Security in SPGBP – Simulation of Protein Generation Bioinformatic Project

Cristian TOMA, Elena PURCARU

Cybernetics and Economic Informatics Faculty,
Academy of Economic Studies Bucharest
Pta. Romana 6, Bucharest, ROMANIA
"CarolDavila" University of Medicine and Pharmacy
Eroii Sanitari Boulevard 8, Bucharest, ROMANIA
cristian.toma@ie.ase.ro, elena.purcaru@gmail.com

Abstract: Bioinformatics has known a rapid growth in the last decade, along with the development of the genomic projects worldwide. The Human Genome Project alone offers the sequences for 70 000 – 100 000 genes, as terra bytes of information. The major challenge nowadays is to process this huge amount of data. This paper comes to suggest a solution for the retrieval and processing of biological data. The main objective of this section is to point a direction in the development of tools that enable manipulation and efficient access to different types of biological data. Also, the major challenge is to use security in the proposed distributed architecture and to import concepts from bioinformatics into cryptography and IT&C security.

Keywords: bioinformatics, security, BioJava, BioSQL, communication protocol, distributed processing.

1. Introduction

Bioinformatics represents the application on information technology in the field of biology, molecular biology in particular. The objective of this field is to ease the understanding of biological processes with special interest in sequence alignment, protein structure and alignment, protein interactions, gene expression, genome assembly, phylogenetic studies etc. In order to understand the bioinformatics approach, some fundamental notions of molecular biology are described in section 3.

Bioinformatics works with biological data. This type of data comes in large volume, which leads to limited possibilities of processing it. Taking as well into consideration the need for intense processing in research purposes, we have identified some of the major issues of biological data manipulation.

This is a post conference paper. Parts of this paper have been published in the Proceedings of the SECITC 2009 Conference (printed version).

2. Distributed Architecture of SPGBP

The Distributed Architecture presented in Figure 1 comes to offer a solution to the mentioned problems. This architecture serves to process biological data such as DNA (Deoxyribonucleic acid) or RNA (Ribonucleic acid). The result of this processing comes as protein sequence. As in real life biological processes, a DNA sequence is first transcribed into RNA sequence which is then translated into a protein (see section 3).

The SPGBP Distributed Architecture implements two major activities:

- Communication with a BioSQL Database to insert/retrieve biological sequences;
- Distributed processing of DNA/RNA sequences in order to transform them into proteins.

All devices participating to these activities and the relations established between them form the Distributed Platform for SPGBP.

ISSN 2067 – 4074

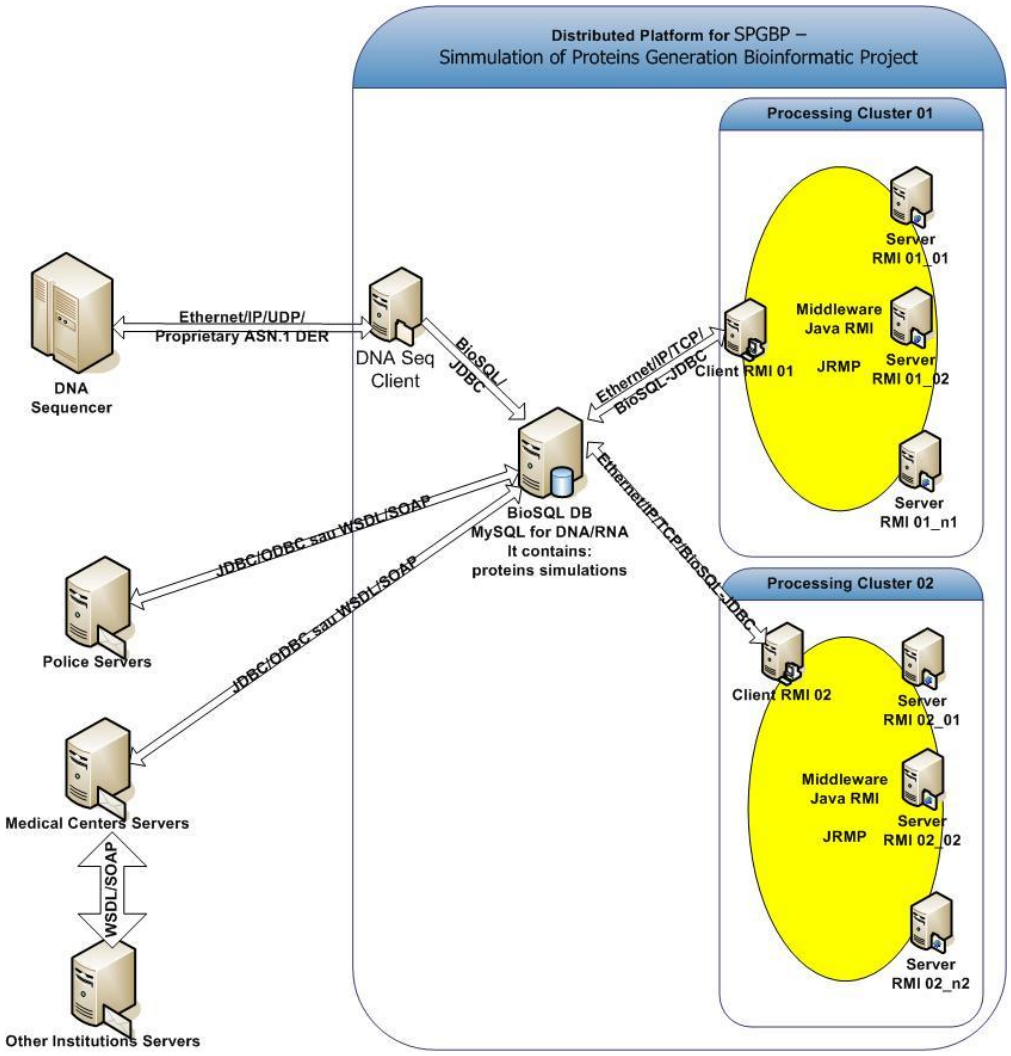

Fig. 1 Architecture of SPGBP

In order to deal with the large volume of the data processed, as well as with an intense data manipulation, this architecture proposes a solution of distributed processing for treating biological data, in this case to transform DNA into protein. Devices that take part into the distributed processing form a Processing Cluster. Each cluster is made up by:

- One Client computer. The Client retrieves a specific DNA or RNA sequence from a database and offers it to the Servers to be handled.
- Several Servers for distributed processing. Server computers cooperate in order to transform the DNA/RNA sequence into a protein and send the result to the Client.

The distributed processing cluster is implemented using Java Remote Method Invocation (JRMI) mechanism. The distributed processing is not using web services because of the network overload and processing speed.

All biological data is stored in a BioSQL Database (developed with MySQL) as DNA, RNA or protein sequences. BioSQL is a standard relational database model dedicated to store biological data [2]. The BioSQL Database stays at the center of the distributed platform that implements the SPGBP Distributed Architecture. The DNA or RNA sequences residing in the database are inserted here after being directly obtained from Sequencers – devices that identify the DNA or RNA in a biological sample (ex.

blood). A Client of a Distributed Processing Cluster retrieves the sequence to be processed from the same database. The result proteins are as well inserted into the database. Therefore, the components of the Distributed Architecture are constantly communicating with the BioSQL Database in order to insert/retrieve sequences of biological data.
The database may as well function as an access point from external computers to biological data. It is unlikely for a user to receive direct access to the resources of a Sequencer. This constraint is required by the sensitivity of the device and its limited resources in terms of network accesses. But a Client may easily have access to a database containing the biological data needed – using web services or for internal use JDBC/ODBC. Such Clients may come as Police Departments, Medical or Research Centers, individuals that had their DNA sequenced or any other institution that has access to the BioSQL Database.

3. SPGBP Communication Protocols and Parallel Processing

The Distributed Platform for SPGBP presented in Figure 1 brings together multiple components that must communicate with each other in order to function successfully in order to:

- Retrieve DNA/RNA sequences from an automated Sequencer;
- Place sequences in a BioDatabase;
- Retrieve sequences for processing, when desired;
- Distributed processing of the DNA/RNA sequences retrieved, into proteins;
- Inserting the protein results into the BioDatabase.

As the IP/TCP communication protocol is the most used protocol in nowadays networks, it is also the one standing at the base of the Distributed Platform. However, as the Platform is made up by heterogeneous devices, communication between them presents several particularities: UDP protocol for Sequencer communications (because the Sequencer is an embedded system and mostly the software implements only UDP over IP and not TCP), JRMP protocol for distributed processing, JDBC driver for communication with the database.

3.1.Sequencer Communications

A Sequencer is a machine used to automate the process of DNA sequencing. In the analysis of biological samples - blood, viruses etc. – the Sequencer will get as result a DNA sequence (or RNA, for viruses and bacteria). The result is stored on a physical medium as a file in FASTA format. In bioinformatics, FASTA format is a text format for the representation of fragments of DNA/RNA, as sequences of nucleotides, or proteins, as sequences of amino acids. Nucleotides or amino acids are represented with single letters. FASTA format is easily manipulated by text processing tools or by using programming languages.
However, a Sequencer is not a device specialized in network communication. Having limited resources in terms of networking, it will more likely communicate through the simpler IP/UDP protocol. We have developed a specific UDP protocol for retrieving DNA/RNA sequences from a Sequencer that has them stored in FASTA format.
After the sequencing process, the result, in FASTA file format is saved on Sequencer's EEPROM. In bioinformatics, FASTA format is a text which represented sequences of nucleotides, so fragments of DNA / RNA, or sequences of amino acids, so proteins. Nucleotides or amino acids are represented by single letters (A,C,G,T for nucleotides - see section 3). The format allows sequences to be preceded by a name and comments. A sequence in FASTA file format starts with a description followed by lines of sequence of amino acids or nucleotides. Description line contains the symbol ">" on the first column. The word which follows this symbol is the identifier

ISSN 2067 – 4074

sequence, and the rest line up is the sequence description. Both the identifier and the sequence descriptions are optional. The sequence ends with a new symbol ">", which marks the beginning of other sequences. FASTA file format is easily manipulated by word processing tools or by using programming languages.

The Sequencer offers the result of it's sequencing activity to different Clients. Therefore it is a Server device. The Server allows multiple customers to simultaneously use its services as it operates through execution threads. A Client may request a sequence from the Server based on a sequence identifier. The request is built in ASN.1 DER language, using an Object Identifier (OID) for identifying Server's services. One specific OID requests a DNA sequence while a different OID requests a RNA sequence. Based on this request, the Server sends its response to the Client, as the sequence required or an informative message if the sequence was not found.

3.1.1. Client 2 Server/Sequencer Request

A Client issues a request for service to the Server. In this case the Client asks the Server for a DNA or RNA sequence. This request is made through an OID defined separately for DNA or RNA. Thus, for DNA sequence is made by OID 1.4.10, and a RNA request is made by OID 1.4.20. These OIDs are defined locally, not officially registered. To build the request message is also the sequence identifier needed. It is transmitted as a string of bytes corresponding to ASCII characters associated. The message is built byte by byte, using the standard ASN.1 DER, as TLV – tag-length-value as follows:

T	L	V						
		T	L	V		T	L	V
30	..	06	02	2c	10/20	04	..	..

The first byte 0x30 indicates that the message is a collection of one or more types of bit length specified by the second ASN.1 SEQUENCE element.

The first element of the collection is of type OID (0x06), with length 0x02. The first byte value is 0x2C (OID (1.4) in ASN.1 DER). The second byte is 0x0A (10 = 0x0A) if requested a DNA sequence, or 0x14 (0x14 = 20) for RNA sequence.

The last element of collection is a range of bytes (type 0x04). It is the only element with variable-length in the message. Length is determined simply as the length of the sequence identifier. The identifier sequence is provided by the user through a user interface. The ASCII codes corresponding to this identifier will form the final "value" segment of the request.

The formed message is transmitted as a UDP packet to the Server, at the IP address and port specified by the user. The message will be verified by the Server (in this scenario the Sequencer device acts as a Server).

3.1.2. Server/Sequencer 2 Client Response

Following the request intiated by the Client, the Server (the Sequencer) sends its response. This response comes in the form of one or more UDP packages.

If the Server finds the requested sequence, the sequence is transmitted in 250 bytes UDP packets until complete transmission. The Server then sends a last package explicitly marking the end of sequence through the word „closed".

If the sequence is not found, the Server sends a message informing the Client, as a single UDP packet with the message „File not found".

The Client receives all packages from the Server and reassembles the sequence received as more than one package if case. The first packet received by the Client with size less than 250 bytes is treated in particular. This package may contain the last part of the sequence transmitted by the Server or, in case the last segment of the sequence had the exact size of 250 bytes, it contains the word "closed". This differential treatment

of the package is justified by the fact that only the message that does not contain the word "closed" will be added to the ongoing reassembling sequence.

3.1.3. Network Traffic Analysis between Client and Server/Sequencer

The communication protocol is basically a set of rules that enable communication between parties to take place. The set of rules created includes all actions between the Server machine and the Client machine to make possible the Client application to receive DNA/RNA sequence from the Server/Sequencer machine. The communication is realized through UDP / IP and includes the following actions, in quasi-chronological order:

- The Server/Sequencer application listens the communication channel on a UDP port (778);
- A Client application sends a request to the Server in order to obtain the DNA/RNA sequence;
- The Server receives the request of the Client and parses it accordingly with ASN.1 TLV from communication protocol;
- The Server application examines the request and, as result of this analysis, it responds to the Client application;
- The Client application receives the reply / replies from the Server application;
- The Server application explicitly marks the end of the transmission;
- The Client receives the end of the transmission and it leaves the communication channel.

The Client request comes in a specially form recognized by the Server (OID ASN.1 presented above). The messages are sent as strings of bytes. Therefore, a significant part of the communication protocol deals with the construction of the correct messages.

Using specialized network analysis software (Wireshark tool) to intercept packets on the network, we can track specific bytes exchanged in the communication between the Server/Sequencer and the Client. The network traffic analysis carried within the Wireshark v1.0.6 shows the Client request to the Server/Sequencer and the Server/Sequencer response – figure 20.7a.

In the test scenario, the communication is realized between two machines-computers with IP addresses 10.4.4.139 and 10.4.4.147. The machine/computer with IP 10.4.4.139 communicates (listen) on UDP port 778 and it is the Server/Sequencer. The machine/computer with IP 10.4.4.147 communicates on UDP port 1428 and is the Client, being the one who launches the Client request. The Client machine/computer UDP port is assigned to Informatik-lm (Informatik License Manager). It can vary at a customer from one run to another. The entire communication flow is presented in Figure 2.

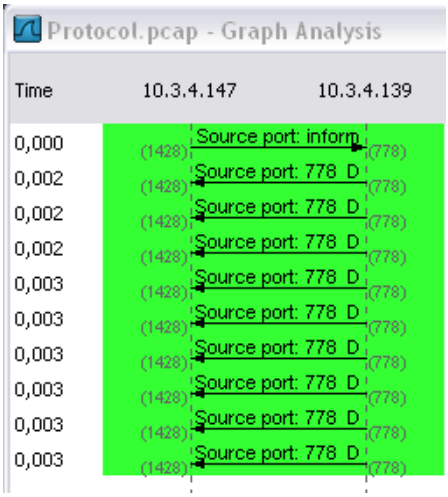

Fig. 2 Communication flow between Client (left) and Sequencer (right)

Only the first packet has the source machine with the IP 10.4.4.147 – the Client. It is the request made by the Client. The other 9 packages are departing from the Server/Sequencer machine/computer with IP 10.4.4.139. Last package will contain the message that marks the end of the transmission, the ASCII codes corresponding to the word "closed".

The entire DNA/RNA sequence is transffered in 9 UDP packets and the entire network communication has 10

ISSN 2067 – 4074

UDP packets within 0,003 seconds as transffer time.
Both Server and Client activity have been implemented in Java language. The Server has been developed using two classes: a main class and class for working with execution threads. A single main class has been sufficient to implement a Client. As UDP protocol does not provide services for dividing messages into packets and their reassembly at the destination, this operation is ensured by the developed protocol, in each main class.

3.2. Communications with the BioDatabase

Once a Client receives a positive response from the Server as a DNA/RNA sequence, the sequence is added into a database (BioSQL). Any communication with the database is realised using a JDBC (Java Database Connectivity) Driver. JDBC is a standard API for an independent connexion between a Java application and a SQL Database. JDBC interface allows the transmission of requests made in SQL language (Structured Query Language) to the program managing the database (Database Managing System – DBMS). Results are also received through JDBC. Depending on DBMS administering the database, there are different JDBC drivers for different DBMS.

3.3. Parallel Processing

Another particular communication protocol used in the Distributed Platform is the JRMP (Java Remote Method Protocol) protocol for distributed processing. JRMP has at base the IP/TCP protocol, specialised for the RMI mechanism. In RMI Architecture, a „Client" object may call a method from a „Server" remote object. The Client locates the remote objects by interrogating the rmiregistry name service. The communication between Server and Client is realized by two components of the RMI system architecture: Stub and Skeleton.

The RMI Client extracts a DNA/RNA sequence from the database. It gives the sequence for processing to a number of RMI Servers that cooperate to convert it into protein.
The RMI Server is implemented using three classes that describe respectively:

- the remote interface – containing prototypes for the methods used to translate DNA/RNA sequences into proteins; this class must reside on both Client and Server machine;
- interface implementation – for each method with prototype described in the remote interface;
- the main Server – where the Server is instantiated.

The RMI Client must also contain the Stub class. This class is dynamically generated on the Server machine at compilation (with rmic.exe tool) and must be explicitly copied on the Client.
The Client is responsible for communicating with the database for sequence retrieval and result insertion. It is the one that monitors/supervises the distributed processing, with the task of sequence disassembly into fragments that will be sent for processing to the remote Servers. The Client must as well assemble the final result.
While RMI provides the mechanism to call the remote methods, how they are called is a task for the Client to decide. It is preferably for the amount of work to be equitably divided between the RMI Servers. Therefore the DNA or RNA sequence must be fragmented in comparable pieces for each Server machine. Moreover, all fragments must have multiple of three symbols number. No symbol must be lost, except for the last one or two symbols from the end of the sequence. These rules are essential for a correct transformation, according to the genetic code (see section 3).
Biological data manipulation and the communication with the BioSQL Database were facilitated by a series of tools developed within the BioJava Project. BioJava is an open-source project dedicated to biological data processing while keeping it in full accordance with the genetic code. It provides a range of analytical and

statistical routines, converters for several well known file formats and allows the manipulation of sequences and three-dimensional structures. The project stands to facilitate the rapid development of bioinformatics applications in the Java language.
BioJava libraries can be downloaded from the official Biojava website [1]. Many of the tools used in the development of this solution are included in org.biojava.bio.seq package from *BioJava* library. This package provides classes and interfaces for managing biosequences (DNA, RNA, proteins). In addition, the package provides tools for working with collections of biological data (databases) through the org.biojava.bio.seq.db component.

4. Molecular Biology Concepts in SPGBP

A molecule is the smallest part from a substance that has the complete characteristics of that substance as a whole. In the living world, the cell is the structural and functional unit of all known living organisms. It is the smallest unit of an organism that is classified as living.

A cell is formed by an outer layer – the membrane – that separates the outer environment from the inner content of the cell. This content is formed by cytoplasm, with cellular organelles, and nucleus, the keeper of genetic material. A cell is approximately 10 μm in diameter. A drop of human blood for example contains up to 5 million cells.

4.1. Genetic material

"The genetic material is a kind of 'operating system' for an organism and contains the blueprints for all body components." [3]
Human genetic material is formed by approximately 25,000 blueprints called genes. The genetic material contains the information for body features such as height, shape of the nose and color of the skin. At least partially, the genes also define our behavior and our abilities – for instance, whether we learn easily or how we handle stressful situations.

Genetic material consists in humans of 23 pairs of chromosomes, of which one pair of sex chromosomes. Each pair of chromosomes is formed by one motherly and one fatherly chromosome. A chromosome is formed by a single piece of DNA. Human DNA is identical in proportion of 99.9%. The rest of 0.1% makes the oneness of every individual.

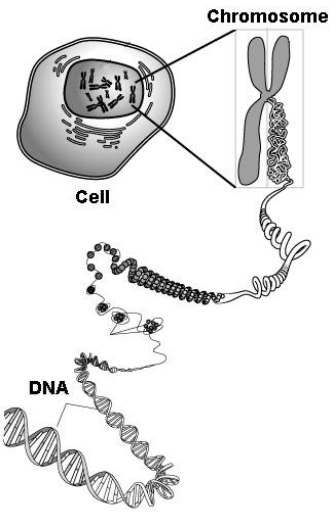

Fig. 3 Genetic material (*www.brynmawr.edu*)

4.2. DNA

Deoxyribonucleic acid (DNA) is a macromolecule that consists of two long chains. Each chain contains simple units called nucleotides. There are only four nucleotides that form all DNA macromolecules. These are: thymine (T), cytosine(C) adenine (A) and guanine(G).
Each DNA chain is formed by the sequence of these four nucleotides, in random order. But the DNA molecule consists of two such chains. How can this be? Each nucleotide has the ability to bond with a complementary nucleotide through links called hydrogen bonds: adenine (A) with thymine (T) (through two hydrogen bonds) and

ISSN 2067 – 4074

guanine (G) with cytosine (C) (through three hydrogen bonds). The two DNA chains are not completely different, but complementary. There are no less than 3,200,000,000 complementary nucleotide pairs in one human cell DNA.
The DNA double chain then spins to the right (due to the orientation of nitrogenous bases on the sugar) forming the famous DNA double helix.

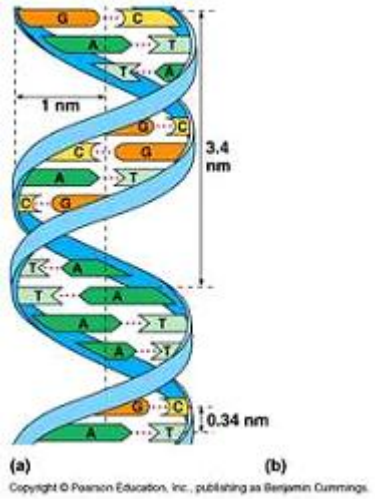

Fig. 4 DNA double helix

4.3. Proteins

Although there are only four types of nucleotides, their sequence in the DNA chain gives DNA the amazing ability to encode information. DNA information storage refers to protein synthesis.
Proteins are fundamental components of all living cells and include many substances, such as enzymes, hormones and antibodies. All these substances are necessary for the proper functioning of an organism. Proteins are organic compounds made of amino acids. Amino acids form short chains called peptides or longer chains called either polypeptides or proteins. Peptides must be grouped to form larger proteins.
Genetic code controls how these amino acids are joined in order to form certain proteins. There are 20 amino acids encoded by the genetic code, called standard amino acids.
An amino acid is encoded in the genetic code by a sequence of three nucleotides, called codon. The codon is the unit of function for all genetic material. In other words, genetic information is read only in nucleotides triplets. Therefore, there are 4x4x4 = 64 possible combination, more than enough to encode the 20 standard amino acids.

4.4. Transcription and Translation

All living things depend on genes. Genes contain genetic information. A gene is a DNA segment that contains the genetic information required to synthesize a polypeptide chain with specific structure and function. In other words, a gene is a DNA segment that, taken as a whole, specifies a trait.
An organism may have hundreds of thousands of genes. Humans for instance are estimated to have near 25 000 genes. Genes are responsible for all biological traits. Some of these traits are immediately visible, such as eye color or number of limbs. Some of them are not, such as blood type or increased risk for specific diseases, or the thousands of basic biochemical processes that sustain life.
Genes, the keepers of the information required to produce proteins, are located on chromosomes, inside the nucleus. But protein synthesis cannot take place in the small nucleus as DNA is a large molecule that cannot travel through the nuclear membrane. In order for genetic information to reach the ribosome, one gene's information is copied into a smaller molecule: messenger RNA (mRNA). RNA (Ribonucleic Acid) is very similar to DNA, with uracil(U) nucleotide replacing timine(T).
Protein synthesis takes place in the cytoplasm, where the factory and the ingredients (amino acids) are found. The cell's protein factory is called ribosome. Ribosome is formed from two parts: a small unit and a large unit. The small unit is responsible for attaching the mRNA to its structure. Keeping the analogy with factory in mind we can say that mRNA is the recipe for protein synthesis. The large unit is responsible for building up the protein. A cell may

contain up to 20 000 ribosomes, found free or membrane-attached.
The process of turning DNA into mRNA is called transcription, while transforming the mRNA into proteins is known as translation.

5. Security Issues and Research Directions

The security issues involved in this type of project give research direction to:
- IT&C Security/Cryptography 2 Bioinformatics project (points A, B, C in this conclusions section).
- Bioinformatics 2 cryptographic algorithms and to IT&C Security.

5.1. IT&C Security/Cryptography 2 Bioinformatics project

A. Because the distributed nature of SPGBP and Sequencer/database centric solution, the security is focused on the network communication and each node in the network, especially at the MySQL database. The network communication between the Sequencer/Server and the Client (UDP connection with ASN.1 DER containing OIDs) is not secured because both components are inside the same company room with a single Ethernet cable. Due to WiFi connection between Sequencer and Client, in the future we will use encryption and digital signatures for this connection too.

B. Security of the database is on-demand on multiple levels:
1. Security of the network communication between the Sequencer Client and database Server –MySQL:
- Communication between Client and database Server database may be based on request / response web services (SOAP over HTTP) that are secured by SSL (Secure Socket Layer 3.0) – and not JDBC over TCP as it is now.
- Client certificates and Server are 'self-signed' version X509 v3 and is created by OpenSSL or Java Keystore Tool – keytool.

2. Security at the database engine:
- Security of the database is provided by the database/operating system administrators who give the access rights on both the database (users management) and the rights on the files of the operating system (minimal solution is based on MySQL database system under Linux)
- For the database tables the data is encrypted for each record and in different columns through insert procedure (SQL INSERT INTO). The decryption is realized through selection (SQL SELECT). Both, encryption and decryption are realized by using the symmetrical key algorithm with 128-bit AES-Rijndael in ECB Electronic Code - Book. Encryption / decryption are realized through stored procedures (AES_ENCRYPT / AES_DECRYPT) which are available and implemented by the MySQL engine databases.

Sample of inserting and retrieving encrypted / decrypted data [5]:

```
INSERT INTO tbltest(u, p) VALUES ('user 1',AES_ENCRYPT('john 1789012345','samplekey012345'));
INSERT INTO tbltest(u, p) VALUES ('user 2',AES_ENCRYPT('john 2789012345','samplekey012345'));
INSERT INTO tbltest(u, p) VALUES ('user 2',AES_ENCRYPT('john 3789012345','samplekey012345'));

SELECT u, AES_DECRYPT(p,'samplekey012345') FROM tbltest;

*p field in the database table has 16 bytes and it is BINARY field
```

3. Exported data security
- Data files are exported in text files (FASTA format). The Client can be configured in order to retrieve/send data via a SMTP/POP3/IMAP4 secure connection (over SSL) or can be configured to use the PGP tool [6].

C. The communication between the nodes of distributed computing is not secure for the moment but JRMP can be

ISSN 2067 – 4074

used over SSL. However, each node in the distributed computing platform is secure by implementing the rules from the Java Policy File. A Java application acts on the privileges required by the interpreter „java.exe" which is run by the operating system. The security policy is established by the Java Security Manager which performs actions accordingly with a security policies file. This security policies file, with name „policy.all", normally resides in the folder „%JAVA_HOME%/jre/lib/security".

It is important for these security policies to be altered in order to permit the operations implemented in the distributed processing. A security policy file which defines a very non secure environment is for instance:

```
grant {
   permission java.security.AllPermission;
};
```

Restrictions can of course be attached for socket and port connections, for the access in the file system of the operating system and even for operations within the network.

5.2. Bioinformatics 2 cryptographic algorithms and to IT&C Security

The second research approach is to use concepts from bioinformatics into cryptographic algorithms, digital signatures and secure communication protocols. This area is in study now and it does not involve processing at biological level but at simulation one.

Because the bioinformatics studies and simulations are used in fields such as:

- Molecular medicine
- Personalised medicine
- Preventative medicine
- Gene therapy
- Drug/Medicine development
- Microbial genome applications
- Waste cleanup
- Climate change Studies
- Alternative energy sources
- Biotechnology and Nanotechnologies
- Antibiotic resistance
- Forensic analysis of microbes
- Bio-weapon defence and creation
- Evolutionary studies
- Crop improvement
- Insect resistance
- Improve nutritional quality
- Development of Drought resistance varieties
- Veterinary Science

a research direction of Bioinformatics IT&C Security will be added.

6. Conclusions

This paper uses only some of the tools offered by BioJava Libraries. Its purpose is merely to indicate a direction for biological sequences processing. BioJava package offers more complex features beyond those for DNA/RNA translation into protein. Any of the BioJava tools can be implemented in the proposed distributed architecture, thus combining the advantages of distributed processing with those of the BioJava project. By implementing work with annotated sequences or with features, become possible genetic analysis based on gene expression or simulations with transcription factors. On such simulations are based techniques for disease diagnosis and treatment and research activities in pathology and immunology.

References

[1] BioJava Project site, http://www.biojava.org

[2] BioSQL Project site, http://www.biosql.org

[3] National Genome Research Network, http://www.ngfn.de/englisch/glossar864.htm

[4] Cristian TOMA, Elena PURCARU – "Security Issues Applied in SPGBP – Simmulation of Proteins Generation

Bioinformatic Project" – based on Elena PURCARU bachelor thesis synthesis published in "Informatics Security Handbook - 2nd Edition", ISBN 978-606-505-246-8, pp. 615-646

[5] MySQL project site: http://dev.mysql.com