpl    cmpl    -1,-1,0,2,0,1,0,1,2,
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ ls
ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ chmod u+x ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ perl -w ExtractRegionsRefGene.pl refGene.txt > 1st_List_ValidGene_Coordinates
Global symbol "$s15" requires explicit package name at ExtractRegionsRefGene.pl line 59.
Execution of ExtractRegionsRefGene.pl aborted due to compilation errors.
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ perl -w ExtractRegionsRefGene.pl refGene.txt > 2st_List_ValidGene_Coordinates
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates  2st_List_ValidGene_Coordinates  ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ less 2st_List_ValidGene_Coordinates
chr5    138751609       138751718       -       NM_016459
chr5    138752049       138752173       -       NM_016459
chr14   87724064        87724204        -       NM_138318
chr14   87728321        87728507        -       NM_138318
chr14   87763472        87763630        -       NM_138318
chr14   87776800        87776917        -       NM_138318
chr14   87799299        87799646        -       NM_138318
chr14   76804562        76804681        -       NM_021257
chr14   76805311        76805421        -       NM_021257
chr12   119136028       119136174       -       NM_001080855
chr12   119136322       119136403       -       NM_001080855
chr12   119137011       119137184       -       NM_001080855
chr12   119137289       119137459       -       NM_001080855
```

Fig. 31 Correct data set related to each human chromosome

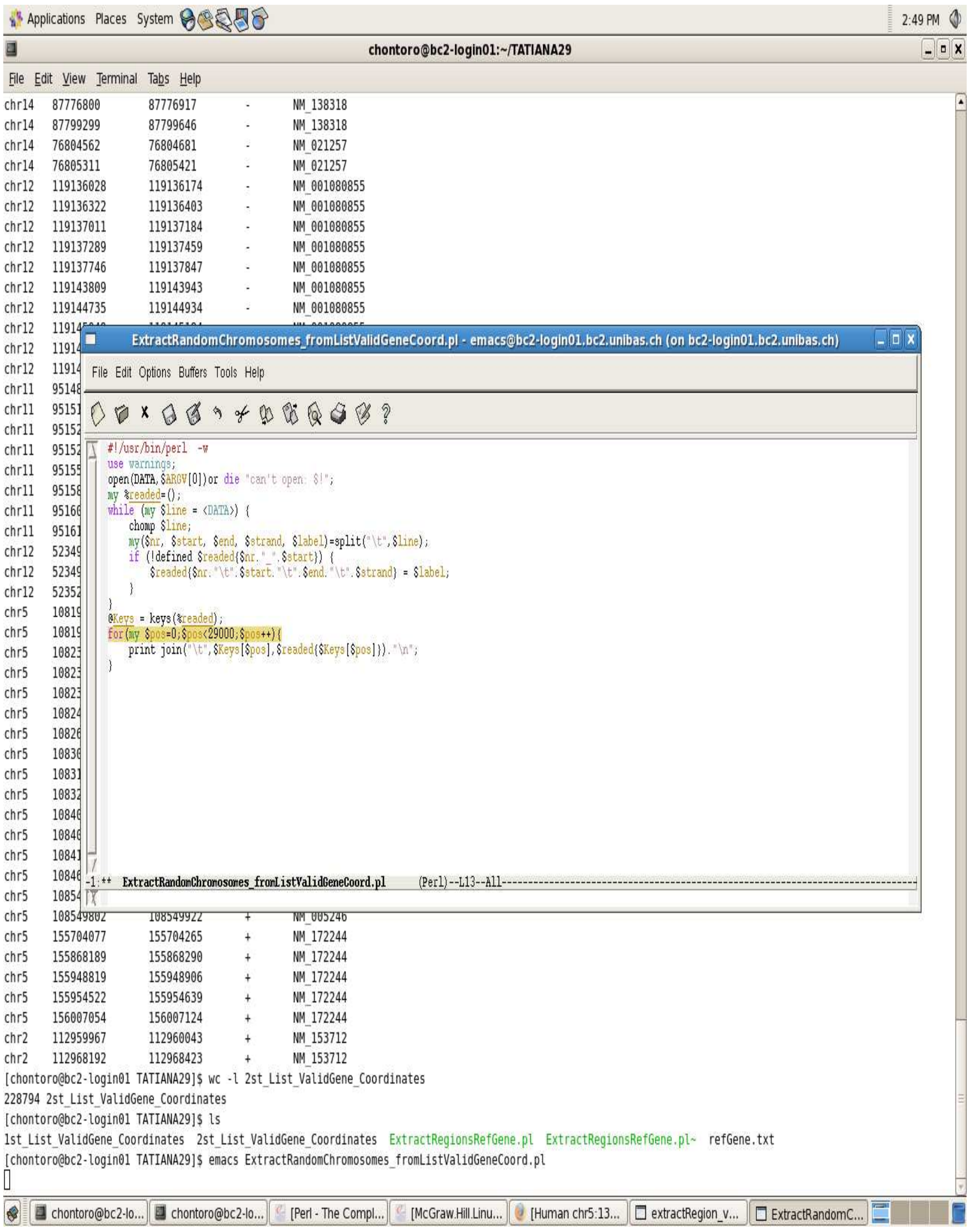


```
chr14   87724064        87724204        -       NM_138318
chr14   87728321        87728507        -       NM_138318
chr14   87763472        87763630        -       NM_138318
chr14   87776800        87776917        -       NM_138318
chr14   87799299        87799646        -       NM_138318
chr14   76804562        76804681        -       NM_021257
chr14   76805311        76805421        -       NM_021257
chr12   119136028       119136174       -       NM_001080855
chr12   119136322       119136403       -       NM_001080855
chr12   119137011       119137184       -       NM_001080855
chr12   119137289       119137459       -       NM_001080855
chr12   119137746       119137847       -       NM_001080855
chr12   119143809       119143943       -       NM_001080855
chr12   119144735       119144934       -       NM_001080855
chr12   119145049       119145184       -       NM_001080855
chr12   119145904       119146019       -       NM_001080855
chr12   119146337       119146561       -       NM_001080855
chr11   95148734        95148835        -       NM_144664
chr11   95151634        95151768        -       NM_144664
chr11   95152419        95152498        -       NM_144664
chr11   95152691        95152737        -       NM_144664
chr11   95155877        95156076        -       NM_144664
chr11   95158931        95159086        -       NM_144664
chr11   95160436        95160489        -       NM_144664
chr11   95161311        95161375        -       NM_144664
chr12   52349199        52349392        -       NM_005176
chr12   52349922        52349999        -       NM_005176
chr12   52352627        52352695        -       NM_005176
chr5    108196370       108196543       +       NM_005246
chr5    108199308       108199407       +       NM_005246
chr5    108231369       108231550       +       NM_005246
chr5    108234966       108235102       +       NM_005246
chr5    108235694       108235812       +       NM_005246
chr5    108246997       108247118       +       NM_005246
chr5    108261260       108261448       +       NM_005246
chr5    108309730       108309822       +       NM_005246
chr5    108318329       108318532       +       NM_005246
chr5    108322825       108322947       +       NM_005246
chr5    108401022       108401078       +       NM_005246
chr5    108408280       108408395       +       NM_005246
chr5    108410705       108410798       +       NM_005246
chr5    108463998       108464119       +       NM_005246
chr5    108544348       108544501       +       NM_005246
chr5    108549802       108549922       +       NM_005246
chr5    155704077       155704265       +       NM_172244
chr5    155868189       155868290       +       NM_172244
chr5    155948819       155948906       +       NM_172244
chr5    155954522       155954639       +       NM_172244
chr5    156007054       156007124       +       NM_172244
chr2    112959967       112960043       +       NM_153712
chr2    112968192       112968423       +       NM_153712
[chontoro@bc2-login01 TATIANA29]$ wc -l 2st_List_ValidGene_Coordinates
228794 2st_List_ValidGene_Coordinates
[chontoro@bc2-login01 TATIANA29]$
```

Fig. 32 Human data set related to each human chromosome

Having the correct start and end coordinates for each coding sequence of entire human genome I proceeded to develop an algorithm which I implemented: to extract the DNA coding sequences for all coding regions from entire human genome.

My file with all extracting coding regions for the entire Human genome contains, (Fig.32) highlighted in black: 228794 lines in printable format 3751 pages a very big amount of data meaning that the amount of data for corresponding coding sequences will be enormous.



```
chr14   87776800     87776917     -      NM_138318
chr14   87799299     87799646     -      NM_138318
chr14   76804562     76804681     -      NM_021257
chr14   76805311     76805421     -      NM_021257
chr12   119136028    119136174    -      NM_001080855
chr12   119136322    119136403    -      NM_001080855
chr12   119137011    119137184    -      NM_001080855
chr12   119137289    119137459    -      NM_001080855
chr12   119137746    119137847    -      NM_001080855
chr12   119143809    119143943    -      NM_001080855
chr12   119144735    119144934    -      NM_001080855
chr12   11914
chr12   11914
chr12   11914
chr11   95148
chr11   95151
chr11   95152
chr11   95152
chr11   95155
chr11   95158
chr11   95160
chr11   95161
chr12   52349
chr12   52349
chr12   52352
chr5    10819
chr5    10819
chr5    10823
chr5    10823
chr5    10823
chr5    10824
chr5    10826
chr5    10830
chr5    10831
chr5    10832
chr5    10840
chr5    10840
chr5    10841
chr5    10846
chr5    10854
chr5    108549802    108549922    +      NM_005246
chr5    155704077    155704265    +      NM_172244
chr5    155868189    155868290    +      NM_172244
chr5    155948819    155948906    +      NM_172244
chr5    155954522    155954639    +      NM_172244
chr5    156007054    156007124    +      NM_172244
chr2    112959967    112960043    +      NM_153712
chr2    112968192    112968423    +      NM_153712
[chontoro@bc2-login01 TATIANA29]$ wc -l 2st_List_ValidGene_Coordinates
228794 2st_List_ValidGene_Coordinates
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates  2st_List_ValidGene_Coordinates  ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRandomChromosomes_fromListValidGeneCoord.pl
```

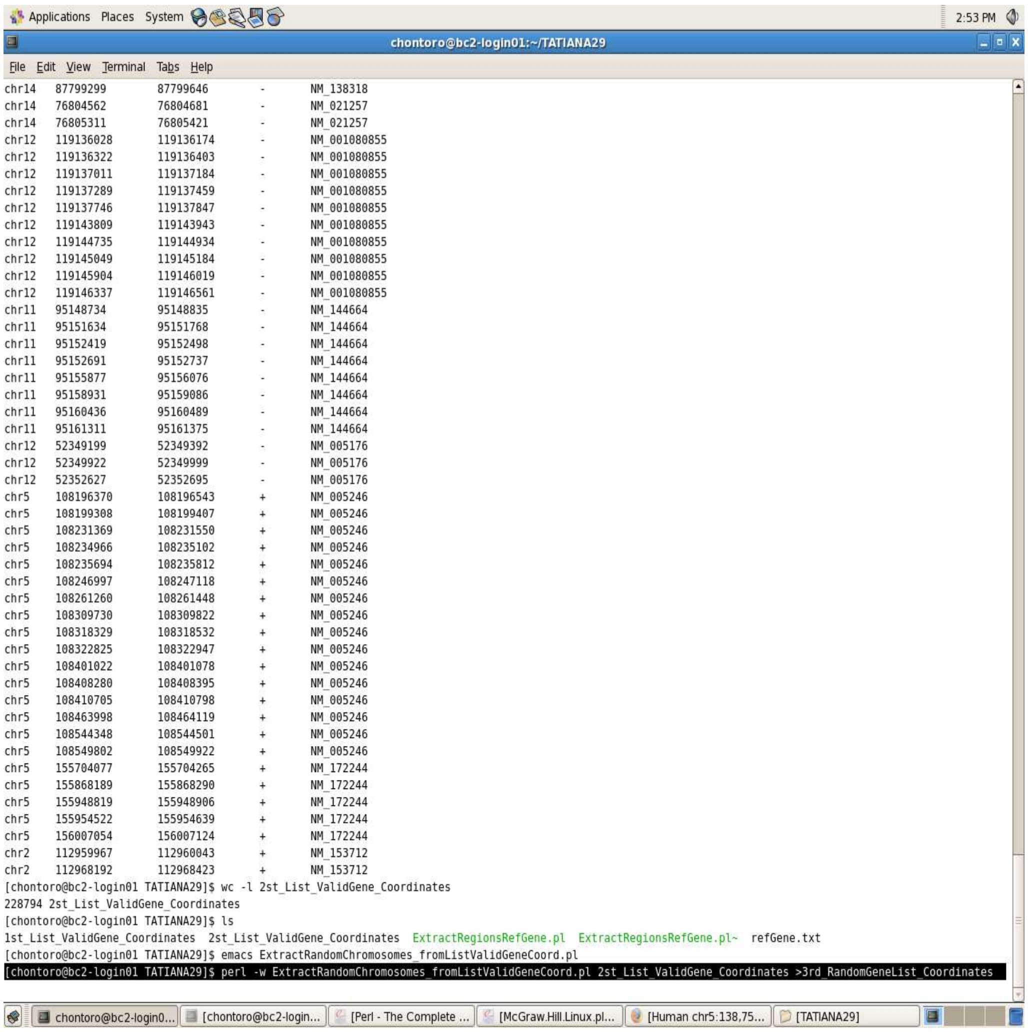
ExtractRandomChromosomes_fromListValidGeneCoord.pl - emacs@bc2-login01.bc2.unibas.ch (on bc2-login01.bc2.unibas.ch)

```perl
#!/usr/bin/perl  -w
use warnings;
open(DATA,$ARGV[0])or die "can't open: $!";
my %readed=();
while (my $line = <DATA>) {
    chomp $line;
    my($nr, $start, $end, $strand, $label)=split("\t",$line);
    if (!defined $readed{$nr."_".$start}) {
        $readed{$nr."\t".$start."\t".$end."\t".$strand} = $label;
    }
}
@Keys = keys(%readed);
for(my $pos=0;$pos<29000;$pos++){
    print join("\t",$Keys[$pos],$readed{$Keys[$pos]})."\n";
}
```

```
-1:**  ExtractRandomChromosomes_fromListValidGeneCoord.pl     (Perl)--L13--All------------
```

Fig. 33 Removing the repeated chromosomes

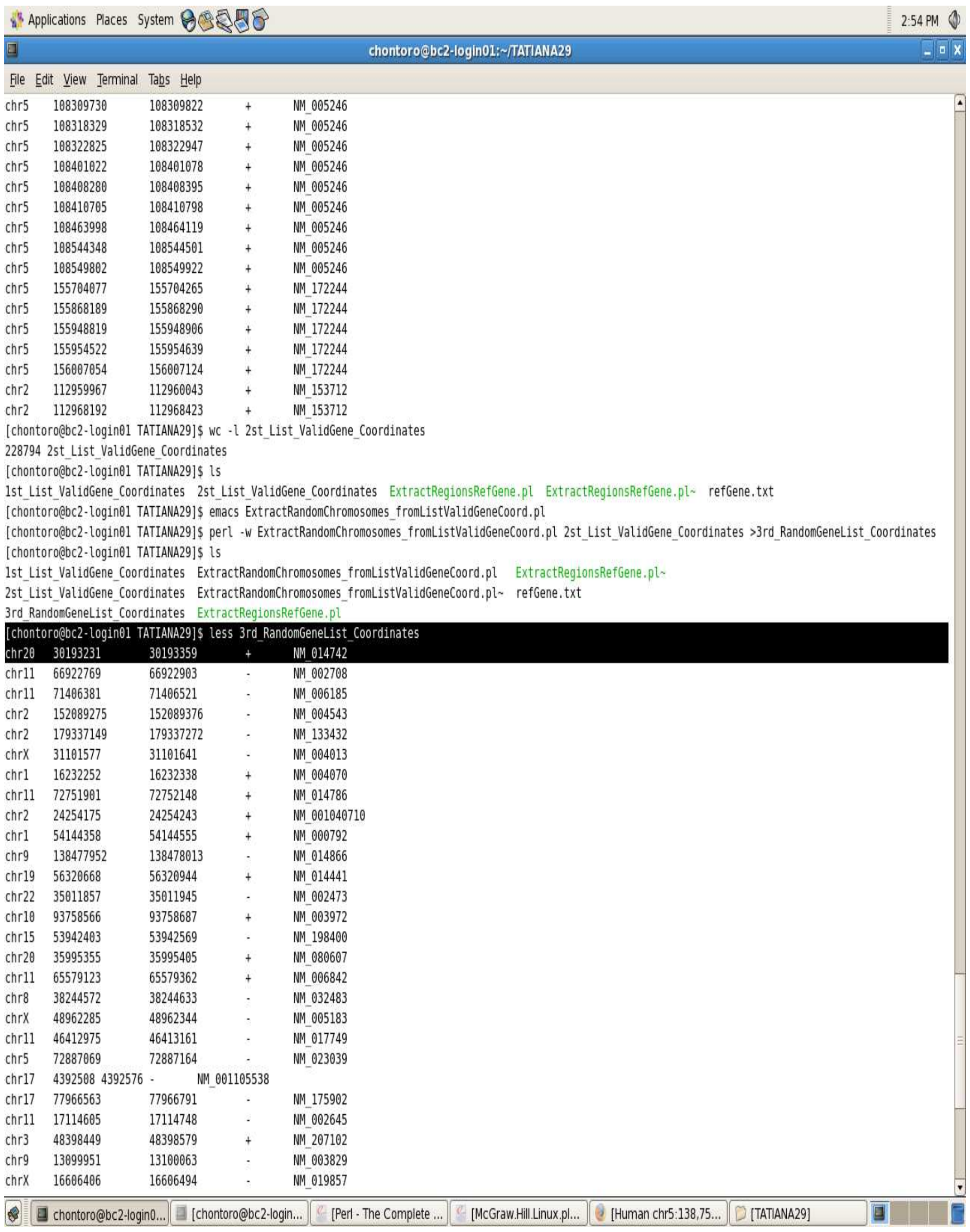
For simplifying the problem I decided to remove from the list of 3751 pages the repeated chromosomes of different genes with the same start and end coordinates. I developed a script named ExtractRandomChromosomes.pl which removed the repeated chromosomes of different genes with the same start and end coordinates, (Fig.33).



```
chr14   87799299        87799646        -       NM_138318
chr14   76804562        76804681        -       NM_021257
chr14   76805311        76805421        -       NM_021257
chr12   119136028       119136174       -       NM_001080855
chr12   119136322       119136403       -       NM_001080855
chr12   119137011       119137184       -       NM_001080855
chr12   119137289       119137459       -       NM_001080855
chr12   119137746       119137847       -       NM_001080855
chr12   119143809       119143943       -       NM_001080855
chr12   119144735       119144934       -       NM_001080855
chr12   119145049       119145184       -       NM_001080855
chr12   119145904       119146019       -       NM_001080855
chr12   119146337       119146561       -       NM_001080855
chr11   95148734        95148835        -       NM_144664
chr11   95151634        95151768        -       NM_144664
chr11   95152419        95152498        -       NM_144664
chr11   95152691        95152737        -       NM_144664
chr11   95155877        95156076        -       NM_144664
chr11   95158931        95159086        -       NM_144664
chr11   95160436        95160489        -       NM_144664
chr11   95161311        95161375        -       NM_144664
chr12   52349199        52349392        -       NM_005176
chr12   52349922        52349999        -       NM_005176
chr12   52352627        52352695        -       NM_005176
chr5    108196370       108196543       +       NM_005246
chr5    108199308       108199407       +       NM_005246
chr5    108231369       108231550       +       NM_005246
chr5    108234966       108235102       +       NM_005246
chr5    108235694       108235812       +       NM_005246
chr5    108246997       108247118       +       NM_005246
chr5    108261260       108261448       +       NM_005246
chr5    108309730       108309822       +       NM_005246
chr5    108318329       108318532       +       NM_005246
chr5    108322825       108322947       +       NM_005246
chr5    108401022       108401078       +       NM_005246
chr5    108408280       108408395       +       NM_005246
chr5    108410705       108410798       +       NM_005246
chr5    108463998       108464119       +       NM_005246
chr5    108544348       108544501       +       NM_005246
chr5    108549802       108549922       +       NM_005246
chr5    155704077       155704265       +       NM_172244
chr5    155868189       155868290       +       NM_172244
chr5    155948819       155948906       +       NM_172244
chr5    155954522       155954639       +       NM_172244
chr5    156007054       156007124       +       NM_172244
chr2    112959967       112960043       +       NM_153712
chr2    112968192       112968423       +       NM_153712
[chontoro@bc2-login01 TATIANA29]$ wc -l 2st_List_ValidGene_Coordinates
228794 2st_List_ValidGene_Coordinates
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates  2st_List_ValidGene_Coordinates  ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRandomChromosomes_fromListValidGeneCoord.pl
[chontoro@bc2-login01 TATIANA29]$ perl -w ExtractRandomChromosomes_fromListValidGeneCoord.pl 2st_List_ValidGene_Coordinates >3rd_RandomGeneList_Coordinates
```

Fig. 34 Creating the random genes list

As a result I obtained a list which I named: *3rd_RandomGeneList_Coordinates, (Fig.34)* with extracted *29*.000 random genes, (Fig. 35), from chromosomes for my father use for deriving of DNA Cryptographic Keys.



```
chr5    108309730       108309822       +       NM_005246
chr5    108318329       108318532       +       NM_005246
chr5    108322825       108322947       +       NM_005246
chr5    108401022       108401078       +       NM_005246
chr5    108408280       108408395       +       NM_005246
chr5    108410705       108410798       +       NM_005246
chr5    108463998       108464119       +       NM_005246
chr5    108544348       108544501       +       NM_005246
chr5    108549802       108549922       +       NM_005246
chr5    155704077       155704265       +       NM_172244
chr5    155868189       155868290       +       NM_172244
chr5    155948819       155948906       +       NM_172244
chr5    155954522       155954639       +       NM_172244
chr5    156007054       156007124       +       NM_172244
chr2    112959967       112960043       +       NM_153712
chr2    112968192       112968423       +       NM_153712
[chontoro@bc2-login01 TATIANA29]$ wc -l 2st_List_ValidGene_Coordinates
228794 2st_List_ValidGene_Coordinates
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates  2st_List_ValidGene_Coordinates  ExtractRegionsRefGene.pl  ExtractRegionsRefGene.pl~  refGene.txt
[chontoro@bc2-login01 TATIANA29]$ emacs ExtractRandomChromosomes_fromListValidGeneCoord.pl
[chontoro@bc2-login01 TATIANA29]$ perl -w ExtractRandomChromosomes_fromListValidGeneCoord.pl 2st_List_ValidGene_Coordinates >3rd_RandomGeneList_Coordinates
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates  ExtractRandomChromosomes_fromListValidGeneCoord.pl   ExtractRegionsRefGene.pl~
2st_List_ValidGene_Coordinates  ExtractRandomChromosomes_fromListValidGeneCoord.pl~  refGene.txt
3rd_RandomGeneList_Coordinates  ExtractRegionsRefGene.pl
[chontoro@bc2-login01 TATIANA29]$ less 3rd_RandomGeneList_Coordinates
chr20   30193231        30193359        +       NM_014742
chr11   66922769        66922903        -       NM_002708
chr11   71406381        71406521        -       NM_006185
chr2    152089275       152089376       -       NM_004543
chr2    179337149       179337272       -       NM_133432
chrX    31101577        31101641        -       NM_004013
chr1    16232252        16232338        +       NM_004070
chr11   72751901        72752148        +       NM_014786
chr2    24254175        24254243        +       NM_001040710
chr1    54144358        54144555        +       NM_000792
chr9    138477952       138478013       -       NM_014866
chr19   56320668        56320944        +       NM_014441
chr22   35011857        35011945        -       NM_002473
chr10   93758566        93758687        +       NM_003972
chr15   53942403        53942569        -       NM_198400
chr20   35995355        35995405        +       NM_080607
chr11   65579123        65579362        +       NM_006842
chr8    38244572        38244633        -       NM_032483
chrX    48962285        48962344        -       NM_005183
chr11   46412975        46413161        -       NM_017749
chr5    72887069        72887164        -       NM_023039
chr17   4392508 4392576 -       NM_001105538
chr17   77966563        77966791        -       NM_175902
chr11   17114605        17114748        -       NM_002645
chr3    48398449        48398579        +       NM_207102
chr9    13099951        13100063        -       NM_003829
chrX    16606406        16606494        -       NM_019857
```

Fig. 35 Random genes in the list

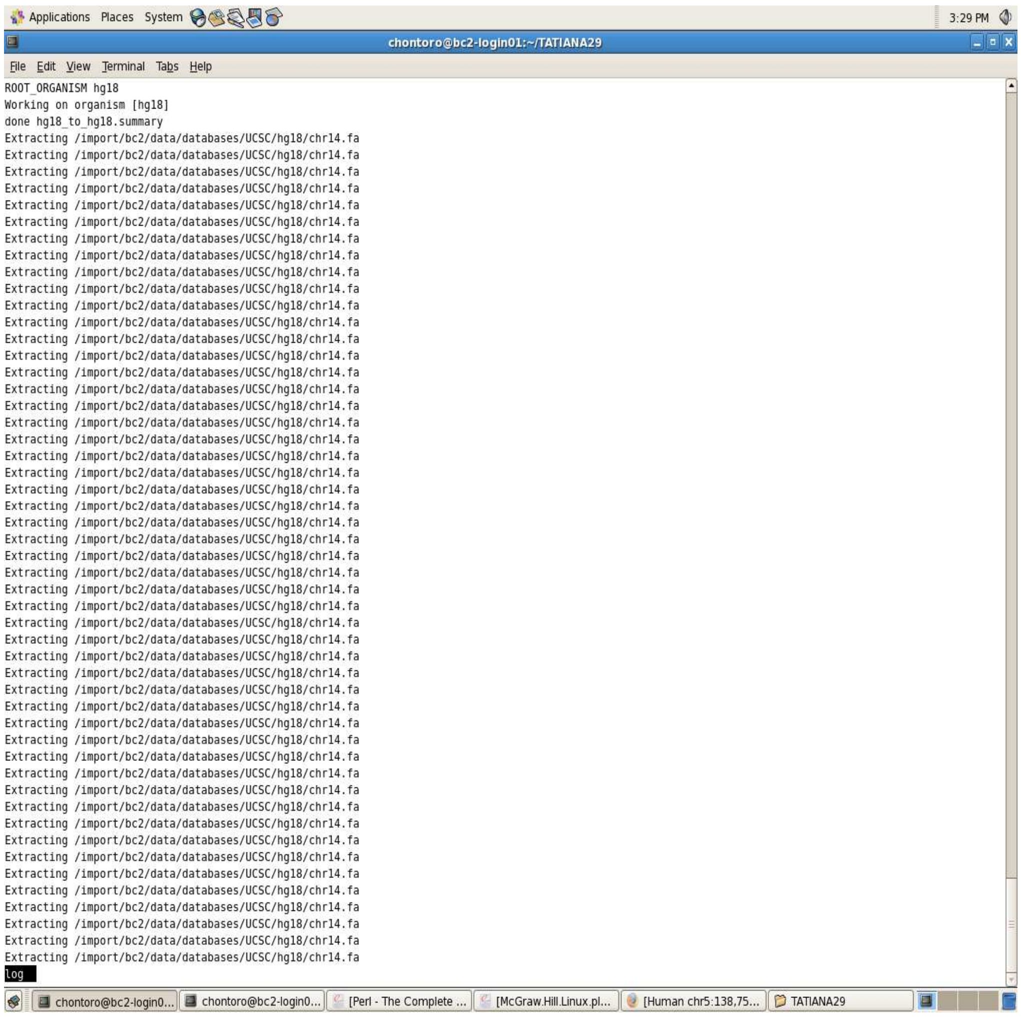
Description of the 2nd Step of implementation:

Based on obtained coordinates from the first step:

- I developed an algorithm to extract all orthologus DNA coding sequences for all genes, from three genomes: HUMAN Genome and related to human genome I used TAURUS Genome and DOG Genome.

-I developed and implemented the software *named pipeline.pl*. to extract all orthologus DNA coding sequences for all genes, from three genomes: HUMAN Genome and related to human genome I used TAURUS Genome and DOG Genome.

The tests were realized first for 7 Genomes, in the second step the tests were realized with six genomes) and three 3 species, with respect to Human sequences extracted in the 1st step. Orhtologus coding regions have and maintain the same function during evolution in all related species. (The process was realized with 7, 6 and 3 species).



```
ROOT_ORGANISM hg18
Working on organism [hg18]
done hg18_to_hg18.summary
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
log
```

Fig. 36 Extracting orthologus DNA coding sequences for all genes Log file

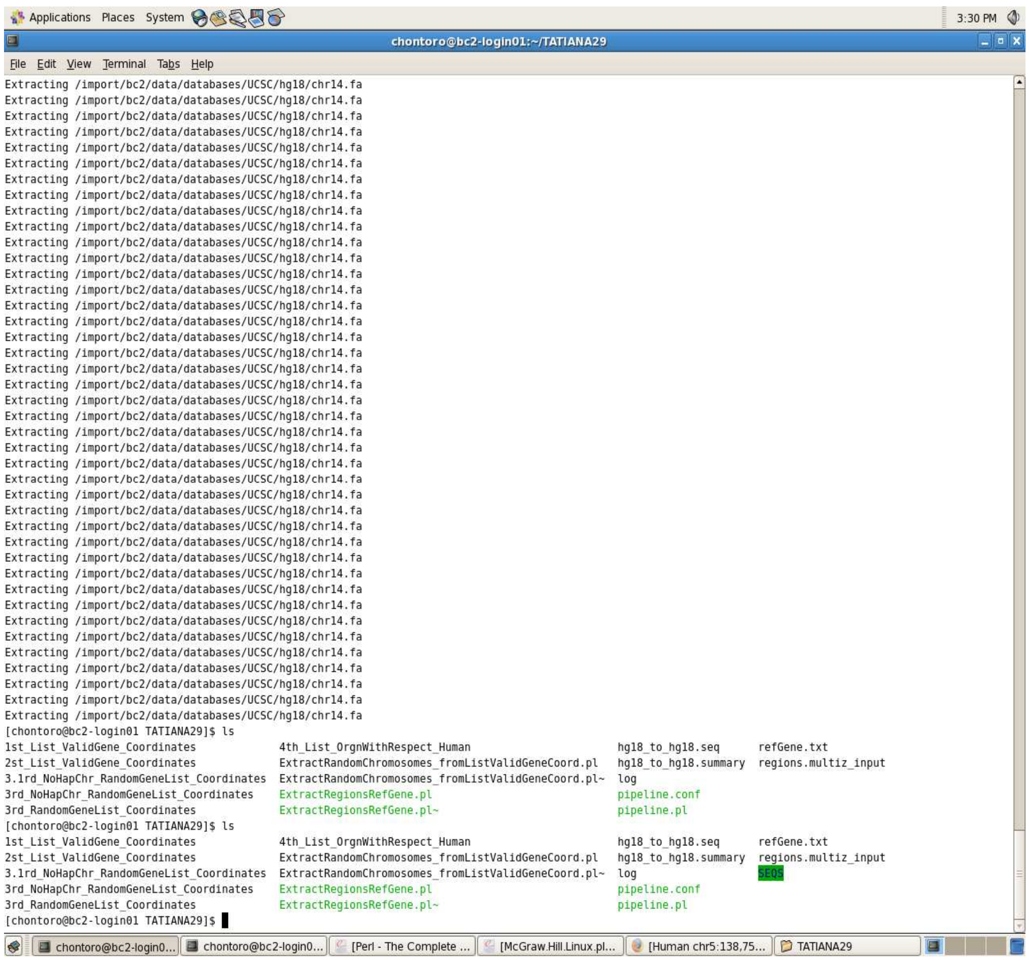
When the mutational rate μ is nonzero the fitness will be:

$$\langle F\rangle = F(A)P(A,\infty) + F(a)(P(a,\infty) = F(A)\left(1-\mu\frac{F(A)}{F(A)-F(a)}\right) + F(a)\mu\frac{F(A)}{F(A)-F(a)} + O(\mu^2) \quad (8.16)$$

The mutation load will be:

$$L = \frac{\langle F\rangle}{F(A)} = (1-\mu) = O(\mu^2) \quad (8.17)$$

The tests and the jobs to extract the sequences during tests were run at Biozentrum, Basel, Switzerland, in parallel on multiple clusters with enough computational power.



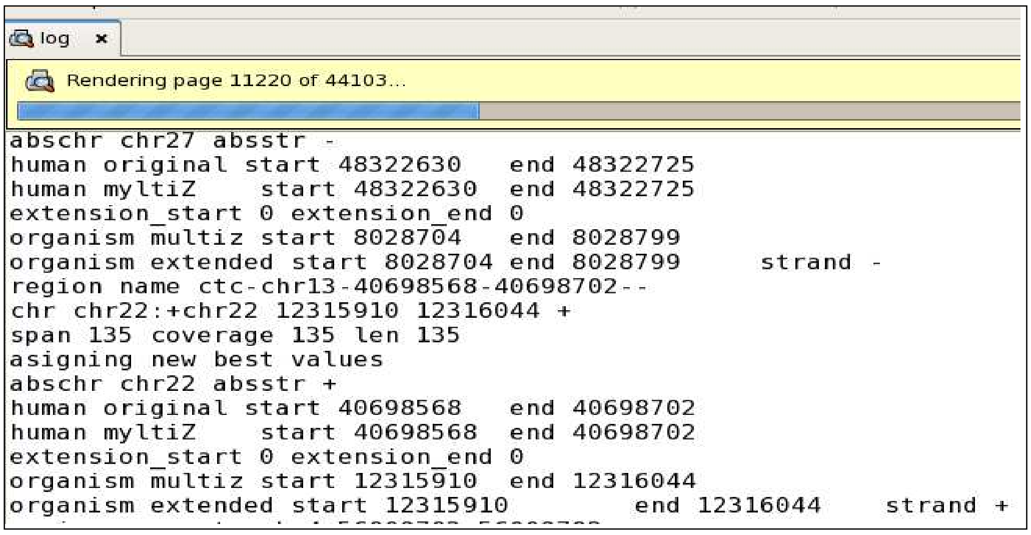
```
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
Extracting /import/bc2/data/databases/UCSC/hg18/chr14.fa
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates          4th_List_OrgnWithRespect_Human                    hg18_to_hg18.seq      refGene.txt
2st_List_ValidGene_Coordinates          ExtractRandomChromosomes_fromListValidGeneCoord.pl  hg18_to_hg18.summary  regions.multiz_input
3.1rd_NoHapChr_RandomGeneList_Coordinates ExtractRandomChromosomes_fromListValidGeneCoord.pl~ log
3rd_NoHapChr_RandomGeneList_Coordinates ExtractRegionsRefGene.pl                          pipeline.conf
3rd_RandomGeneList_Coordinates          ExtractRegionsRefGene.pl~                         pipeline.pl
[chontoro@bc2-login01 TATIANA29]$ ls
1st_List_ValidGene_Coordinates          4th_List_OrgnWithRespect_Human                    hg18_to_hg18.seq      refGene.txt
2st_List_ValidGene_Coordinates          ExtractRandomChromosomes_fromListValidGeneCoord.pl  hg18_to_hg18.summary  regions.multiz_input
3.1rd_NoHapChr_RandomGeneList_Coordinates ExtractRandomChromosomes_fromListValidGeneCoord.pl~ log                   SEQS
3rd_NoHapChr_RandomGeneList_Coordinates ExtractRegionsRefGene.pl                          pipeline.conf
3rd_RandomGeneList_Coordinates          ExtractRegionsRefGene.pl~                         pipeline.pl
[chontoro@bc2-login01 TATIANA29]$ 
```

Fig.37 Extracting the DNA sequencing

It took more than 24 hours to extract all DNA coding sequences for 29.000 genes.

We can see the head of our log file, (Fig. 36) during more than 24 hours process when our script was running and extracting the DNA sequences for 29.000 genes in SEQS directory, (Fig.37).

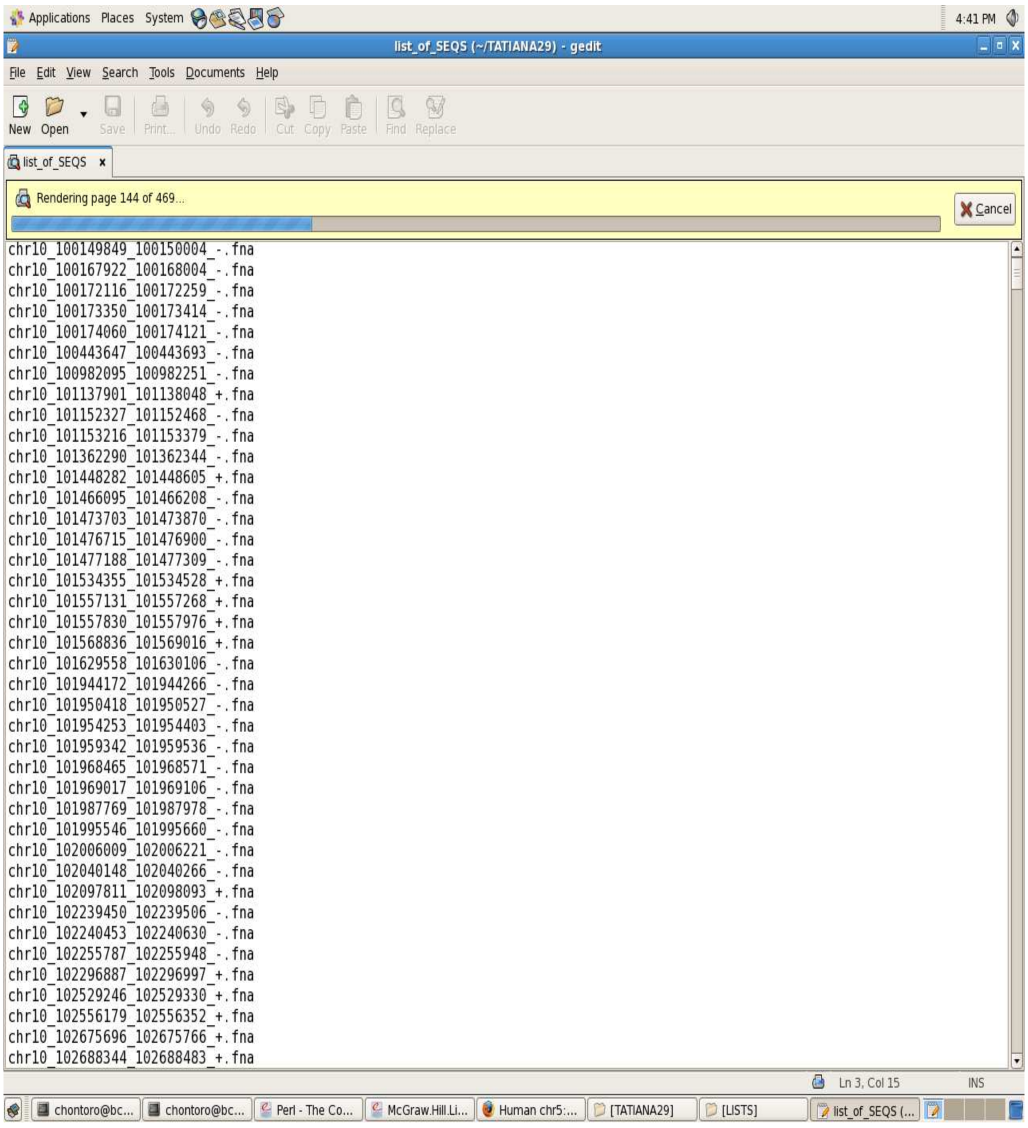
The screen shoot shows that for 29.000 records I extracted in printable format 44103 pages of all orthologus DNA coding sequences from my chosen 3 genomes: HUMAN, TAURUS, DOG, (Fig.38).



```
abschr chr27 absstr -
human original start 48322630   end 48322725
human myltiZ    start 48322630   end 48322725
extension_start 0 extension_end 0
organism multiz start 8028704   end 8028799
organism extended start 8028704 end 8028799     strand -
region name ctc-chr13-40698568-40698702--
chr chr22:+chr22 12315910 12316044 +
span 135 coverage 135 len 135
asigning new best values
abschr chr22 absstr +
human original start 40698568   end 40698702
human myltiZ    start 40698568   end 40698702
extension_start 0 extension_end 0
organism multiz start 12315910  end 12316044
organism extended start 12315910        end 12316044    strand +
```

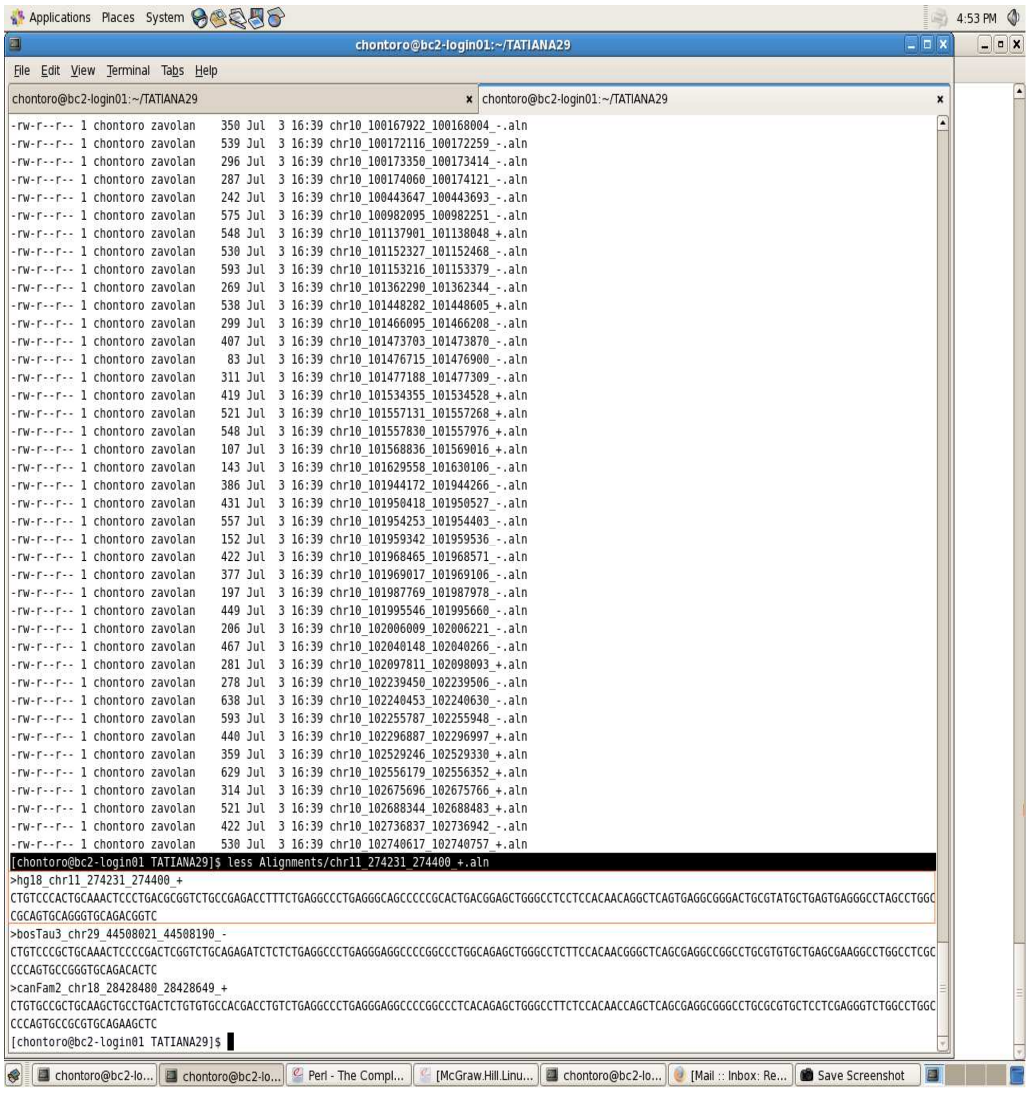
Fig. 38 Screen shoot Log file dimension printable format

As a result, as it was my purpose I extracted in fasta format files all orthologus DNA coding sequences for my 3 chosen genomes, (Fig.39).
The listed records of fasta file in printable format contain 469 pages. An open record from total amount of fasta files I show at the bottom of figure 40.


Applications Places System
4:41 PM
list_of_SEQS (~/TATIANA29) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
list_of_SEQS
Rendering page 144 of 469...
Cancel
Ln 3, Col 15
INS
chontoro@bc... chontoro@bc... Perl - The Co... McGraw.Hill.Li... Human chr5:... [TATIANA29] [LISTS] list_of_SEQS (...

```
chr10_100149849_100150004_-.fna
chr10_100167922_100168004_-.fna
chr10_100172116_100172259_-.fna
chr10_100173350_100173414_-.fna
chr10_100174060_100174121_-.fna
chr10_100443647_100443693_-.fna
chr10_100982095_100982251_-.fna
chr10_101137901_101138048_+.fna
chr10_101152327_101152468_-.fna
chr10_101153216_101153379_-.fna
chr10_101362290_101362344_-.fna
chr10_101448282_101448605_+.fna
chr10_101466095_101466208_-.fna
chr10_101473703_101473870_-.fna
chr10_101476715_101476900_-.fna
chr10_101477188_101477309_-.fna
chr10_101534355_101534528_+.fna
chr10_101557131_101557268_+.fna
chr10_101557830_101557976_+.fna
chr10_101568836_101569016_+.fna
chr10_101629558_101630106_-.fna
chr10_101944172_101944266_-.fna
chr10_101950418_101950527_-.fna
chr10_101954253_101954403_-.fna
chr10_101959342_101959536_-.fna
chr10_101968465_101968571_-.fna
chr10_101969017_101969106_-.fna
chr10_101987769_101987978_-.fna
chr10_101995546_101995660_-.fna
chr10_102006009_102006221_-.fna
chr10_102040148_102040266_-.fna
chr10_102097811_102098093_+.fna
chr10_102239450_102239506_-.fna
chr10_102240453_102240630_-.fna
chr10_102255787_102255948_-.fna
chr10_102296887_102296997_+.fna
chr10_102529246_102529330_+.fna
chr10_102556179_102556352_+.fna
chr10_102675696_102675766_+.fna
chr10_102688344_102688483_+.fna
```

Fig. 39 Extracted Human Coding DNA Sequences in Fasta Format



```
-rw-r--r-- 1 chontoro zavolan    350 Jul  3 16:39 chr10_100167922_100168004_-.aln
-rw-r--r-- 1 chontoro zavolan    539 Jul  3 16:39 chr10_100172116_100172259_-.aln
-rw-r--r-- 1 chontoro zavolan    296 Jul  3 16:39 chr10_100173350_100173414_-.aln
-rw-r--r-- 1 chontoro zavolan    287 Jul  3 16:39 chr10_100174060_100174121_-.aln
-rw-r--r-- 1 chontoro zavolan    242 Jul  3 16:39 chr10_100443647_100443693_-.aln
-rw-r--r-- 1 chontoro zavolan    575 Jul  3 16:39 chr10_100982095_100982251_-.aln
-rw-r--r-- 1 chontoro zavolan    548 Jul  3 16:39 chr10_101137901_101138048_+.aln
-rw-r--r-- 1 chontoro zavolan    530 Jul  3 16:39 chr10_101152327_101152468_-.aln
-rw-r--r-- 1 chontoro zavolan    593 Jul  3 16:39 chr10_101153216_101153379_-.aln
-rw-r--r-- 1 chontoro zavolan    269 Jul  3 16:39 chr10_101362290_101362344_-.aln
-rw-r--r-- 1 chontoro zavolan    538 Jul  3 16:39 chr10_101448282_101448605_+.aln
-rw-r--r-- 1 chontoro zavolan    299 Jul  3 16:39 chr10_101466095_101466208_-.aln
-rw-r--r-- 1 chontoro zavolan    407 Jul  3 16:39 chr10_101473703_101473870_-.aln
-rw-r--r-- 1 chontoro zavolan     83 Jul  3 16:39 chr10_101476715_101476900_-.aln
-rw-r--r-- 1 chontoro zavolan    311 Jul  3 16:39 chr10_101477188_101477309_-.aln
-rw-r--r-- 1 chontoro zavolan    419 Jul  3 16:39 chr10_101534355_101534528_+.aln
-rw-r--r-- 1 chontoro zavolan    521 Jul  3 16:39 chr10_101557131_101557268_+.aln
-rw-r--r-- 1 chontoro zavolan    548 Jul  3 16:39 chr10_101557830_101557976_+.aln
-rw-r--r-- 1 chontoro zavolan    107 Jul  3 16:39 chr10_101568836_101569016_+.aln
-rw-r--r-- 1 chontoro zavolan    143 Jul  3 16:39 chr10_101629558_101630106_-.aln
-rw-r--r-- 1 chontoro zavolan    386 Jul  3 16:39 chr10_101944172_101944266_-.aln
-rw-r--r-- 1 chontoro zavolan    431 Jul  3 16:39 chr10_101950418_101950527_-.aln
-rw-r--r-- 1 chontoro zavolan    557 Jul  3 16:39 chr10_101954253_101954403_-.aln
-rw-r--r-- 1 chontoro zavolan    152 Jul  3 16:39 chr10_101959342_101959536_-.aln
-rw-r--r-- 1 chontoro zavolan    422 Jul  3 16:39 chr10_101968465_101968571_-.aln
-rw-r--r-- 1 chontoro zavolan    377 Jul  3 16:39 chr10_101969017_101969106_-.aln
-rw-r--r-- 1 chontoro zavolan    197 Jul  3 16:39 chr10_101987769_101987978_-.aln
-rw-r--r-- 1 chontoro zavolan    449 Jul  3 16:39 chr10_101995546_101995660_-.aln
-rw-r--r-- 1 chontoro zavolan    206 Jul  3 16:39 chr10_102006009_102006221_-.aln
-rw-r--r-- 1 chontoro zavolan    467 Jul  3 16:39 chr10_102040148_102040266_-.aln
-rw-r--r-- 1 chontoro zavolan    281 Jul  3 16:39 chr10_102097811_102098093_+.aln
-rw-r--r-- 1 chontoro zavolan    278 Jul  3 16:39 chr10_102239450_102239506_-.aln
-rw-r--r-- 1 chontoro zavolan    638 Jul  3 16:39 chr10_102240453_102240630_-.aln
-rw-r--r-- 1 chontoro zavolan    593 Jul  3 16:39 chr10_102255787_102255948_-.aln
-rw-r--r-- 1 chontoro zavolan    440 Jul  3 16:39 chr10_102296887_102296997_+.aln
-rw-r--r-- 1 chontoro zavolan    359 Jul  3 16:39 chr10_102529246_102529330_+.aln
-rw-r--r-- 1 chontoro zavolan    629 Jul  3 16:39 chr10_102556179_102556352_+.aln
-rw-r--r-- 1 chontoro zavolan    314 Jul  3 16:39 chr10_102675696_102675766_+.aln
-rw-r--r-- 1 chontoro zavolan    521 Jul  3 16:39 chr10_102688344_102688483_+.aln
-rw-r--r-- 1 chontoro zavolan    422 Jul  3 16:39 chr10_102736837_102736942_-.aln
-rw-r--r-- 1 chontoro zavolan    530 Jul  3 16:39 chr10_102740617_102740757_+.aln
[chontoro@bc2-login01 TATIANA29]$ less Alignments/chr11_274231_274400_+.aln
>hg18_chr11_274231_274400_+
CTGTCCCACTGCAAACTCCCTGACGCGGTCTGCCGAGACCTTTCTGAGGCCCTGAGGGCAGCCCCCGCACTGACGGAGCTGGGCCTCCTCCACAACAGGCTCAGTGAGGCGGGACTGCGTATGCTGAGTGAGGGCCTAGCCTGGC
CGCAGTGCAGGGTGCAGACGGTC
>bosTau3_chr29_44508021_44508190_-
CTGTCCCGCTGCAAACTCCCCGACTCGGTCTGCAGAGATCTCTCTGAGGCCCTGAGGGAGGCCCCGGCCCTGGCAGAGCTGGGCCTCTTCCACAACGGGCTCAGCGAGGCCGGCCTGCGTGTGCTGAGCGAAGGCCTGGCCTCGC
CCCAGTGCCGGGTGCAGACACTC
>canFam2_chr18_28428480_28428649_+
CTGTGCCGCTGCAAGCTGCCTGACTCTGTGTGCCACGACCTGTCTGAGGCCCTGAGGGAGGCCCCGGCCCTCACAGAGCTGGGCCTTCTCCACAACCAGCTCAGCGAGGCGGGCCTGCGCGTGCTCCTCGAGGGTCTGGCCTGGC
CCCAGTGCCGCGTGCAGAAGCTC
[chontoro@bc2-login01 TATIANA29]$
```

Fig. 40 Open file with Extracted DNA coding DNA sequences in Fasta Format

The first line following the highlighted black line indicates:

- The DNA sequence from HUMAN Genom(*hg 18*), located in chromosome 11(chr 11),and the dna coding sequence starts at the position 274231 and it ends at the position 274400, located on the positive strand of twisted DNA.
  The DNA coding sequence is 169 bases long (274400-274231=169, counted and extracted automatically by my implemented software).
  >hg18_chr11_ 274231_274400_+
- The software extracted the orthologus DNA coding sequence with the same function in TAURUS Genome, as in human chromosome 11.

As seen in figure 40, the DNA orthologus sequence in Taurus, responsible for the same function as in human is located in Taurus(*bosTau3*) in not in chromosome 11 but in chromosome 29 with start coordinate position 44508021 and end position 44508190, is located on negative strand of twisted Taurus DNA.

>bosTau3_chr29_44508190_44508021_-

- The software extracted the orthologus DNA coding sequence with the same function in DOG Genome (*canFam2*), as in human chromosome 11.

As seen in figure 40, the DNA orthologus sequence in DOG, responsible for the same function as in human is located in DOG Genome not in chromosome 11 or chromosome 29 as in Taurus, but is located in Dog chromosome 18, with start coordinate position 28428480 and end position 28428649 the dna sequence is located on the positive strand of twisted Dog DNA

> canFam2__chr18_ 28428480_28428649_+

Beeing explicit about the same function of an ortholog DNA Coding Sequence, as an example of the same function I will consider the DNA Coding Sequence responsible for developing the eyes:

The DNA Coding Sequence responsible for developing the eyes in human is located in chromosome 11, in Taurus is located in chromosome 29 and in Dog is located in chromosome 18.

For every gene and every DNA coding sequence of HUMAN Genome I extracted in fasta files the DNA coding sequence of human and the ortholog TAURUS DNA coding sequence and DOG DNA coding sequence, (fig 40):

>hg18_chr11_ 274231_274400_+
>bosTau3_chr29_44508190_44508021_-
>canFam2_chr18_28428480_28428649_+

The extracted DNA coding sequence I call **DNA Cryptographic Keys Sequences**

Description of the 3rd Step of implementation:

I developed and implemented a software tool to aligning the DNA Cryptographic Keys Sequences (obtained in the second step of implementation) of the same gene from related chosen species with respect to Human DNA Sequences. The alignment in the evolutionary system pipeline of DNA Cryptographic Keys Sequences is realized with trained ProbCons tool, a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy.

As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution, (Ochman, 2003). Some of these will lead to differences in the amino acids of the encoded protein (non-synonymous changes) Because of the degeneracy of the genetic code leave the protein unchanged (synonymous, or silent changes). If Ka/Ks< 1 *Purifying (negative) selection,* most proteins are well adapted to carry out their function change would not lead to the creation of selective advantage. If *Ka/Ks >1 Diversifying (positive)*, selection has acted to change the protein and if *Ka/Ks= 1 Neutral evolution*, (Mustonen, Lässig, 2005). After aligning my extracted DNA Sequences with ProbCons tool, I derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

ProbCons is a tool for generating multiple alignments of protein sequences. It uses a combination of probabilistic modeling and consistency-based alignment techniques and has achieved the highest accuracies of all alignments methods. The basic for ProbCons algorithm is the computation of pairwise posterior probability matrices, P(xi ~ yi |x, y), which give the probability that one should match letters *xi* and *yi* when aligning two sequences *x* and *y*. ProbCons uses a simple probabilistic model that allows for efficient computation of this probabilities. Given a set of sequences ProbCons computes the posterior probability matrices for each pair of sequences and computes the expected accuracy of each alignment, (Moses, Chiang, Pollard, Iyer, Eisen, 2004). It applies the probabilistic consistency transformation to posterior matrices and computes a guide tree using the expected accuracies, progressively aligning the sequences using the guide tree.

ProbCons is a pair-hidden Markov model-based progressive alignment algorithm that primarily differs from most typical approaches in its use of *maximum expected accuracy* rather than Viterbi alignment, and of the *probabilistic consistency transformation* to incorporate multiple sequence conservation information during pairwise alignment. ProbCons uses the HMM shown in (Fig.41), to specify the probability distribution over all alignments between a pair of sequences.
Emission probabilities, which correspond to traditional substitution scores, are based on the BLOSUM62 matrix. Transition probabilities, which correspond to gap penalties, are trained with unsupervised expectation maximization (EM).

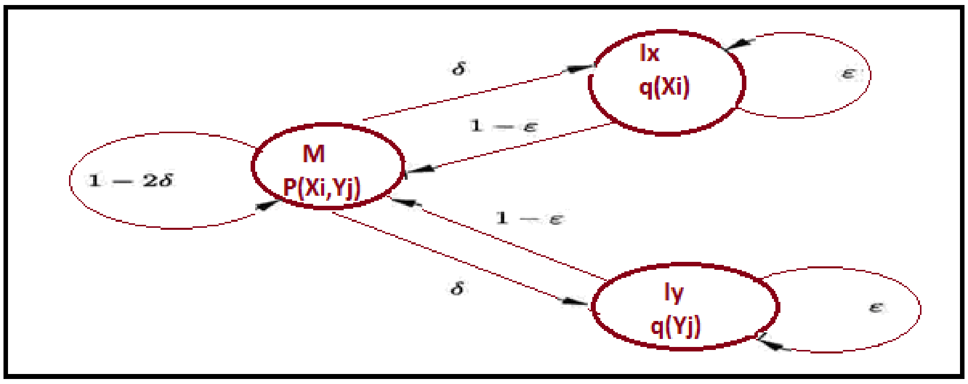


Fig. 41 Basic pair-HMM for sequence alignment between two sequences

State *M* emits two letters, one from each sequence, and corresponds to the two letters being aligned together. State *Ix* emits a letter in sequence *x* that is aligned to a gap, and similarly state *Iy* emits a letter in sequence *y* that is aligned to a gap, (Halpern, Bruno, 1998). Finding the most likely alignment according to this model by using the Viterbi algorithm corresponds to applying Needleman–Wunsch with appropriate parameters. The logarithm of the emission probability function *p*(.,.) at *M* corresponds to a substitution scoring matrix, while affine gap penalty parameters can be derived from the transition probabilities. In order to find the best set of parameter file probcons needs to be trained.

Description of the 4th Step of implementation:

I implemented a software tool to train ProbCons tool in order to obtain the best parameter set I need to use for the accurate alignment in the 3rd step.

The best alignment for coding regions (*DNA Cryptographic Keys Sequences),* of the same gene, from related chosen species, with respect to Human DNA can be realized only with a trained best parameter set of ProbCons tool. The software uses my obtained coding sequences from 1st and 2nd step of implementation.

ProbCons tool, a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy.

In order to find the best set of parameter file probcons needs to be trained. The trained parameter file I used for aligning DNA sequences, specifies the initial/final probabilities and transition probabilities for the Hidden Marko Model, model used by aligner.

The HMM consists of Match state, Insert state X and Insert state Y. The file consists of three lines, containing:

1. initMatchProb, initInsertXprob, initInsertYProb
2. startInsertXProb, startInsertYProb
3. extendInsertXProb, extendInsertYProb

When the sum of this parameter set is close to 0 (less than 0.1), ProbCons converges considering that it has found the best parameter set needed to align our DNA sequences.

In order we developed the script to train ProbCons and find the best trained parameter file we used for aligning our sequences.

In my case ProbCons converges at FILEPARAM.105, considering that it has found the best parameter set I need to align my DNA sequences.

FILEPARAM.105 is the best trained parameter file which I used for aligning my DNAsequences.

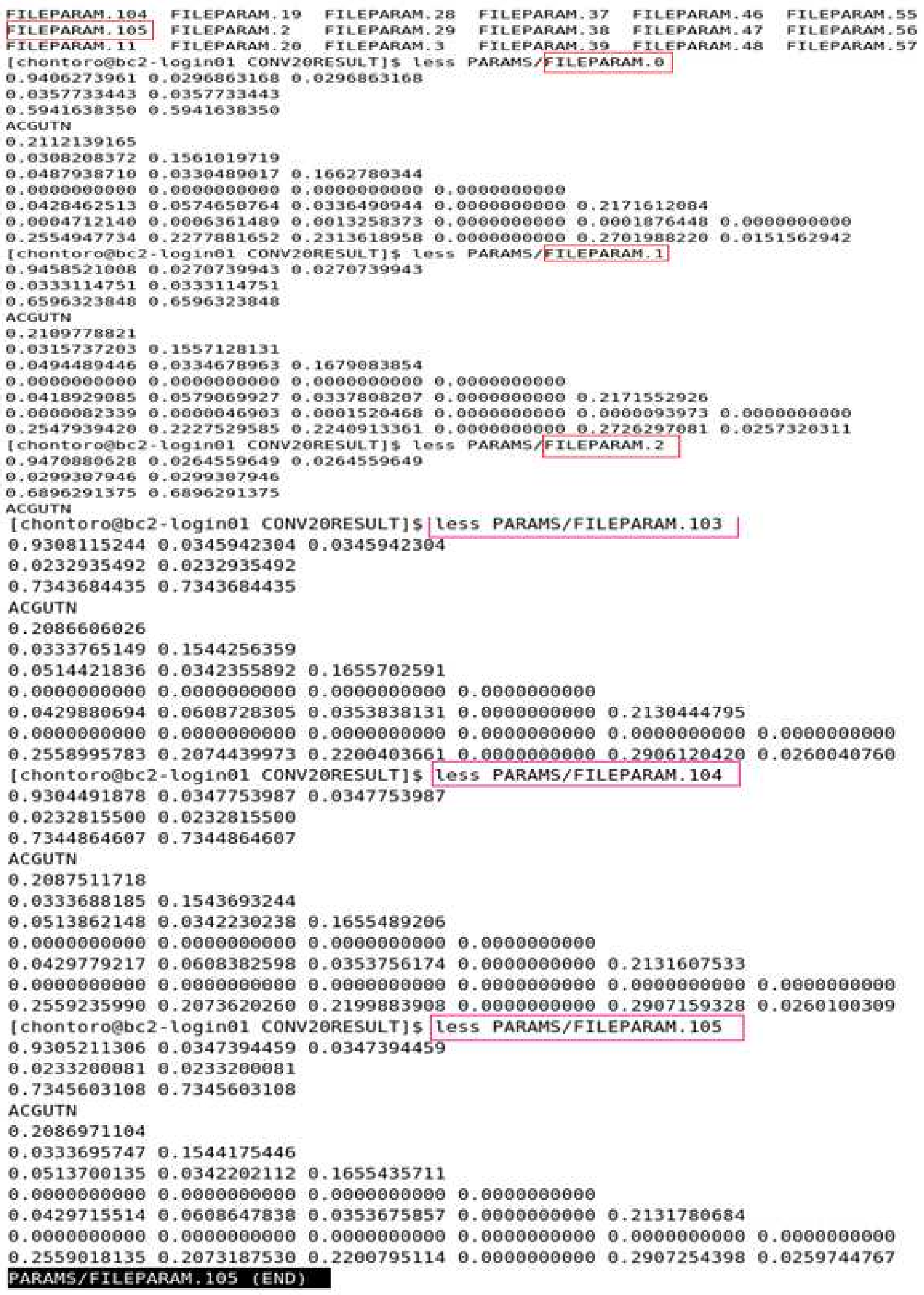
As seen in Figure 42, the sum of the parameter set in the FILEPARAM.105 is close to 0 and less than 0.1.

In my case the sum of the parameter set in FILEPARAM.105 is close to 0 (less than 0.1) and ProbCons converges considering that it has found the best parameter set I need to align my DNA sequences.

After obtaining the best trained parameter set, (Fig.42) I used it for aligning extracted DNA Sequences which I use to derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

Using the best parameter set in my case FILEPARAM.105, I aligned my extracted DNA Sequences (which are fasta format files).

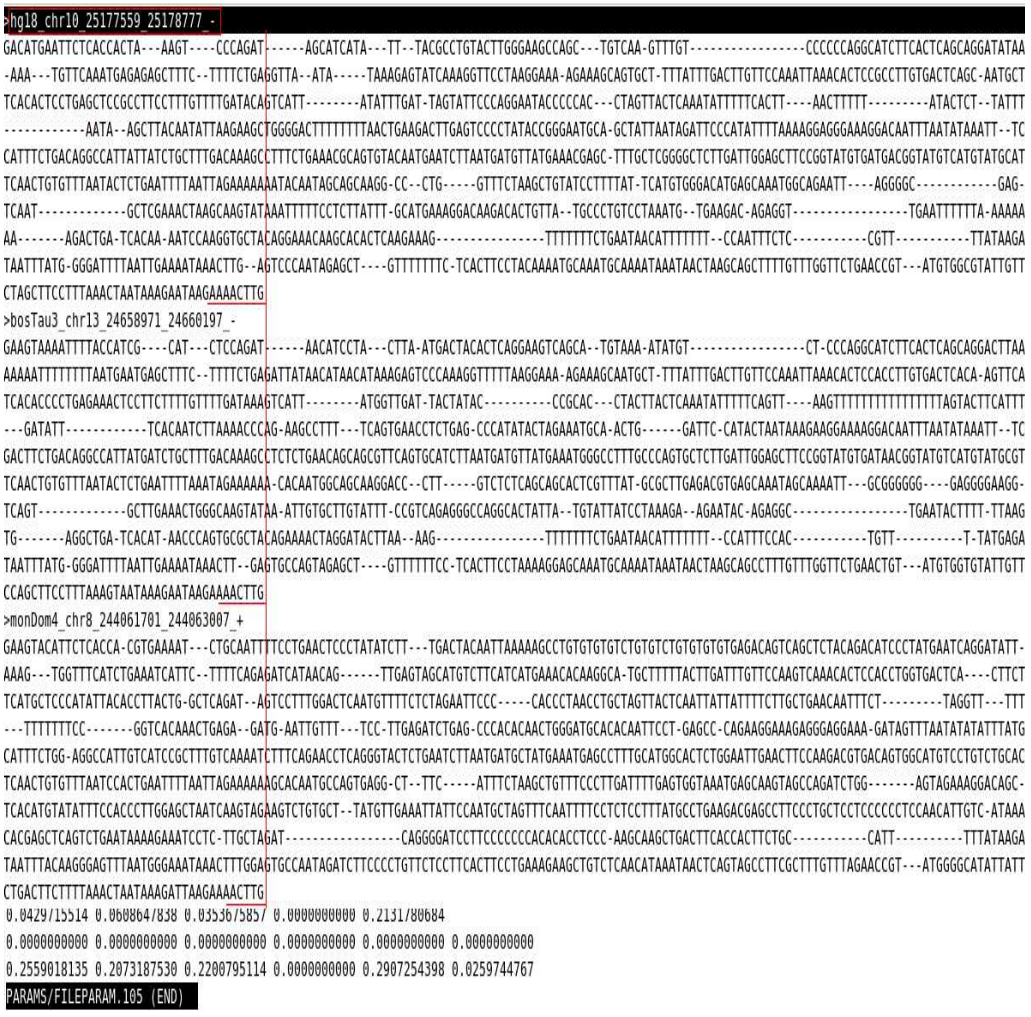
This DNA sequence files I use to derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions, containing information for all human genes related to the chromosome.



```
FILEPARAM.104   FILEPARAM.19   FILEPARAM.28   FILEPARAM.37   FILEPARAM.46   FILEPARAM.55
FILEPARAM.105   FILEPARAM.2    FILEPARAM.29   FILEPARAM.38   FILEPARAM.47   FILEPARAM.56
FILEPARAM.11    FILEPARAM.20   FILEPARAM.3    FILEPARAM.39   FILEPARAM.48   FILEPARAM.57
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.0
0.9406273961 0.0296863168 0.0296863168
0.0357733443 0.0357733443
0.5941638350 0.5941638350
ACGUTN
0.2112139165
0.0308208372 0.1561019719
0.0487938710 0.0330489017 0.1662780344
0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.0428462513 0.0574650764 0.0336490944 0.0000000000 0.2171612084
0.0004712140 0.0006361489 0.0013258373 0.0000000000 0.0001876448 0.0000000000
0.2554947734 0.2277881652 0.2313618958 0.0000000000 0.2701988220 0.0151562942
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.1
0.9458521008 0.0270739943 0.0270739943
0.0333114751 0.0333114751
0.6596323848 0.6596323848
ACGUTN
0.2109778821
0.0315737203 0.1557128131
0.0494489446 0.0334678963 0.1679083854
0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.0418929085 0.0579069927 0.0337808207 0.0000000000 0.2171552926
0.0000082339 0.0000046903 0.0001520468 0.0000000000 0.0000093973 0.0000000000
0.2547939420 0.2227529585 0.2240913361 0.0000000000 0.2726297081 0.0257320311
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.2
0.9470880628 0.0264559649 0.0264559649
0.0299307946 0.0299307946
0.6896291375 0.6896291375
ACGUTN
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.103
0.9308115244 0.0345942304 0.0345942304
0.0232935492 0.0232935492
0.7343684435 0.7343684435
ACGUTN
0.2086606026
0.0333765149 0.1544256359
0.0514421836 0.0342355892 0.1655702591
0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.0429880694 0.0608728305 0.0353838131 0.0000000000 0.2130444795
0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.2558995783 0.2074439973 0.2200403661 0.0000000000 0.2906120420 0.0260040760
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.104
0.9304491878 0.0347753987 0.0347753987
0.0232815500 0.0232815500
0.7344864607 0.7344864607
ACGUTN
0.2087511718
0.0333688185 0.1543693244
0.0513862148 0.0342230238 0.1655489206
0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.0429779217 0.0608382598 0.0353756174 0.0000000000 0.2131607533
0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.2559235990 0.2073620260 0.2199883908 0.0000000000 0.2907159328 0.0260100309
[chontoro@bc2-login01 CONV20RESULT]$ less PARAMS/FILEPARAM.105
0.9305211306 0.0347394459 0.0347394459
0.0233200081 0.0233200081
0.7345603108 0.7345603108
ACGUTN
0.2086971104
0.0333695747 0.1544175446
0.0513700135 0.0342202112 0.1655435711
0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.0429715514 0.0608647838 0.0353675857 0.0000000000 0.2131780684
0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.2559018135 0.2073187530 0.2200795114 0.0000000000 0.2907254398 0.0259744767
PARAMS/FILEPARAM.105 (END)
```

Fig. 42 Trained Parameter Set

Figure 43 shows one of my open file in fasta format file with aligned DNA sequences as seen.
The open fasta file contains the ortholog DNA sequence, **for Human (>hg18) Chromosome10** (**chr10**), with **start coordinate 25177559** and the **end coordinate is 25178777**. This ortholog DNA sequence is 1218 base long.
>hg18 chr10_25177559_25178777
The ortholog DNA sequence with the same function as in Human:
hg18 chr10_25177559_25178777, in Taurus is (as seen in fig 43)**,**
>bosTaur3_chr13_24658971_24660197



```
>hg18_chr10_25177559_25178777_-
GACATGAATTCTCACCACTA---AAGT----CCCAGAT------AGCATCATA---TT--TACGCCTGTACTTGGGAAGCCAGC---TGTCAA-GTTTGT----------------CCCCCCAGGCATCTTCACTCAGCAGGATATAA
-AAA---TGTTCAAATGAGAGAGCTTTC--TTTTCTGAGGTTA--ATA-----TAAAGAGTATCAAAGGTTCCTAAGGAAA-AGAAAGCAGTGCT-TTTATTTGACTTGTTCCAAATTAAACACTCCGCCTTGTGACTCAGC-AATGCT
TCACACTCCTGAGCTCCGCCTTCCTTTGTTTTGATACAGTCATT--------ATATTTGAT-TAGTATTCCCAGGAATACCCCCAC---CTAGTTACTCAAATATTTTTCACTT----AACTTTTT---------ATACTCT--TATTT
------------AATA--AGCTTACAATATTAAGAAGCTGGGGACTTTTTTTTAACTGAAGACTTGAGTCCCCTATACCGGGAATGCA-GCTATTAATAGATTCCCATATTTTAAAAGGAGGGAAAGGACAATTTAATATAAATT--TC
CATTTCTGACAGGCCATTATTATCTGCTTTGACAAAGCCTTTCTGAAACGCAGTGTACAATGAATCTTAATGATGTTATGAAACGAGC-TTTGCTCGGGGCTCTTGATTGGAGCTTCCGGTATGTGATGACGGTATGTCATGTATGCAT
TCAACTGTGTTTAATACTCTGAATTTTAATTAGAAAAAAATACAATAGCAGCAAGG-CC--CTG-----GTTTCTAAGCTGTATCCTTTTAT-TCATGTGGGACATGAGCAAATGGCAGAATT----AGGGGC------------GAG-
TCAAT-------------GCTCGAAACTAAGCAAGTATAAATTTTTCCTCTTATTT-GCATGAAAGGACAAGACACTGTTA--TGCCCTGTCCTAAATG--TGAAGAC-AGAGGT----------------TGAATTTTTTA-AAAAA
AA-------AGACTGA-TCACAA-AATCCAAGGTGCTACAGGAAACAAGCACACTCAAGAAAG----------------TTTTTTTCTGAATAACATTTTTTT--CCAATTTCTC-----------CGTT-----------TTATAAGA
TAATTTATG-GGGATTTTAATTGAAAATAAACTTG--AGTCCCAATAGAGCT----GTTTTTTTC-TCACTTCCTACAAAATGCAAATGCAAAATAAATAACTAAGCAGCTTTTGTTTGGTTCTGAACCGT---ATGTGGCGTATTGTT
CTAGCTTCCTTTAAACTAATAAAGAATAAGAAAACTTG
>bosTau3_chr13_24658971_24660197_-
GAAGTAAAATTTTACCATCG----CAT---CTCCAGAT------AACATCCTA---CTTA-ATGACTACACTCAGGAAGTCAGCA--TGTAAA-ATATGT----------------CT-CCCAGGCATCTTCACTCAGCAGGACTTAA
AAAAATTTTTTTTAATGAATGAGCTTTC--TTTTCTGAGATTATAACATAACATAAAGAGTCCCAAAGGTTTTTAAGGAAA-AGAAAGCAATGCT-TTTATTTGACTTGTTCCAAATTAAACACTCCACCTTGTGACTCACA-AGTTCA
TCACACCCCTGAGAAACTCCTTCTTTTGTTTTGATAAAGTCATT--------ATGGTTGAT-TACTATAC----------CCGCAC---CTACTTACTCAAATATTTTTCAGTT----AAGTTTTTTTTTTTTTTTTAGTACTTCATTT
---GATATT------------TCACAATCTTAAAACCCAG-AAGCCTTT---TCAGTGAACCTCTGAG-CCCATATACTAGAAATGCA-ACTG------GATTC-CATACTAATAAAGAAGGAAAAGGACAATTTAATATAAATT--TC
GACTTCTGACAGGCCATTATGATCTGCTTTGACAAAGCCTCTCTGAACAGCAGCGTTCAGTGCATCTTAATGATGTTATGAAATGGGCCTTTGCCCAGTGCTCTTGATTGGAGCTTCCGGTATGTGATAACGGTATGTCATGTATGCGT
TCAACTGTGTTTAATACTCTGAATTTTAAATAGAAAAAA-CACAATGGCAGCAAGGACC--CTT-----GTCTCTCAGCAGCACTCGTTTAT-GCGCTTGAGACGTGAGCAAATAGCAAAATT---GCGGGGGG----GAGGGGAAGG-
TCAGT-------------GCTTGAAACTGGGCAAGTATAA-ATTGTGCTTGTATTT-CCGTCAGAGGGCCAGGCACTATTA--TGTATTATCCTAAAGA--AGAATAC-AGAGGC----------------TGAATACTTTT-TTAAG
TG-------AGGCTGA-TCACAT-AACCCAGTGCGCTACAGAAAACTAGGATACTTAA--AAG----------------TTTTTTTCTGAATAACATTTTTTT--CCATTTCCAC-----------TGTT----------T-TATGAGA
TAATTTATG-GGGATTTTAATTGAAAATAAACTT--GAGTGCCAGTAGAGCT----GTTTTTTCC-TCACTTCCTAAAAGGAGCAAATGCAAAATAAATAACTAAGCAGCCTTTGTTTGGTTCTGAACTGT---ATGTGGTGTATTGTT
CCAGCTTCCTTTAAAGTAATAAAGAATAAGAAAACTTG
>monDom4_chr8_244061701_244063007_+
GAAGTACATTCTCACCA-CGTGAAAAT---CTGCAATTTTCCTGAACTCCCTATATCTT---TGACTACAATTAAAAAGCCTGTGTGTGTCTGTGTCTGTGTGTGTGAGACAGTCAGCTCTACAGACATCCCTATGAATCAGGATATT-
AAAG---TGGTTTCATCTGAAATCATTC--TTTTCAGAGATCATAACAG------TTGAGTAGCATGTCTTCATCATGAAACACAAGGCA-TGCTTTTTACTTGATTTGTTCCAAGTCAAACACTCCACCTGGTGACTCA----CTTCT
TCATGCTCCCATATTACACCTTACTG-GCTCAGAT--AGTCCTTTGGACTCAATGTTTTCTCTAGAATTCCC-----CACCCTAACCTGCTAGTTACTCAATTATTATTTTCTTGCTGAACAATTTCT---------TAGGTT---TTT
---TTTTTTTCC-------GGTCACAAACTGAGA--GATG-AATTGTTT---TCC-TTGAGATCTGAG-CCCACACAACTGGGATGCACACAATTCCT-GAGCC-CAGAAGGAAAGAGGGAGGAAA-GATAGTTTAATATATATTTATG
CATTTCTGG-AGGCCATTGTCATCCGCTTTGTCAAAATCTTTCAGAACCTCAGGGTACTCTGAATCTTAATGATGCTATGAAATGAGCCTTTGCATGGCACTCTGGAATTGAACTTCCAAGACGTGACAGTGGCATGTCCTGTCTGCAC
TCAACTGTGTTTAATCCACTGAATTTTAATTAGAAAAAAGCACAATGCCAGTGAGG-CT--TTC-----ATTTCTAAGCTGTTTCCCTTGATTTTGAGTGGTAAATGAGCAAGTAGCCAGATCTGG-------AGTAGAAAGGACAGC-
TCACATGTATATTTCCACCCTTGGAGCTAATCAAGTAGAAGTCTGTGCT--TATGTTGAAATTATTCCAATGCTAGTTTCAATTTTCCTCTCCTTTATGCCTGAAGACGAGCCTTCCCTGCTCCTCCCCCCTCCAACATTGTC-ATAAA
CACGAGCTCAGTCTGAATAAAAGAAATCCTC-TTGCTAGAT----------------CAGGGGATCCTTCCCCCCCACACACCTCCC-AAGCAAGCTGACTTCACCACTTCTGC-----------CATT----------TTTATAAGA
TAATTTACAAGGGAGTTTAATGGGAAATAAACTTTGGAGTGCCAATAGATCTTCCCCTGTTCTCCTTCACTTCCTGAAAGAAGCTGTCTCAACATAAATAACTCAGTAGCCTTCGCTTTGTTTAGAACCGT---ATGGGGCATATTATT
CTGACTTCTTTTAAACTAATAAAGATTAAGAAAACTTG
0.0429715514 0.0608647838 0.0353675857 0.0000000000 0.2131780684
0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000
0.2559018135 0.2073187530 0.2200795114 0.0000000000 0.2907254398 0.0259744767
PARAMS/FILEPARAM.105 (END)
```

Fig. 43 Fasta file with Aligned Ortholog DNA Sequences

I use the fasta files with Aligned DNA Orhtologus Coding Sequences from related species to infer the philogenetic tree, in order to derive the private/public pair DNA cryptographic keys sequences.

Description of the 5th Step of implementation

I developed a*n algorithm and implemented a software(Process System Pipeline Evolutionary Models) for computing the philogenetic tree relating my chosen species and the branch length during evolution for chosen species deriving DNA Cryptographic Keys Sequences from Human Genome Analysis* as described in the following section 8.2, of my thesis.

Models of DNA evolution were first proposed in 1969 by Jukes and Cantor, assuming equal transition rates and equal equilibrium frequencies for all bases.
In 1980 Kimura-Ohta introduced a model of DNA Evolution with two parameters: one for the transition and one for the transversion rate.

To estimate evolutionary distances in terms of the number of nucleotide substitutions and the evolutionary rates when the divergence times are known by comparing a pair of nucleotide sequences. There are two types of differences when homologous sites are occupied by different nucleotide bases and both are purines or both are pyrimidines. The difference is called Transition type when one of the two is a purine and the other is a pyrimidine then the difference is called transversion type.
Let *P* and *Q* be the fractions of nucleotide sites, showing between two sequences compared the transition and transversion type differences, then:
The *Evolutionary Distance per Site* is:

$$K = -(1/2)\ln\{(1 - 2P - Q)\} \qquad (8.18)$$

The *Evolutionary Rate per Year* is then given by:

$$K = K/(2T) \qquad (8.19)$$

T is the time since the divergence of the two sequences. If only the third codon positions are compared, then *the Synonymous Component of Evolutionary Base Substitutions per Site* is:

$$K'_S = -(1/2)\ln(1 - 2P - Q) \qquad (8.20)$$

In biology, a substitution model describes the process from which a sequence of characters changes into another set of traits.
Each position in the sequence corresponds to a property of a species which can either be present or absent. In phylogenetics sequences are obtained in the following order:
1. Obtaining a protein or nucleotide sequence alignment
2. Taking in the alignment as the characters the amino acids or bases at corresponding positions.
In phylogenetics for constructing evolutionary trees substitution models are used.
Substitution models are:
Neutral Models, where selection does not operate on substitutions.
Independent Models, where changes in one site do not affect the probability of changes in another site.
Finite Sites Models, there are finitely many sites and over evolution, a single site can be changed multiple times.

A branch length of a phylogenetic tree is expressed as the expected number of substitutions per site. If the evolutionary model indicates that each site within an ancestral sequence will typically experience *x* substitutions by the time it evolves to a particular descendant's sequence then the ancestor and descendant are considered to be separated by branch length *x*.

Sometimes a branch length is measured in terms of geological years.
The number of substitutions per site per year is indicated with the letter (μ).
A model has a molecular clock if the expected number of substitutions per year μ is constant regarding species' evolution examination.

Many substitution models are time-reversible, in mathematics, the model does not depend of which sequence is the ancestor and which sequence is the descendant, so long as all other parameter as the expected number of substitutions per site between two sequences is constant.

The phylogenetic tree can be rooted using any of the species, re-rooted later based on new knowledge, or left unrooted, because there is no special species, all species will derive from one another with the same probability.
A *Model is Time Reversible* if and only if it satisfies the property:

$$\pi_i Q_{ij} = \pi_j Q_{ji} \tag{8.21}$$

or, equivalently, the *Detailed Balance Property*:

$$\pi_i P(t)_{ij} = \pi_j P(t)_{ji} \tag{8.22}$$

where *P* is the *Markov Transition Matrix* (transition probability),

$$P_{ij} = P(X_t = j \mid X_{t-1} = i) \tag{8.23}$$

and π_i and π_j are the *Equilibrium Probabilities* of being in states *i* and *j*, respectively.
When for all *i*:

$$Pr(X_{t-1} = i) = \pi_i \tag{8.24}$$

this is equivalent to the *Probability Matrix*:

$$Pr(X_{t-1} = i, X_t = j) \tag{8.25}$$

being symmetric in *i* and *j*; or symmetric in $t-1$ and *t*.

The definition carries over straight forwardly to continuous variables, where π becomes a *Probability Density* and $P(s', s)$ a *Transition Kernel Probability Density* from state *s′* to state *s*:

$$\pi(s')P(s',s) = \pi(s)P(s,s') \tag{8.26}$$

A Markov process that has detailed balance is said to be a *Reversible Markov Process or Reversible Markov Chain.*

A Markov chain is said to be *reversible* if there is a *Probability Distribution* over states, **π**, such that for all times *n* and all states *i* and *j*:

$$\pi_i \Pr(X_{n+1} = j \mid X_n = i) = \pi_j \Pr(X_{n+1} = i \mid X_n = j) \tag{8.27}$$

This condition is also known as the ***Detailed Balance Condition***.
With a *Time Homogeneous Markov Chain*

$$Pr(X_{n+1} = j \mid X_n = i) \tag{8.28}$$

does not change with time *n* and it can be written more simply as p_{ij}.
And the *Detailed Balance Equation* can be written as:

$$\pi_i p_{ij=}\pi_j p_{ji} \tag{8.29}$$

Summing the original equation over *i* gives:

$$\sum_i \pi_i \Pr(X_{n+1} = j \mid X_n = i) = \sum_i \pi_j \Pr(X_{n+1} = i \mid X_n = j) = \pi_j \sum_i \Pr(X_{n+1} = i \mid X_n = j) = \pi_j \tag{8.30}$$

for reversible Markov chains, **π** is always a *Steady-State Distribution* of

$$Pr(X_{n+1} = j \mid X_n = i), \text{ for every } n \tag{8.31}$$

If the Markov chain begins in the Steady-State Distribution and if:

$$Pr(X_0 = i) = \pi_i, \tag{8.32}$$

Then:

$$Pr(X_n = i) = \pi_i \text{ for all } n \tag{8.33}$$

The Detailed Balance Equation can be written as:

$$\Pr(X_n = i, X_{n+1} = j) = \Pr(X_{n+1} = i, X_n = j) \tag{8.34}$$

The left and right part sides of this equation are identical except for a reversing of the time indices *n* and *n* + 1.

Detailed balance implies that around any closed cycle of states, there is no net flow of probability and implies that for all *a*, *b* and *c*:

$$P(a,b)P(b,c)P(c,a) = P(a,c)P(c,b)P(b,a) \tag{8.35}$$

In the case of a *Positive Transition Matrix*, the condition implies a Detailed Balance.

A *Markov Process* with detailed balance can be described in terms of a *Relative Entropy Function*:

$$H(x) = -\sum_i x_i \ln \frac{x_i}{\pi_i} \tag{8.36}$$

which acts like a potential in that the relative entropy of the Markov Process is always increasing, reaches its maximum at the stationary distribution.

The proof of this is a mild generalization of *Boltzmann's H-Theorem*.
Transition Matrices that are symmetric always and have detailed balance:

$$(P_{ij} = P_{ji} \text{ or } P(s', s) = P(s, s')) \tag{8.37}$$

In these cases, a uniform distribution over the states represents an equilibrium distribution.

For continuous systems with detailed balance it is possible to continuously transform the coordinates until the equilibrium distribution is uniform.

For the case of discrete states it is be possible to achieve similarity by breaking the Markov states into a degeneracy of substates.

Stationary, Neutral, Independent and Finite Sites Models, assuming a constant rate of evolution, have two parameters:

п: an *Equilibrium Vector of Base Frequencies*

Q: a Rate Matrix,

which describes the rate at which bases of one type change into bases of another type; element Q_{ij} for $i \neq j$ is the rate at which base *i* goes to base *j*.

The diagonals of the *Q* matrix are chosen so that the rows sum to zero:

$$Q_{ij} = -\sum_{\{j|j \neq i\}} Q_{ij} \tag{8.38}$$

The *Equilibrium Row Vector* п must be annihilated by the rate matrix *Q*:

$$\pi Q = 0 \tag{8.39}$$

The *Transition Matrix Function* is a function from the branch lengths to a matrix of conditional probabilities, denoted as *P*(*t*).

The entry in the *i*th column and the *j*th row, $P_{ij}(t)$, is the probability, after time *t*, that there is a base *j* at a given position, conditional on there being a base *i* in that position at time 0. When the model is time reversible it can be performed between any two sequences and if the total branch length between them is known.

The asymptotic properties of P_{ij}(t) are such that:

$$P_{ij}(0) = \delta_{ij} \text{ , } \quad \delta_{ij} \text{ is the } \textit{Kronecker Delta Function} \tag{8.40}$$

That is there is no change in base composition between a sequence and itself.

In mathematics *Kronecker Delta Function* is a function of two variables usually integers which is 1 if they are equal and 0 if not:.

$$\delta_{ij} = \begin{Bmatrix} 1, i = j \\ 0, i \neq j \end{Bmatrix} \tag{8.41}$$

Written as the symbol δ_{ij}, and treated as notational shorthand rather than as a function.

At the other extreme:

$$\text{s}\lim_{t \to \infty} P_{ij}(t) = \pi_j \tag{8.42}$$

As time goes to infinity, the probability of finding base *j* at a position given there was a base *i* at that position, goes to the equilibrium probability that there is base *j* at that position, regardless of the original base.

It follows that:

$$\pi P(t) = \pi \text{ for all } t \tag{8.43}$$

The *Transition Matrix* can be computed from the rate matrix via matrix exponentiation:

$$P(t) = e^{Qt} = \sum_{n=0}^{\infty} Q^n \frac{t^n}{n!} \tag{8.44}$$

where Q^n is the matrix Q multiplied by itself, enough times to produce its nth power. If Q is diagonalizable, the matrix exponential can be computed directly
Let:

$$Q = U^{-1} \Lambda U \tag{8.45}$$

a diagonalization of *Q*, with:

$$\Lambda = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_4 \end{pmatrix} \tag{8.46}$$

Where *Λ* is a *Diagonal Matrix* and where *{λ_i}* are the *eigenvalues* of *Q*. Each repeated according to its multiplicity.
Then:

$$\mathrm{P}(t) = e^{Qt} = e^{U-1(\Lambda t)U} = U^{-1} e^{\Lambda t} U \tag{8.47}$$

and the *Diagonal Matrix* $e^{\Lambda t}$ is given by

$$e^{\Lambda t} = \begin{pmatrix} e^{\Lambda 1t} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & e^{\Lambda 4t} \end{pmatrix} \tag{8.48}$$

Generalized Time Reversible (GTR) is the most general neutral, independent, finite-sites, time-reversible model possible, (Simon Tavaré, 1986).
The GTR parameters for nucleotides consist of an:
Equilibrium Base Frequency Vector:

$$\vec{\pi} = (\pi_{1,} \pi_{2,} \pi_{3,} \pi_4) \tag{8.49}$$

Gives the frequency at which each base occurs at each site and the *Rate Matrix:*

$$Q = \begin{pmatrix} -(x_1 + x_2 + x_3) & x_1 & x_2 & x_3 \\ \frac{\pi_1 x_1}{\pi_2} & -(\frac{\pi_1 x_1}{\pi_2} + x_4 + x_5) & x_4 & x_5 \\ \frac{\pi_1 x_2}{\pi_3} & \frac{\pi_2 x_4}{\pi_3} & -(\frac{\pi_1 x_2}{\pi_3} + \frac{\pi_2 x_4}{\pi_3} + x_6) & x_6 \\ \frac{\pi_1 x_3}{\pi_4} & \frac{\pi_2 x_5}{\pi_4} & \frac{\pi_3 x_6}{\pi_4} & -(\frac{\pi_1 x_3}{\pi_4} + \frac{\pi_2 x_5}{\pi_4} + \frac{\pi_3 x_6}{\pi_4}) \end{pmatrix} \tag{8.50}$$

As the model must be time reversible, must approach the equilibrium nucleotide (base) frequencies at long times, each rate below the diagonal equals the reciprocal rate above the diagonal multiplied by the equilibrium ratio of the two bases.

As such, the nucleotide GTR requires 6 substitution rate parameters and 4 equilibrium base frequency parameters. All 4 frequency parameters must sum to 1 and there are only 3 free frequency parameters. The total of nine free parameters is further reduced to 8 parameters plus μ, the overall number of substitutions per unit time.

When measuring time in substitutions (μ=1) only 8 free parameters remain. To compute the number of parameters we count the number of entries above the diagonal in the matrix for n trait values per site:

$$\frac{n^2-n}{2} \tag{8.51}$$

Then I add *n-1* for the equilibrium frequencies, and subtract 1 because μ is fixed.
I get:

$$\frac{n^2-n}{2}+(n-1)-1=\frac{1}{2}n^2+\frac{1}{2}n\ \ -2 \tag{8.52}$$

Description of the 6th Step of implementation

I developed and implemented a software pipeline by which I computed the ***DNA Public Keys.***
Public-key algorithms are based on mathematical functions, rather than on substitution and permutation and involve the use of two separate keys in contrast to symmetric encryption, (Fig. 44).

For every alignment column I calculated the likelihood under two evolutionary models: a "foreground" and a "background" model.
The background model assumes a rate model, (Felsenstein 1981), parameterized by the branch lengths of the phylogenetic tree:
w is a vector of nucleotide frequencies, with w_α the frequency of nucleotide α,
$r_{\alpha\beta}$ -the rate of substitution from base β to base α which is proportional to w_α, independent of β.

For every background evolution models I have a corresponding foreground model. The difference between the foreground model and background model is that the background model assumes that all positions have substitutions from base β to base α at the same rate $r_{\alpha\beta}\lambda\omega_\alpha$

In the foreground model I assume that, at a position *i*, the substitution rates $r^i{}_{\alpha\beta}\lambda\omega^i{}_\alpha$ are altered because of specific selection preferences for given bases at this position, parameterized by nucleotide frequencies $\omega^i{}_\alpha$. The parameters $\omega^i{}_\alpha$, at each position are unknown.
For every alignment column of the reference species in genes, I calculated the ratio *R,* representing the likelihoods of foreground and background evolutionary models.
Halpern and Bruno in 1998 estimated the evolutionary distances from coding sequences taking into account protein-level selection.

The equilibrium frequencies determine the maximum dissimilarity expected for diverged and functionally sequences (Molina, Nimwegen 2008).

Halpern and Bruno introduced a codon-level model of coding sequence evolution, they demonstrated the importance of modeling this behavior and the model produced linear distance estimated over a wide range of distances. If *r* is the rate of substitution from a base a to a base b at position *i*, μ is the rate of mutation from a to b and *w* is the equilibrium frequency of nucleotide i, at this position, (Halpern AL, Bruno WJ, 1998).

Following Golding and Felsenstein (1990), Halpern and Bruno (1998) who have shown that mutation limit of the standard Kimura-Ohta theory, we can determine substitution rates in terms of the mutation rates and the equilibrium frequencies $\omega^i{}_\alpha$ if $r^i{}_{\alpha\beta}$ is the rate of substitution from β to α at position *i*, $\mu_{\alpha\beta}$ the rate of mutation from β to α, and $\omega^i{}_\alpha$ the equilibrium frequency of α at this position, we have (Halpern and Bruno 1998)

$$r^i{}_{\alpha\beta} = \mu_{\alpha\beta} \frac{\log\left[\dfrac{\mu_{\beta\alpha}\omega^i{}_\alpha}{\mu_{\alpha\beta}\omega^i{}_\beta}\right]}{1-\dfrac{\mu_{\alpha\beta}\omega^i{}_\beta}{\mu_{\beta\alpha}\omega^i{}_\alpha}} \tag{8.53}$$

The probability to evolve from nucleotide β in the ancestor to nucleotide α in the descendant over a time *t* is:

$$P_{HB}(\alpha \mid \beta, \mu, \omega^i, t) = (e^{r^i t})_{\alpha\beta} \tag{8.54}$$

μ is the matrix of mutation rates, w^i is the vector of equilibrium frequencies at position *i*, $\mathbf{r}^i$ is the matrix of substitution rates at this position. The exponential matrix $e^{r^i t}$ is calculated by the matrix r^i.
Given the transition probabilities from 8.54 equation and given a phylogenetic tree *T*, I can calculate for an alignment column *C,* the likelihood $L_{HB}(C \mid \omega, \mu, T)$.

Likelihood is the product of transition probabilities $P_{HB}(\alpha \mid \beta, \mu, \omega^i, t)$ for every branch of the tree and summed for all nucleotides for internal nodes, calculated efficiently using the recursive algorithm introduced by (Felsenstein (1981))

The calculation requires the matrix μ and the equilibrium frequencies $\omega^i{}_\alpha$.
Given a prior probability distribution *P*(*w*) over possible equilibrium frequencies I calculate:

$$L_{HB}(C \mid \mu, T) = \int L_{HB}(C \mid \omega, \mu, T) P(w) dw \tag{8.55}$$

The integral is over all vectors *w*, that $w_\alpha \geq 0$ for all α, and $\sum_\alpha w_\alpha = 1$.
For approximating Halpern and Bruno model with a simpler model I used the substitution rate model introduced by (Felsenstein, 1981), known as the F81 model.

$$r^i{}_{\alpha\beta} = \mu w^i{}_\alpha \tag{8.56}$$

In 1981, Felsenstein introduced the model in which the substitution rates correspond to the equilibrium frequency of the target nucleotide. In the F81 model the simplification is that the substitution rate is dependent only on the identity of the target base. In the F81 mutation rate is a single parameter μ and subsumes mutational biases and position-dependent selection into the position-dependent equilibrium frequencies w^i_α, assuming equal rates of mutations; at position *i*, the probability of a mutation to base α has a probability w^i_α.

The probability *P*(α|β, *t*, *w*), to evolve from ancestral base β to base α over a time *t* is:

$$P_{F81}(\alpha \mid \beta, q, w) = e^{-\mu t}\delta_{\alpha\beta} + (1 - e^{-\mu t})w_\alpha \tag{8.57}$$

To calculate the likelihood $L_{F81}(C|\mu, T)$ of an alignment column *C,* I calculate the integral:

$$L_{F81}(C \mid \mu, T) = \int L_{F81}(C \mid \mu, T)P(w)dw \tag{8.58}$$

I use standard Dirichlet priors of the form:

$$P(w)\alpha \prod_\alpha (w_\alpha)^{\lambda_\alpha - 1} \tag{8.59}$$

λ_α is the pseudocounts. The likelihood $L_{F81}(C|w, \mu, T)$ is polynomial in the equilibrium frequencies w_α. I calculated the integral, by using general identity:

$$\int \prod_\alpha (w_\alpha)^{n_\alpha - 1} dw = \frac{\prod_\alpha \Gamma(n_\alpha)}{\Gamma(\sum_\alpha n_\alpha)} \tag{8.60}$$

I used a simplified version of the general Halpern-Bruno model, the F81 Model. I calculate the likelihood $L_{F81}(C|\mu, T)$ of any alignment column *C* as a function of the mutation rate μ and phylogenetic tree *T*.

The likelihoods of foreground model and background model for each branch of the tree depend on the product μ*t* of mutation rate μ and branch length *t*.

To estimate the distances between every pair of species I compute the branch lengths *t* for each branch of the tree.

In every multiple-alignment column I calculated the likelihood for the foreground model and likelihood for the background models.

In the genome of the reference species for each alignment column *C, I* computed the likelihoods $L_{fg}(C|c)$ and $L_{bg}(C|c)$ of the foreground and background models. The likelihood ratio:

$$R(C \mid c) = \frac{L_{fg}(C \mid c)}{L_{bg}(C \mid c)} \tag{8.61}$$

shows that column *C* is evolved with different substitution rates from the background model.

In Table 4, Second Colum (C2) model represents the computed *DNA Public Keys*, with respect to Colum C1 and assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, (Table 4) and represents the public keys.

Given the transition probabilities and given a phylogenetic tree, I calculated the likelihood ratio for an alignment column C3/C2, which is the product over transition probabilities for each branch of the tree, summed over all possible nucleotides for the internal nodes and calculated using the recursive algorithm introduced by Felsenstein (1981).

The columns in fig 44, present private/public pair of DNA cryptographic keys and mathematical show the selection for different classes of silent position within genes, (Fig.44).

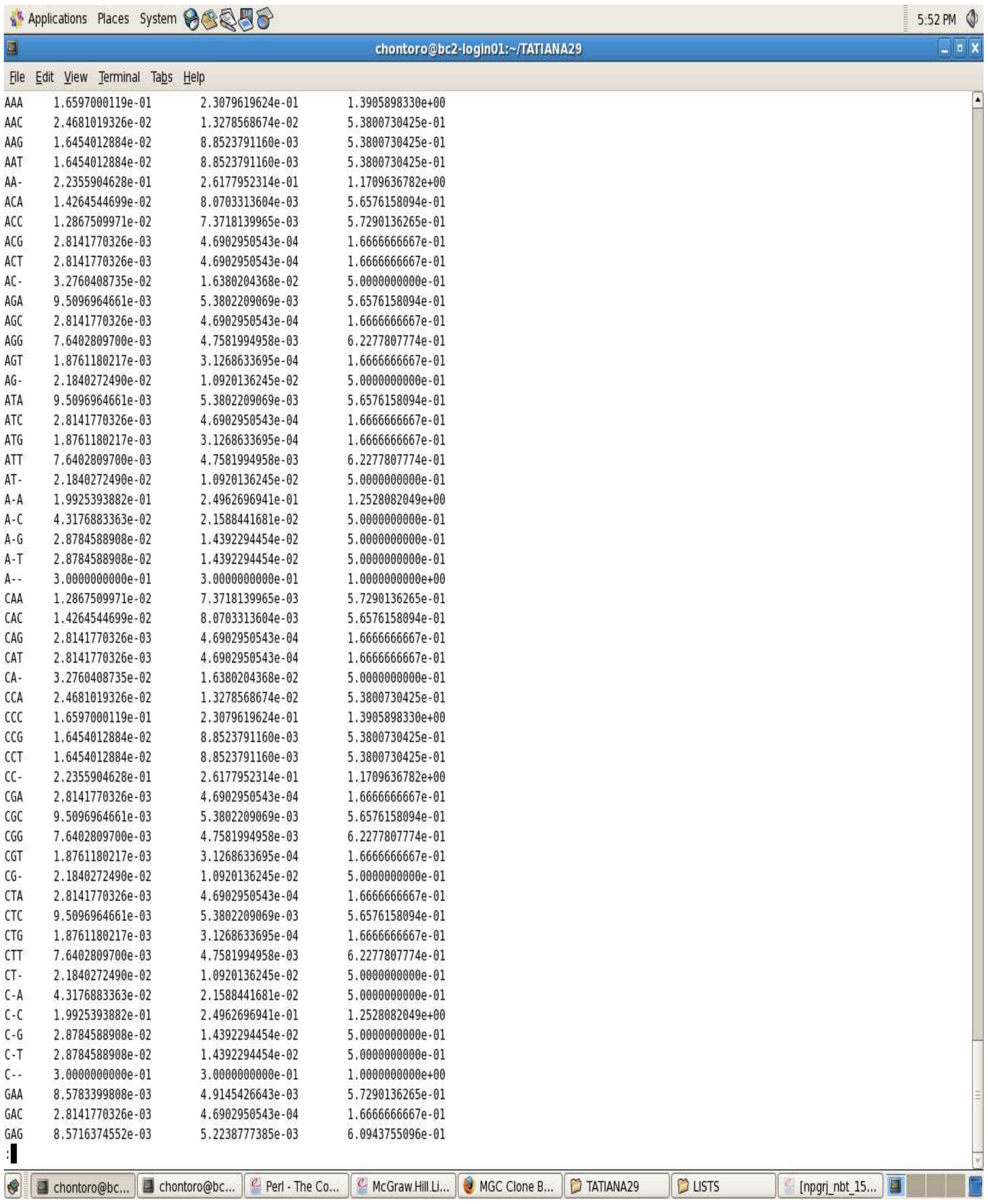
The first column C1 represents all possible three base sequences with respect to human species. Second Colum (C2) model, with respect to C1 assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, (Table 4 and fig 44) and represents the public key.

The third Colum (C3) assumes that at a given position substitution rates are altered because of selection preferences for a base. The last Colum is the likelihood ratio C3/C2 and represents the private key, (Hodorogea, Ionas, 2012).

Description of the 7th Step of implementation

I developed and implemented a software pipeline by which I computed the ***DNA Private Keys.*** The third Colum (C3) assumes that at a given position, the substitution rates are altered during due to specific selection preferences for a certain base, (Hodorogea, Ionas 2012).

Given the transition probabilities and given a phylogenetic tree, I calculate the ratio for an alignment column C3/C2, which is the product over transition probabilities for each branch of the tree, summed over all possible nucleotides for the internal nodes, and can be calculated efficiently using the recursive algorithm introduced by Felsenstein (1981).

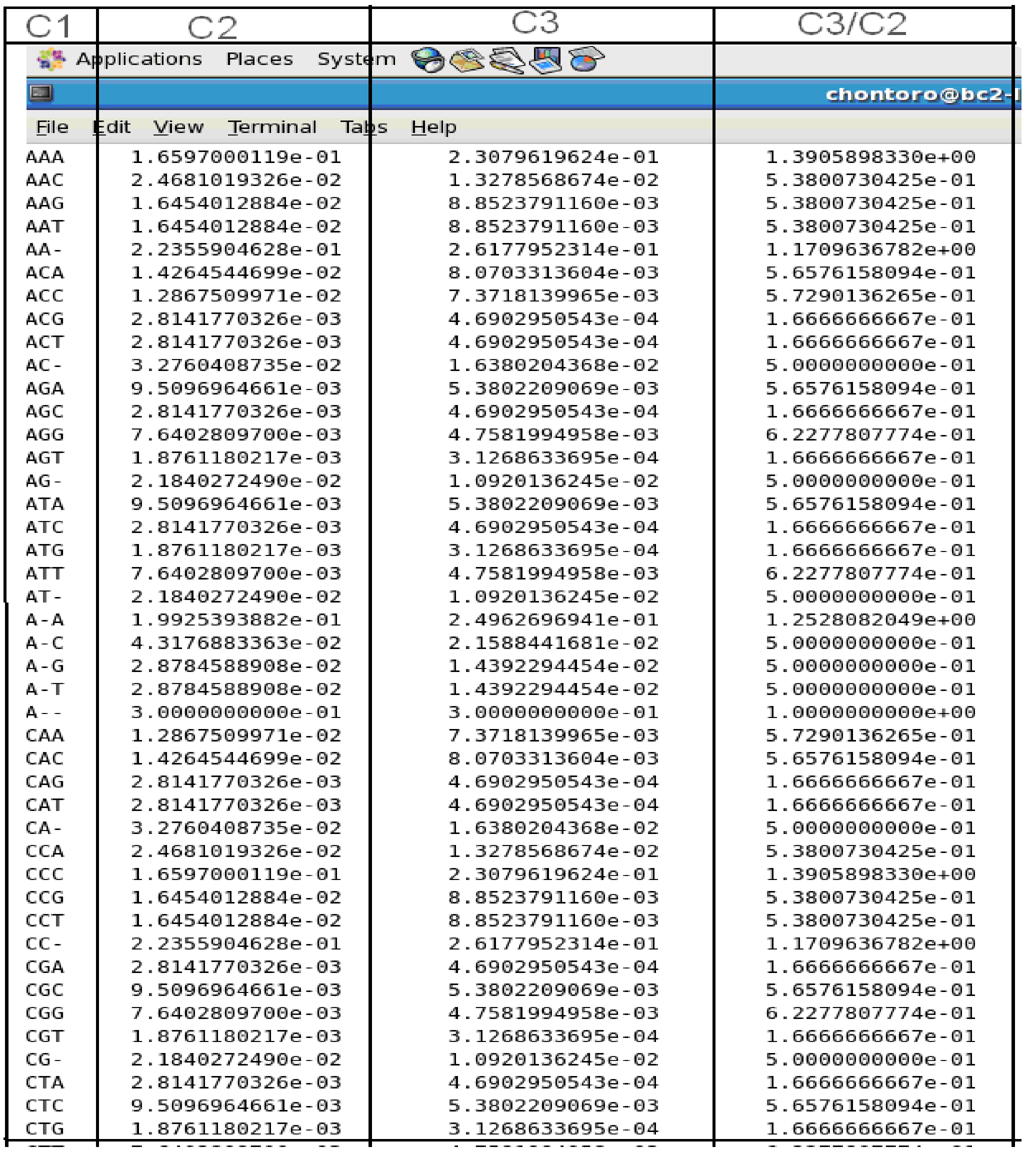


```
AAA     1.6597000119e-01        2.3079619624e-01        1.3905898330e+00
AAC     2.4681019326e-02        1.3278568674e-02        5.3800730425e-01
AAG     1.6454012884e-02        8.8523791160e-03        5.3800730425e-01
AAT     1.6454012884e-02        8.8523791160e-03        5.3800730425e-01
AA-     2.2355904628e-01        2.6177952314e-01        1.1709636782e+00
ACA     1.4264544699e-02        8.0703313604e-03        5.6576158094e-01
ACC     1.2867509971e-02        7.3718139965e-03        5.7290136265e-01
ACG     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
ACT     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
AC-     3.2760408735e-02        1.6380204368e-02        5.0000000000e-01
AGA     9.5096964661e-03        5.3802209069e-03        5.6576158094e-01
AGC     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
AGG     7.6402809700e-03        4.7581994958e-03        6.2277807774e-01
AGT     1.8761180217e-03        3.1268633695e-04        1.6666666667e-01
AG-     2.1840272490e-02        1.0920136245e-02        5.0000000000e-01
ATA     9.5096964661e-03        5.3802209069e-03        5.6576158094e-01
ATC     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
ATG     1.8761180217e-03        3.1268633695e-04        1.6666666667e-01
ATT     7.6402809700e-03        4.7581994958e-03        6.2277807774e-01
AT-     2.1840272490e-02        1.0920136245e-02        5.0000000000e-01
A-A     1.9925393882e-01        2.4962696941e-01        1.2528082049e+00
A-C     4.3176883363e-02        2.1588441681e-02        5.0000000000e-01
A-G     2.8784588908e-02        1.4392294454e-02        5.0000000000e-01
A-T     2.8784588908e-02        1.4392294454e-02        5.0000000000e-01
A--     3.0000000000e-01        3.0000000000e-01        1.0000000000e+00
CAA     1.2867509971e-02        7.3718139965e-03        5.7290136265e-01
CAC     1.4264544699e-02        8.0703313604e-03        5.6576158094e-01
CAG     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
CAT     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
CA-     3.2760408735e-02        1.6380204368e-02        5.0000000000e-01
CCA     2.4681019326e-02        1.3278568674e-02        5.3800730425e-01
CCC     1.6597000119e-01        2.3079619624e-01        1.3905898330e+00
CCG     1.6454012884e-02        8.8523791160e-03        5.3800730425e-01
CCT     1.6454012884e-02        8.8523791160e-03        5.3800730425e-01
CC-     2.2355904628e-01        2.6177952314e-01        1.1709636782e+00
CGA     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
CGC     9.5096964661e-03        5.3802209069e-03        5.6576158094e-01
CGG     7.6402809700e-03        4.7581994958e-03        6.2277807774e-01
CGT     1.8761180217e-03        3.1268633695e-04        1.6666666667e-01
CG-     2.1840272490e-02        1.0920136245e-02        5.0000000000e-01
CTA     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
CTC     9.5096964661e-03        5.3802209069e-03        5.6576158094e-01
CTG     1.8761180217e-03        3.1268633695e-04        1.6666666667e-01
CTT     7.6402809700e-03        4.7581994958e-03        6.2277807774e-01
CT-     2.1840272490e-02        1.0920136245e-02        5.0000000000e-01
C-A     4.3176883363e-02        2.1588441681e-02        5.0000000000e-01
C-C     1.9925393882e-01        2.4962696941e-01        1.2528082049e+00
C-G     2.8784588908e-02        1.4392294454e-02        5.0000000000e-01
C-T     2.8784588908e-02        1.4392294454e-02        5.0000000000e-01
C--     3.0000000000e-01        3.0000000000e-01        1.0000000000e+00
GAA     8.5783399808e-03        4.9145426643e-03        5.7290136265e-01
GAC     2.8141770326e-03        4.6902950543e-04        1.6666666667e-01
GAG     8.5716374552e-03        5.2238777385e-03        6.0943755096e-01
:
```

Fig. 44 Private/public pair of DNA cryptographic keys

The last Colum is the ratio C3/C2 and represents the private key, (Table 4).

Table 4, Public/Private DNA Cryptographic Keys, [60]



C1	C2	C3	C3/C2
AAA	1.6597000119e-01	2.3079619624e-01	1.3905898330e+00
AAC	2.4681019326e-02	1.3278568674e-02	5.3800730425e-01
AAG	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
AAT	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
AA-	2.2355904628e-01	2.6177952314e-01	1.1709636782e+00
ACA	1.4264544699e-02	8.0703313604e-03	5.6576158094e-01
ACC	1.2867509971e-02	7.3718139965e-03	5.7290136265e-01
ACG	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
ACT	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
AC-	3.2760408735e-02	1.6380204368e-02	5.0000000000e-01
AGA	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
AGC	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
AGG	7.6402809700e-03	4.7581994958e-03	6.2277807774e-01
AGT	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
AG-	2.1840272490e-02	1.0920136245e-02	5.0000000000e-01
ATA	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
ATC	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
ATG	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
ATT	7.6402809700e-03	4.7581994958e-03	6.2277807774e-01
AT-	2.1840272490e-02	1.0920136245e-02	5.0000000000e-01
A-A	1.9925393882e-01	2.4962696941e-01	1.2528082049e+00
A-C	4.3176883363e-02	2.1588441681e-02	5.0000000000e-01
A-G	2.8784588908e-02	1.4392294454e-02	5.0000000000e-01
A-T	2.8784588908e-02	1.4392294454e-02	5.0000000000e-01
A--	3.0000000000e-01	3.0000000000e-01	1.0000000000e+00
CAA	1.2867509971e-02	7.3718139965e-03	5.7290136265e-01
CAC	1.4264544699e-02	8.0703313604e-03	5.6576158094e-01
CAG	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
CAT	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
CA-	3.2760408735e-02	1.6380204368e-02	5.0000000000e-01
CCA	2.4681019326e-02	1.3278568674e-02	5.3800730425e-01
CCC	1.6597000119e-01	2.3079619624e-01	1.3905898330e+00
CCG	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
CCT	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
CC-	2.2355904628e-01	2.6177952314e-01	1.1709636782e+00
CGA	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
CGC	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
CGG	7.6402809700e-03	4.7581994958e-03	6.2277807774e-01
CGT	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
CG-	2.1840272490e-02	1.0920136245e-02	5.0000000000e-01
CTA	2.8141770326e-03	4.6902950543e-04	1.6666666667e-01
CTC	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
CTG	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01

Using the same model and desired length of bases from the first column I can derive the public/private keys used in Java KeyStore with respect to human or desired species. Resulting in new set of desired public/private *DNA Cryptographic Keys for Java DNA KeyStore usage, (Table 4).*

8.2 Unique Process System for Evolutionary Models

I developed a Unique Process System for Evolutionary Models of deriving DNA Cryptographic Keys Sequences by deriving the DNA secret keys from human genome analysis by computing the philogenetic tree relating and the branch length during evolution for chosen species.

In this Unique Process System for Evolutionary Models developed and implemented I extract and aligning the DNA Cryptographic Keys Sequences of the same gene from related chosen species with respect to Human DNA Sequences, [60].

The molecular evolution model is a complex process, involving mutational biases, interactions, heterogeneous recombination rates, population mixing patterns, time-dependent selection, and frequency-dependent selection, [96].

These models assign probabilities to multiple-alignment columns in terms of the substitution rates and lengths of the phylogenetic tree branches.

Following Golding and Felsenstein (1990), Halpern and Bruno (1998) who have shown that mutation limit of the standard Kimura-Ohta theory I can uniquely determine substitution rates in terms of the mutation rates and the equilibrium frequencies, [48], [83].

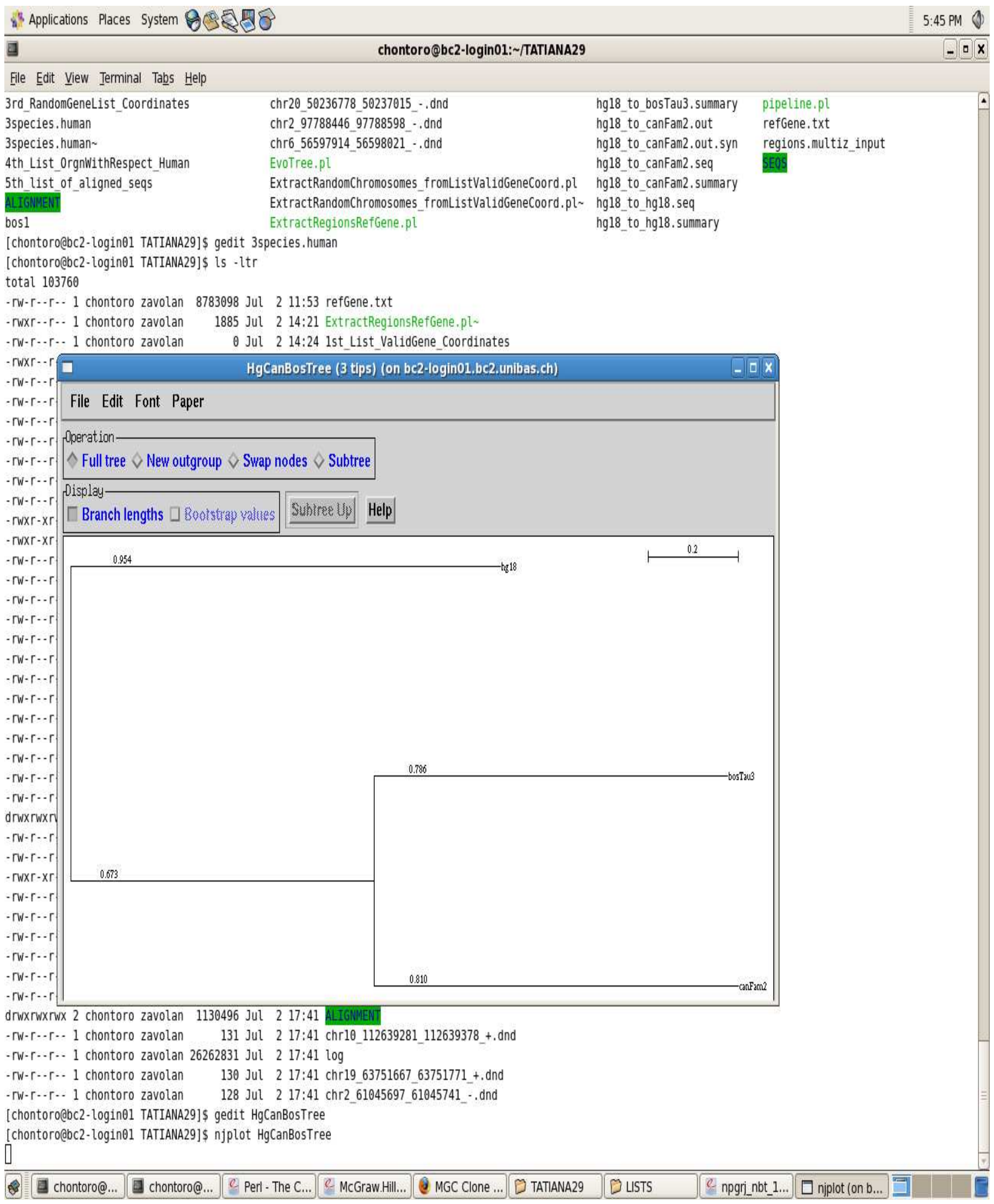
If r is the rate of substitution from a base a to a base b at position i, μ is the rate of mutation from a to b and w is the equilibrium frequency of nucleotide i, at this position, [48].

I computed the philogenitic tree (for my three chosen species) and the branch length during evolution, (Fig 45) for my chosen species with respect to human (hg18).

There are different approaches to detect natural selection from sequence data, [87].

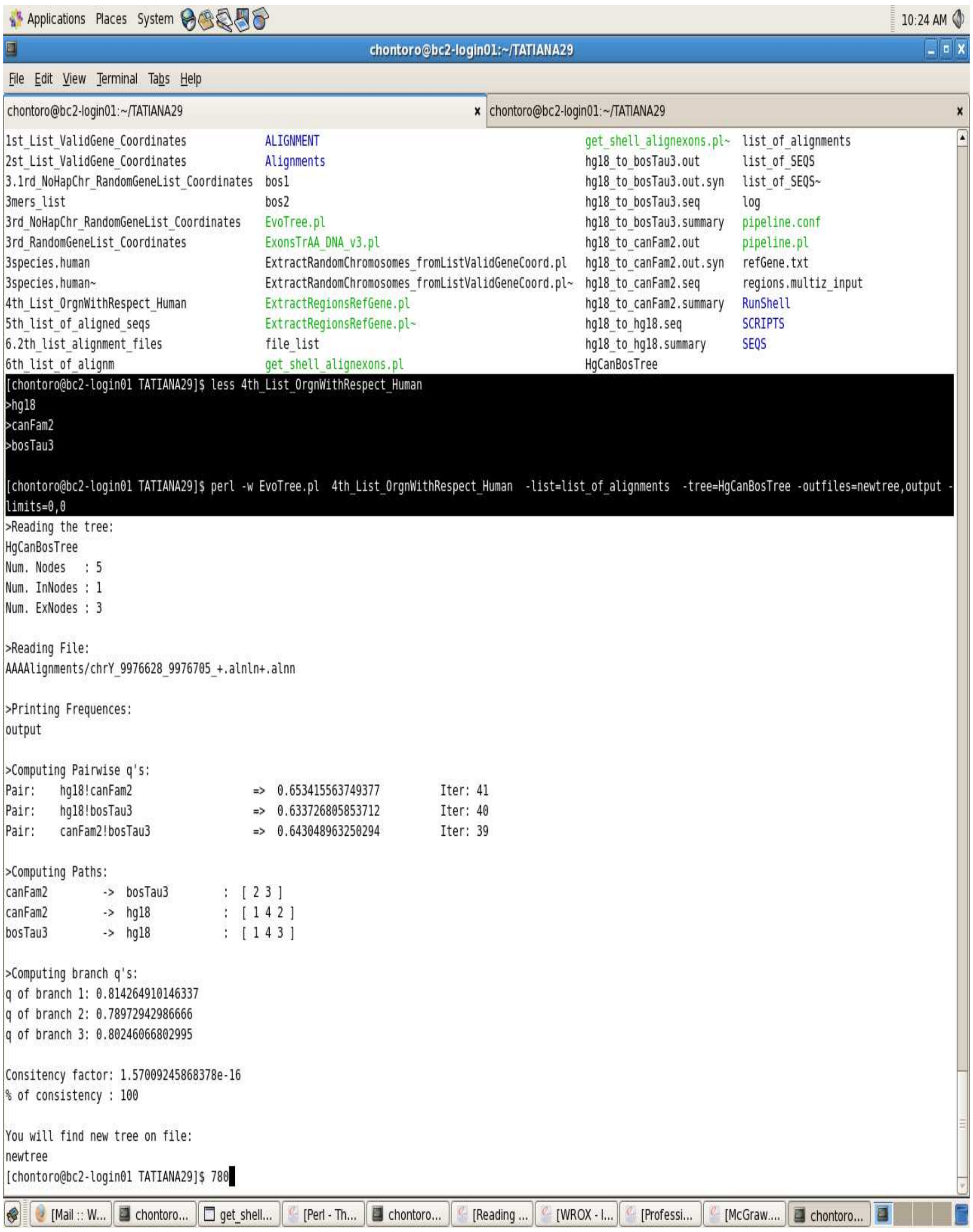
Detecting sequence substitutions positively selected, requires the comparison of polymorphism data in a given species with substitution data in a related species, [87].

I used conservation statistics of my multiple alignments for coding orthologous DNA sequences from related species to infer the philogenetic tree.



```
3rd_RandomGeneList_Coordinates      chr20_50236778_50237015_-.dnd                         hg18_to_bosTau3.summary   pipeline.pl
3species.human                      chr2_97788446_97788598_-.dnd                          hg18_to_canFam2.out       refGene.txt
3species.human~                     chr6_56597914_56598021_-.dnd                          hg18_to_canFam2.out.syn   regions.multiz_input
4th_List_OrgnWithRespect_Human      EvoTree.pl                                            hg18_to_canFam2.seq       SEQS
5th_list_of_aligned_seqs            ExtractRandomChromosomes_fromListValidGeneCoord.pl    hg18_to_canFam2.summary
ALIGNMENT                           ExtractRandomChromosomes_fromListValidGeneCoord.pl~   hg18_to_hg18.seq
bos1                                ExtractRegionsRefGene.pl                              hg18_to_hg18.summary
[chontoro@bc2-login01 TATIANA29]$ gedit 3species.human
[chontoro@bc2-login01 TATIANA29]$ ls -ltr
total 103760
-rw-r--r-- 1 chontoro zavolan  8783098 Jul  2 11:53 refGene.txt
-rwxr--r-- 1 chontoro zavolan     1885 Jul  2 14:21 ExtractRegionsRefGene.pl~
-rw-r--r-- 1 chontoro zavolan        0 Jul  2 14:24 1st_List_ValidGene_Coordinates
-rwxr--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rwxr-xr
-rwxr-xr
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
drwxrwxr
-rw-r--r
-rw-r--r
-rwxr-xr
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
-rw-r--r
drwxrwxrwx 2 chontoro zavolan  1130496 Jul  2 17:41 ALIGNMENT
-rw-r--r-- 1 chontoro zavolan      131 Jul  2 17:41 chr10_112639281_112639378_+.dnd
-rw-r--r-- 1 chontoro zavolan 26262831 Jul  2 17:41 log
-rw-r--r-- 1 chontoro zavolan      130 Jul  2 17:41 chr19_63751667_63751771_+.dnd
-rw-r--r-- 1 chontoro zavolan      128 Jul  2 17:41 chr2_61045697_61045741_-.dnd
[chontoro@bc2-login01 TATIANA29]$ gedit HgCanBosTree
[chontoro@bc2-login01 TATIANA29]$ njplot HgCanBosTree
```

Fig. 45 Philogenetic tree and the branch length, [60]



```
1st_List_ValidGene_Coordinates               ALIGNMENT                                      get_shell_alignexons.pl~  list_of_alignments
2st_List_ValidGene_Coordinates               Alignments                                     hg18_to_bosTau3.out       list_of_SEQS
3.1rd_NoHapChr_RandomGeneList_Coordinates    bos1                                           hg18_to_bosTau3.out.syn   list_of_SEQS~
3mers_list                                   bos2                                           hg18_to_bosTau3.seq       log
3rd_NoHapChr_RandomGeneList_Coordinates      EvoTree.pl                                     hg18_to_bosTau3.summary   pipeline.conf
3rd_RandomGeneList_Coordinates               ExonsTrAA_DNA_v3.pl                            hg18_to_canFam2.out       pipeline.pl
3species.human                               ExtractRandomChromosomes_fromListValidGeneCoord.pl   hg18_to_canFam2.out.syn   refGene.txt
3species.human~                              ExtractRandomChromosomes_fromListValidGeneCoord.pl~  hg18_to_canFam2.seq       regions.multiz_input
4th_List_OrgnWithRespect_Human               ExtractRegionsRefGene.pl                       hg18_to_canFam2.summary   RunShell
5th_list_of_aligned_seqs                     ExtractRegionsRefGene.pl~                      hg18_to_hg18.seq          SCRIPTS
6.2th_list_alignment_files                   file_list                                      hg18_to_hg18.summary      SEQS
6th_list_of_alignm                           get_shell_alignexons.pl                        HgCanBosTree
[chontoro@bc2-login01 TATIANA29]$ less 4th_List_OrgnWithRespect_Human
>hg18
>canFam2
>bosTau3

[chontoro@bc2-login01 TATIANA29]$ perl -w EvoTree.pl  4th_List_OrgnWithRespect_Human  -list=list_of_alignments  -tree=HgCanBosTree -outfiles=newtree,output -
limits=0,0
>Reading the tree:
HgCanBosTree
Num. Nodes   : 5
Num. InNodes : 1
Num. ExNodes : 3

>Reading File:
AAAAlignments/chrY_9976628_9976705_+.alnln+.alnn

>Printing Frequences:
output

>Computing Pairwise q's:
Pair:    hg18!canFam2                 =>  0.653415563749377        Iter: 41
Pair:    hg18!bosTau3                 =>  0.633726805853712        Iter: 40
Pair:    canFam2!bosTau3              =>  0.643048963250294        Iter: 39

>Computing Paths:
canFam2        ->  bosTau3        :  [ 2 3 ]
canFam2        ->  hg18           :  [ 1 4 2 ]
bosTau3        ->  hg18           :  [ 1 4 3 ]

>Computing branch q's:
q of branch 1: 0.814264910146337
q of branch 2: 0.78972942986666
q of branch 3: 0.80246066802995

Consitency factor: 1.57009245868378e-16
% of consistency : 100

You will find new tree on file:
newtree
[chontoro@bc2-login01 TATIANA29]$ 780
```

Fig. 46 Computing the philogenetic tree and branch length

The Perl soft developed, reeds the tree, computes the pairwise alignment, computes the branches of the tree for our chosen mammalian species, (Fig.46), [59].

A public-key encryption scheme is vulnerable to brute-force attack and depends on the use of some sort of invertible mathematical function, [99].

The most widely used public-key cryptosystem is RSA and the difficulty of attacking RSA is based on the difficulty of finding the prime factors of a composite number and with a very small public key, RSA becomes vulnerable to a simple attack.

A problem that can be solved in polynomial time is considered feasible and exponential time is considered infeasible.

Compared to existed public-key algorithms where the difficulty of attacking is easy, based on the problem Y=f(x) and finding the problem solution can be solved easily in polynomial time as a function of input length, the time to compute the function is proportional to n^a, where *a* is a fixed constant, [99].

In my case the difficulty of attacking the derived DNA asymmetric cryptographic keys is based on the mathematical functions of probability theory and the complexity to compute the key for a cryptanalyst is NP complete (which means it can be not solved in exponential time) and the complexity of breaking the algorithm with brute force attack is proportional with the quantity of possible keys depending exponential of the key length.

According to Stalings: a common measure of the efficiency of an algorithm is its time complexity defined as f(*n*) if, for all *n* and all inputs of length *n*, the execution of the algorithm takes at most f(*n*) steps.

The difficulty of attacking the derived DNA asymmetric cryptographic keys is based on the difficulty to solve the infeasible problem $X = f^{-1}(Y)$, as the problem to be solved is infeasible as the effort to solve the problem grows faster than polynomial time as a function of input size, resulting in the best algorithms time complexity efficiency compared to the efficiency of the exited algorithms, [99].

8.3 SmartCipher Application Integrating DNAProvider with DNA Encryption (DNAE) System

I implemented a cryptographic package provider, named DNAProvider as Java Cryptographic Extension (JCE), which allowed me to extend the JCE by implementing faster and more secure DNA cryptographic algorithm.

I implemented the DNA cryptographic algorithm which encodes the records of an individual in DNA data strand flanked by unique primer sequences which represent the DNA Cryptographic Keys Sequences
In the DNA Encryption (DNAE) system I derive the DNA Cryptographic Keys based on Evolutionary Models.

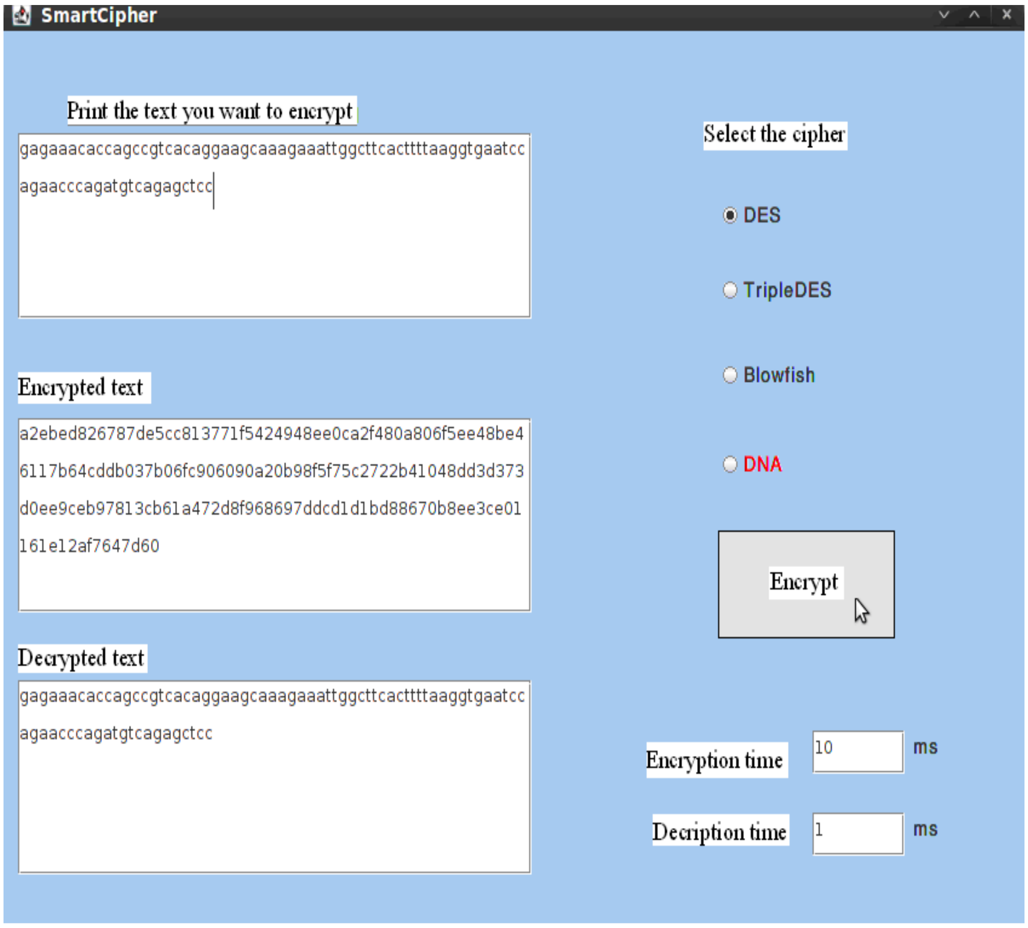
I developed, implemented and tested a novel idea and algorithm: a UniqPprocess System Pipeline Evolutionary Models of deriving DNA Cryptographic Keys Sequences by deriving the DNA secret keys from human genome analysis.

I developed and implemented the UniqPprocess System Pipeline Evolutionary Models with which I extracted and aligned the DNA Cryptographic Keys Sequences of the same gene from related chosen species with respect to Human DNA Sequences.

I realized the alignment in the evolutionary system pipeline of DNA Cryptographic Keys Sequences with trained ProbCons tool, a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy, (described in all details in Chapter 7). Sun Microsystems certified and signed our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm. We got the Code Signing Certificate from Sun Microsystems for our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm which is available for 5 years, until with the reference #679, when renewing it.

Based on this in Helios research team from Technical University of Cluj-Napoca, we developed in Java Language a software application named SmartCipher where I integrated our DNAProvider with DNA Encryption (DNAE) system based on the Central Dogma of Molecular Biology (CDMB).
The SmartCipher application has a following interface, (Fig. 47) and integrates my DNAProvider in it.

The interface was developed using Java IDE Jigloo GUI Builder plugin for Eclipse, containing a CTextArea where the user can introduce the desired text and desired algorithm for encryption: Des, TripleDes, Blowfish and DNA,



Print the text you want to encrypt

gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc

Encrypted text

a2ebed826787de5cc813771f5424948ee0ca2f480a806f5ee48be4
6117b64cddb037b06fc906090a20b98f5f75c2722b41048dd3d373
d0ee9ceb97813cb61a472d8f968697ddcd1d1bd88670b8ee3ce01
161e12af7647d60

Decrypted text

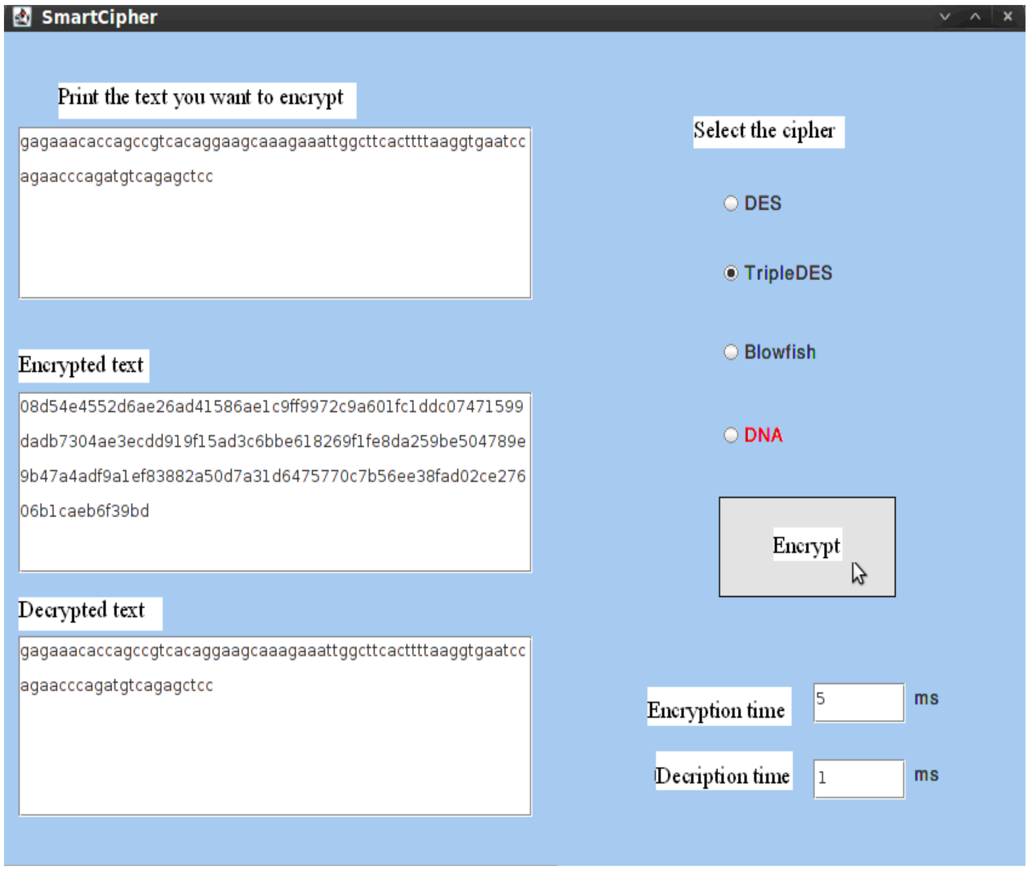
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc

[✓ DES] TripleDES Blowfish DNA

Encryption time 10 ms

Deciption time 1 ms

Fig. 47 SmartCipher Interface –DES Encryption/Decryption



Print the text you want to encrypt

```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

Encrypted text

```
08d54e4552d6ae26ad41586ae1c9ff9972c9a601fc1ddc07471599
dadb7304ae3ecdd919f15ad3c6bbe618269f1fe8da259be504789e
9b47a4adf9a1ef83882a50d7a31d6475770c7b56ee38fad02ce276
06b1caeb6f39bd
```

Decrypted text

```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

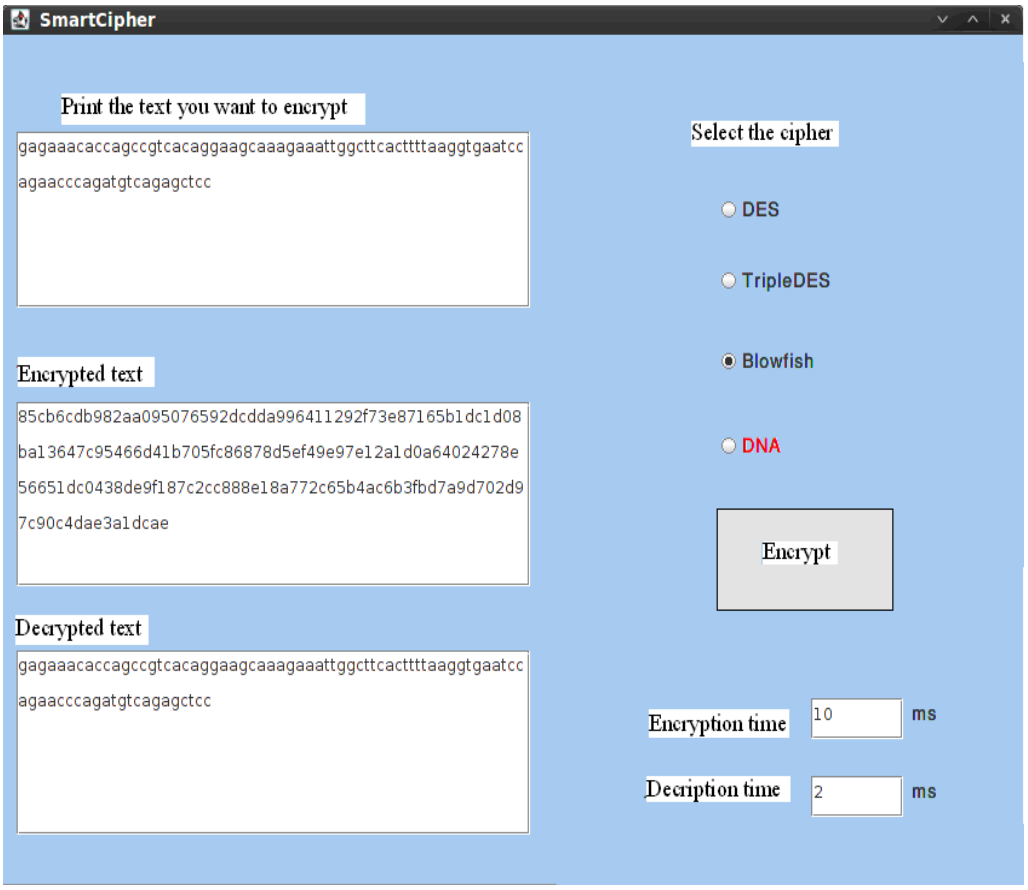
Select the cipher

- ○ DES
- ● TripleDES
- ○ Blowfish
- ○ DNA

Encryption time: 5 ms

Decription time: 1 ms

Fig. 48 SmartCipher Interface –TripleDES Encryption/Decryption



Print the text you want to encrypt

```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

Encrypted text

```
85cb6cdb982aa095076592dcdda996411292f73e87165b1dc1d08
ba13647c95466d41b705fc86878d5ef49e97e12a1d0a64024278e
56651dc0438de9f187c2cc888e18a772c65b4ac6b3fbd7a9d702d9
7c90c4dae3a1dcae
```

Decrypted text

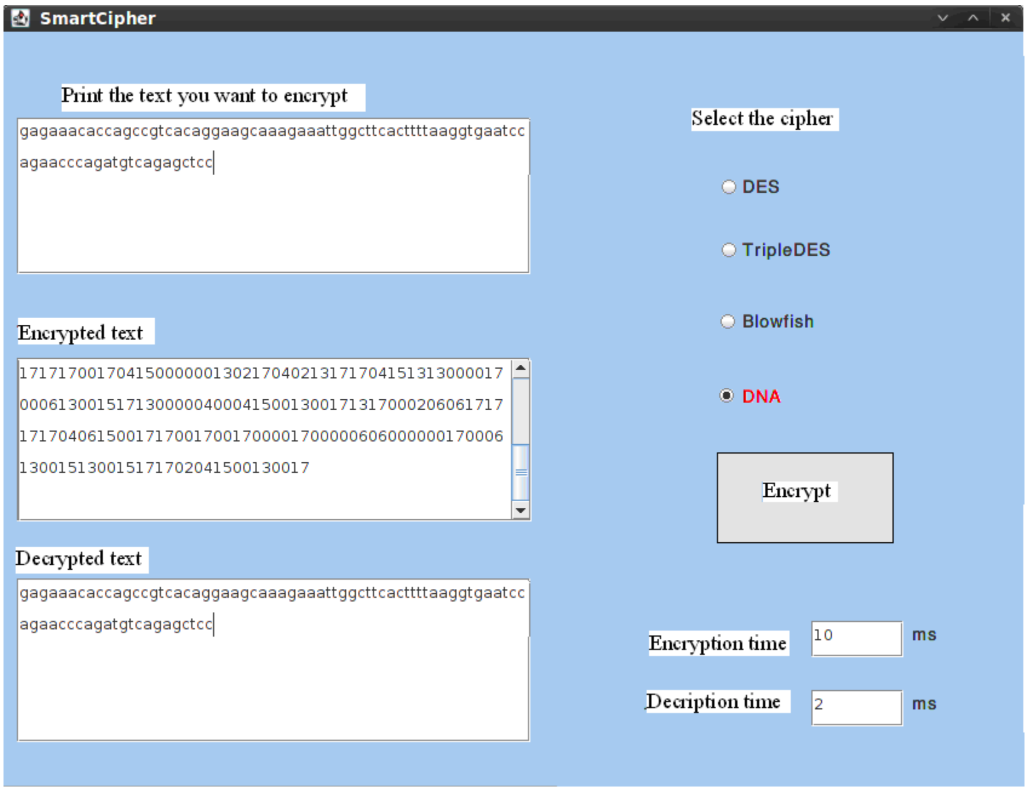
```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

Select the cipher: DES / TripleDES / [✓ Blowfish] / DNA

Encryption time: 10 ms

Decription time: 2 ms

Fig. 49 SmartCipher Interface – Blowfish Encryption/Decryption



Print the text you want to encrypt

```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

Encrypted text

```
171717001704150000001302170402131717041513130000017
000613001517130000040004150013001713170002060617 17
1717040615001717001700170000170000060600000017000 6
13001513001517170204150013001 7
```

Decrypted text

```
gagaaacaccagccgtcacaggaagcaaagaaattggcttcacttttaaggtgaatcc
agaacccagatgtcagagctcc
```

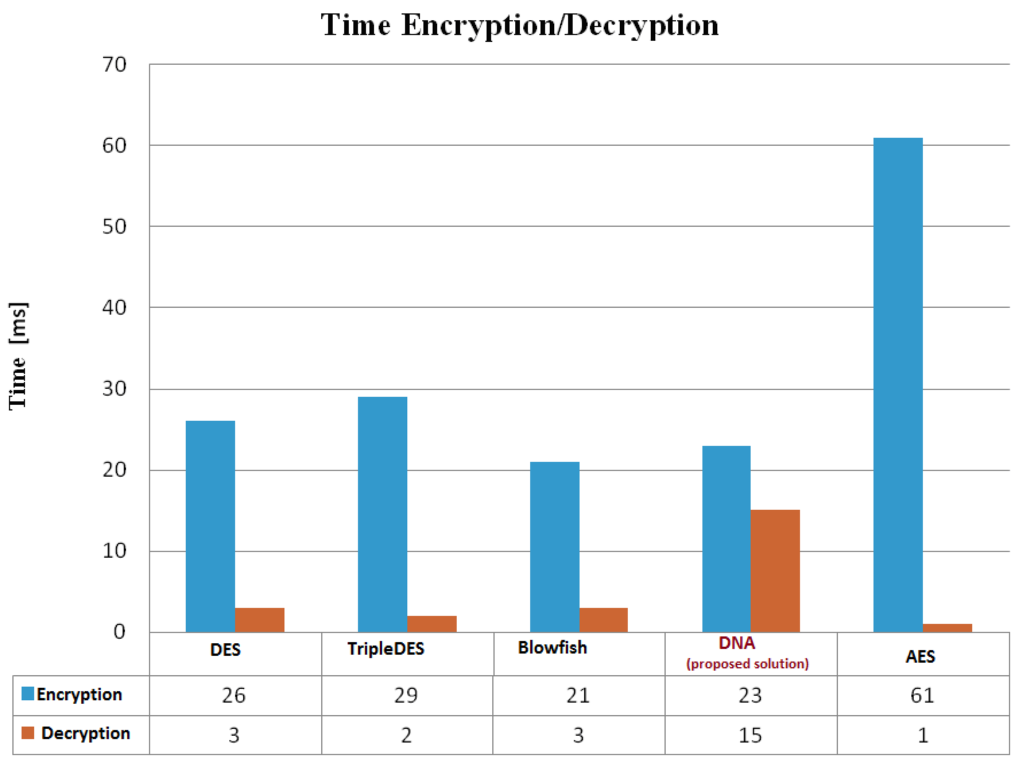
[✓ DNA]

Encryption time 10 ms

Decription time 2 ms

Fig. 50 SmartCipher Interface –DNA Encryption/Decryption

SmartCipher application where I integrated my DNAProvider with DNA Encryption (DNAE) system based on the Central Dogma of Molecular Biology (CDMB), and association between classes and UML diagrams, (Anex.1 Fig.52-54). Tests were executed on a Intel Pentium 4 CPU, 3.00 GHz, RAM: 1,5GB,OS: Ubuntu 10.04.



	DES	TripleDES	Blowfish	DNA (proposed solution)	AES
🟦 Encryption	26	29	21	23	61
🟧 Decryption	3	2	3	15	1

Fig. 51 Necessary time for encryption decryption for different ciphers

DNA encryption needs a shorter time for encryption compare with algorithms: DES, TripleDES, AES and a longer time for decryption process, (Fig. 51).

Increased time is explained by the string conversion to array of bytes and vice versa as for implemented cryptographic algorithm the cryptographic provider implements the abstract methods of SPI class processing array of bytes but the implemented DNA cryptographic algorithm process array of strings, the strength and efficiency of the proposed solution is in its time complexity defined as f(n) if, for all n and all inputs of length n, the execution of the algorithm takes at most f(n) steps.

9. Conclusions – Personal Contributions- Future Directions

The thesis consists of the very valuable practical work as every novel proposed idea, method and algorithm has been developed implemented and tested. The major results and contribution of this thesis have been published in March, 2012 in the book: "Modern Cryptography".

The main theoretical contributions of the thesis are:

The thesis consists of an important theoretical signification dedicated to the development of the new encryption methods and algorithms using bioinformatics with a higher degree of security and reliability in the encryption systems.

-A study regarding data security in current networks, Internet and Distributed Systems.

-The description of DNA Engineering and Bioinformatics Science which brings together the areas of molecular biology, chemistry, mathematics physics, and computer science.

As my research result I present a short synthesis of my main practical contribution to this thesis:

The aim and motivation of the thesis consisted in the development of the novel modern encryption techniques based on bioinformatic science. I proposed and implemented a new idea of a development of a cryptographic package provider; named *DNAProvider as Java Cryptographic Extension (JCE)* which extends the JCE by implementing a more secure *DNA cryptographic algorithm* based on bioinformatic science.

The originality and scientific novelty of the thesis and obtained results exceed current available methods by security and time complexity, consisting in the development and elaboration of the novel and modern methods for data protection based on bioinformatics, novel ideas and novel algorithms which I implemented and tested to derive the asymmetric DNA Public/Private *Cryptographic Keys* based on mathematics of the *Evolutionary Models.*

1. I proposed a new idea and implemented a DNA *cryptographic algorithm* which encodes the medical records of an individual in *DNA data strand* flanked by unique primer sequences.

2. I proposed a new idea which I implemented and tested the *DNA Encryption System (DNAE)* based on the Central Dogma of Molecular Biology (CDMB).

3. I proposed and implemented a new idea of a development of a cryptographic package provider, named *DNAProvider as Java Cryptographic Extension (JCE)*, extending the JCE by implementing faster and more secure *DNA cryptographic algorithm*.

***4.** Following all the necessary steps required by* Sun Microsystems and as a result of my research work for extending the JCE with DNA Encryption Algorithm as part of Java Cryptographic Security Provider, named DNAProvider, I obtained from Sun

Microsystems, Inc, the: *JCE Code Signing Certificate Issuance, Nr.<679>, valid for 5 years until 01/26/2013.*
5. In collaboration with Helios research team from Technical University of Cluj-Napoca, we developed in Java Language a software application named *SmartCipher* where we integrated my *DNAProvider with DNA Encryption (DNAE)* system based on the Central Dogma of Molecular Biology (CDMB).
6. Using the same modern technologies as bioinformatics and biotechnology in the field of cryptography I developed, implemented and tested a cryptographic package provider ***DNAProvider as Java Cryptographic Extension (JCE) based on the Central Dogma of Molecular Biology***, used for security of software applications.
7. I extended the JCE with DNA Encryption Algorithm as part of Java Cryptographic Security Provider, named ***DNAProvider***, which I use for security of software applications.

***The originality and scientific novelty* of the thesis and obtained results exceed current available methods by efficiency of the algorithm measured in time complexity and security**, consisting in the development and elaboration of the novel and modern methods for data protection based on bioinformatics, novel ideas and novel algorithms which I implemented and tested:
To derive the asymmetric DNA *Cryptographic Public/Private Keys* based on mathematics of the *Evolutionary Models* for use in Public Key Infrastructure
8. I developed and implemented a unique innovative method to derive the cryptographic public/private keys called the DNA Cryptographic Keys based on Evolutionary Models.
-I came with the novel idea and a new algorithm which I implemented and tested: I developed and implemented a unique innovative method to derive the asymmetric DNA *Cryptographic Public/Private Keys* from human genome analysis, based on the complexity of using Bioinformatics, Sequence analysis, Phylogenetic tree analysis and Mathematical functions of Bioinformatics Evolutionary Models, for the author's DNA Encryption (DNAE) system.

The difficulty of attacking the derived DNA asymmetric cryptographic keys is based on the mathematical functions of probability theory and the complexity to compute the key for a cryptanalyst is NP complete (which means it cannot be solved in exponential time) and the complexity of breaking the algorithm with brute force attack is proportional with the quantity of possible keys depending exponential of the key length.

The difficulty of attacking the derived DNA asymmetric cryptographic keys is based: on the difficulty to solve the exponential infeasible problem, as the effort to solve the problem grows faster than polynomial time as a function of input size, resulting in the best algorithms time complexity efficiency compared to the efficiency of the exited algorithms.

My work opens a future direction for research in data security based on bioinformatics which will be in high demand with the advent of high-speed next-generation communication systems and networks.

10. References

1. C. Abad, J. Taylor, C. Sengul, W. Yurcik, Y. Zhou, K. Rowe, Log correlation for intrusion detection,In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003). Los Alamitos, CA: IEEE Computer Society Press, 2003
2. M. Almgren, E. Jonsson, Using active learning in intrusion detection, 17th IEEE Computer Security Foundations Workshop (CSFW'04), Los Alamitos, CA: IEEE Computer Society Press, 2003
3. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, Molecular Biology of the Cell 4th edition, Garland Science, ISBN-10: 0-8153-3218-1, 2002
4. R. Bace, P. Mell, Intrusion detection systems, NIST special publication in intrusion detection systems, from http://csrc.nist,gov/publications/nistpubs/800-31/sp800-31.pdf, 2001
5. D. Bales, Java Programming with Oracle JDBC, O'Reilly, December 2001.
6. E. Biham, A. Shamir, Differential Cryptanalysis of the Full 16-Round DES, Advances in Cryptology, Proceedings of CRYPTO '92, 1992
7. E. Biham, A. Shamir A., Differential Cryptanalysis of the Data Encryption Standard, Springer Verlag, 1993
8. E. Biham, A. Shamir, Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI andLucifer, CRYPTO 1991:
9. B. Blobel, Authorisation and access control for electronic health record system, InternationalJournal of Medical Informatics, 73, 251-257.
10. B. Blobel, P. Hoepner, R. Joop, S. Karnouskos, G. Kleinhuis, G. Stassinopoulos, Using a privilege management infrastructure for secure Web-based e-health applications. Computer Communication, 26(16), 1863-1872, 2003
11. M.B. Eisen et al., Proc. Natl. Acad. Sci. USA 95:14863 14868, National Academy of Sciences,1998
12. M. Bishop, Introduction to Computer Security, Prentice Hall PTR, October 2004
13. A. F. BOREM, R.Santos, Understanding Biotechnology, Prentice Hall, PTR, 2003
14. W. Cheswick, S. Bellovin, A. D Rubin, Firewalls and Internet security: Repelling the wily hacker, Addison-Wesley, 2003
15. CSI Publications, CSI/FBI computer crime and security survey. Retrieved from http://www.GoCSI.com, 2005
16. M. Cade, S. Roberts, Sun Certified Enterprise Architect for J2EE™ Technology Study Guide, Prentice Hall, 2002.
17. J. Chen, R. Deaton, DNA based memory, Nat. Comput. 4(2)2:83-101, 2005
18. R. Colburn, Sams Teach Yourself CGI in 24 Hours, Second Edition, Sams, September 2002.
19. S. Convery, Network Security Architectures, Cisco Press, April 19, 2004. September 12, 2003
20. T. Clelland Catherine, V. Risca, Carter Bancroft, 1999, Hiding Messages in DNA Micodots, Nature Magazine Vol. 399, June 10, 1999.
21. W. Crawford, Java Enterprise in a Nutshell, 3rd Edition, Jim Farley, O'Reilly, November 2005.
22. A. Das, C. E. Veni Madhavan, Public-key Cryptography, Theory and Practice, Pub, Pearson Education India, ISBN: 81-3170-832-2, 2009

23. J. Davies, Implementing SSL/TLS Using Cryptography and PKI, Pub, John Wiley & Sons, ISBN: 978-0-470-92041-1, 2011
24. H. Feistel, Cryptography and computer privacy, Scientific American 1973;
25. J. Felsenstein, Evolutionary trees from DNA sequences: a maximum likelihood approach, J. Mol. Evol. **17** (6): 368–76. doi:10.1007/BF01734359, PMID 7288891, (1981).
26. N. Ferguson, B. Schneir, Practical Cryptography, Wiley, 2003
27. N. Ferguson, B. Schneir, T. Kohno, Cryptography Engineering Desing Principle and Practical Applications, Wiley, ISBN: 978-0-470-47424-2, 2010
28. N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, Improved Cryptanalysis of Rijndael, Berlin: Springer, 2001
29. N. Ferguson, R. Schroeppel, D. Whiting, A simple algebraic representation of Rijndael, LNCS, vol. 2259. Berlin: Springer; 2001
30. R. Forré, Methods and instruments for designing S-boxes, Journal of Cryptology 1990
31. H. Fourche, Cryptanalysis A Study of Ciphers and their Solution, Dover Publications, 1939
32. R. Flenner, M. Abbott, T. Boubez, F. Cohen, N. Krishnan, A. Moffet, R. Ramamurti, B. Siddiqui, F. Sommers, Java™ P2P Unleashed, Sams, 2002
33. D. Flanagan, Java in a Nutshell, 5th Edition, March 2005
34. R. Flickenger, R. Weeks, Wireless Hacks, 2nd Edition, O'Reilly, November 2005
35. M. Fisher, R. Lai, S. Sharma, L. Moroney, Java EE and .Net Interoperability: Integration Strategies, Patterns, and Best Practices, Prentice Hall, April 21, 2006
36. R. Flenner, Jini™ and JavaSpaces™ Application Development, Sams, December, 2006
37. J. Farley, Java Distributed Computing, O'Reilly, January, 1998
38. J. Gustavus, Contemporary cryptology: the science of information integrity, IEEE Press, 1992
39. A. Gehani, T.H LaBean, DNA Based Cryptography, 5th DIMACS Workshop, MIT, Cambridge, Volume 54, pp. 141-151, ISBN 0-8218-2053-2, 1999
40. S. Garfinkel, Web Security, Privacy & Commerce, 2nd Edition, O'Reilly, November 2001
41. S. Garfinkel, A. Schwartz, Gene Spafford, Practical UNIX & Internet Security, 3rd Edition, O'Reilly, February 2003
42. M. Gregg, D. Kim, Inside Network Security Assessment: Guarding your IT Infrastructure, Sams, November 18, 2005
43. M. G. Graff, K. R. van Wyk, Secure Coding: Principles & Practices, O'Reilly, June 2003
44. W. Grosso, Java RMI, October 2001
45. M. Gupta, J. Walp, R. Sharman, Strategic and Practical Approaches for Information Security Governance, Pub. IGI Global, ISBN: 1-4666-0197-3, Feb. 2012
46. A. Hadlerman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino,A. Feldman, Attacks on Encryption Keys,17th USENIX Security Symposium, San Jose, CA, USA, ISBN 978-1-931971-60-7, 2008
47. D.L. Halligan, A. Eyre-Walker, P. Andolfatto, P.D. Keightley, Patterns of evolutionary constraints in intronic and intergenic DNA of Drosophila. Genome Res. 14: 273–279, 2004
48. A.L. Halpern, W.J. Bruno, Evolutionary distances for protein-coding sequences: Modeling site-specific residue frequencies.Mol. Biol. Evol. 5: 910–917, 1998

49. R. Hamid Nemati; Li Yang, Applied Cryptography for Cyber Security and Defense: Information Encryption and Cyphering, Pub. IGI Global, ISBN: 978-1-61520-783-1, Aug. 2010
50. E. R. Harold, Java Network Programming, 3rd Edition, O'Reilly, October 2004
51. E. R. Harold, Java I/O, O'Reilly, September 1999
52. M. Hasegawa, H. Kishino, T. Yano, Dating the human-ape splitting by a molecular clock of mitochondrial DNA, J. Mol. Evol**.22:** 160–174, 1985
53. W. Diffie, M. E. Hellman, New Directions in Cryptography, IEEE Transactions Information Theory, vol. IT-22, Nov. 1976
54. W. Diffie, M. E. Hellman, Exhaustive Cryptanalysis of the NBS Data Encryption Standard, IEEE Computer 10 (6), 1977
55. W. Diffie, M. E. Hellman, Privacy and authentication: An introduction to cryptography, Proc. IEEE, vol. 67, 1979
56. C. A. Henk, Encyclopedia of Cryptography and Security, Springer Science, 2005
57. T. Hodorogea, M. Vaida, M. Borda, "A Java Crypto Implementation of DNAProvider Featuring Complexity in Theory and Practice", IEEE Explore (ITI) 2008 International Conference on Information Technology Interfaces, 23-26 June, 2008, Cavtat, Croatia, pp. 607-612
58. T. Hodorogea, O. Ionas, J. Sarbu, "DERIVING DNA CRYPTOGRAPHIC KEYS BASED ON EVOLUTIONARY MODELS", IEEE Explore, IEEE Catalog Number: CFP1142L-CDR, ISBN: 978-1-61284-359-9, 2011 IEEE, pp.148-151
59. T. Hodorogea, O. Ionas, "Security of Business to Business and Business to Customer Software Applications Based on the Central Dogma of Molecular Biology (CDMB) and Evolutionary Models", Information Technology Interface IEEE Explore (ITI) 2011 International Conferince on Information Technology Interfaces, 27-30 June, 2011, Cavtat, Croatia
60. T. Hodorogea, O. Ionas, Chapter, Modern Technologies Used for Security of Software Applications, Cryptography / Book 2, Pub. InTech, ISBN 979-953-307-863-1, edited by, Dr. Jaydip Sen, March, 2012
61. J. Hoffstein, J. Pipher, J.H. Silverman, Introduction to Mathematical Cryptography, Springer Science, 2008
62. I. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, M. Tackera, P. Schuster, Fast folding and comparison of RNA secondary structures, Monatsh. Chem. 125: 167–188, 1994
63. C. S. Horstmann, G. Cornell, Core Java™ 2 Volume II - Advanced Features, Seventh Edition, Prentice Hall, November 2004
64. M. Howard, D. LeBlanc, Writing Secure Code, Microsoft Press, December 4, 2002
65. D. Kahn, Codebreaking and the Battle of the Atlantic, US Air Force Academy, Colorado, Springs, CO, 1994
66. D. Khadraoui, F. Herrman, Advances in Enterprise Information Technology Security, Pub. IGI Global, ISBN 978-1-59904-090-3
67. D. Kahn, Codebrakers, McMillan, New York, 1967
68. S. Kalman, Web Security Field Guide, Cisco Press, November 08, 2002
69. D. Kerr; John G. Gammack; Kay Bryant, Business Security Development: Management Technologies, Pub. IGI Global, ISBN 978-1-60566-806-2, 2010
70. J. B. Knudsen," Java Cryptography", O'Reilly, 1998
71. A. G. Konheim, Computer Security and Cryptography, John Wiley & Sons, Inc., 2007
72. P. Kumar, J2EE™ Security for Servlets, EJBs and Web Services: Applying Theory and Standards to Practice, Prentice Hall, September 2004.,Professional, June 2000

73. A.V. Senthil Kumar, Hakikur Rahman, Mobile Computing Techniques in Emerging Markets, Pub. IGI Global, ISBN: 1-4666-0080-2, Jan 2012
74. R.D. Knight, S.J. Freeland, L.F. Landweber, A simple model based on mutation and selection explains trends in codon andamino-acid usage and GC composition within and across genomes.Genome Biol. doi: 10.1186/gb-2001-2-4-research0010, 2001
75. S.K. Kummerfeld, S.A. Teichmann, A transcription factor prediction database. Nucleic Acids Res. **34:** D74–D81. doi:10.1093/nar/gkj131, 2006
76. M. Kimura, A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences". J. Mol. Evol. **16** (2): 111–doi:10.1007/BF01731581.PMID 7463489, 1980
77. D. Kim, Fundamentals of Information Systems Security, ISBN-13: 978-0-7637-9025-7, 2010
78. M. MacDonald Creating Web Sites: The Missing Manual, O'Reilly 2005
79. M. Mogollon, Cryptography and Security Services: Mechanisms and applications, Pub. IGI Global, ISBN: 978-1-59904-837-6, 2008
80. J. W. Muchow Core J2ME™ Technology & MIDP, Prentice Hall, December 2001
81. A. Menezes, P.C. Van Oorschot, S. A. Vanstone, Handbook of applied cryptography, CRC Press, 1997
82. P. Minuţ, Teoria numerelor, Vol. 1, Editura Crenguţa Gâldău, Iaşi 1997
83. N. Molina, E. Nimwegen, Universal patterns of purifying selection at noncoding positions in bacteria, PMC2134783, Genome Res. 2008
84. A.M. Moses, D.Y. Chiang, D.A. Pollard, V.N. Iyer, M.B. Eisen, Identifying conserved transcription-factor bindingsites in multiple alignments using a binding site-specificevolutionary model. Genome Biol. **5**. R98. doi:10.1186/gb-2004-5-12-r98, 2004
85. V. Mustonen, M. Lässig, Evolutionary population genetics of promoters: Predicting binding sites and functional phylogenies. Proc.Natl. Acad. Sci. **102:** 15936–15941, 2005
86. H. Nahari, R. L. Krutz, Web Commerce Security Design and Development, Pub. John Wiley & Sons, ISBN: 978-0-470-62446-3, 2011
87. E. Nimwegen, Scaling laws in the functional content of genomes, Trends Genet. 19: 479–484, 2003
88. J. Zdziarski, Hacking and Securing iOS Applications, Pub: O'Reilly Media, Inc, Jan. 2012
89. H. Ochman, Neutral mutations and neutral substitutions in bacterial genomes. Mol. Biol. Evol. **20:** 2091–2096, 2003
90. C. Onwubiko, Thomas Owens Situational Awareness in Computer Network Defense, Pub. IGI Global, ISBN: 1-4666-0104-3, Jan. 2012
91. C. Paar, J. Pelzl, Understanding Cryptography. A Textbook for Students and Practitioners, Springer, 2009
92. P. J. Perrone, S. R. Venkata, Building Java™ Enterprise Systems with J2EE™, Sams, June 07, 2000
93. W. Phil, "Digital Identity", O'Reilly, August 2005
94. P. Pfleeger, Pfleeger Consulting Group, Shari Lawrence Pfleeger Dartmouth College, Analyzing Computer Security, A Threat/Vulnerability/Countermeasure Approach, Pub. Prentice Hall, ISBN: 0-13-278946-9, Aug. 2011
95. N. Rajewsky, N. D. Socci, M. Zapotocky, E.D. Siggia, The evolution of DNA regulatory regions for proteo-gamma bacteria by interspecies comparisons. Genome Res. 12: 298–308, 2002

96. I.B. Rogozin, K.S. Makarova, D.A. Natale, A.N. Spiridonov, R.L. Tatusov, Y.I. Wolf, J. Yin, E.V. Koonin, Congruent evolution of different classes of non-coding DNA in prokaryotic genomes.Nucleic Acids Res. 30: 4264–4271. doi, 2002
97. George Reese, Java Database Best Practices, O'Reilly, May 2003
98. W. Stallings, Cryptography and Network Security, Pearson Prentice Hall, 2006
99. W. Stallings, Cryptography and Network Security, Pearson Prentice Hall, ISBN: 1-13-609704, 2011
100. I. Singh, B. Stearns, M. Johnson, Applications with the J2EE™ Platform, Second Edition, Addison Wesley Professional, 2002
101. C. Steel, R. Nagappan, R. Lai Core Security Patterns: Best Practices and Strategies for J2EE™, Web Services, and Identity Management, Prentice Hall, October 2005
102. R.L. Rivest, A. Shamir, L.A. Adleman, A method for obtaining digital signatures and publickey cryptosystems, Comm ACM 21, 1978
103. B. Schneier, Applied Cryptography, Second Edition. John Wiley & Sons, 1996
104. B. Schneier, Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish), Fast Software Encryption, 1993
105. B. Schneier, The Blowfish Encryption Algorithm. One Year Later, Dr. Dobb's Journal, September 1995.
106. C. E. Shannon, A Mathematical Theory of Communication, Bell System Technical Journal, v. 27, n. 4, 1948
107. C. E. Shannon, Communication Theory of Secrecy Systems, Bell System Technical Journal, v. 28, n. 4, 1949
108. C. E. Shannon, Predication and Entropy in Printed English, Bell System Technical Journal, v. 30, n. 1, 1951
109. V. Shoup, A computational introduction to number theory and algebra, Cambridge University Press, 2005
110. M. Stewart, Network Security, Firewalls, and VPNs, Pub. Jones & Bartlett Learning, ISBN: 978-0-7637-9130-8, 2010
111. D. Tynan, "Computer Privacy Annoyances", O'Reilly, July 2005.
112. D. Teare, C. Paquet, "Campus Network Design Fundamentals", Cisco 4th IASTED International, Conference on Web-Based Education, Grindelwald, Switzerland, pp. 667-672, 2005
113. J. R. Vacca, Network and System Security, Syngress, ISBN-13: 978-1-59749-535-6, 2010

11. List of Research Activity

Journals:
[1] Tatiana **Hodorogea,** Mircea-Florin Vaida, Biometric Security Using Blood Analysis, **Best Paper**, Transactions of the VSB-Technical University of Ostrava, Mechanical Series, Czech Republic, Vol. LII, 2/2006, pp. 53-58, ISBN 80-248-1211-8, ISSN 1210-0471, ISI Journal
[2] Mircea-Florin Vaida, Tatiana **Hodorogea**, DNA Security between Technical and Spiritual Concepts, Acta Tehnica Napocensis Electronics and Telecommunications, Cluj-Napoca, Vol. 47, No.2, 2006, pp. 1-8, ISSN 1221-6542
[3] Tatiana **Hodorogea**, Mircea-Florin Vaida, Deriving DNA Public Keys from Blood Analysis, International Conference on Computers and Communications, ICCC'06, Baile Felix, June 1-3, 2006, pp. 262-267, ISSN 1841-9836 (print version), ISSN 1841-9844 (online version), INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL Volume: 1 Pages: 262-267 Published: 2006, ISI Journal
[4] Monica E. Borda, Olga Tornea, Tatiana **Hodorogea**, Secret Writing by DNA Hybridization , Acta Technical Napocensis, Volum 49, Number 1, 2009, pp. 21-24, ISSN 1221-6542

Conferences:
[1] Tatiana **Hodorogea**, Mircea-Florin Vaida, Alternate Cryptography Techniques, ICCC05, Miskolc-Lillafured, Hungary, 24-27 may 2005, Vol. 1, pp. 513-518, ISBN 963-661-644-2
[2] Tatiana **Hodorogea**, Mircea-Florin Vaida, BLOOD ANALYSIS AS BIOMETRIC SELECTION OF PUBLICKEYS, 7 th International Carpathian Control Conference ICCC'2006, Ostrava – Beskydy, Czech Republic, May 29-31, 2006, pp. 675-678, ISBN 80-248-1066-2
[3] Tatiana **Hodorogea**, Mircea-Florin Vaida, Deriving DNA Public Keys from Blood Analysis, International Conference on Computers and Communications, ICCC'06, Baile Felix, June 1-3, 2006, pp. 262-267, ISSN 1841-9836 (print version), ISSN 1841-9844 (online version), INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL Volume: 1 Pages: 262-267 Published: 2006, ISI Journal
[4] Mircea-Florin Vaida, Tatiana **Hodorogea**, Coding Documents using Alternative Techniques, CSNDSP 2006 PROCEEDINGS, July 2006, Patras, Greece, pp.359-361, ISBN 960-89282-0-6
[5**]**Tatiana **Hodorogea**, Vaida Mircea-Florin, SECURITY PROVIDER WITH DNA CRYPTOGRAPHY ALGORITHM AS A SUBSET OF JAVA SECURITY API, 8 th International Carpathian Control Conference ICCC'2007, Štrbské Pleso, Slovak Republic, May 24-27, 2007, pp. 175-178, ISBN 978-80-8073-805-1
[6] Ligia Chiorean, Aron Sipos, Mircea-Florin Vaida, Tatiana **Hodorogea**, Content Based Medical Image Retrieval Using Oracle Intemedia, 8th International Carpathian

Control Conference ICCC'2007, Štrbské Pleso, Slovak Republic, May 24-27, 2007, pp. 207-210, ISBN 978-80-8073-805-1
[7] Tatiana **Hodorogea**, Mircea-Florin Vaida, SECURITY PROVIDER WITH ALTERNATE CRYPTOGRAPHY TECHNIQUE, ICMCS'2007, 5 th International Conference on Microelectronics and Computer Science, Chisinau, Moldova, September 19-21, 2007, pp. 318-321, ISBN 978-9975-45-046-1
[8] Tatiana **Hodorogea**, Mircea-Florin Vaida, Biomolecular Cryptographic Algorithm as a Subset of Java Cryptography Extension, Regional Conference on Embedded and Ambient Systems - RCEAS, 22-24 November 2007, Budapesta
[9] Tatiana **Hodorogea**, Mircea-Florin Vaida, COMPLEXITY OF DNA ENCRYPTION SYSTEM AS A SUBSET OF JAVA CRYPTOGRAPHY EXTENSION, IASTED International Conference on Biomedical Engineering (BioMed 2008), 13-15 Feb., Innsbruck, Austria, paper 601-167, pp. 19-24, ISBN: 978-0-88986-082-2, (Acta Press), http://www.actapress.com/Abstract.aspx?paperId=32643
[10] Tatiana **Hodorogea**, Mircea-Florin Vaida, Monica Borda, Cosmin Striletchi, A Java Crypto Implementation of DNAProvider Featuring Complexity in Theory and Practice, Proceedings of 30-th Int. Conf. on ITI 2008, June 23-26, 2008, Cavtat/Dubrovnik, Croatia, Univ. of Zagreb, Univ. Computing Center, pp. 607-612, IEEE Catalog Number CFP08498-PRT, ISBN: 978-953-7138-12-7,ISSN: 1330-1012 (indexare IEEExplore), INSPEC Database
[11] Olga Tornea, Monica Borda, Tatiana **Hodorogea**, Mircea-Florin Vaida, ENCRYPTION SYSTEM WITH INDEXING DNA CHROMOSOMES CRYPTOGRAPHIC ALGORITHM, IASTED International Conference on Biomedical Engineering (BioMed 2010), 15-18 Feb., Innsbruck, Austria, paper 680-099, pp. 12-15,
ISBN: 978-0-88986-825-0, 975-0-88986-826-7, Indexare IEEE Explore, Acta Press
[12] Tatiana **Hodorogea**, Szilard Otto Ionas, "Deriving DNA Cryptographic Keys Based on Evolutionary Models for Security of Web-based Business Processes", Paper accepted for publication, (ID 40086), The Third International Conferences on Advances in Multimedia, MMEDIA 2011, April 17-22, 2011, Budapest, Hungary.
[13] Tatiana **Hodorogea**, Szilard Otto Ionas, Janetta Sarbu, "DERIVING DNA CRYPTOGRAPHIC KEYS BASED ON EVOLUTIONARY MODELS", IEEE Explore, IEEE Catalog Number: CFP1142L-CDR, ISBN: 978-1-61284-359-9, 2011 IEEE, pp.148-151
[14] Tatiana **Hodorogea**, Szilard Otto Ionas**,** Security of Business to Business and Business to Customer Software Applications based on the Central Dogma of Molecular Biology (CDMB) and evolutionary models, Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces (June 2011), pg. 403-408 IEEE Catalog Number: CFPI 1498-PRT, ISBN: 978-1-61284-897-6

Workshops:
[1] Tatiana Hodorogea, Mircea-Florin Vaida, DNA CRYPTOGRAPHY TECHNIQUE, "VERIFICATORI BIOMETRICI" Workshop 26-27 mai 2005, Cluj-Napoca, pp. 175-182, ISBN 973-656-918-7
[2] Tatiana **Hodorogea**, Mircea-Florin Vaida, DAS Clients to view annotation, feature and structural information of proteins, 25.02.2008, EBI, Cambridge, UK
[3] Tatiana **Hodorogea,** Mircea-Florin Vaida, Sequence annotations for the purpose of sharing annotations of public genome or protein sequences using a DAS-enabled browser application, 02.04.2008, Brussels, Belgium

Reports:

[1] Mircea –Florin Vaida, Tatiana Hodorogea, Biotechnologies for Software Security, Technical Report, THR 05007, Technical University of Cluj-Napoca, November 2005, ISSN 1454-0665

[2] Tatiana **Hodorogea**, Mircea-Florin Vaida, Technics for Developing Secure Software Applications, doctoral report, Technical University of Cluj-Napoca, 24.03.2006

[3] Tatiana **Hodorogea**, Alternate Technologies used for Data Security, doctoral report, The Technical University of Cluj-Napoca, March, 2007.

[4] Tatiana Hodorogea, Mircea-Florin Vaida, Unconditional Secure DNAE System as part of security provider, named DNAProvider, 07.12.2007, Leeds, UK

[5] Tatiana **Hodorogea**, Developing computational models and tools to study RNA-based regulation of gene expression, Scientific report, 23.03.2009, Swiss Institute of Bioinformatics and Biozentrum, University of Basel, Switzerland.

[6] Tatiana **Hodorogea**, Developing computational models and tools to study RNA-based regulation of gene expression, Scientific report, 23.03.2009, Swiss Institute of Bioinformatics and Biozentrum, University of Basel, Switzerland.

[7] Tatiana **Hodorogea**, Relation between Chemical Components, Structure and Function of Cellular Membranes, Scientific Report, 19.05.2009, Biozentrum, University of Basel, Switzerland

[8] Tatiana **Hodorogea,** Role of Potassium Channels in Pharmacology and Drug Discovery, Scientific Report, 26.05.2009, Biozentrum, University of Basel, Switzerland

Books:

Tatiana **Hodorogea**, Szilard Otto Ionas, Chapter, "Modern Technologies Used for Security of Software Applications", **Cryptography / Book 2, ISBN 979-953-307-863-1, edited by, Dr. Jaydip Sen, March 2012**

12. Appendix

A1. SmartCipher Application UML Diagrams

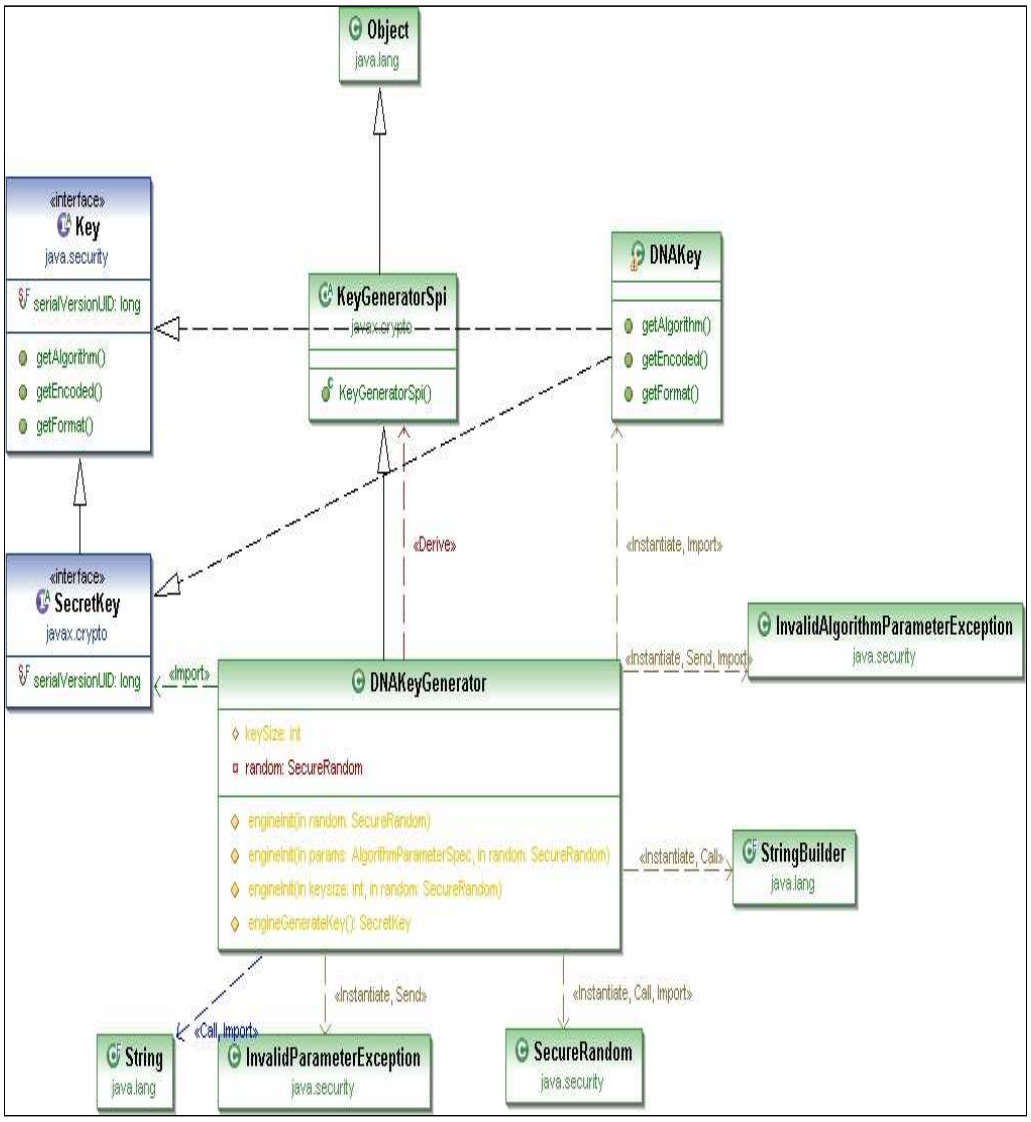


Fig. 52 DNAKeyGenerator, UML Diagram

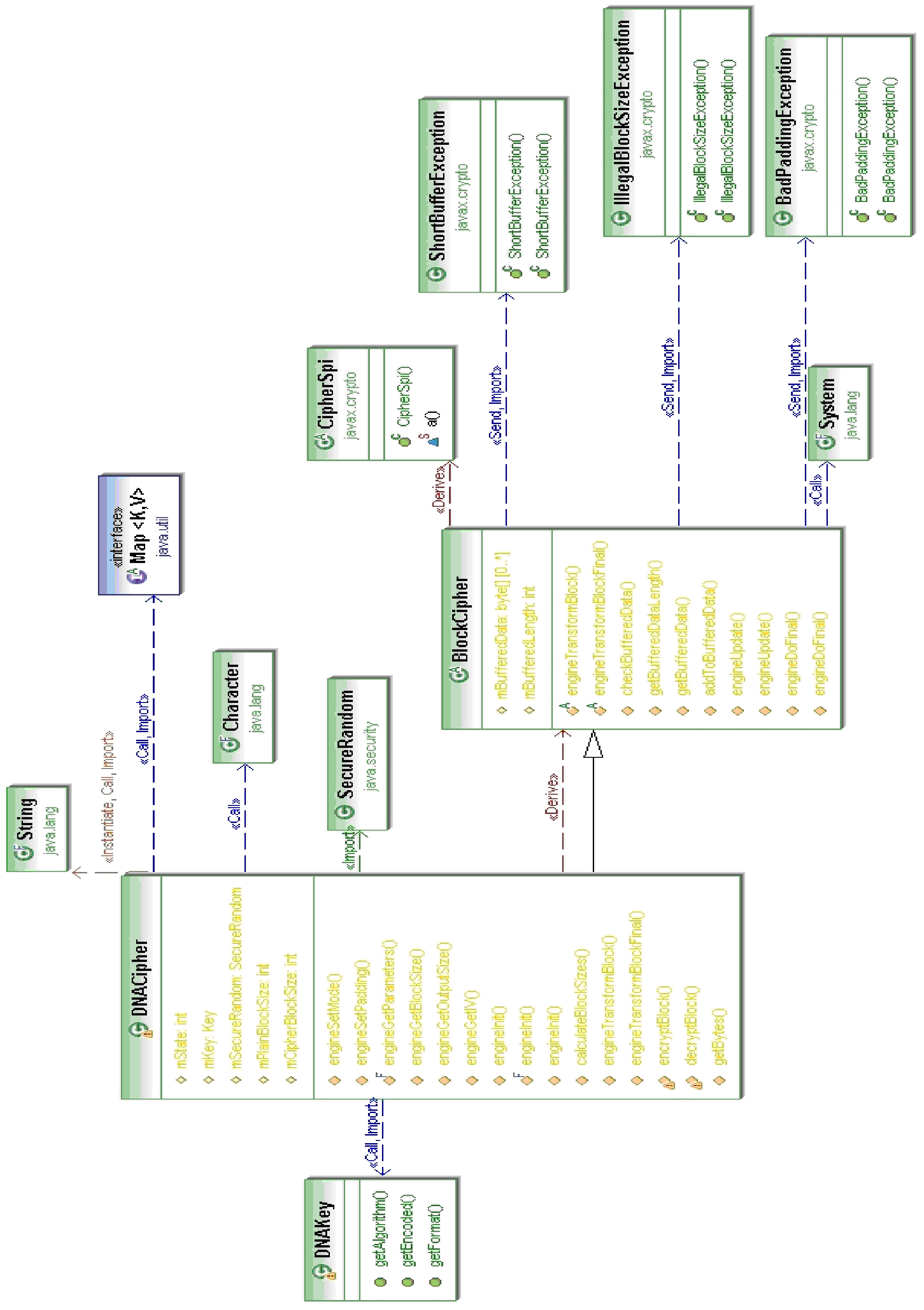


Fig. 53 DNA encryption/decryption UML diagram

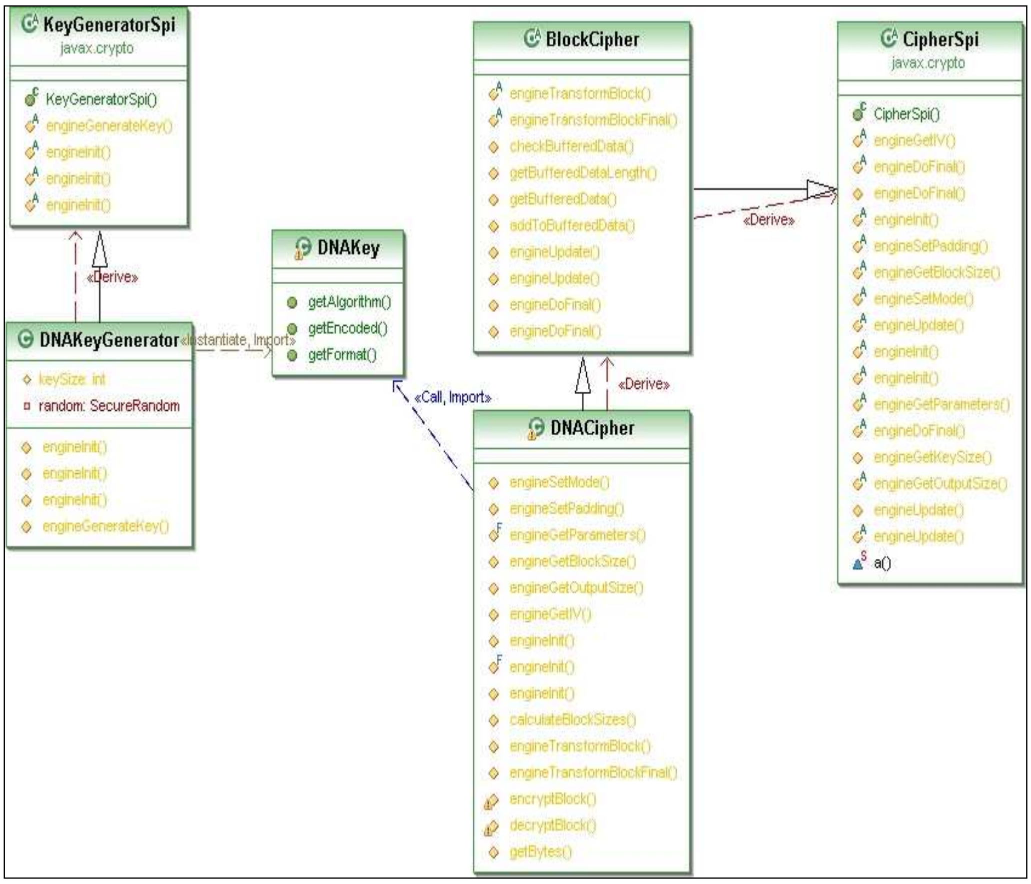


Fig. 54 CipherSpi and KeyGeneratorSpi, UML Diagram

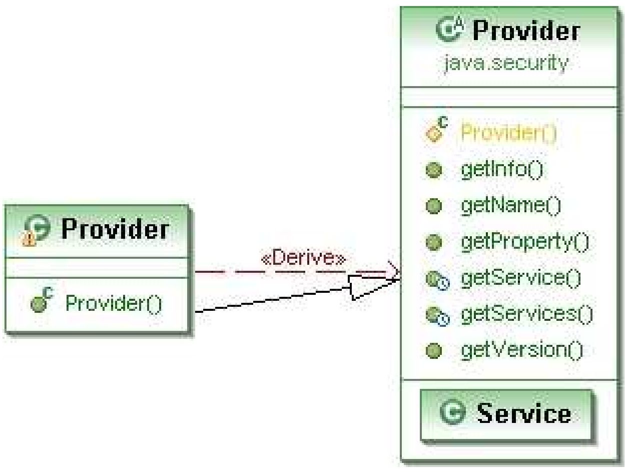


Fig. 55 Deriving Provider class from java.security.Provider

A2. DNA Provider Certificates

***** JCE Code Signing Certificate Issuance *****
<679>

<Provider Code Signing Cert>
CN=Tatiana Hodorogea,OU=Technical University Cluj-Napoca,O=UTCN,L=Cluj-Napoca,ST=Cluj,C=RO

-----BEGIN CERTIFICATE-----
MIID2zCCA5mgAwIBAgICApEwCwYHKoZIzjgEAwUAMIGQMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCQ0ExEjAQBgNVBAcTCVBhbG8gQWx0bzEdMBsGA1UEChMUU3VuIE1
p
Y3Jvc3lzdGVtcyBJbmMxIzAhBgNVBAsTGkphdmEgU29mdHdhcmUgQ29kZSBTaWdu
aW5nMRwwGgYDVQQDExNKQ0UgQ29kZSBTaWduaW5nIENBMB4XDTA4MDEyMjE5M
DIy
N1oXDTEzMDEyNTE5MDIyN1owgYgxCzAJBgNVBAYTAlJPMQ0wCwYDVQQIEwRDbHVq
MRQwEgYDVQQHEwtDbHVqLU5hcG9jYTENMAsGA1UEChMEVVRDTjEpMCcGA1UECxM
g
VGVjaG5pY2FsIFVuaXZlcnNpdHkgQ2x1ai1OYXBvY2ExGjAYBgNVBAMTEVRhdGlh
bmEgSG9kb3JvZ2VhMIIBtzCCASwGByqGSM44BAEwggEfAoGBAP1/U4EddRIpUt9K
nC7s5Of2EbdSPO9EAMMeP4C2USZpRV1AIlH7WT2NWPq/xfW6MPbLm1Vs14E7gB00
b/JmYLdrmVClpJ+f6AR7ECLCT7up1/63xhv4O1fnxqimFQ8E+4P208UewwI1VBNa
FpEy9nXzrith1yrv8iIDGZ3RSAHHAhUAl2BQjxUjC8yykrmCouuEC/BYHPUCgYEA
9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRWVeOutRZT+ZxBxCBgLRJ
FnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuzpnWRbqN/C/ohNWLx+2J6ASQ7
zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSoDgYQAAoGAbyV11UdeygOC
BT2h8xrvjlOcpYQjtqLndYlbqivdiLj9NKzn+7qN2vxepacCbe8Mnkf3IKqY48kp
BItL5xH/i//34TeOmcz65G42Nng9U6rmw9CqjVNOgEp84UBuMGIRBybBJIxweddC
01SvcXMKQ8kPhoUPYiZms+i/n/kKIt2jgYcwgYQwEQYJYIZIAYb4QgEBBAQDAgQQ
MA4GA1UdDwEB/wQEAwIF4DAdBgNVHQ4EFgQUB23QpB1u4BuMb/6LZPocWlhvnxUw
HwYDVR0jBBgwFoAUZeL0hsnTTvCRTliiavXYeFqawaYwHwYDVR0RBBgwFoEUdGhv
ZG9yb2dlYUB5YWhvby5jb20wCwYHKoZIzjgEAwUAAy8AMCwCFFocW73se4eHudRw
9VttquJWNjjGAhRM5nNv/0qMOA/sBuIao3uS3NNAwQ==
-----END CERTIFICATE-----

<JCE Root CA Cert>

CN=JCE Code Signing CA,OU=Java Software Code Signing,O=Sun Microsystems Inc,L=Palo Alto,ST=CA,C=US

-----BEGIN CERTIFICATE-----
MIIDwDCCA36gAwIBAgIBEDALBgcqhkjOOAQDBQAwgZAxCzAJBgNVBAYTAlVTMQsw
CQYDVQQIEwJDQTESMBAGA1UEBxMJUGFsbyBBbHRvMR0wGwYDVQQKExRTdW4gT
Wlj
cm9zeXN0ZW1zIEluYzEjMCEGA1UECxMaSmF2YSBTb2Z0d2FyZSBDb2RlIFNpZ25p
bmcxHDAaBgNVBAMTE0pDRSBDb2RlIFNpZ25pbmcgQ0EwHhcNMDEwNDI1MDcwMD
Aw
WhcNMjAwNDI1MDcwMDAwWjCBkDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMR
Iw

EAYDVQQHEwlQYWxvIEFsdG8xHTAbBgNVBAoTFFN1biBNaWNyb3N5c3RlbXMgSW5j
MSMwIQYDVQQLExpKYXZhIFNvZnR3YXJlIENvZGUgU2lnbmluZzEcMBoGA1UEAxMT
SkNFIENvZGUgU2lnbmluZyBDQTCCAbcwggEsBgcqhkjOOAQBMIIBHwKBgQDrrzcE
HspRHmldsPKP9rVJH8akmQXXKb90t2r1Gdge5Bv4CgGamP9wq+JKVoZsU7P84ciB
jDHwxPOwi+ZwBuz3aWjbg0xyKYkpNhdcO0oHoCACKkaXUR1wyAgYC84Mbpt29wXj
5/vTYXnhYJokjQaVgzxRIOEwzzhXgqYacg3O0wIVAIQlReG6ualiq3noWzC4iWsb
/3t1AoGBAKvJdHt07+5CtWpTTTvdkAZyaJEPC6Qpdi5VO9WuTWVcfio6BKZnptBx
qqXXt+LBcg2k0aoeklRMIAAJorAJQRkzALLDXK5C+LGLynyW2BB/N0Rbqsx4yNdy
djdrQJmoVWb6qAMei0oRAmnLTLglBhygd9LJrNI96QoQ+nZwt/vcA4GEAAKBgC0J
mFysuJzHmX7uIBkqNJD516urrt1rcpUNZvjvJ49Esu0oRMf+r7CmJ28AZ0WCWweo
VlY70ilRYV5pOdcudHcSzxlK9S3Iy3JhxE5v+kdDPxS7+rwYZijC2WaLei0vwmCS
SxT+WD4hf2hivmxISfmgS16FnRkQ+RVFURtx1PcLo2YwZDARBglghkgBhvhCAQEE
BAMCAAcwDwYDVR0TAQH/BAUwAwEB/zAfBgNVHSMEGDAWgBRl4vSGydNO8JFOWKJ
q
9dh4WprBpjAdBgNVHQ4EFgQUZeL0hsnTTvCRTliiavXYeFqawaYwCwYHKoZIzjgE
AwUAAy8AMCwCFCr3zzyXXfl4tgjXQbTZDUVM5LScAhRFzXVpDiH6HdazKbLp9zMd
M/38SQ==
-----END CERTIFICATE-----

Sun’s Certification Form for DNA Cryptographic Service Provider (CSP)

Sun's JCE implements encryption using Cryptographic Service
Providers (CSPs). Only CSPs signed by a trusted entity can be
plugged into the JCE framework. You have indicated that you
will be providing a DNA CSP for JCE. Please read, sign, and return
this form to:

Sun Microsystems, Inc.
International Trade Services/Export Compliance
Attn: Encryption Export
10 Network Circle MS: UMPK10-144
Menlo Park, CA 94025

Also fax paperwork to the follwing number, with the following headings:

Fax #: 408-668-0918
Subject Line: CSR # **<679>**
Attention: Encryption Export
Sun Microsystems, Inc.
International Trade Services/Export Compliance

Before you return this form, you must send, via email, a Certificate Signing Request (CSR) and contact information to javasoft-cert-request@sun.com. In response, you will receive a request number, also via email.
You may then mail this form. Include the request number here:

Request Number: **<679>**

Including the request number ensures this form can be matched to your email message.

Overview

Export controls on commercial encryption products are administered by the Bureau of Industry and Security (BIS) in the U.S. Department of Commerce. Rules governing exports of encryption are found in the Export Administration Regulations (EAR), 15 CFR Parts 730-774 which implements the Export Administration Act ("EAA" 50 U.S.C. App. 2401 et seq.).
BIS requires that each entity exporting products be familiar with and comply with their affirmative obligations set forth in the Export Administration Regulations. Please note that the regulations are subject to change. We recommend that you obtain your own legal advice when attempting to export. In addition some countries may restrict certain levels of encryption imported into their country. We recommend consulting legal counsel in the appropriate country or the applicable governmental agencies in the particular country.
If you developed your CSP inside the United States, and you want to export your CSP, it must be reviewed and approved for export by the Bureau of Industry and Security.
If you developed your CSP outside the United States, then you may nothave to submit your product for a technical review, but it may be subject to the requirements of the Export Administration Regulations.

Certification

I certify that I have read the above "Overview" and will comply with the Export Administration Regulations. I understand that any CSP developed in the United States may not be exported from the United States without prior approval by the US Government and that such approval is my responsibility. I further understand that any CSP developed outside the United States nevertheless may be subject to the Export Administration Regulations.

Company: **Technical University of Cluj-Napoca, Romania**

Address: **26-28, George Baritiu, Street, Cluj-Napoca, Romania**

Signature: ______________ Date: **19 January 2008**

Printed Name: **TATIANA HODOROGEA**

Title: **Assistant Researcher, PhD, Eng.**

A3. Derived List of DNA Cryptographic Keys Sequences from Human Genome Analysis with three mers DNA letters length

AAA	1.9194786002e-01	2.4454800665e-01	1.2740335142e+00
AAC	1.4182466683e-02	7.7023433519e-03	5.4308912011e-01
AAG	9.4549777886e-03	5.1348955679e-03	5.4308912011e-01
AAT	9.4549777886e-03	5.1348955679e-03	5.4308912011e-01
AA-	2.2504028228e-01	2.6252014114e-01	1.1665473331e+00
ACA	1.5115460728e-02	8.1688403743e-03	5.4042946632e-01
ACC	1.3343472517e-02	7.2828462690e-03	5.4579842388e-01
ACG	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
ACT	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
AC-	3.2125593307e-02	1.6062796654e-02	5.0000000000e-01
AGA	1.0076973819e-02	5.4458935829e-03	5.4042946632e-01
AGC	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
AGG	8.2845383345e-03	4.7533791776e-03	5.7376512555e-01
AGT	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
AG-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
ATA	1.0076973819e-02	5.4458935829e-03	5.4042946632e-01
ATC	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
ATG	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
ATT	8.2845383345e-03	4.7533791776e-03	5.7376512555e-01
AT-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
A-A	2.2721726839e-01	2.6360863419e-01	1.1601610919e+00
A-C	3.1192599263e-02	1.5596299631e-02	5.0000000000e-01
A-G	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
A-T	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
A--	3.0000000000e-01	3.0000000000e-01	1.0000000000e+00
CAA	1.3343472517e-02	7.2828462690e-03	5.4579842388e-01
CAC	1.5115460728e-02	8.1688403743e-03	5.4042946632e-01
CAG	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
CAT	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
CA-	3.2125593307e-02	1.6062796654e-02	5.0000000000e-01
CCA	1.4182466683e-02	7.7023433519e-03	5.4308912011e-01
CCC	1.9194786002e-01	2.4454800665e-01	1.2740335142e+00
CCG	9.4549777886e-03	5.1348955679e-03	5.4308912011e-01
CCT	9.4549777886e-03	5.1348955679e-03	5.4308912011e-01
CC-	2.2504028228e-01	2.6252014114e-01	1.1665473331e+00
CGA	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
CGC	1.0076973819e-02	5.4458935829e-03	5.4042946632e-01
CGG	8.2845383345e-03	4.7533791776e-03	5.7376512555e-01
CGT	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
CG-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
CTA	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
CTC	1.0076973819e-02	5.4458935829e-03	5.4042946632e-01
CTG	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
CTT	8.2845383345e-03	4.7533791776e-03	5.7376512555e-01
CT-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01

C-A	3.1192599263e-02	1.5596299631e-02	5.0000000000e-01
C-C	2.2721726839e-01	2.6360863419e-01	1.1601610919e+00C-G
	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
C-T	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
C--	3.0000000000e-01	3.0000000000e-01	1.0000000000e+00
GAA	8.8956483448e-03	4.8552308460e-03	5.4579842388e-01
GAC	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
GAG	9.4658638082e-03	5.3440419145e-03	5.6455934955e-01
GAT	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
GA-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
GCA	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
GCC	8.8956483448e-03	4.8552308460e-03	5.4579842388e-01
GCG	9.4658638082e-03	5.3440419145e-03	5.6455934955e-01
GCT	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
GC-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
GGA	8.8438677782e-03	5.0330438995e-03	5.6909985831e-01
GGC	8.8438677782e-03	5.0330438995e-03	5.6909985831e-01
GGG	1.1930418671e-01	1.5802246666e-01	1.3245341259e+00
GGT	5.8959118522e-03	3.3553625997e-03	5.6909985831e-01
GG-	1.4288783412e-01	1.7144391706e-01	1.1998496451e+00
GTA	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
GTC	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
GTG	6.3105758721e-03	3.5626946097e-03	5.6455934955e-01
GTT	5.5230255563e-03	3.1689194517e-03	5.7376512555e-01
GT-	1.4278041470e-02	7.1390207350e-03	5.0000000000e-01
G-A	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
G-C	2.0795066175e-02	1.0397533088e-02	5.0000000000e-01
G-G	1.4454649020e-01	1.7227324510e-01	1.1918189426e+00
G-T	1.3863377450e-02	6.9316887250e-03	5.0000000000e-01
G--	2.0000000000e-01	2.0000000000e-01	1.0000000000e+00
TAA	8.8956483448e-03	4.8552308460e-03	5.4579842388e-01
TAC	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
TAG	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
TAT	9.4658638082e-03	5.3440419145e-03	5.6455934955e-01
TA-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
TCA	1.8333300312e-03	3.0555500519e-04	1.6666666667e-01
TCC	8.8956483448e-03	4.8552308460e-03	5.4579842388e-01
TCG	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
TCT	9.4658638082e-03	5.3440419145e-03	5.6455934955e-01
TC-	2.1417062205e-02	1.0708531102e-02	5.0000000000e-01
TGA	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
TGC	1.2222200208e-03	2.0370333680e-04	1.6666666667e-01
TGG	5.5230255563e-03	3.1689194517e-03	5.7376512555e-01
TGT	6.3105758721e-03	3.5626946097e-03	5.6455934955e-01
TG-	1.4278041470e-02	7.1390207350e-03	5.0000000000e-01
TTA	8.8438677782e-03	5.0330438995e-03	5.6909985831e-01
TTC	8.8438677782e-03	5.0330438995e-03	5.6909985831e-01
TTG	5.8959118522e-03	3.3553625997e-03	5.6909985831e-01
TTT	1.1930418671e-01	1.5802246666e-01	1.3245341259e+00
TT-	1.4288783412e-01	1.7144391706e-01	1.1998496451e+00

```
T-A     2.0795066175e-02    1.0397533088e-02    5.0000000000e-01
T-C     2.0795066175e-02    1.0397533088e-02    5.0000000000e-01
T-G     1.3863377450e-02    6.9316887250e-03    5.0000000000e-01T-T
        1.4454649020e-01    1.7227324510e-01    1.1918189426e+00
T--     2.0000000000e-01    2.0000000000e-01    1.0000000000e+00
-AA     2.2308262923e-01    2.6154131461e-01    1.1723965937e+00
-AC     3.2964587473e-02    1.6482293737e-02    5.0000000000e-01
-AG     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-AT     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-A-     3.0000000000e-01    3.0000000000e-01    1.0000000000e+00
-CA     3.2964587473e-02    1.6482293737e-02    5.0000000000e-01
-CC     2.2308262923e-01    2.6154131461e-01    1.1723965937e+00
-CG     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-CT     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-C-     3.0000000000e-01    3.0000000000e-01    1.0000000000e+00
-GA     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-GC     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-GG     1.4139628894e-01    1.7069814447e-01    1.2072321399e+00
-GT     1.4650927766e-02    7.3254638829e-03    5.0000000000e-01
-G-     2.0000000000e-01    2.0000000000e-01    1.0000000000e+00
-TA     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-TC     2.1976391649e-02    1.0988195824e-02    5.0000000000e-01
-TG     1.4650927766e-02    7.3254638829e-03    5.0000000000e-01
-TT     1.4139628894e-01    1.7069814447e-01    1.2072321399e+00
-T-     2.0000000000e-01    2.0000000000e-01    1.0000000000e+00
--A     3.0000000000e-01    3.0000000000e-01    1.0000000000e+00
--C     3.0000000000e-01    3.0000000000e-01    1.0000000000e+00
--G     2.0000000000e-01    2.0000000000e-01    1.0000000000e+00
--T     2.0000000000e-01    2.0000000000e-01    1.0000000000e+00
---     1.0000000000e+00    1.0000000000e+00    1.0000000000e+00
```

A4. Publications

iTi 2008
30th International Conference on
INFORMATION TECHNOLOGY INTERFACES
June 23 - 26, 2008, Cavtat/Dubrovnik, Croatia
30
iTi

1. Tatiana **Hodorogea**, Mircea-Florin Vaida, Monica Borda, Cosmin Striletchi, A Java Crypto Implementation of DNAProvider Featuring Complexity in Theory and Practice, Proceedings of 30-th Int. Conf. on ITI 2008, June 23-26, 2008, Cavtat/Dubrovnik, Croatia, Univ. of Zagreb, Univ. Computing Center, pp. 607-612, IEEE Catalog Number CFP08498-PRT, ISBN: 978-953-7138-12-7, ISSN: 1330-1012, (IEEExplore), INSPEC Database

A Java Crypto Implementation of DNAProvider Featuring Complexity in Theory and Practice

Hodorogea Tatiana, Vaida Mircea-Florin , Borda Monica, Streletchi Cosmin
The Faculty of Electronics, Telecommunications and Information Technologies, Technical University of Cluj-Napoca, 26-28 George Baritiu Street Cluj-Napoca, 400027, Romania
thodorogea@yahoo.com, Mircea.Vaida@com.utcluj.ro, Monica.Borda@com.utcluj.ro, Cosmin.Streletchi@com.utcluj.ro

Abstract. *Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing us to extend the JCE by implementing faster or more secure cryptographic algorithms. By the same means we shall provide our independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB).*
In this work we present a technical process for protecting data assets such as personal medical information using a DNA cryptography technique in which a person's own blood mineral levels serve as a seed for selecting, transmitting, and recovering his sensitive personal data. As we know that the management of security keys remains a challenge, we also propose a mechanism to generate encrypt-decrypt keys by taking into consideration specifics of the cryptography method and the individual's blood analysis.

Our work is based on the complexity of developing, as a subset of JCE, an unconditionally secure DNAE System as part of our security provider, named DNAProvider. We intend to use DNA Provider with the DNAE System in medical applications to ensure security of medical

Keywords: DNA Encryption (DNAE) system, Central Dogma of Molecular Biology (CDMB).

1. Introduction

Why we need data security is already well-known. Do we need to find alternative, more secure encryption techniques for protecting sensitive data? With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's information infrastructure. A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory. When an attacker is theoretically able to intrude, but it is computationally infeasible with available resources, the cryptographic system is said to be *conditionally secure*. The mathematics in such systems is based on computational complexity theory. To design a secure cryptographic system is a very challenging. A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output. If an algorithm's behavior is completely determined by the input, it is called *deterministic*, and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*. A distributed algorithm in which two or more entities take part is defined as a protocol including a set of communicational and computational steps. Each communicational step requires data to be transferred from one side to the other and each computational step may occur only on one side of the protocol. The goal of every cryptographer is to reduce the probability of a successful attack against the security of an encryption system – to zero. Probability theory provides the answer for this goal. Our work is based on the complexity of developing an unconditionally-secure DNA Encryption System as part of DNA Provider. We aim to use DNA Provider with unconditional secure DNAE system in medical applications to ensure security of medical information. However, DNA Provider with DNAE will also have applications in many aspects of today's web-based business processes such as e-commerce, Internet banking, and email.

2. Java Cryptography Extension Architectural Model with DNA Encryption Security Provider

The goal of the security provider interface is to allow a means whereby specific algorithm implementations can be substituted for the default provider, SUN JCE, (Fig. 1). JCE was developed as an extension package which includes implementation for cryptographic services. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. Thus, JCE offers support for developing alternative cryptographic package providers. This support allows us to provide our independent implementation of DNAE System, based on the of the CDMB [11].

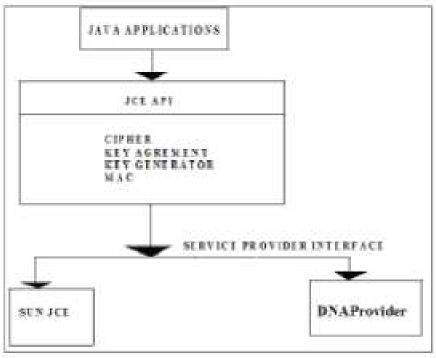


Figure 1. Java Cryptography Extensions architectural model with unconditional secure DNA Encryption as part of our security provider (DNAProvider)

The application code calls the appropriate JCE API classes. The JCE API classes invoke the classes in a provider that implements the interface classes, JCE SPI. The JCE SPI classes, in turn, invoke the the requested functionality of the DNA Provider (Fig.2).
When the Java Virtual Machine starts execution, it examines the user's properties to determine which security providers should be used. The user's properties are located in the file *java.security*, in which each provider is also enumerated. If users prefer to use DNAProvider as an additional security provider they can edit this file and add the DNA Provider. When the Security Class is asked to provide a particular engine and algorithm, it searches the listed providers for the first that can supply the desired operation.

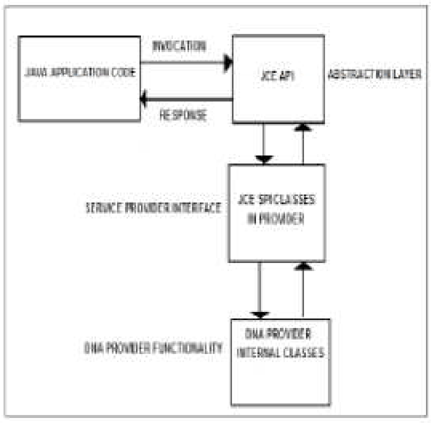

JAVA APPLICATION CODE
INVOCATION
RESPONSE
JCE API
ABSTRACTION LAYER
SERVICE PROVIDER INTERFACE
JCE SPICLASSES IN PROVIDER
DNA PROVIDER FUNCTIONALITY
DNA PROVIDER INTERNAL CLASSES

Figure 2. Invocation of DNAProvider for providing requested functionality

3. Creating the DNA Security Provider with DNA Encryption

An Engine Class defines an abstract cryptographic service, without its concrete implementation. JCE 1.2.2 is provided as an optional package and adds engine classes such as: Cipher, KeyAgreement, KeyGenerator, MAC, and SecretKeyFactory. The application interfaces given by an engine class are implemented and referred to as the *Service Provider Interface*. We wrote the service implementation code and implemented the master class, which is a subclass of DNAProvider. As a next step, we prepared to request a code-signing certificate from SUN for testing. This code-signing certificate will be used to sign DNAProvider. The certificate is valid for five years for both testing and production. Prior to requesting the certificate we will continue work on the complexity of DNA Encryption Protocol, as well as the theoretical and practical possibilities it offers. To provide secure communications based on unconditionally-secure DNA Cryptography, we apply the ramifications of information theory and probability theory. Unconditional security, in a theoretical sense, can only be achieved if the entropy of the secret key is greater than or equal to the entropy of the plain-text message. [8]. Information theory and probability theory continue to have a deep impact on modern cryptography; for example, Quantum Cryptography applies the Heisenberg uncertainty principle of quantum physics to provide a secure channel.

4. Data Hiding in DNA

Recent research considers the use of the Human genome in cryptography. In 2000, the Junior Nobel Prize was awarded to Romanian-American student, Viviana Risca, for her work in DNA steganography [10]. A DNA-encoded message is first camouflaged within the enormous complexity of human genomic DNA, and then further concealed by confining this sample to a microdot. A prototypical 'secret message' DNA strand contains an encoded message flanked by polymerase chain reaction (PCR) primer sequences. Denatured human DNA provides a very complex background for concealing a secret-message.

Risca, knowing both the secret-message DNA, PCR primer sequences and the encryption key could readily amplify the DNA and then proposed a mechanism to read and decode the message. We propose to encode the medical records of an individual in DNA data strand flanked by unique primer sequences, which we obtain in the process of deriving a DNA secret key from blood analysis [9]. The specific mineral levels and their deviation from normal values are considered as a first step. We then mix the message-encoded DNA strand among other decoy DNA strands that will together be sent to a receiver through a public channel.

5. Biological Principles

A DNA segment that constitutes a gene is read, starting from the promoter (starting position) of the DNA segmen, (Fig. 3). The non-coding areas (introns) are removed according to certain tags. The remaining coding areas (extrons) are rejoined and capped. Then the sequence is transcribed into a single stranded sequence of mRNA (messenger RNA). The mRNA moves from the nucleus into the cytoplasm. In chromosomes, DNA acts as a template for the synthesis of RNA in a process called transcription. During RNA Synthesis and Processing in the transcription and splicing steps, introns are excised and extrons are retained to form mRNA, which will perform the translation work.

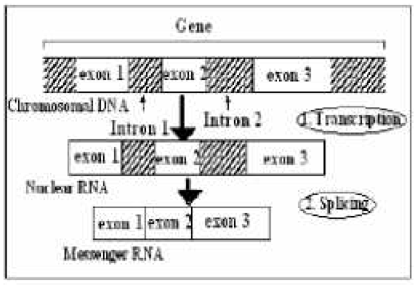


Figure 3. Central Dogma of Molecular Biology

In the translation process, codons are translated into the amino acids according to the genetic code. The DNA form of information is scanned by a hypothetical operator, Stefani, to find the locations of the introns, which she then records [15]. She cuts out the introns according to the specified pattern so that the DNA form of data is translated into the mRNA form. The mRNA form then translates into the protein form of data according to the genetic code table (64 codons to 20 amino acids).

6. The DNA Encryption Protocol

Adleman began the new field of bio-molecular computing research. His idea was to use DNA biochemistry for solving problems that are impossible to solve by conventional computers, or that require an enormous number of computation steps. The DNAE technique simulates the CDMB steps: transcription, splicing, and translation process. The time complexity of an attack on a message of length n, is $O(2^n)$. DNA computing takes advantages of combinatorial properties of DNA for massively-parallel computation.

Introducing DNA cryptography into the common PKI scenario, it is possible to follow the pattern of PKI, while also exploiting the inherent massively-parallel computing properties of DNA bonding to perform the encryption and decryption of the public and private keys [6]. The resulting encryption algorithm used in the transaction is much more complex than the one used by conventional encryption methods.

an intruder, it would prove extremely difficult to read and detect the DNA strand that contains someone's medical history, without knowing the specific unique primer sequences of the specific person. Performing a quick search by a program we will get the chosen mineral (M).

Considering a dedicated medical application with security facilities developed by our research team, we are able to incorporate the blood analysis results of an individual into the security process using the following algorithm:

If the mineral level (L) does not correspond to a normal level (NL) and is equal to value: L=X.YZ mmol/l (1)

Then:

1st step: We associate X times the nucleotide sequence corresponding to the synergetic mineral of the selected mineral, obtaining a sequence S1.

2nd step: In the second step, to S1 we add Y+Z numbers of nucleotide from the original sequence of a synergetic mineral, resulting sequence S that will constitute the unique primer sequence.

Otherwise:

If all blood results are in a normal level we will choose a minimal encoding to generate the unique primer sequences because we don't have to increase the hiding process for an individual of normal health. In this case we will just associate the nucleotide sequence corresponding to a chosen mineral.

As in this heading, they should be Times 12-point boldface, initially capitalized, flush left, with one blank line before, and one after. Use a hanging indent for long headings.

6.1.1 Complexity of Secure DNA Encryption System Based on Probabilistic Theory

Given $n \in N$ as a composite positive integer and $x \in Z^*n$, the Quadratic Residuosity Problem (QRP) is the problem of deciding whether $x \in QR_n$. If an arbitrary element of J_n *is* given, it is computationally difficult to decide whether it is a square or it is a pseudo-square modulo n. Probabilistic DNA encryption can exploit this computational difficulty [3].

The DNA Encrypt algorithm based on CDMB, employed by probabilistic DNA encryption, must specify how a k-bit plain-text message $m = m_1m_2 \ldots m_k$ is DNA-encrypted according to CDMB, so that only Otto, the recipient (or a person holding the Otto's private key), is able to decrypt it. The DNA Encryption algorithm needs to be a probabilistic one, to take as input, a public key (n, y) and a message m, generating the DNA cipher-text c as output. For every message bit m_i (i= 1, ... , k), the DNA Encrypt algorithm needs to choose $x_i \in R\ Z^*n$ and compute: $c_i \equiv x^{2i}(\text{mod } n)$ if $m_i = 0$ and $yx^{2i}(\text{mod } n)$ if $m_i = 1$. In both cases, each message bit m_i must be DNA-encrypted with an element of $Z^*\ n$. The resulting DNA cipher-text c is the k-tuple $c = (c_1, \ldots , c_k)$. For DNA c_i, the DNA decryption algorithm needs to evaluate $e_i = c_i/\ p_i$ and compute $m_i = 0$ if $e_i = 1$, otherwise, $m_i = 1$.

As the last step, the plain-text message m is set to $m = m_1m_2 \ldots m_k$. The DNA Probabilistic encryption is semantically secure.

Probabilistic encryption as proposed by Goldwasser and Micali, is the best notion of security we currently have for asymmetric encryption systems [3].

From a theoretical point of view of the complexity in DNA Probabilistic encryption, there is not much to add. From a practical point of view, the complexity in DNA probabilistic encryption based on CDMB includes the fact that every plain-text message bit m_i is encrypted with an element of $Z^*\ n$, resulting in a considerable message expansion. We need to improve the minimization of message expansion to a constant number of bits.

7. Conclusion

We hope to find that our DNA Encryption System with DNA Cryptography, based on CDMB, probability theory, and information theory, implemented as a subset of the Java Cryptography Extension, may prove to be an unconditionally secure encryption system.

8. Acknowledgements

The research activity is supported by PN2 project 1083 from National Authority for Scientific Research that will be realised during 3 years (2007-2010).

Research activity is supported by Research project PN2-Nr.332/2007(code:909/2007), "cDNA Microarray Image Acquisition and Processing System"- supported by the National Authority of Romania.

9. References

[1.] Titu Bajenescu, Monica Borda " Securitatea in informatica si telecomunicatii", Editura Dacia, 2001

[2.] Boneh, D., and H. Shacham, "Fast Variants of RSA," *CryptoBytes*, Vol. 5, No. 1, 2002

[3.] Goldwasser, S., and S. Micali, "Probabilistic Encryption," Journal of Computer and System Sciences, Vol. 28, No. 2, April 1984,

[4.] Garfinkel Simson, Web Security, Privacy & Commerce, 2nd Edition, O'Reilly Publisher, November 2001

[5.] Gehani Ashish; La Bean, Thomas H.; Reif, John H, "DNA-Based Cryptography," Department of Computer Science, Duke University, June 1999

[6.] Tatiana Hodorogea, Mircea-Florin Vaida, Alternate Cryptography Techniques, ICCC05, Miskolc-Lillafured, Hungary, 24-27 May 2005, Vol. 1, pp. 513-518

[7.] Shoup, V., "OAEP Reconsidered," *Proceedings of CRYPTO '01*, Springer-Verlag, LNCS 2139,2001

[8.] Tatiana Hodorogea, Mircea-Florin Vaida, Blood Analysis as biometric selection of Public Keys, 7th International Carpathian Control Conference ICCC'2006, Ostrava – Beskydy, Czech Republic, May 29-31, 2006, pp. 675-678

[9.] Taylor Clelland Catherine, Viviana Risca, Carter Bancroft, "Hiding Messages in DNA Microdots", Nature Magazine Vol., 399, June 10,

[10.] Tatiana Hodorogea, Mircea-Florin Vaida, Security Provider with DNA cryptography algorithm as a subset of Java Security API, ICCC'2007, Slovakia,Technical University of Kosice, 24-27 may 2007.

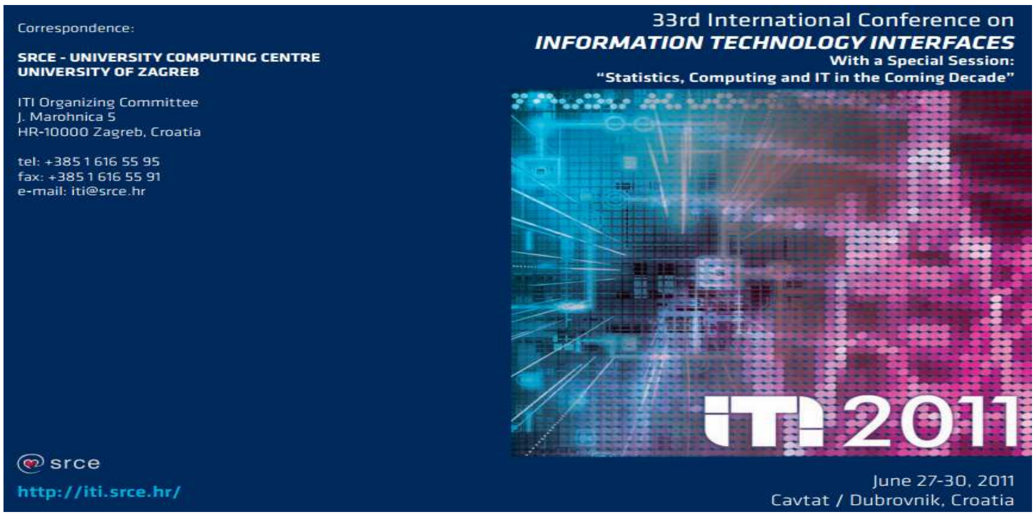
[11.] Kahn D., The Codebrakers, McMillan, New York, 1967

[12.] C. Striletchi, M. Vaida, Enhancing the Security of Web Applications, Information Technology Interfaces, ITI 2003, Proceedings of the 25th International Conference on Knowledge Management and E-Commerce, pp. 463-468, June 2003, Cavtat, Croatia

[13.] Pointcheval, D., "How to Encrypt Properly with RSA," *CryptoBytes*, Vol. 5, No. 1, 2002, Norwood, MA, 2003.

[14.] Vaida Mircea-Florin, Tatiana Hodorogea, Coding Documents using Alternative Techniques, CSNDSP 2006 PROCEEDINGS, July 2006, Patras, Greece, pp.359-361

[15.] Vaida Mircea-Florin, Teaching Computers as a Human Spiritual Evolution, the 4th IASTED International Conference on Web-Based Education, WBE2005, February 21-23, 2005, Grindelwald, Switzerland, pp. 667-672



Correspondence:

SRCE - UNIVERSITY COMPUTING CENTRE
UNIVERSITY OF ZAGREB

ITI Organizing Committee
J. Marohnica 5
HR-10000 Zagreb, Croatia

tel: +385 1 616 55 95
fax: +385 1 616 55 91
e-mail: iti@srce.hr

srce

http://iti.srce.hr/

33rd International Conference on
INFORMATION TECHNOLOGY INTERFACES
With a Special Session:
"Statistics, Computing and IT in the Coming Decade"

ITI 2011

June 27-30, 2011
Cavtat / Dubrovnik, Croatia

2. Tatiana Hodorogea, Ionas Otto**, Security of Business to Business and Business to Customer Software Applications based on the Central Dogma of Molecular Biology (CDMB) and evolutionary models,** Proceedings of the ITI 2011, 33rd International Conference on Information Technology Interfaces (June 2011), pg. 403-408, IEEE Catalog Number: CFPl 1498-PRT, ISBN: 978-1-61284-897-6

Security of Business to Business and Business to Customer Software Applications Based on the Central Dogma of Molecular Biology (CDMB) and Evolutionary Models

Tatiana Hodorogea [1], Ionas Szilard Otto [2]
[1]*INNOVA BIOTECH, Cluj-Napoca, Romania*
[2]*INNOVA BIOTECH, Cluj-Napoca, Romania*
E-mail(s):thodorogea@yahoo.com, otto.ionas@yahoo.com

Abstract.
The development, implementation and testing for Security of Business to Business and Business to Customer Software Applications System is based on the Central Dogma of Molecular Biology (CDMB), where we derive DNA Cryptographic Keys based on evolutionary models as public-key algorithms are based on mathematical functions rather than on substitution and permutation involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution and because of the degeneracy of the genetic. Based on evolutionary models we extract and align the DNA sequences of the same gene from related chosen species with respect to human DNA Sequences. The alignment in the evolutionary system pipeline is realized with ProbCons tool, which is a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy. After aligning our extracted DNA Sequences with ProbCons we derive the private/public pair DNA cryptographic keys based on evolutionary models mathematical functions used for Security of Web-based Business Processes Software Applications

Keywords DNA cryptographic keys, evolutionary models, Hidden Markov Model, Public-key Introduction

1. Introduction

The aim of this paper is the development of a DNA Cryptographic Keys Based on Evolutionary Models, for the integration in our DNAProvider as Java Cryptographic Extension (JCE) with DNA Encryption (DNAE) system for use in security of our developed Web-based Business Processes Software Applications.

Nowadays information systems involve more complexity because of there heterogeneity involving very big threats on, networks which are widely spread open and interconnected. The security attacks and the technologies to exploit security attacks are growing continuously.

The importance of providing and maintaining the data and information security across networks is a major enterprise business activity resulting in a big demand and need to ensure and maintain information security. With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's web-based Business to Business and Business to Customer Software Applications. A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory, [4]. The development, implementation and testing for Security of Business to Business and Business to Customer Software Applications System is based on the Central Dogma of Molecular Biology (CDMB), where we derive DNA Cryptographic Keys based on evolutionary models as Public-Key algorithms are based on mathematical functions rather than on substitution and permutation involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output.

If an algorithm's behavior is completely determined by the input, it is called *deterministic*, and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*.

Our work is based on Deriving DNA Cryptographic Keys Based on Evolutionary

Models for Security of Business to Business and Business to Customer Software Applications.

When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution and because of the degeneracy of the genetic. Based on evolutionary models we extract and align the DNA sequences of the same gene from related chosen species with respect to human DNA Sequences. The alignment in the evolutionary system pipeline is realized with ProbCons tool, which is a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy. After aligning our extracted DNA Sequences with ProbCons we derive the private/public pair DNA cryptographic keys based on evolutionary models mathematical functions. Our work is based on the complexity of deriving DNA Cryptographic Keys Based on Evolutionary Models for Security of Web-based Business Processes.

2. Java Data Security

Java Cryptography Extension (JCE) was developed as an extension package which includes implementation for cryptographic services. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. The security provider interface the means by which different security implementations may be plugged into the security package as message digests, encryption, digital signatures and keys, through JCE, JSSE and authentication through JAAS.

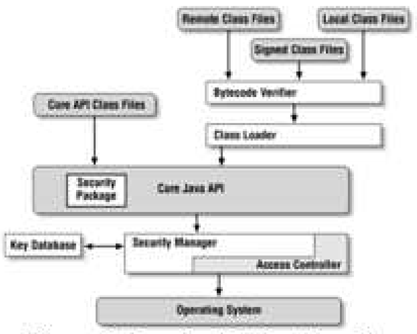


Figure 1. Java Application Security

Thus, JCE support allowed us to provide our independent implementation of DNA Cryptographic Keys Based on Evolutionary Models used for Security of Web-based Business Processes.

The classes necessary to handle secret keys come only with JCE. Keys and certificates are normally associated with some person or organization, and the way in which keys are stored, transmitted, and shared is an important topic in the security package

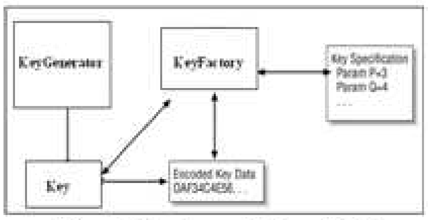


Figure 2.Implementation of DNA Cryptographic Keys Based on Evolutionary Models

A generator class creates DNA Cryptographic Keys from scratch, (Fig 2). Symmetric keys are generated by the KeyGenerator class while asymmetric key pairs are generated by the KeyPairGenerator class.

The KeyFactory class translates between key objects and their external representations. There are two abstract public methods of the key pair generator SPI: the initialize() method and the generateKeyPair() method. The KeyGenerator class (javax.crypto.KeyGenerator) is used to generate secret keys

The important operations done are the ability to create new DNA Cryptographic Keys based on Evolutionary Models from scratch, using the key pair generator or the key generator and the ability to export our keys, either as a parameter specification or as a set of bytes, and the corresponding ability to import that data in order to create a DNA cryptographic key. A key may be transmitted by its encoded format or by the parameters that were used to generate the key. Either of these representations may be encapsulated in a key specification, which is used to interact with the *KeyFactory class (java.security.KeyFactory)* and the *SecretKeyFactory class (javax.crypto.SecretKeyFactory).*
Key management system we built around the notion of a keystore. Key stores are created and

manipulated though an administrative tool, keytool and there is a Java API that allows us to use key stores programmatically.

3. DNA Biology and Evolutionary Models

Resent research considers the use of the Human genome in cryptography and the famous DNA one-time-pad encryption schemes utilizing the indexed of random key string was first developed by Ashish Gehani, Thomas H. LaBean and John H. Reif.

At the lowest level, a genome can be described as a long string of nucleotides. It could be compared to a very long text made of four letters (strings of DNA). All living organisms consist of cells and in each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism made from genes, which are made from blocks of DNA. Complete set of genetic material (all chromosomes) is called genome. The assumption of evolutionary models is that biological systems have evolved from the same origin, constantly reusing some basic building blocks and through the cycles of mutation and selection that constitute evolution, new functions have been created by reusing pieces of already existing DNA machinery. If we consider this problem in terms of sequences, this means that two sequences responsible for similar functions may be different, depending on how long they have been diverging. Many of the problems in bioinformatics and more specifically in sequence alignment are said to be NP complete as the number of potential solutions rises exponentially with the number of sequences and their length and the solution cannot be found in polynomial time and space. A sequence alignment is the representation of two sequences in a way that reflects their relationship and if the alignment is designed to reflect phylogenetic relationships, the residues will be aligned when they originate from the same residue in the common ancestor. If a given sequence lacks one residue, a gap will be inserted in its place at the corresponding position, in an evolutionary model context, a null sign means that a residue was inserted in one of the sequences or deleted in the other while the sequences were diverging from their common ancestor.

4. Secure data transmission and reception in a Web-Based Business Process

To put this into the common description of secure data transmission and reception in a Web-Based Business Process, let us say Stefani is the sender, and Otto, the receiver. Stefani provides Otto her DNA Cryptographic Public key derived based on evolutionary models, while only he knows his secret key, [3]. Thus, anyone can use Otto's public key to send him an encrypted message, but only Otto knows the secret key to decrypt it.

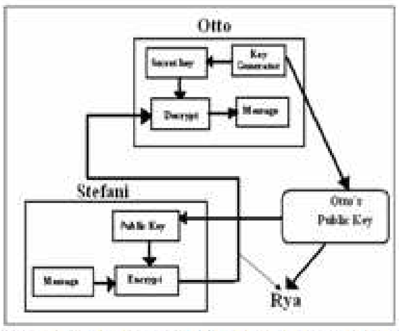


Figure 3. Implementation security for data transmission and reception

Thus, anyone can use Otto's public key to send him an encrypted message, but only Otto knows the secret key to decrypt it.

5. The Technique of Deriving DNA Cryptographic Keys Based on Evolutionary Models

As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key, [4]. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution.

We developed a Unique Process System Pipeline Evolutionary Models of deriving DNA Cryptographic Keys Sequences by deriving the DNA private/public keys from human genome analysis by computing the philogenetic tree relating and the branch length during evolution for chosen species. The molecular evolution model assigns probabilities to multiple-alignment columns in terms of the the philogenetic tree branches and is time dependent of frequency selections. Based on Kimura-Ohta theory Halpern and Bruno, [1] have shown that mutation limit can be determined by substitution rates in terms of the mutation rates and equilibrium frequencies.

We derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

We started with extracting from public available database all orthologus DNA coding sequences for all genes, from related species with respect to Human Genome sequences. A genome of a reference species in our case is Human Genom (hg18) and two more additional genomes are: Taurus Genome (bosTau3) and Dog Genome (canFam2). We extracted the DNA sequences for 29.000 genes which equals to 44103 pages in printable format. Using a trained parameter set for ProbCons tool we aligned all orthologus DNA coding sequences of our choosen species for all genes with respect to Human DNA coding sequences.

ProbCons achieved the highest accuracies of all multiple alignments methods as it uses probabilistic modeling and consistency-based alignment techniques.

We computed the philogenitic tree for our chosen species and the branch length during evolution, (Fig. 4) with respect to human genome (hg18).

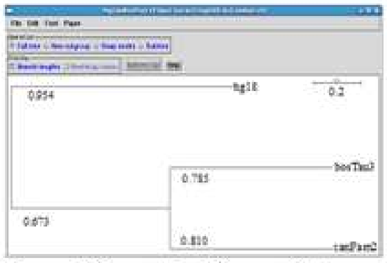


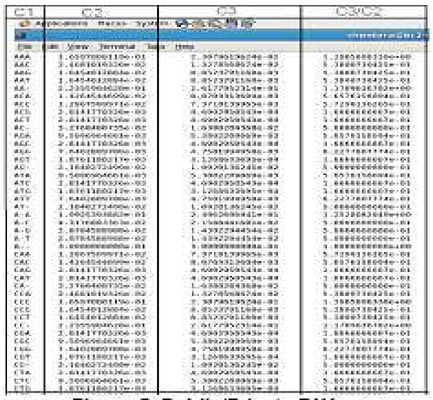
Figure 4. Computed philogenetic tree and the branch length

A Software Application, reeds the tree, computes the pairwise alignment, computes the branches of the tree for our chosen mammalian species. Public-key algorithms are based on mathematical functions, rather than on substitution and permutation and involve the use of two separate keys in contrast to symmetric encryption, (Fig. 4).

In (Fig. 5), Second Column (C2) model represents the computed DNA Public Keys, with respect to Column C1 and assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, (Fig. 5), and represents the public keys.

Given the transition probabilities and given a phylogenetic tree we calculated the ratio for an alignment column C3/C2, which is the product over transition probabilities for each branch of the tree we summed over all possible nucleotides for internal nodes calculated by recursive algorithm introduced by Felsenstein.

The first column C1 represents all possible three base sequences with respect to human species. Second Column (C2) model, with respect to C1 assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, and represents the public key. The third Column (C3) assumes that at a given position, the substitution rates are altered during due to specific selection preferences for a certain base. The last Column is the ratio C3/C2 and represents the private key, (Fig. 5).



C1	C2	C3	C3/C2
AAA	[illegible]	[illegible]	[illegible]
AAC	[illegible]	[illegible]	[illegible]
AAG	[illegible]	[illegible]	[illegible]
AAT	[illegible]	[illegible]	[illegible]
AA-	[illegible]	[illegible]	[illegible]
ACA	[illegible]	[illegible]	[illegible]
ACC	[illegible]	[illegible]	[illegible]
ACG	[illegible]	[illegible]	[illegible]
ACT	[illegible]	[illegible]	[illegible]
AC-	[illegible]	[illegible]	[illegible]
AGA	[illegible]	[illegible]	[illegible]
AGC	[illegible]	[illegible]	[illegible]
AGG	[illegible]	[illegible]	[illegible]
AGT	[illegible]	[illegible]	[illegible]
AG-	[illegible]	[illegible]	[illegible]
ATA	[illegible]	[illegible]	[illegible]
ATC	[illegible]	[illegible]	[illegible]
ATG	[illegible]	[illegible]	[illegible]
ATT	[illegible]	[illegible]	[illegible]
AT-	[illegible]	[illegible]	[illegible]
A-A	[illegible]	[illegible]	[illegible]
A-C	[illegible]	[illegible]	[illegible]
A-G	[illegible]	[illegible]	[illegible]
A-T	[illegible]	[illegible]	[illegible]
A--	[illegible]	[illegible]	[illegible]
CAA	[illegible]	[illegible]	[illegible]
CAC	[illegible]	[illegible]	[illegible]
CAG	[illegible]	[illegible]	[illegible]
CAT	[illegible]	[illegible]	[illegible]
CA-	[illegible]	[illegible]	[illegible]
CCA	[illegible]	[illegible]	[illegible]
CCC	[illegible]	[illegible]	[illegible]
CCG	[illegible]	[illegible]	[illegible]
CCT	[illegible]	[illegible]	[illegible]
CC-	[illegible]	[illegible]	[illegible]
CGA	[illegible]	[illegible]	[illegible]
CGC	[illegible]	[illegible]	[illegible]
CGG	[illegible]	[illegible]	[illegible]
CGT	[illegible]	[illegible]	[illegible]
CG-	[illegible]	[illegible]	[illegible]
CTA	[illegible]	[illegible]	[illegible]
CTC	[illegible]	[illegible]	[illegible]
CTG	[illegible]	[illegible]	[illegible]

Figure 5. Public/Private DNA Cryptographic Keys

Using the same model and desired length of bases from the first column we can derive the public/private keys used in Java KeyStore with respect to human or desired number of species. Resulting in new set of public/private *DNA Cryptographic Keys for our Java DNA KeyStore usage.*

5. Conclusions and Future Work

We developed the cryptographic package provider, named DNAProvider as Java Cryptographic Extension (JCE), extending the JCE by implementing faster and more secure DNA Encryption (DNAE) system based on the Central Dogma of Molecular Biology (CDMB), [5]. Sun Microsystems certified and signed our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm. We got the Code Signing Certificate from Sun Microsystems for our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm which is available for 5 years, until with the reference #679, when renewing it in 2013. We intend to integrate the derived DNA Cryptographic Keys Based on Evolutionary Models, in our DNAProvider as Java Cryptographic Extension (JCE) with DNA Encryption (DNAE) system for use in security of our developed Web-based Business Processes Software Applications.

6. Acknowledgements

This research work is supported by the Company INNOVA BIOTECH, Cluj-Napoca, Romania.

7. References

[1]Halpern, A.L. and Bruno, W.J. 1998. Evolutionary distances forprotein-coding sequences: Modeling site-specific residue frequencies.Mol. Biol. Evol. 5: 910–917.

[2]Halligan, D.L., Eyre-Walker, A., Andolfatto, P., and Keightley, P.D. 2004.Patterns of evolutionary constraints in intronic and intergenic DNA of *Drosophila*. *Genome Res.* **14:** 273–27

[3]Tatiana Hodorogea, Mircea-Florin Vaida, Security Provider with DNA Cryptography Algorithm as a Subset of Java Security API, ICCC`2007, Slovakia, Technical University of Kosice, 24-27 may 2007, pp. 175-178.

[4]Tatiana Hodorogea, Mircea-Florin Vaida, Complexity of DNA Encryption System as a Subset of Java Cryptography Extension, IASTED International Conference on Biomedical Engineering (BioMed 2008), 13-15 Feb., Innsbruck, Austria, Acta Press From Proceeding (601) Biomedical Engineering - 2008, pp. 19-24

[5]Tatiana Hodorogea, Mircea-Florin Vaida, Monica Borda, "A Java Crypto Implementation of DNAProvider Featuring Complexity in Theory and Practice", IEEE Explore (ITI) 2008 International Conference on Information Technology Interfaces, 23-26 June, 2008, Cavtat, Croatia, pp. 607-612

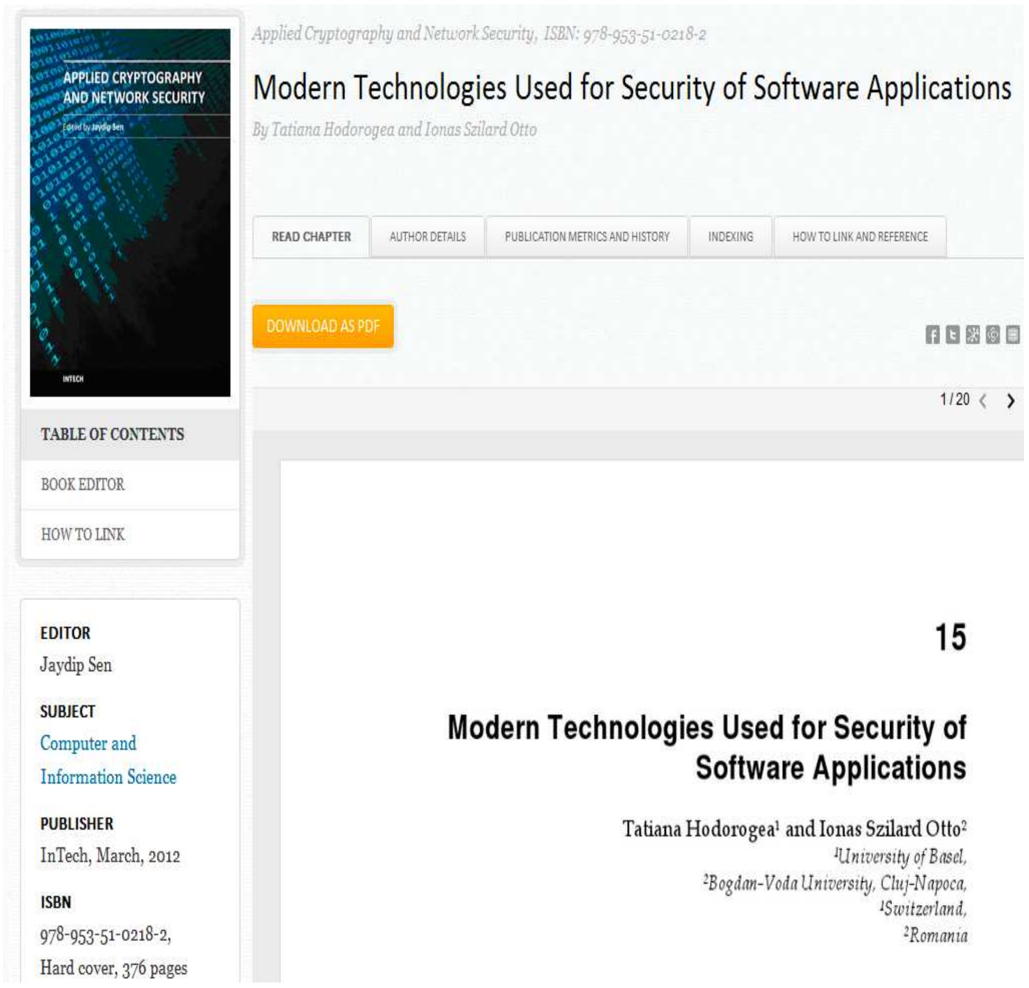
[6] Rajewsky, N., Socci, N.D., Zapotocky, M., and Siggia, E.D. 2002. The evolution of DNA regulatory regions for proteo-gamma bacteria by interspecies comparisons. *Genome Res.* **12:** 298–308

[7] Rogozin, I.B., Makarova, K.S., Natale, D.A., Spiridonov, A.N., Tatusov, R.L., Wolf, Y.I., Yin, J., and

[8]Koonin, E.V. 2002. Congruent evolution of different classes of non-coding DNA in prokaryoticgenomes.Nucleic Acids Res. 30: 4264–4271. doi:2001. Codon bias at the 3_-side of the initiation codon is correlated

[9]van Nimwegen, E. 2003. Scaling laws in the functional content of genomes. Trends Genet. 19:

[10]van Nimwegen, E. 2004. Scaling laws in the functional content of genomes: Fundamental constants of evolution In Power laws, scale-free networks and genome biology (eds. E. Koonin et al.), pp.236–253 Landes Bioscience, Austin, TX.



Applied Cryptography and Network Security, ISBN: 978-953-51-0218-2

Modern Technologies Used for Security of Software Applications

By Tatiana Hodorogea and Ionas Szilard Otto

EDITOR
Jaydip Sen

SUBJECT
Computer and Information Science

PUBLISHER
InTech, March, 2012

ISBN
978-953-51-0218-2,
Hard cover, 376 pages

15

Modern Technologies Used for Security of Software Applications

Tatiana Hodorogea[1] and Ionas Szilard Otto[2]
[1]*University of Basel,*
[2]*Bogdan-Voda University, Cluj-Napoca,*
[1]*Switzerland,*
[2]*Romania*

3. Tatiana Hodorogea, Szilard Otto Ionas**, Chapter, "Modern Technologies Used for Security of Software Applications",** Cryptography/Book 2, Pub: InTech, ISBN 979-953-307-863-1, edited by, Dr. Jaydip Sen, March 2012

15

Modern Technologies Used for Security of Software Applications

Tatiana Hodorogea[1] and Ionas Szilard Otto[2]
[1]*University of Basel,*
[2]*Bogdan-Voda University, Cluj-Napoca,*
[1]*Switzerland,*
[2]*Romania*

1. Introduction

Nowadays information systems security services involve more complexity because of there heterogeneity involving very big threats and attacks on such kind of networks, which are widely spread, open and interconnected. The security attacks and the technologies to exploit security attacks are growing continuously.

The importance of providing and maintaining the data and information security across networks is a major enterprise business activity, resulting in a big demand and need to ensure and maintain information security.

Cryptographic algorithms for confidentiality and authentication play a major importance role in nowadays information security.

With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's web-Software Applications. A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory. The goal of every cryptographer is to reduce the probability of a successful attack against the security of an encryption system – to zero and the probability theory provides the answer for this goal.

The aim and objective of this chapter is the development of a DNA Cryptographic Keys Based on Evolutionary Models, for the integration in our DNAProvider as Java Cryptographic Extension (JCE) with DNA Encryption (DNAE) system for use in security of our developed Web-based Software Applications.

Java Cryptography Extension (JCE) was developed as an extension package which includes implementation for cryptographic services. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. The security provider interface the means by which different security implementations may be plugged into the security package as message digests, encryption, digital signatures and keys, through JCE, JSSE and authentication through JAAS. Thus, JCE support allowed us to provide our independent implementation of DNA Cryptographic Keys Based on Evolutionary Models used for Security of Web-based Business Processes.

As Public-Key algorithms are based on mathematical functions rather than on substitution and permutation involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key we developed, implemented and tested the Security System Software Applications based on the Central Dogma of Molecular Biology (CDMB), where we derived DNA Cryptographic Keys based on evolutionary models. Our cryptographic system has one or more algorithms which implements a computational procedure by taking a variable input and generating a corresponding output.

If an algorithm's behavior is completely determined by the input, it is called *deterministic*, and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*.

Our work was based on the complexity of developing, as a subset of JCE, an unconditionally secure DNAE System as part of our security provider, named DNAProvider, (Hodorogea, Ionas, 2011).

Our work is based on Deriving DNA Cryptographic Keys Based on Evolutionary Models for Security of Software Applications.

Biotechnological Methods as recombinant DNA have been developed for a wide class of operations on DNA and RNA strands.

When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution and because of the degeneracy of the genetic code. Based on evolutionary models we extract and align the DNA sequences of the same gene from related chosen species with respect to human DNA Sequences. The alignment in the evolutionary system pipeline is realized with ProbCons tool, which is a pair-hidden Markov model-based on progressive alignment algorithm that primarily differs from most typical approaches in its use of maximum expected accuracy. After aligning our extracted DNA Sequences with ProbCons we derive the private/public pair DNA cryptographic keys based on evolutionary models mathematical functions. The molecular evolution model assigns probabilities to multiple-alignment columns in terms of the the philogenetic tree branches and is time dependent of frequency selections. Based on Kimura-Ohta theory Halpern and Bruno who have shown that mutation limit can be determined by substitution rates in terms of the mutation rates and equilibrium frequencies. Our work described in this chapter was based on the complexity of deriving DNA Cryptographic Keys Based on Evolutionary Models for Security of Software Applications.

2. Data security and cryptography

Networks are based on a number of network level equipments and servers as:

- **Dynamic host configuration protocol (DHCP)**, server dynamically assigns an IP address.
- **Domain name system (DNS)**, server translates a domain name (URL) into an IP address.

- **Network address translation (NAT)**, performs translation between private and public addresses.
- **E-mail server** supports electronic mailing
- Internet/Intranet/Extranet Web servers
- **Access points (AP)**, giving wireless equipments access to wired network.
- **Virtual LAN (VLAN)** which virtually separate flows over the same physical network, so that direct communications between equipments from different VLANs could be restricted and required to go through a router for filtering purposes
- **Network access server (NAS) / Broadband access server (BAS)**, gateways between the switched phone network and an IP-based network
- **Intrusion detection system (IDS) / Intrusion prevention system (IPS)** used to detect intrusions based on known intrusion scenario signatures.

More than 20 years information security considers confidentiality, integrity and availability, known as CIA as the base of information security. Cryptography gives us all of these services, linked with transmitted or stored data.

Considering GRID computing security where the heterogeneous resources are shared and located in different places belonging to different administrative domains over a heterogeneous network, additional security requirements must be satisfied compare to classical network security.

A GRID is a software toolbox and provides services for managing distributed software resources. Securing information in GRID computing encompasses verifying the integrity of the message against malicious modification, authenticating the source of a message and assuring the confidentiality of the message being sent.

The key points of information security are:

- Confidentiality- keeping the data secret
- Integrity - keeping the data unmodified
- Authentication - certifies the source of the data
- Non-repudiation - the process of the sent data can't be negated

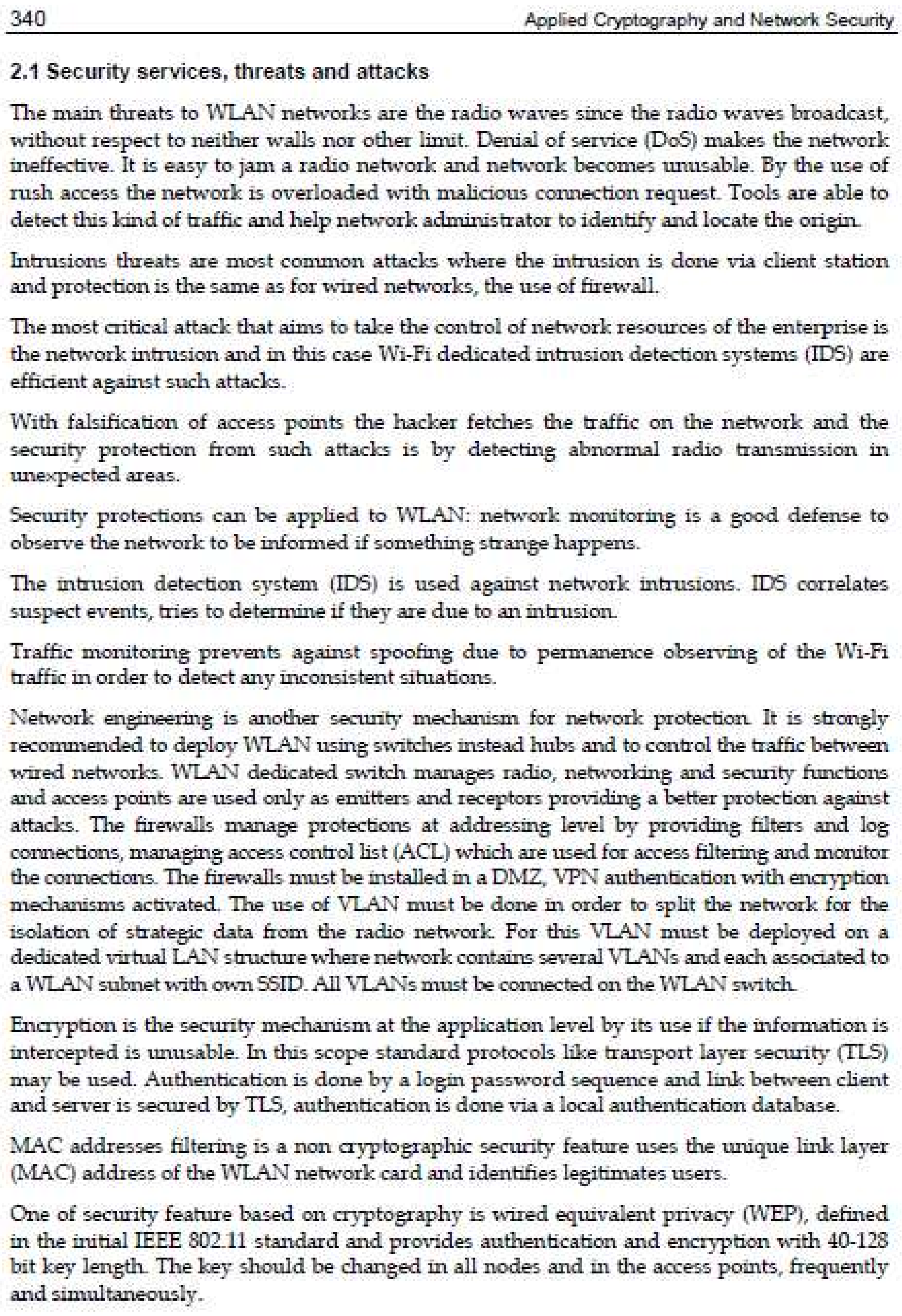
Confidentiality implies the prevention to disclosure information by individuals and unauthorized systems.

In information security integrity implies the impossibility of data modification without the authorization and keeping the data unchanged. Authentication means the knowledge of the source, from where the data was received.

The information needs to be available when necessary. The assurance of availability implies the prevention of denial of service attacks.

Communication between GRID entities must be secure and confidentiality must be ensured for sensitive data, from communication stage, to potential storage stage. Problems of integrity should be detected in order to avoid treatment faults, availability is directly linked to performance and cost in GRID environment.

Cryptographic algorithms for confidentiality and authentication play a major importance role in nowadays information security.



2.1 Security services, threats and attacks

The main threats to WLAN networks are the radio waves since the radio waves broadcast, without respect to neither walls nor other limit. Denial of service (DoS) makes the network ineffective. It is easy to jam a radio network and network becomes unusable. By the use of rush access the network is overloaded with malicious connection request. Tools are able to detect this kind of traffic and help network administrator to identify and locate the origin.

Intrusions threats are most common attacks where the intrusion is done via client station and protection is the same as for wired networks, the use of firewall.

The most critical attack that aims to take the control of network resources of the enterprise is the network intrusion and in this case Wi-Fi dedicated intrusion detection systems (IDS) are efficient against such attacks.

With falsification of access points the hacker fetches the traffic on the network and the security protection from such attacks is by detecting abnormal radio transmission in unexpected areas.

Security protections can be applied to WLAN: network monitoring is a good defense to observe the network to be informed if something strange happens.

The intrusion detection system (IDS) is used against network intrusions. IDS correlates suspect events, tries to determine if they are due to an intrusion.

Traffic monitoring prevents against spoofing due to permanence observing of the Wi-Fi traffic in order to detect any inconsistent situations.

Network engineering is another security mechanism for network protection. It is strongly recommended to deploy WLAN using switches instead hubs and to control the traffic between wired networks. WLAN dedicated switch manages radio, networking and security functions and access points are used only as emitters and receptors providing a better protection against attacks. The firewalls manage protections at addressing level by providing filters and log connections, managing access control list (ACL) which are used for access filtering and monitor the connections. The firewalls must be installed in a DMZ, VPN authentication with encryption mechanisms activated. The use of VLAN must be done in order to split the network for the isolation of strategic data from the radio network. For this VLAN must be deployed on a dedicated virtual LAN structure where network contains several VLANs and each associated to a WLAN subnet with own SSID. All VLANs must be connected on the WLAN switch.

Encryption is the security mechanism at the application level by its use if the information is intercepted is unusable. In this scope standard protocols like transport layer security (TLS) may be used. Authentication is done by a login password sequence and link between client and server is secured by TLS, authentication is done via a local authentication database.

MAC addresses filtering is a non cryptographic security feature uses the unique link layer (MAC) address of the WLAN network card and identifies legitimates users.

One of security feature based on cryptography is wired equivalent privacy (WEP), defined in the initial IEEE 802.11 standard and provides authentication and encryption with 40-128 bit key length. The key should be changed in all nodes and in the access points, frequently and simultaneously.

Because of WEP weakness, the IEEE designed a protocol named 802.11i, known as WPA 2 (Wi-Fi Protected Access 2). Temporal key integrity protocol (TKIP) is used for generating per-packet keys for the RC4 ciphering used. The key is called temporal because is changed frequently and is combined with the sender's MAC address using the exclusive OR-operation. Resulting in the usage of different keys for upstream and downstream transmissions,

Two types of security mechanisms are known: first type is the one which are implemented in a certain protocol layer. Second type of the security mechanisms are not related to protocol layers or any security services.

Cryptography encrypts the data by the mean of using encryption security mechanism.

Encryption security mechanism is an encryption algorithm which encrypts and decrypts the data, transforming it into unreadable format.

The encryption mechanism depends on encryption keys being used (zero or more) and encryption algorithm. After the readable data is cryptographically transformed, digital signature is appended to it as a second security mechanism. *Digital signature security mechanism* proves the integrity of the data, the source of the data and protects the information send against forgery.

The access right to information and resources is realized thought the third security mechanism known as: *access control security mechanism*.

For preventing traffic analysis attempts the bits are inserted into the gaps of the data stream and this constitutes the *traffic padding security mechanism*.

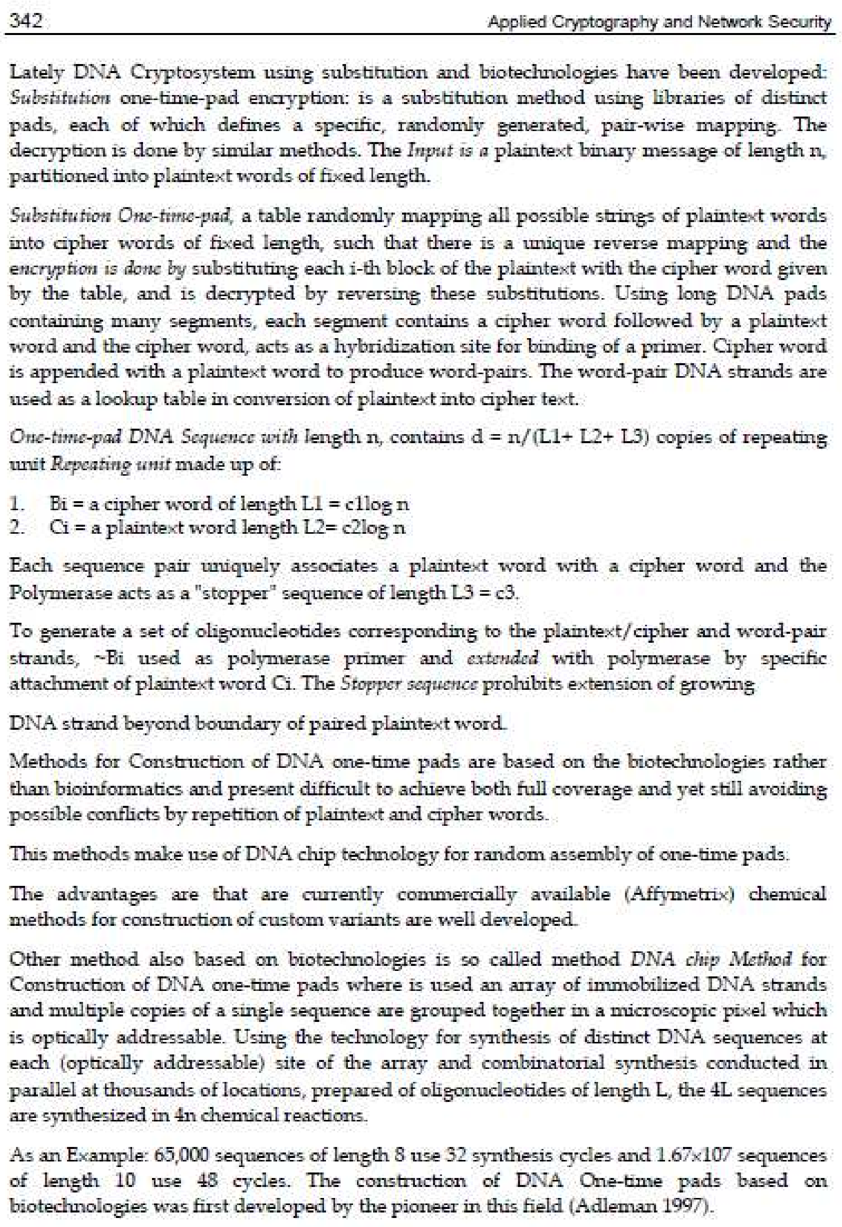
Data security model represents a secure transfer of information across information channel (internet), between two principals: sender and receiver, by the use of communication protocols. Data security model implies the protection of data against confidentiality and authentication threats coming from an opponent. Security related transformation is needed to satisfy these conditions of data protection during transfer through information channel. Encryption transforms the message in an unreadable format, by the opponent. The additional code is added to the secret information based on the content of the message and this way the identity of the sender is verified.

3. DNA cryptography model

With current network, Internet, and distributed systems, cryptography has become a key technology to ensure the security of today's information infrastructure.

Biotechnological Methods as recombinant DNA have been developed for a wide class of operations on DNA and RNA strands. Bio Molecular Computation (BMC) makes use of biotechnological methods for doing computation and splicing operations allow for universal computation.

The first applications of DNA-based cryptography systems using biotechnologies techniques included: methods for 2D data input and output by use of chip-based DNA micro-array technology and transformation between conventional binary storage media via (photo-sensitive and/or photo emitting) DNA chip arrays



Lately DNA Cryptosystem using substitution and biotechnologies have been developed: *Substitution* one-time-pad encryption: is a substitution method using libraries of distinct pads, each of which defines a specific, randomly generated, pair-wise mapping. The decryption is done by similar methods. The *Input is a* plaintext binary message of length n, partitioned into plaintext words of fixed length.

Substitution One-time-pad, a table randomly mapping all possible strings of plaintext words into cipher words of fixed length, such that there is a unique reverse mapping and the *encryption is done by* substituting each i-th block of the plaintext with the cipher word given by the table, and is decrypted by reversing these substitutions. Using long DNA pads containing many segments, each segment contains a cipher word followed by a plaintext word and the cipher word, acts as a hybridization site for binding of a primer. Cipher word is appended with a plaintext word to produce word-pairs. The word-pair DNA strands are used as a lookup table in conversion of plaintext into cipher text.

One-time-pad DNA Sequence with length n, contains d = n/(L1+ L2+ L3) copies of repeating unit *Repeating unit* made up of:

1. Bi = a cipher word of length L1 = c1log n
2. Ci = a plaintext word length L2= c2log n

Each sequence pair uniquely associates a plaintext word with a cipher word and the Polymerase acts as a "stopper" sequence of length L3 = c3.

To generate a set of oligonucleotides corresponding to the plaintext/cipher and word-pair strands, ~Bi used as polymerase primer and *extended* with polymerase by specific attachment of plaintext word Ci. The *Stopper sequence* prohibits extension of growing

DNA strand beyond boundary of paired plaintext word.

Methods for Construction of DNA one-time pads are based on the biotechnologies rather than bioinformatics and present difficult to achieve both full coverage and yet still avoiding possible conflicts by repetition of plaintext and cipher words.

This methods make use of DNA chip technology for random assembly of one-time pads.

The advantages are that are currently commercially available (Affymetrix) chemical methods for construction of custom variants are well developed.

Other method also based on biotechnologies is so called method *DNA chip Method* for Construction of DNA one-time pads where is used an array of immobilized DNA strands and multiple copies of a single sequence are grouped together in a microscopic pixel which is optically addressable. Using the technology for synthesis of distinct DNA sequences at each (optically addressable) site of the array and combinatorial synthesis conducted in parallel at thousands of locations, prepared of oligonucleotides of length L, the 4L sequences are synthesized in 4n chemical reactions.

As an Example: 65,000 sequences of length 8 use 32 synthesis cycles and 1.67×107 sequences of length 10 use 48 cycles. The construction of DNA One-time pads based on biotechnologies was first developed by the pioneer in this field (Adleman 1997).

XOR One-time-pad (Vernam Cipher) Cryptosystem based on biotechnologies One-time-pad: S is a sequence of independently distributed random bits

M is a plaintext binary message of n bits resulting in the following cipher text ,

Ci = Mi XOR Si for = 1,...,n .

Decrypted bits, use commutative property of XOR Ci XOR resulting in:

Si = (Mi XOR Si) XOR Si= Mi XOR (Si XOR Si)= Mi.

DNA Implementation of XOR One-time-pad Cryptosystem:

The *plaintext messages is* one test tube of short DNA strands

The *encrypted message is* another test tube of different short DNA strands

Encryption by XOR One-time-pad maps these in a random and reversible way such as plaintext is converted to cipher strands and plaintext strands are removed. For the *efficient* DNA encoding Adleman proposed to use *modular base 4 as* DNA has four nucleotides. Encryption constitutes the addition of one-time-pad elements modulo 4 and decryption is the subtract one-time-pad elements modulo.

Details of DNA Implementation of XOR One-time-pad Cryptosystem based on biotechnologies:

Each plaintext message has appended a unique *prefix index tag* of length L indexing it.

Each of one-time-pad DNA sequence has appended unique *prefix index tag* of same length L, forming *complements* of plaintext message tags. Using recombinant DNA bio techniques such as annealing and ligation in order to *concatenate into a single DNA strand* each corresponding pair of a plaintext message and a one-time-pad sequence resulting in *enciphered by bit-wise XOR computation and* fragments of the plaintext are converted to cipher strands using the one-time-pad DNA sequences, and plaintext strands are removed.

The reverse decryption is similar using commutative property of bit-wise XOR operation.

BMC Methods to effect bit-wise XOR on Vectors. This method can adapt BMC methods for binary addition and similar to bit-wise XOR computation can disable carry-sums logic to do XOR

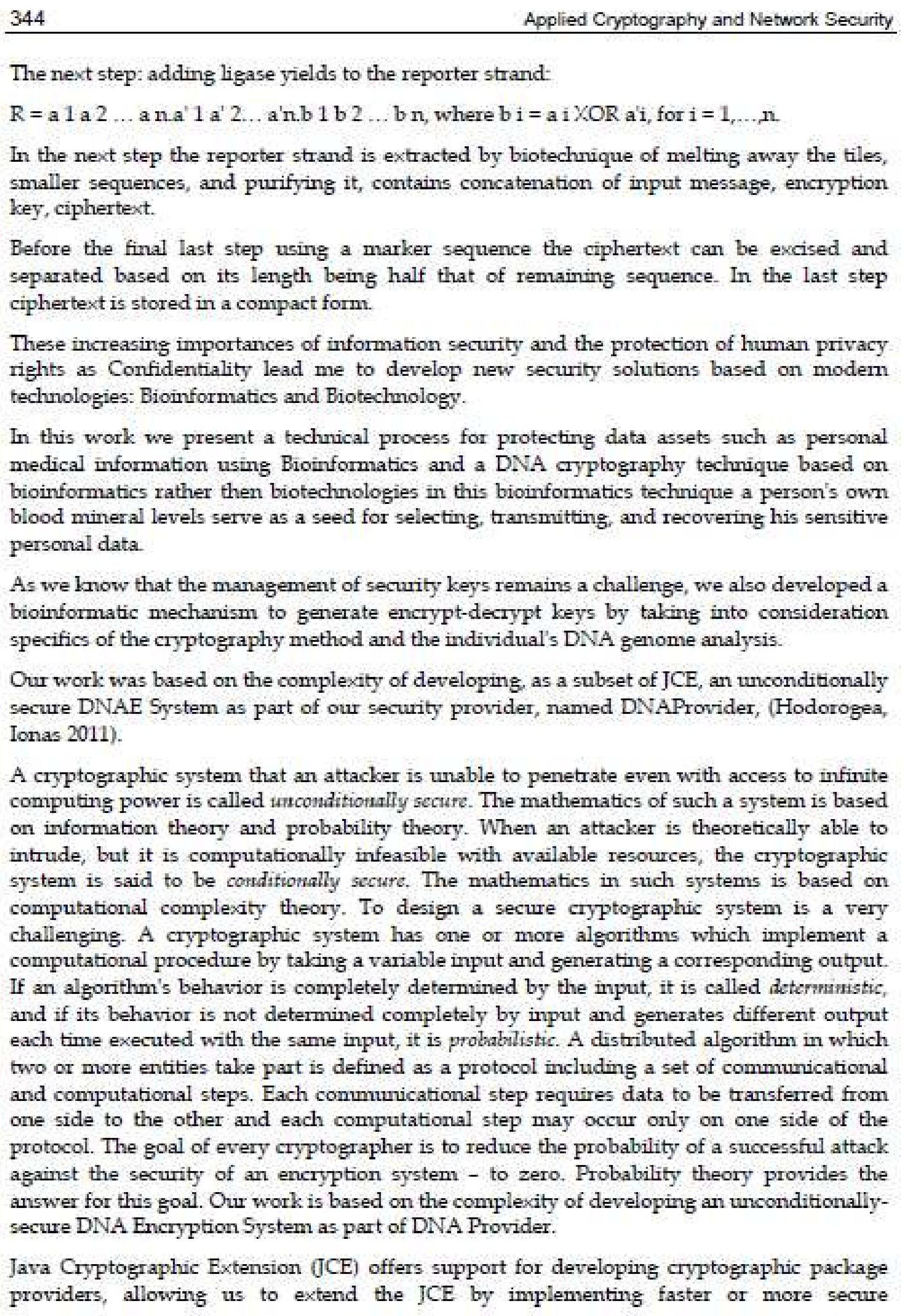
BMC techniques for Integer Addition were implemented by (Guarnieri, Fliss, and Bancroft 96), first BMC addition operations (on single bits) by (Rubin el al 98, OGB97, LKSR97, GPZ97) permit chaining on n bits.

Addition by *Self Assembly* of DNA tiles was exploited by (Reif, 97) and (LaBean, 99)

XOR by Self Assembly of DNA tiles (LaBean, 99): *XOR by Self Assembly of DNA tiles includes that for* each bit Mi of the message, construct sequence ai that represents the ith bit.

Scaffold strands for binary inputs to the XOR are the usage of linkers to assemble the message M's n bits into scaffold strand sequence *a1, a 2 ... a n.*

The One-time-pad is further portion scaffold strand *a' 1a' 2... a'n* and is created from random inputs add output tiles, the annealing give self assembly of the tiling.



The next step: adding ligase yields to the reporter strand:

R = a 1 a 2 … a n.a' 1 a' 2… a'n.b 1 b 2 … b n, where b i = a i XOR a'i, for i = 1,…,n.

In the next step the reporter strand is extracted by biotechnique of melting away the tiles, smaller sequences, and purifying it, contains concatenation of input message, encryption key, ciphertext.

Before the final last step using a marker sequence the ciphertext can be excised and separated based on its length being half that of remaining sequence. In the last step ciphertext is stored in a compact form.

These increasing importances of information security and the protection of human privacy rights as Confidentiality lead me to develop new security solutions based on modern technologies: Bioinformatics and Biotechnology.

In this work we present a technical process for protecting data assets such as personal medical information using Bioinformatics and a DNA cryptography technique based on bioinformatics rather then biotechnologies in this bioinformatics technique a person's own blood mineral levels serve as a seed for selecting, transmitting, and recovering his sensitive personal data.

As we know that the management of security keys remains a challenge, we also developed a bioinformatic mechanism to generate encrypt-decrypt keys by taking into consideration specifics of the cryptography method and the individual's DNA genome analysis.

Our work was based on the complexity of developing, as a subset of JCE, an unconditionally secure DNAE System as part of our security provider, named DNAProvider, (Hodorogea, Ionas 2011).

A cryptographic system that an attacker is unable to penetrate even with access to infinite computing power is called *unconditionally secure*. The mathematics of such a system is based on information theory and probability theory. When an attacker is theoretically able to intrude, but it is computationally infeasible with available resources, the cryptographic system is said to be *conditionally secure*. The mathematics in such systems is based on computational complexity theory. To design a secure cryptographic system is a very challenging. A cryptographic system has one or more algorithms which implement a computational procedure by taking a variable input and generating a corresponding output. If an algorithm's behavior is completely determined by the input, it is called *deterministic*, and if its behavior is not determined completely by input and generates different output each time executed with the same input, it is *probabilistic*. A distributed algorithm in which two or more entities take part is defined as a protocol including a set of communicational and computational steps. Each communicational step requires data to be transferred from one side to the other and each computational step may occur only on one side of the protocol. The goal of every cryptographer is to reduce the probability of a successful attack against the security of an encryption system – to zero. Probability theory provides the answer for this goal. Our work is based on the complexity of developing an unconditionally-secure DNA Encryption System as part of DNA Provider.

Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing us to extend the JCE by implementing faster or more secure cryptographic algorithms. By the same means we shall provide our independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB).

4. Complexity of DNA encryption system as a subset of Java cryptography extension

Java Cryptography Extension (JCE) was developed as an extension package which includes implementation for cryptographic services.

The goal of the security provider interface is to allow a means whereby specific algorithm implementations can be substituted for the default provider, SUN JCE. JCE was developed as an extension package which includes implementation for cryptographic services. JCE offers a provider implementation plus API packages providing support for key agreement, encryption, decryption and secret key generation. Thus, JCE offers support for developing alternative cryptographic package providers, (Fig.1)

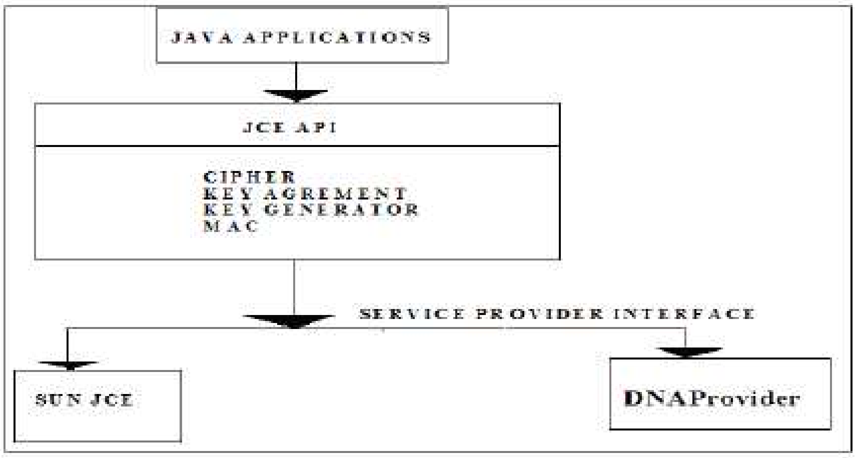


Fig. 1. Java Cryptography Extensions architectural model with unconditional secure DNA Encryption as part of our security provider (DNAProvider)

This support allows us to provide our independent implementation of DNAE System, based on the CDMB (Central Dogma of Molecular Biology).

The application code calls the appropriate JCE API classes. The JCE API classes invoke the classes in a provider that implements the interface classes, JCE SPI. The JCE SPI classes, in turn, invoke the requested functionality of the DNA Provider.

The security provider interface the means by which different security implementations may be plugged into the security package as message digests, encryption, digital signatures and keys, through JCE, JSSE and authentication through JAAS. Thus, JCE support allowed us to provide our independent implementation of DNA Cryptographic Keys Based on Evolutionary Models used for Security of Web-based Business Processes.

The classes necessary to handle secret keys come only with JCE. Keys and certificates are normally associated with some person or organization, and the way in which keys are stored, transmitted, and shared is an important topic in the security package.

When the Java Virtual Machine starts execution, it examines the user's properties to determine which security providers should be used. The user's properties are located in the file *java.security*, in which each provider is also enumerated. If users prefer to use DNAProvider as an additional security provider they can edit this file and add the DNA Provider. When the Security Class is asked to provide a particular engine and algorithm, it searches the listed providers for the first that can supply the desired operation,(Fig.2).

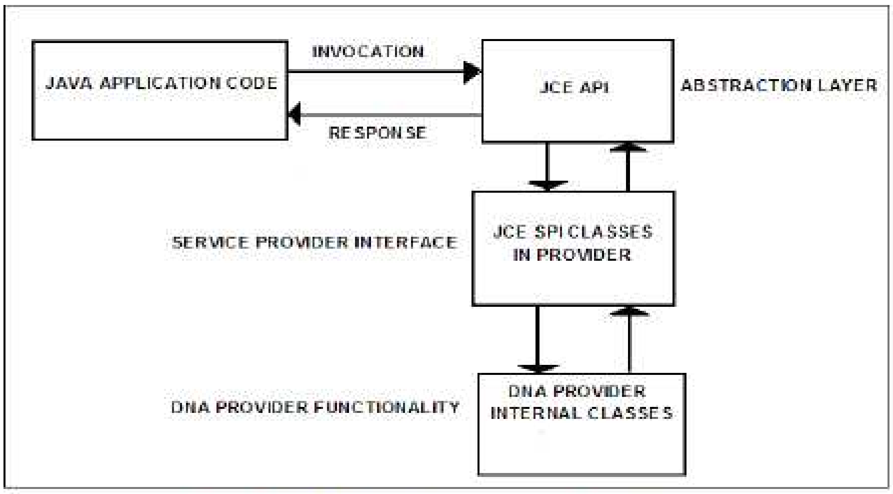

JAVA APPLICATION CODE
INVOCATION
RESPONSE
JCE API
ABSTRACTION LAYER
SERVICE PROVIDER INTERFACE
JCE SPI CLASSES IN PROVIDER
DNA PROVIDER FUNCTIONALITY
DNA PROVIDER INTERNAL CLASSES

Fig. 2. Invocation of DNAProvider for providing requested functionality

The security provider abstracts two ideas: engines and algorithms. An Engine Class defines an abstract cryptographic service, without its concrete implementation. The goal of the security provider interface is to allow an easy mechanism where the specific algorithms and their implementations can be easily changed or substituted. The architecture including all of this contains:

Engine classes, these classes come with the Java virtual machine as part of the core API.

Algorithm classes, at the basic level, there is a set of classes that implement particular algorithms for particular engines.

A default set of these classes is provided by the supplier of the Java platform. Other third-party organizations or individual can supply additional sets of algorithm classes. These classes may implement one or more algorithms for one or more engines.

Going to provide my own set of classes to perform security operations, I must extend the Provider class and register that class with the security infrastructure

Provider class is abstract, none of its methods are abstract, I need do is subclass the Provider class and provide an appropriate constructor.

The basic implementation of a DNAProvider security provider is:

```
public class DNAProvider extends Provider
{
public DNAProvider( )
{
super("DNAProvider", 1.0, "DNA Security Provider v1.0");
}
}
```

Here we define the skeleton of a DNAProvider that is going to provide certain facilities based on Central Dogma of Molecular Biology(CDMB).

Java Cryptographic Extension (JCE) offers support for developing cryptographic package providers, allowing us to extend the JCE by implementing faster or more secure cryptographic algorithms. By the same means we provide our independent implementation of a DNA Encryption (DNAE) system, based on the Central Dogma of Molecular Biology (CDMB). In this work we present a technical process for protecting data assets such as personal information using a DNA cryptography technique in which a person's own blood mineral levels serve as a seed for selecting, transmitting, and recovering his sensitive personal data.

Adleman began the new field of bio-molecular computing research. His idea was to use DNA biochemistry for solving problems that are impossible to solve by conventional computers, or that require an enormous number of computation steps. The DNAE technique simulates the CDMB steps: transcription, splicing, and translation process. The time complexity of an attack on a message of length n, is $O(2^n)$. DNA computing takes advantages of combinatorial properties of DNA for massively-parallel computation.

Introducing DNA cryptography into the common PKI scenario, it is possible to follow the pattern of PKI, while also exploiting the inherent massively-parallel computing properties of DNA bonding to perform the encryption and decryption of the public and private keys. The resulting encryption algorithm used in the transaction is much more complex than the one used by conventional encryption methods.

To put this into the common description of secure data transmission and reception with respect to DNA cryptography, let us say Stefani is the sender, and Otto, the receiver. Stefani provides Otto her public key which will comprise someone's unique blood analysis. The Public Key (PK) encryption technique splits the key into a public key for encryption and a secret key for decryption. As an example: Otto generates a pair of keys and publishes his public key, while only he knows his secret key. Thus, anyone can use Otto's public key to send him an encrypted message, but only Otto knows the secret key to decrypt it.

A secret DNA data strand contains three parts: a secret DNA data strand in the middle, and unique primer sequences on each side S1. Stefani uses the technique of deriving DNA private key.

Using an information conversion program, Stefani encodes the medical records in a DNA data strand flanked by unique primer sequences S1 and mixes it among other decoy DNA strands.

According to the CDMB, during the process of transcription, Stefani removes the introns from the data-encoded DNA, resulting in encryption key 1, E1 (starting and pattern codes of introns). Thus, E1 => C1 = E1(P), where P is plain-text and C is the cipher-text. Stefani translates the resulting spliced form of the data from which she derives Encryption key 2, E2 (codon-amino acid mapping). E2 => C = E2(C1) obtains the data-encoded protein after the translation process. Stefani sends Otto the keys E1 and E2 through a public channel.

Then she sends Otto the encoded protein form of the data through a public channel. Otto uses the key E2 to recover the mRNA form of the data from the protein form of the data. Decryption key, D1 = E2 => P1=D1(C). Otto recovers the DNA form of the data in the reverse order that Stefani encrypted it. Decryption key, D2 = E1 => P = D2(P1). Otto identifies the secret data-carrying DNA strand using the program that associates the nucleotide sequence based on someone's blood mineral analysis.

He obtains the unique primer sequences S1 that mark the beginning and end of the secret data DNA strand hidden among the decoy strands. In this last step, Otto uses the information conversion program and reads the medical record of the individual.

4.1 The DNA encryption protocol

Resent research considers the use of the Human genome in cryptography and the famous DNA one-time-pad encryption schemes utilizing the indexed of random key string was first developed by Ashish Gehani, Thomas H. LaBean and John H. Reif.

At the lowest level, a genome can be described as a long string of nucleotides. It could be compared to a very long text made of four letters (strings of DNA). All living organisms consist of cells and in each cell there is the same set of chromosomes. Chromosomes are strings of DNA and serve as a model for the whole organism made from genes, which are made from blocks of DNA. Complete set of genetic material (all chromosomes) is called genome. The assumption of evolutionary models is that biological systems have evolved from the same origin, constantly reusing some basic building blocks and through the cycles of mutation and selection that constitute evolution, new functions have been created by reusing pieces of already existing DNA machinery. If we consider this problem in terms of sequences, this means that two sequences responsible for similar functions may be different, depending on how long they have been diverging. Many of the problems in bioinformatics and more specifically in sequence alignment are said to be NP complete as the number of potential solutions rises exponentially with the number of sequences and their length and the solution cannot be found in polynomial time and space. A sequence alignment is the representation of two sequences in a way that reflects their relationship and if the alignment is designed to reflect phylogenetic relationships, the residues will be aligned when they originate from the same residue in the common ancestor. If a given sequence lacks one residue, a gap will be inserted in its place at the corresponding position, in an evolutionary model context, a null sign means that a residue was inserted in one of the sequences or deleted in the other while the sequences were diverging from their common ancestor.

As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution.

We developed a Unique Process System Pipeline Evolutionary Models of deriving DNA Cryptographic

Keys Sequences by deriving the DNA private/public keys from human genome analysis by computing the philogenetic tree relating and the branch length during evolution for chosen species. The molecular evolution model assigns probabilities to multiple-alignment columns in terms of the the philogenetic tree branches and is time dependent of frequency selections. Based on Kimura-Ohta theory Halpern and Bruno, have shown that mutation limit can be determined by substitution rates in terms of the mutation rates and equilibrium frequencies.

Models of DNA evolution were first proposed in 1969 by Jukes and Cantor, assuming equal transition rates and equal equilibrium frequencies for all bases.

In 1980 Kimura-Ohta introduced a model of DNA Evolution with two parameters: one for the transition and one for the transversion rate.

To estimate evolutionary distances in terms of the number of nucleotide substitutions and the evolutionary rates when the divergence times are known by comparing a pair of nucleotide sequences. There are two types of differences when homologous sites are occupied by different nucleotide bases and both are purines or both are pyrimidines. The difference is called Transition type when one of the two is a purine and the other is a pyrimidine then the difference is called transversion type.

Let P and Q be the fractions of nucleotide sites, showing between two sequences compared the transition and transversion type differences, then:

The Evolutionary Distance per Site is:

$$K = -(1/2)\ln\{(1-2P-Q)\} \tag{1}$$

The Evolutionary Rate per Year is then given by:

$$k = K/(2T) \tag{2}$$

T is the time since the divergence of the two sequences. If only the third codon positions are compared, then *the Synonymous Component of Evolutionary Base Substitutions per Site* is:

$$K'_S = -(1/2)\ln(1-2P-Q) \tag{3}$$

In biology, a substitution model describes the process from which a sequence of characters changes into another set of traits.

Each position in the sequence corresponds to a property of a species which can either be present or absent.

4.2 The technique of deriving DNA cryptographic keys based on evolutionary models

We developed and implemented a software tool for aligning the DNA Cryptographic Keys Sequences of the same gene from related chosen species with respect to Human DNA Sequences.

The alignment in the evolutionary system pipeline of DNA Cryptographic Keys Sequences was realized with trained ProbCons tool which is a pair-hidden Markov model-based on progressive alignment algorithm, that primarily differs from most typical approaches in its use of maximum expected accuracy.

As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution, (Ochman, 2003). Some of these will lead to differences in the amino acids of the encoded protein (non-synonymous changes). Because of the degeneracy of the genetic code leave the protein unchanged (synonymous, or silent changes). If Ka/Ks< 1 *Purifying (negative) selection*, most proteins are well adapted to carry out their function change would not lead to the creation of selective advantage. If *Ka/Ks >1 Diversifying (positive)*, selection has acted to change the protein and if *Ka/Ks= 1 Neutral evolution*, (Mustonen, Lässig, 2005). After aligning our extracted DNA Sequences with ProbCons tool, we derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

ProbCons is a tool for generating multiple alignments of protein sequences. It uses a combination of probabilistic modeling and consistency-based alignment techniques and has achieved the highest accuracies of all alignments methods. The basic for ProbCons algorithm is the computation of pairwise posterior probability matrices, P(xi ~ yi |x, y), which give the probability that one should match letters *xi* and *yi* when aligning two sequences *x* and *y*. ProbCons uses a simple probabilistic model that allows for efficient computation of this probabilities. Given a set of sequences ProbCons computes the posterior probability matrices for each pair of sequences and computes the expected accuracy of each alignment.

As Public-key algorithms are based on mathematical functions rather than on substitution and permutation and involves the use of two separate keys, in contrast to symmetric encryption, which uses only one key. When aligning the DNA sequences of the same gene from related species, there will usually be differences between the sequences because of evolution.

We developed a Unique Process System Pipeline Evolutionary Models of deriving DNA Cryptographic

Keys Sequences by deriving the DNA private/public keys from human genome analysis by computing the philogenetic tree relating and the branch length during evolution for chosen species. The molecular evolution model assigns probabilities to multiple-alignment columns in terms of the the philogenetic tree branches and is time dependent of frequency selections. Based on Kimura-Ohta theory Halpern and Bruno, have shown that mutation limit can be determined by substitution rates in terms of the mutation rates and equilibrium frequencies.

For every alignment column, we calculated the likelihood under two evolutionary models: a "foreground" and a "background" model.

The background model assumes a rate model (Felsenstein 1981), parameterized by the branch lengths of the phylogenetic tree:

w is a vector of nucleotide frequencies, with w_α the frequency of nucleotide α,

$r_{\alpha\beta}$ -the rate of substitution from base β to base α which is proportional to w_α, independent of β.

For every background evolution models we have a corresponding foreground model. The difference between the foreground model and background model is that the background model assumes that all positions undergo substitutions from base β to base α at the same rate $r_{\alpha\beta} \propto w_\alpha$.

The foreground model I assume that, at a given position i, the substitution rates $r^i_{\alpha\beta} \propto w^i_\alpha$ are altered due to specific selection preferences for certain bases at this position, parameterized by nucleotide frequencies w^i_α .

The parameters w^i_α , at each position are unknown, integrated out of the likelihood.

For each alignment column of the reference species, in intergenic regions and in genes, we calculate the ratio R, representing the likelihoods of foreground and background evolutionary models.

Halpern and Bruno in 1998 estimated the evolutionary distances from coding sequences taking into account protein-level selection to avoid relative underestimation of longer evolutionary distances.

The equilibrium frequencies determine the maximum dissimilarity expected for highly diverged but functionally and structurally conserved sequences and crucial for estimating long distances (Molina, Nimwegen 2008).

Halpern and Bruno introduced a codon-level model of coding sequence evolution in which position-specific amino acid equilibrium frequencies were free parameters. They demonstrated the importance and feasibility of modeling such behavior as the model produced linear distance estimated over a wide range of distances. Some alternative models underestimated long distances, relative to short distances.

If r is the rate of substitution from a base a to a base b at position i, μ is the rate of mutation from a to b and w is the equilibrium frequency of nucleotide i, at this position, (Halpern AL, Bruno WJ, 1998).

Following Golding and Felsenstein (1990), Halpern and Bruno (1998) who have shown that mutation limit of the standard Kimura-Ohta theory, one can uniquely determine substitution rates in terms of the mutation rates and the equilibrium frequencies w^i_α if $r^i_{\alpha\beta}$ is the rate of substitution from β to α at position i, $\mu_{\alpha\beta}$ the rate of mutation from β to α, and w^i_α the equilibrium frequency of α at this position, we have (Halpern and Bruno 1998).

We derive the private/public pair DNA cryptographic keys based on evolutionary models and based on mathematical functions.

We started with extracting from public available database all orthologus DNA coding sequences for all genes, from related species with respect to Human Genome sequences. A genome of a reference species in our case is Human Genom (hg18) and two more additional genomes are: Taurus Genome (bosTau3) and Dog Genome (canFam2). We extracted the DNA sequences for 29.000 genes which equals to 44103 pages in printable format. Using a trained parameter set for ProbCons tool we aligned all orthologus DNA coding sequences of our choosen species for all genes with respect to Human DNA coding sequences.

ProbCons achieved the highest accuracies of all multiple alignments methods as it uses probabilistic modeling and consistency-based alignment techniques.

We computed the philogenitic tree for our chosen species and the branch length during evolution, (Fig. 4) with respect to human genome (hg18).

A Software Application, reeds the tree, computes the pairwise alignment, computes the branches of the tree for our

chosen mammalian species. Public-key algorithms are based on mathematical functions, rather than on substitution and permutation and involve the use of two separate keys in contrast to symmetric encryption, (Fig. 3).

In Table 1, Second Column (C2) model represents the computed DNA Public Keys, with respect to Colum C1 and assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, (Table 1) and represents the public keys.

Given the transition probabilities and given a phylogenetic tree we calculated the ratio for an alignment column C3/C2, which is the product over transition probabilities for each branch of the tree we summed over all possible nucleotides for internal nodes calculated by recursive algorithm introduced by Felsenstein.

The first column C1 represents all possible three base sequences with respect to human species. Second Colum (C2) model, with respect to C1 assumes substitution rate model which is calculated by the branch lengths of the phylogenetic tree and a vector of nucleotide frequencies, and represents the public key. The third Colum (C3) assumes that at a given position, the substitution rates are altered during due to specific selection preferences for a certain base. The last Colum is the ratio C3/C2 and represents the private key, (Table 1).

Using the same model and desired length of bases from the first column we can derive the public/private keys used in Java KeyStore with respect to human or desired number of species. Resulting in new set of public/private *DNA Cryptographic Keys for our Java DNA KeyStore usage.*

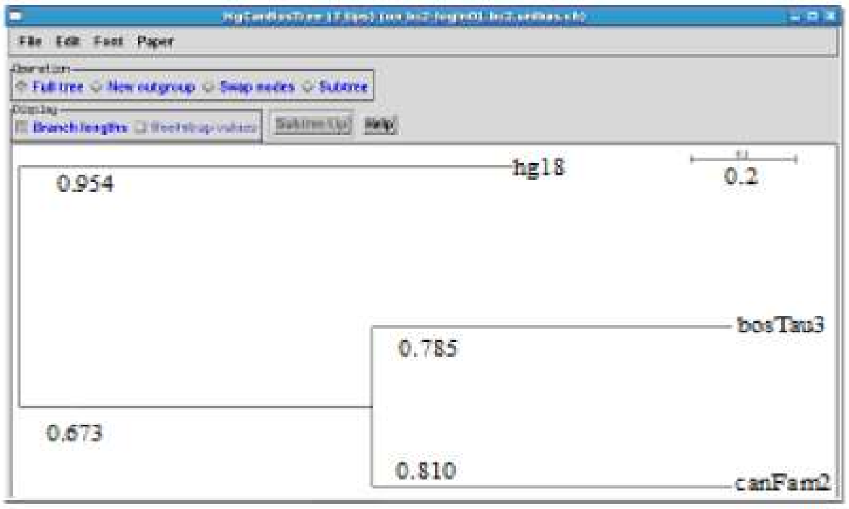
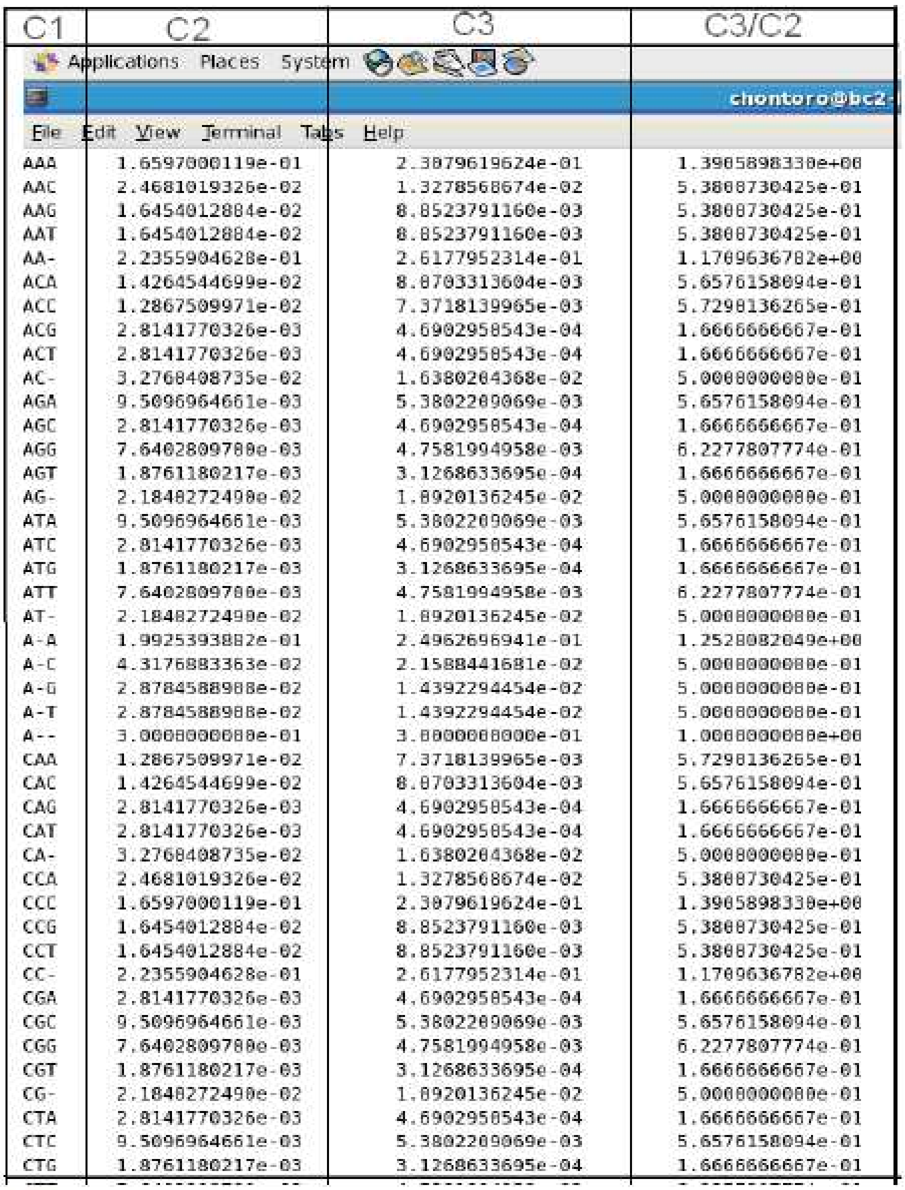


Fig. 3. Computed philogenetic tree and the branch length



C1	C2	C3	C3/C2
AAA	1.6597000119e-01	2.3079619624e-01	1.3905898330e+00
AAC	2.4681019326e-02	1.3278568674e-02	5.3800730425e-01
AAG	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
AAT	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
AA-	2.2355904628e-01	2.6177952314e-01	1.1709636782e+00
ACA	1.4264544699e-02	8.0703313604e-03	5.6576158094e-01
ACC	1.2867509971e-02	7.3718139965e-03	5.7298136265e-01
ACG	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
ACT	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
AC-	3.2768408735e-02	1.6380204368e-02	5.0000000000e-01
AGA	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
AGC	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
AGG	7.6402809780e-03	4.7581994958e-03	6.2277807774e-01
AGT	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
AG-	2.1840272490e-02	1.8920136245e-02	5.0000000000e-01
ATA	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
ATC	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
ATG	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
ATT	7.6402809780e-03	4.7581994958e-03	6.2277807774e-01
AT-	2.1840272490e-02	1.8920136245e-02	5.0000000000e-01
A-A	1.9925393882e-01	2.4962696941e-01	1.2528082049e+00
A-C	4.3176883363e-02	2.1588441681e-02	5.0000000000e-01
A-G	2.8784588908e-02	1.4392294454e-02	5.0000000000e-01
A-T	2.8784588908e-02	1.4392294454e-02	5.0000000000e-01
A--	3.0000000000e-01	3.0000000000e-01	1.0000000000e+00
CAA	1.2867509971e-02	7.3718139965e-03	5.7298136265e-01
CAC	1.4264544699e-02	8.0703313604e-03	5.6576158094e-01
CAG	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
CAT	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
CA-	3.2768408735e-02	1.6380204368e-02	5.0000000000e-01
CCA	2.4681019326e-02	1.3278568674e-02	5.3800730425e-01
CCC	1.6597000119e-01	2.3079619624e-01	1.3905898330e+00
CCG	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
CCT	1.6454012884e-02	8.8523791160e-03	5.3800730425e-01
CC-	2.2355904628e-01	2.6177952314e-01	1.1709636782e+00
CGA	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
CGC	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
CGG	7.6402809780e-03	4.7581994958e-03	6.2277807774e-01
CGT	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01
CG-	2.1840272490e-02	1.8920136245e-02	5.0000000000e-01
CTA	2.8141770326e-03	4.6902958543e-04	1.6666666667e-01
CTC	9.5096964661e-03	5.3802209069e-03	5.6576158094e-01
CTG	1.8761180217e-03	3.1268633695e-04	1.6666666667e-01

Table 1. Public/Private DNA Cryptographic Keys

5. Conclusion

Considering GRID computing security where the heterogeneous resources are shared and located in different places belonging to different administrative domains over a heterogeneous network, additional security requirements must be satisfied compare to classical network security. Communication between GRID entities must be secure and confidentiality must be ensured for sensitive data, from communication stage, to potential storage stage. Cryptographic algorithms for confidentiality play a major importance role in nowadays information security.

Our work described in this chapter was based on the complexity of developing the cryptographic package provider, named DNAProvider as Java Cryptographic Extension (JCE), where we derive the DNA Cryptographic Keys Based on Evolutionary Models for Security of Software Applications, extending the JCE by implementing faster and more secure DNA Encryption (DNAE) system based on the Central Dogma of Molecular Biology (CDMB). Sun Microsystems certified and signed our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm. We got the Code Signing Certificate from Sun Microsystems for our DNAProvider as Java Cryptographic Extension (JCE) with DNA cryptographic algorithm which is available for 5 years, until with the reference #679, when renewing it in 2013.

In our future research work we intend to integrate our developed system pipeline of deriving DNA Cryptographic Keys Based on Evolutionary Models implemented and tested at University of Basel, Switzerland, in our DNAProvider as Java Cryptographic Extension (JCE) with DNA Encryption (DNAE) system for use in security of our developed Web-based Business Processes Software Applications. We aim to use DNA Provider with unconditional secure DNAE system to ensure security of today's web-based business processes. as e-commerce and Internet banking. (Hodorogea, Ionas, 2011).

6. Acknowledgment

This research work is supported by the Company INNOVA BIOTECH, Cluj-Napoca, Romania.

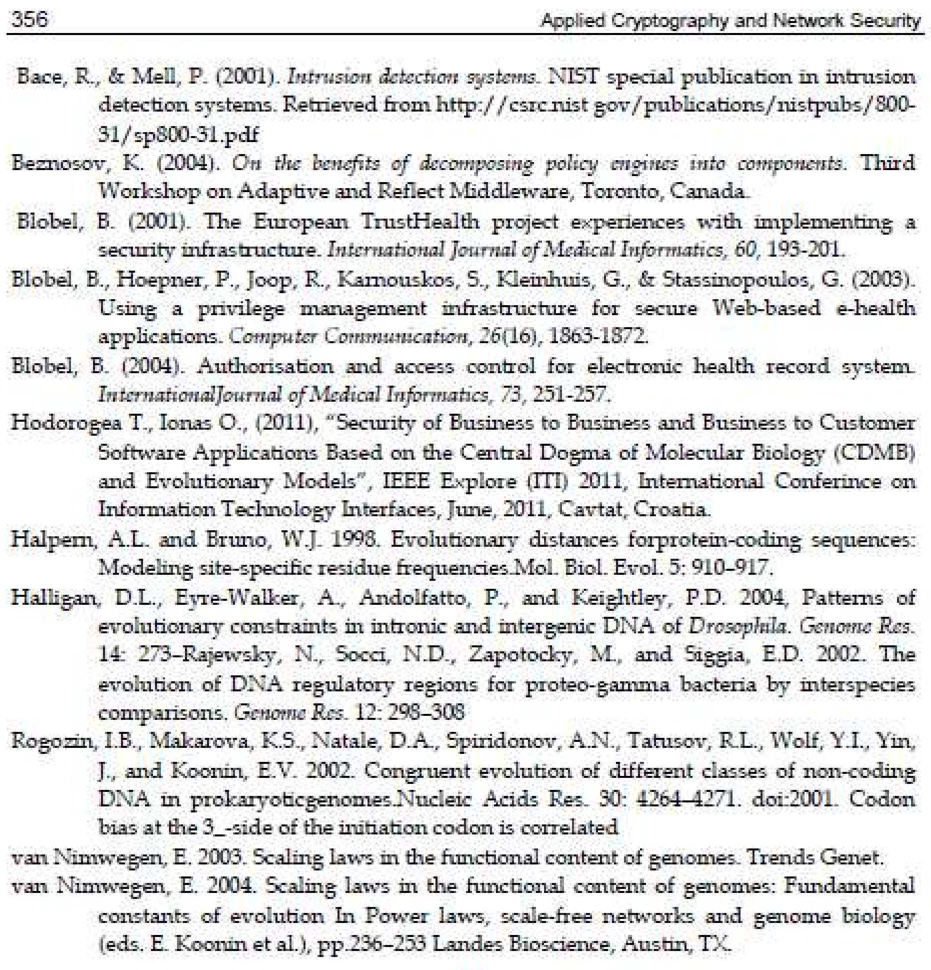
7. References

Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou,Y., & Rowe, K. (2003). Log correlation for intrusion detection: A proof of concept. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC 2003)*. Los Alamitos, CA: IEEE Computer Society Press.

Almgren, M., & Jonsson, E. (2004). Using active learning in intrusion detection. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*. Los Alamitos, CA: IEEE Computer Society Press.

Anderson, J. P. (1980). *Computer security threat monitoring and surveillance* (Tech.l Rep.). FortWashington, PA: James P. Anderson.

Alberts C., Audrey D., "Managing Information Security Risks: The OCTAVESM Approach ", Addison Wesley Professional, July 09, 2002.



Bace, R., & Mell, P. (2001). *Intrusion detection systems.* NIST special publication in intrusion detection systems. Retrieved from http://csrc.nist gov/publications/nistpubs/800-31/sp800-31.pdf

Beznosov, K. (2004). *On the benefits of decomposing policy engines into components.* Third Workshop on Adaptive and Reflect Middleware, Toronto, Canada.

Blobel, B. (2001). The European TrustHealth project experiences with implementing a security infrastructure. *International Journal of Medical Informatics, 60,* 193-201.

Blobel, B., Hoepner, P., Joop, R., Karnouskos, S., Kleinhuis, G., & Stassinopoulos, G. (2003). Using a privilege management infrastructure for secure Web-based e-health applications. *Computer Communication, 26*(16), 1863-1872.

Blobel, B. (2004). Authorisation and access control for electronic health record system. *InternationalJournal of Medical Informatics, 73,* 251-257.

Hodorogea T., Ionas O., (2011), "Security of Business to Business and Business to Customer Software Applications Based on the Central Dogma of Molecular Biology (CDMB) and Evolutionary Models", IEEE Explore (ITI) 2011, International Conferince on Information Technology Interfaces, June, 2011, Cavtat, Croatia.

Halpern, A.L. and Bruno, W.J. 1998. Evolutionary distances forprotein-coding sequences: Modeling site-specific residue frequencies.Mol. Biol. Evol. 5: 910–917.

Halligan, D.L., Eyre-Walker, A., Andolfatto, P., and Keightley, P.D. 2004, Patterns of evolutionary constraints in intronic and intergenic DNA of *Drosophila. Genome Res.* 14: 273–Rajewsky, N., Socci, N.D., Zapotocky, M., and Siggia, E.D. 2002. The evolution of DNA regulatory regions for proteo-gamma bacteria by interspecies comparisons. *Genome Res.* 12: 298–308

Rogozin, I.B., Makarova, K.S., Natale, D.A., Spiridonov, A.N., Tatusov, R.L., Wolf, Y.I., Yin, J., and Koonin, E.V. 2002. Congruent evolution of different classes of non-coding DNA in prokaryoticgenomes.Nucleic Acids Res. 30: 4264–4271. doi:2001. Codon bias at the 3_-side of the initiation codon is correlated

van Nimwegen, E. 2003. Scaling laws in the functional content of genomes. Trends Genet.

van Nimwegen, E. 2004. Scaling laws in the functional content of genomes: Fundamental constants of evolution In Power laws, scale-free networks and genome biology (eds. E. Koonin et al.), pp.236–253 Landes Bioscience, Austin, TX.