Project number: 317871
Project acronym: BIOBANKCLOUD

**Project title:** Scalable, Secure Storage of Biobank Data
**Project website:** http://www.biobankcloud.eu
**Project coordinator:** Jim Dowling (KTH)
**Coordinator e-mail:** jdowling@kth.se

**WORK PACKAGE 4:**
**Inter-connection of Biobanks and Clouds**

WP leader: Alysson Bessani
WP leader organization: FFCUL
WP leader e-mail: bessani@di.fc.ul.pt

**PROJECT DELIVERABLE**

**D4.1**

**State of the Art and Preliminary Architecture**

Due date: 30th November, 2013 (M12)
Deliverable version: 1.0
Editor
Alysson Bessani (FFCUL)

Contributors
Alysson Bessani, Vinicius Cogo, Paulo Verissimo, Ricardo Mendes, Tiago Oliveira, Ricardo Fonseca, Fernando Ramos, Nuno Neves (FFCUL)

Disclaimer
This work was partially supported by the European Commission through the FP7-ICT program under project BiobankCloud, number 317871.

The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose.

The user thereof uses the information at its sole risk and liability. The opinions expressed in this deliverable are those of the authors. They do not necessarily represent the views of all BiobankCloud partners.
Executive Summary

This deliverable presents an overview of the state of the art in the integration of Biobanks and sequence data repositories together with the preliminary OVERBANK architecture and its main components. In particular, we propose to implement our sharing infrastructure as a dropbox-like system in which authorized researches can store and share information between biobanks, using also public clouds as additional storage.
Contents

1 Introduction

2 A vision of e-Biobanking
   2.1 Introduction ......................................................... 3
   2.2 The advent of e-Biobanking ........................................ 5
   2.3 Security and dependability challenges of e-Biobanking .... 6
   2.4 The BiobankCloud approach to secure and dependable biobanking .... 8
   2.5 Final Remarks ....................................................... 10

3 State of the Art in Biobank Integration
   3.1 Introduction ............................................................. 11
   3.2 Background ............................................................. 11
      3.2.1 Biobank Data .................................................... 11
      3.2.2 Data Integration ............................................... 12
   3.3 Biobank Integration in the Real World ......................... 16
      3.3.1 INSDC: Integrating Public Sequences ....................... 16
      3.3.2 UK Biobank: Integrating Biobanks from a Country ......... 18
      3.3.3 BBMRI and ELIXIR: Integrating Biobanks from a Continent ..... 18
      3.3.4 GenomEUtwin: Integrating Biobanks with Similar Missions .... 19
      3.3.5 COMMIT: Integrating Classical Biobanks in Virtual Biobanks .... 19
      3.3.6 Cloud Initiatives: DNAnexus, BaseSpace and Galaxy ......... 19
   3.4 Final Remarks ....................................................... 22

4 OVERBANK Storage Architecture
   4.1 Introduction ............................................................. 23
   4.2 Charon Architecture ................................................. 26
      4.2.1 Background ....................................................... 26
      4.2.2 Charon Overview ............................................... 27
      4.2.3 Cloud-of-Clouds Data Store .................................. 27
      4.2.4 Caching ............................................................. 29
      4.2.5 Sharing with Charon .......................................... 30
      4.2.6 Garbage Collector ............................................... 32
      4.2.7 Dealing with Big Files ....................................... 32
   4.3 RANC Architecture ................................................... 33
      4.3.1 Integrating the RANC with Charon ......................... 35
   4.4 An SDN architecture for RANC ..................................... 35
      4.4.1 Rationale for an SDN infrastructure ....................... 36
   4.5 Final Remarks ....................................................... 37
5 A Privacy-assuring Disclosure Filter for Genetic Information
  5.1 Introduction ................................................................. 38
  5.2 Related Work ............................................................. 39
  5.3 Disclosure Filter .......................................................... 40
    5.3.1 Adversary Model ..................................................... 40
    5.3.2 Populating the Filter: Algorithms .............................. 41
    5.3.3 Discussion ............................................................ 44
  5.4 Implementation .......................................................... 45
  5.5 Experimental Evaluation .............................................. 46
    5.5.1 Experimental Environment ....................................... 46
    5.5.2 Disclosure Filter Input Data .................................... 47
    5.5.3 Bloom Filter Size .................................................. 47
    5.5.4 Local Performance .................................................. 48
    5.5.5 Sensitive Reads ..................................................... 49
  5.6 Use Cases ................................................................. 49
    5.6.1 Archival of Raw Data .............................................. 49
    5.6.2 Hybrid Sequence Alignment ....................................... 50
  5.7 Final Remarks .......................................................... 51

6 Conclusions and Future Work ........................................... 52

A Erasure and Network Codes .............................................. 54
  A.1 Introduction ............................................................ 54
  A.2 Distributed data storage in cloud-of-clouds ..................... 54
  A.3 Resilient data transmission in multipath communication .......... 55

B Software-Defined Networks .............................................. 56
  B.1 An introduction to Software Defined Networking ............... 56
  B.2 SDN Controllers and Programming Languages .................... 57
  B.3 SDN Applications ...................................................... 58
  B.4 SDN Main Challenges .................................................. 59
Chapter 1

Introduction

Chapter Authors:
Alysson Bessani (FFCUL).

Biobanks are repositories for storing biological samples and genetic information for use in research. The decreasing cost of sequencing genomes is increasing exponentially the amount of data to be stored and analyzed [140], whereas biobanks are still increasing linearly in terms of storage and processing capacity. Moreover, geographically dispersed biobanks are isolated data islands, while medicine needs their integration to improve genetic associations with complex diseases [41]. This association depends on understanding the interactions between gene, environment, lifestyle and disease. For instance, there are estimates that genetic main-effect studies require 2–5k samples to make strong associations, lifestyle main-effect studies require 2–20k samples, and gene-lifestyle interaction studies require 20–50k samples [139].

Integrating biobanks assists the evolution of several sciences by increasing the number and diversity of biological samples available for researchers. However, it is not an easy task since biobanks can store distinct data, employ different data models and formats, as well as efficiency and security requirements must be correctly addressed in such integration.

The BiobankCloud project aims to develop a cloud-computing platform as a service (PaaS) for biobanking. Such platform will provide scalable and secure storage, data-intensive tools and algorithms, support for data sharing between biobanks and for using public clouds without endangering data privacy. The workpackage 4 (WP4) is particularly concerned with the last two features.

WP4 aims to interconnect individual biobank clouds (i.e., the instances of the BiobankCloud PaaS deployed within a biobank information infrastructure) through a semi-structured loosely coupled shared storage abstraction that allow a federation of biobank clouds to share information between them. This abstraction will be based on a data-centric coordination model with the objective to externalize a (anonymized) subset of the data stored in a participant biobank to other biobanks of the federation. The idea is to build an OVERBANK, an overlay biobank composed by several biobank clouds that allows access to the high volume of data stored in these biobanks. Given the privacy sensitivity of sequenced data, all data stored in the OVERBANK need to respect the access control framework (as defined in WP3) and the privacy constraints related with such data (as defined in WP6).

A second objective of WP4 is to integrate the storage and processing capacity of public clouds in the OVERBANK without putting the privacy of donors at risk. The key idea here is
to allow individual biobank clouds to increase their storage and analysis capacity by moving part of its data to public storage clouds (e.g., Amazon S3), even if temporarily, to make room for load spikes (e.g., when new sequenced data set comes in for analysis) or unplanned growth. Furthermore, such public clouds can also be used as archival storage for compressed data.

**Deliverable 4.1.** This document describes the work done within the context of WP4 in the 1st year of the BiobankCloud project. In particular, it focuses on the definition of a vision for secure and dependable e-Biobanking for the project and on the design of a preliminary architecture for implementing such vision. This system architecture considers state-of-the-art techniques and tools for building wide-area storage systems, e.g., cloud-of-clouds storage, erasure and network coding, personal cloud storage services, and software-defined networks. Furthermore, in this deliverable we also describe a new technique for filtering privacy-sensitive sequence data, which can enable the use of public clouds without endangering the privacy of sequence donors.

Finally, the document also presents a study of the current practices and methods for integrating bioinformatics databases, biobanks and sequence repositories. The study shows that the techniques we are proposing are capable of revolutionizing the integration of biobanks and the use of public clouds for storing privacy-sensitive information. The ideas introduced in this deliverable are mostly related with the preliminary architecture of the OVERBANK and its subsystems. It is important to recall these ideas are still under development and will be fully described (maybe with minor modifications), implemented and even extended in the 2nd and 3rd year deliverables of the project.

**Organization of the document.** Apart from this introduction, the other four chapters of this deliverable have the following contents.

- **Chapter 2** gives an overview of the main trends in biobanking and presents our vision for secure and dependable e-Biobanking in the project.
- **Chapter 3** describes some concepts and methods for integration of bioinformatics data and a brief survey of the current initiatives for integrating biobanks and sequence repositories, including some recent cloud-based services.
- **Chapter 4** presents an overview of the OVERBANK as a Dropbox-like infrastructure for sharing data between biobanks and federated research institutions. The chapter also gives some details about the two fundamental sub-systems of this infrastructure: the *Charon distributed file system* and the *RANC communication system*.
- **Chapter 5** deals with the specific problem of identifying portions of sequence data that can be used to identify its donor. The chapter introduces a disclosure filter for genetic information – a kind of black list of short sequences that can contain privacy-sensitive information – and shows that this filter can be implemented efficiently in modern hardware. Using this filter, we expect to separate sequence data in what can be stored and processed in the public clouds and privacy-sensitive data that should be kept locally.
- **Chapter 6** presents a summary of this deliverable, relates our contributions with other BiobankCloud deliverables and discuss the next steps in WP 4.

The text concludes with two appendixes describing some techniques used in the OVERBANK: erasure codes (Appendix A) and software-defined networks (Appendix B).
Chapter 2

A vision of e-Biobanking

Chapter Authors:
Paulo Veríssimo, Alysson Bessani (FFCUL).

2.1 Introduction

Biobanking is becoming an extremely important activity, with deep societal, medical and scientific implications. Storage of biological material is getting systematic (tissues, fluids, etc.), both for clinical and research purposes. One of the recent major driving forces has been the quest for finding correlations between genomic features and diseases through large scale genome sequencing. This scenario may have two interesting outcomes.

Firstly, the capital and operational expenditure with biobanks will increase exponentially, given the considerable costs of the life-cycle of generating, pre-freezing and manipulating (−20 °C), and deep-freezing for storage (−80 °C), of the biological material samples. According to some predictions [33] the number of tissue samples in US banks alone was estimated at more than 300 million at the turn of the century and is increasing by 20 million a year.

Secondly, DNA sequencing has undergone a revolution in the past half-a-dozen years, with the advent of a breed of so-called Next-Generation-Sequencing (NGS) machines. These machines have lowered the price and increased the speed of sequencing by several orders of magnitude. Figure 2.1 shows the impressive evolution of publicly available DNA base pairs. The predictable result is that not only the rate of sequencing of stored samples is bound to increase – the latter themselves augmenting by the reasons explained in the previous paragraph – but also a market for individual genome sequencing, and a drive for large population genome sequencing for research purposes, will make the number of produced and stored genomes sky-rocket. Figure 2.2 shows some simple statistics about the recent evolution of the cost of genome sequencing.

Furthermore, although the current state of the art points to only sequencing a panel of selected genomic positions or of exomes (genome subsets, genes), it is foreseeable that the cost of selecting specific genome parts will soon be higher than simply sequencing the entire genome. This is also biologically advantageous, as there are many indications that many more parts of a genome are functional than thought in the last decades [56].

At first sight, it would be a matter of having the concerned organizations – hospitals, genomics, biomedical and bioinformatics research labs, etc. – invest more on their IT (see for example the Biobanking and Biomolecular Resources Research Infrastructure (BBMRI) [4], or
Chapter 3 – A vision of e-Biobanking

1. Introduction
As discussed in the preceding sections, Biobanking is becoming an extremely important system for collecting and storing biological material samples. According to some estimates, the number of tissue samples in US banks alone was estimated at more than 300 million at the turn of the century and is increasing by 20 million a year. This scenario may have two consequences: the capital and operational expenditure with biobanks will increase exponentially, given the quest for finding correlations between genomic features and diseases through large scale genome sequencing. The recent major driving forces has been systematic (tissues, fluids, etc.), both for clinical and research purposes.

However, a disturbing trend has recently been unveiled [128], explained in Figure 2.3: the rate of growth of storage capacity under fixed cost (dictated by Kryder’s law, the homologue of the famous Moore’s Law for digital storage [20]), was surpassed by the rate of the curve of genome sequencing capacity at fixed cost. The figure shows a comparison, over time, of sequenced DNA base-pairs per dollar, versus stored megabytes per dollar. This trend is bound to continue with no basic technology solution in sight on the horizon.

As the figure shows, every day we have a few dozens of thousands more tissue and fluid samples stored, and quite a few additional dozens of terabytes of stored raw data. The predicted growth of storage capacity under fixed cost (dictated by Kryder’s law) was surpassed by the rate of the curve of genome sequencing capacity at fixed cost.

---

![Figure 2.1: Evolution of the number of publicly available DNA sequences and base pairs (from the GenBank, http://www.ncbi.nlm.nih.gov/genbank/statistics).](image1)

![Figure 2.2: Evolution of the cost of genome sequencing (from the National Human Genome Research Institute, http://www.genome.gov/sequencingcosts).](image2)
On the other hand, cloud-based ecosystems may drastically improve data sharing and collaboration. This is very relevant in the context of the previously discussed solution avenues, promoting the comprehensive digitalization of biological samples, facilitating access and manipulation. Cloud computing will bring other benefits, especially for research, by alleviating the pressure on IT and helping put the base-pair vs. megabyte cost gap of Figure 2.3 back on the right side. Cloud computing will definitely alleviate the pressure on IT and help put the base-pair vs. megabyte cost gap of Figure 2.3 back on the right side.

Likewise, since we can sequence physical material cheaper and quicker, the burden that the digital counterpart is imposing on the IT systems of the organizations is also a main concern today, because it will become unsustainable at a short term.

2.2 The advent of e-Biobanking

An obvious first instance of solutions to the problem of cost and space for storing physical samples would be to consider classes of physical samples that, in time, might be replaced by their digital representation, and stored in computer media, thus freeing (more expensive) space in physical biobanks. In fact, less radical solutions are being pursued already, where digital copies of (harder to manipulate) physical samples are made available to researchers, in what have been called virtual tissue banks (see the COMMIT project [5], described in Chapter 3).

Another avenue further down this road would be to aggressively use compression in order to improve the storage factor of digital representations of biological samples, attacking the problem of disk space. DNA samples, for example, could be classified in hierarchies of criticality, allowing choosing between lossless and lossy compression for their computer representations. These issues are currently being considered to improve the utilization of IT resources [45], but more radical approaches, where physical samples may actually be selectively disposed of after being digitally processed – again freeing space in physical biobanks – are, in our opinion, inevitable, in order to ease the burden on the physical sample storage side as well.

On the other hand, NGS and the challenges for storage and post-processing it have put cloud computing and web services on the agenda [128], and this may well be the means to definitely alleviate the pressure on IT and help put the base-pair vs. megabyte cost gap of Figure 2.3 back on the right side. Cloud computing will bring other benefits, especially for research, by promoting the comprehensive digitalization of biological samples, facilitating access and manipulation. This is very relevant in the context of the previously discussed solution avenues. On the other hand, cloud-based ecosystems may drastically improve data sharing and collabo-
ration amongst organizations, which is currently developing (see BBMRI [4] or the e-BioGrid infrastructure for life-science research [8]).

In summary, we believe that we are observing the advent of the new era of e-Biobanking, in terms of the creation of genuine hybrid ecosystems composed of interplaying physical and computer storage and processing infrastructures, handling physical and computerized samples of biological data, in a symbiotic and seamless way.

In the short term, systems will still be largely decoupled from each other and there will be a complete separation of processes between the physical and digital domains. However, whilst digital information is currently stored and processed mostly in classical data centers, with public and/or private access, there will be a progressive evolution towards the use of public clouds, coexisting with dedicated data centers, as Figure 2.4(a) suggests. As time goes by, we foresee the e-BioBank vision come true, through the integration of what are today physical and virtual biobank processes in a completely symbiotic and seamless way (Figure 2.4(b)). Along this road, we predict the development of private-public cloud ecosystems, as a way to promote sharing and simultaneously address the multiple security and dependability challenges that will ensue.

![Figure 2.4: A vision of the road towards e-Biobanking.](image-url)

### 2.3 Security and dependability challenges of e-Biobanking

Whilst the above is a vision worthwhile being pursued, it brings significant security and dependability challenges.

As the dependence on computerized representations of physical samples grows, there is a need to guarantee incremental levels of dependability (reliability and availability). Laboratories will assume they can access any catalogued sample at the click of a finger, all day, every day, implying that challenging standards of availability are met, not only by the storage but also by the computing infrastructure. Furthermore, there will be a possibly very large collection of samples which were stored with lossy compression, or samples about which there will only be digital representations (remember the cost factor of physical biobanking) after the former have been completely analyzed and represented digitally. These classes of samples would have to
be stored in extremely reliable modules of the infrastructure, since their physical counterparts would no longer exist.

But even more worrying is the fact that a perhaps significant part of this material is security and privacy sensitive, such as information on individual’s diseases, or individual genomes. There are strict regulations on data protection that include DNA and disease information, especially in European countries, which would ultimately prevent the use of public clouds for those data. However, the situation is not homogeneous all over the world and, even in the more concerned blocks, there is enormous pressure for easing on out of strict protection, to the detriment of citizens’ privacy [40]. The fact often forgotten is that a genome or a chronic disease is not like a password: once compromised, you cannot change it, the harm is done forever and, to some extent, also to descendants, ancestors, and other family members.

When trying to meet the economics or usability challenges, like lowering storage costs or improving accessibility and sharing of data, we should always keep in sight these non-functional aspects, like reliability, availability, integrity or privacy, else the promised shiny e-Biobanking future may become very cloudy.

In fact, recent developments show that the movement towards using clouds and the web is irreversible. It is in the roadmaps of many players, from Biobanking consortia, through genomics and bioinformatics communities, to NGS machine vendors (see, e.g., BaseSpace from Illumina [75], discussed in Chapter 3). Not just any cloud, but including public clouds. Not just very restricted access, but including Internet web-based access. On the other hand, even if regulations are relaxed, we foresee that significant data will be restrained to the premises of their owners, even if its manipulation is authorized by third parties, creating two problems: secure sharing of that data; and the resulting lack of integration with less critical data housed in public clouds. The move to public clouds cannot be seen in isolation from the greater dependence on IT that the advent of e-Biobanking is bringing. The challenges expressed in the previous section have to be contrasted with a worrying roster of failure events suffered by major cloud providers over the past few years — including data and privacy loss — calling for dramatically better cloud resilience, especially when critical applications are at stake [131].

Last but not least, it is possible that a false feeling of security brought by the announced use of standard protection techniques (SSL, firewalls, anonymization, etc.) will impel people to subject critical data to a high level of threat. The well-known Attack-Vulnerability-Intrusion fault model [132], explains that the level of risk in the protection of digital assets is commensurate with the combined level of threat and the degree of vulnerability those assets are subject to. Critical biological data were reasonably screened from threats until now, even if kept in vulnerable systems, because they were isolated. The pressure to expose data and thus increase the level of threat, without completely understanding the risk that derives from the combination with the resulting vulnerabilities, may have serious consequences, especially societal, in what concerns privacy of individuals.

In order to give just an example, parts of the move to massively create e-Biobanking data are genome databases, where essentially citizens give consent that their genomes or exomes be published in public databases, under the assumption that data are anonymized. The same happens with cohort studies of patients suffering from a similar disease. This is as precious for science, as it may be damaging for those willing citizens if something called re-identification occurs: somehow it becomes possible to track from the public genomics or clinical data, back to the individuals.

And it is easier than it looks: it is called a re-identification attack, and has been shown to have happened in real data (including health) that had been publicly released – anonymized health data on state employees in the state of Massachusetts, US; anonymized AOL user data;
a huge database of anonymized movie recommendations by Netflix [110]. The problem is that current DNA and clinical data anonymization procedures essentially follow similar methods to the ones used in those earlier experiments known to have fallen prey to re-identification attacks. The intention is good but dangerous in a public release scenario: participants remain re-identifiable in order to provide clinically relevant information back to the donor. This is a vulnerability that can be exploited in several ways and in fact can even hit more sophisticated anonymization approaches, as shown by a team of researchers who could recently identify the surnames of 12% of the sequence donors from the 1000 Genome project [92]. In consequence, 131 citizens have lost the privacy of their genome forever.

Going a bit further, in a world of massive and not so careful storage and manipulation of genomic data, the prospect of it being possible to massively connect DNA samples to individuals or populations thereof, is in itself a daunting vulnerability. Threats can get much more worrying, such as the possibility of engineering a targeted attack on a person’s genetic fragilities, found from her hacked DNA footprint. Generalizing, we could have the massive propagation of diseases carefully crafted so that certain population groups would have no immunity, for terrorism or illegal profit (e.g., remedy drug sales). Likewise, cybercriminals might be hired to modify suspect’s DNA profiles stored in police databases. Synthetic DNA technologies soon to become reality, planting false evidence connecting a person to a scene without ever having met her will no longer be a sci-fi subject: suffices to hack her DNA from a database [66].

2.4 The BiobankCloud approach to secure and dependable biobanking

One of the key objectives of the BiobankCloud project (and in particular WP4) is to address these major challenges, in the context of open biobanking, that is, one privileging cloud ecosystems, and data sharing and collaboration amongst organizations. In this context, we are seeking to preserve the following key security and dependability objectives, despite threats:

- promote easy but secure access by both occasional and sophisticated users;
- enforce dependability (reliability and availability) against cloud outages and other failures;
- guarantee security (integrity and privacy) of data against non-authorized users and malicious insiders in cloud providers.
- enforce long-term protection against re-identification and linkage attacks (In which some known person in linked to a certain study or anonymized database).

A high-level view of the BiobankCloud storage architecture, or OVERBANK, is depicted in Figure 2.5. BiobankCloud draws on recent results from yet another European project, TClouds [27], to create storage architectures which are based on clouds-of-clouds. That is, multiple instances of clouds, both private and public, from several stakeholders and providers, respectively, are involved in creating the abstraction of a single cloud-based key-value store or file system, which is perceived seamlessly as a single cloud, by the e-biobank users and administrators. However, underneath, powerful encryption and fault-tolerance mechanisms ensure that the several components interoperate so that no part is a single-point-of-failure, in the presence of accidental or malicious threats.
Figure 2.5: Overview of OVERBANK: the BiobankCloud integrated architecture.

On the other hand, the modularity of the architecture makes it easy to set up secure and dependable constellations of private and public clouds belonging to diverse stakeholders, with separation of risk and concerns. BiobankCloud technology will hopefully help the e-biobanking vision come true, enabling the creation of ecosystems involving coalitions or consortia of: hospitals, biomedical or bioinformatics research institutions, non-profit organizations for biobanking and genomics research support, and even NGS vendors’ own infrastructures. Occasional and power users may easily use the infrastructure in an integrated way, through comfortable Platform-as-a-Service interfaces. Experiments may be set up to automatically share data belonging to several realms, improving throughput and turn-around of experiments. For example, bioinformatics applications may perform complex computations on mix-criticality data very efficiently, by manipulating critical clinical data without it ever leaving the private cloud where it is stored, whilst simultaneously accessing archive data in a public cloud.

However, security and dependability must not be compromised. First, data of different levels of criticality will reside in different cloud subsets with adequate levels of protection, preserving liability and regulations. For example, privacy-critical data will never leave private clouds, yet may processed by having authorized users ship the adequate certified computing functions to those clouds. Figure 2.6(a) suggests a typical example: bioinformaticians at University Z being able to perform operations on mix-criticality data residing in public and private clouds. Supposing that critical data cannot leave some private clouds, trust would be achieved by having the job dispatcher ship certified functions to those clouds, running them, and receive the results back. Second, BiobankCloud technology will also significantly leverage the availability of a competitive market of public clouds to optimize the cost factor of e-biobanking, using them as is for a start, yielding direct access to public data by large communities of users, e.g., data anonymized with innovative algorithms, which we call privacy filters (see Chapter 5), being developed in the project to overcome vulnerabilities leading to re-identification.

A third important development is the ability to use generic public clouds as artifacts of back-end very resilient and highly secure cloud-of-clouds storage, to address the fast growing need for storing critical data in a cost-effective way, that is, complementing private clouds. To achieve this objective, powerful state-of-the art encryption, coding and dispersion mechanisms [34], will be used. Figure 2.6(b) illustrates the basic mechanism in a simple way: data, whilst still

---

1 Although this problem is not being addressed in this deliverable, we expect to exploit the OVERBANK capabilities for providing such feature in the 2nd and 3rd year of the project.
in a trusted environment (the private cloud) is allocated a key and encrypted. Next, it is subject to erasure-coding, a form of coding that produces m coded chunks from the initial data, with enough redundancy that the whole data can be recovered from just a few chunks. Then, the key is subject to secret-sharing, a cryptographic operation that “breaks” the key into several shares, m in this case, again ensuring that the key can be recovered from just, say, k < m shares, whilst guaranteeing that it cannot be recovered from less than k shares.

This method resists attacks and failures, both to the integrity of the data and the confidentiality of the key, as long as their severity (the number of chunks or shares affected) is not higher than a pre-defined protection threshold, let us call it f. In the example of the figure, f = 1, and it is enough to recover f + 1 i.e., two chunks and two key shares, to restore the data. On the other hand, for privacy to be broken, the attacker would have to simultaneously hack into two different clouds. Likewise, for the data to become irrecoverable, the attacker would have to compromise all clouds but one, and destroy the respective chunks. On the other hand, cloud-of-clouds diversity reduces failure probability and increases the attacker working factor. Note that, despite the simplicity of the example, the scheme is parametric; f can take any value that the designer chooses, allowing to attain arbitrarily high levels of resilience and security, at the cost of higher levels of redundancy. In Chapter 4 we give more details about how this scheme will be employed in the OVERBANK.

2.5 Final Remarks

In this chapter we presented our vision of e-Biobanking, in which a federation of biobanks storing not only physical samples, but also related datasets, operated can share data between themselves and use public clouds and shared infrastructures for storing such data securely. In the next chapters we present more details about such vision, and how we plan to implement it in the BiobankCloud software platform. However, before delving into these details, the next chapter of this deliverable presents an overview of the state of the art in the integration of bioinformatics data repositories and biobanks.

---

2See Appendix A for more details about these codes.
Chapter 3

State of the Art in Biobank Integration

Chapter Authors:
Vinicius Cogo, Alysson Bessani (FFCUL).

3.1 Introduction

In this chapter, we present the current state of biobank integration and discuss the opportunities arising from the employment of state-of-the-art techniques from existing initiatives in this context. The remaining three sections of this chapter are organized as follows. Section 3.2 contains a review of data handled by BiobankCloud project, and data integration classifications and methods. Section 3.3 analyzes cases of biological and biobank data integration in the real world, from international consortia to companies, passing through national and European projects. We conclude the chapter in Section 3.4 with some final remarks about what we could learn from the state of the art.

3.2 Background

This section contains a background review of data handled by BiobankCloud project together with a description of concepts, classifications and methods for data integration.

3.2.1 Biobank Data

A biobank is a repository that stores a sample collection composed by several studies. Today, a large portion of biobanks are mostly concerned with the storage of physical samples (e.g., blood samples), however, there is an ongoing transition to use such infrastructures also to archive the data associated with these samples. Such data is usually represented by the concept of study. Each study contains two types of data: the study metadata and the omics data [97]. The former describes donors and samples in general terms, whereas the latter is the real study data and is comprised of omics descriptive metadata, datasets and analysis results. In the BiobankCloud project we are mostly interested in dealing with data from human donors and samples, since there are important challenges in storing and sharing such datasets due to its privacy sensitivity.

Figure 3.1 presents a graphical representation of the biobank data classification adopted in the BiobankCloud project.
In practical terms, from all types of data in the bottom of figure, only datasets can contain sensitive information that may compromise donor’s privacy, whereas all other data must be anonymized or public. These datasets can contain whole genome sequences and short DNA or RNA sequences. They can not be downloaded from the biobank by researchers and need to be analyzed locally where they are stored. In terms of data integration, both the study metadata and the omics datasets are important. The latter is important for obvious reasons: the main objective of integration is to allow researchers to have access to a rich collection of datasets from the federated biobanks. The study metadata is also important because it enables researchers to discover the existence of datasets of interest to carry on studies considering correlated samples in different biobanks. Within the BiobankCloud project, the MIABIS (Minimum Information About Biobank data Sharing) [109] data model is the intermediate solution to integrate biobanks metadata.

### 3.2.2 Data Integration

Data integration roughly consists in providing an unified view and access of multiple datasets to users [86]. More specifically, it aims at (1) finding reliable sources and pulling the needed information from the databases designed for this purpose and (2) understanding the power of data at large when a number of data sources are available [38]. Data from multiple sources can be heterogeneous, with different structure, models, query languages, semantics, etc.

#### Classifications

The integration problem is a long-standing concern in computing. There are several ways to classify and organize solutions to integrate data, from which the first classification is based on interoperability layers [124]. It is divided into four conceptual groups characterized by known problems in data integration: *system, syntax, structure* and *semantic* interoperability [124]. In Figure 3.2, we present these layers and some examples of aspects that are considered important within each one of them. In the following we briefly describe these layers.
System interoperability involves dealing with differences of hardware, operating systems, database management systems, and communication protocols [124]. It can be subdivided into three classifications: by user model [72], system transparency [72] and exchange regime [104]. The user model determines the expected knowledge level of system users and can be classified in: inexpert users, interactive queries, query experts and language expert users [72]. The system transparency refers to the extent to which the user has control over or is required to specify the particular sources to be used in answering a query. It can be divided into sources specified by users, hard-wired specified by the system or dynamically selected by the system [72]. The exchange regime defines the means of access and relations between partners of a collaboration. It can be divided into three categories: the cooperation (samples freely circulate), the reciprocity (only the agreed samples are exchanged) and the subcontract (samples are provided per contract) [104].

Syntactic interoperability refers to the integration of different data models, types and formats [124]. The data model determines whether data is structured or not, and the possible groups are structured, semi-structured and unstructured data. The data type defines the type of data that will be applied for each data field. One problem in this case is when systems need to integrate semantically-related fields that were declared with different data types, for example, using float and integers for the weight of a person. The data format defines the encoding format applied to a data field. One problem in this case, which is very similar to the previous case, is when a system needs to integrate fields encoded differently, for example, the donor name is encoded using Unicode in one system and ISO-8859 in another.

Structural interoperability refers to the integration of different data schema [78] and representation in structured databases [124]. A data schema describes a database structure in a formal language supported by the database management system (DBMS) and a classification based on it is subdivided into three main integration levels: conceptual, logical and physical.
schema integration [78]. In representation terms, data describing the same object can be represented in diverse ways and their integration is a specific problem that need to be addressed case by case.

Semantic interoperability focuses on integrating the meaning of information and content from different data sources [124]. It is subdivided into two classifications: by source complementarity and mediation approach. In source complementarity, data sources can be complementary (they do not have the same data) or overlapping (they might have the same data). If they are complementary, we can employ horizontal integration by only concatenating data from each source to create an answer to a query. In the overlapping case, we can employ vertical integration [72, 129] by merging data from the sources to create an answer. A mediation approach [119] defines how semantic integration will be performed and it is directly related to ontologies. Ontology is an important concept for semantic integration, and can be defined as a specific vocabulary and the relationships used to describe and store data about certain aspects of reality. It can be more complete and precise than simple keywords, thesauri or taxonomies, because it can create several levels of relation between internal terms or even between terms from different ontologies. Mediation approaches can vary from an alignment, to the partial compatibility and the unification approach [119]. The first mediation approach is a mapping of concepts and relations from one ontology to other, indicating equivalence. The second is an alignment that supports equivalent inferences and computations on those equivalent concepts and relations. The last is an one-to-one alignment of all concepts and relations that allows any inference or computation expressed in one ontology to be mapped to an equivalent inference or computation in another.

Methods

There are different integration approaches that consider the different ways to provide a global view of data. Some of the first surveys [72, 127] defined three main categories of data integration: the data warehouse, the mediator-based and the navigational integration models. Another survey [90] separated mediator-based integration from federated databases and included a category for peer data management systems. Finally, a more recent survey [100] presented 9 categories of data integration, being those three (warehouse, mediator and navigational) the main known data integration models. Furthermore, it is also interesting for BiobankCloud to consider models based in loosely coupled systems and semantic web technologies.

We start by presenting two of the best known methods in literature: the data warehouse and the mediator-based models. In Figure 3.3, we compare all these methods with a non-integrated scenario. In the non-integrated case (Figure 3.3(a)), clients must know the location of all data sources, query each one of them, and correlate the resulting data from each query to obtain the answer. Data sources do not exchange information in this scenario, which can lead to problems, such as, heterogeneous answers (non-harmonized data), heterogeneous semantics, and duplicated answers.

The first data integration model considered in this study is the data warehouse [72, 127, 90, 100] (Figure 3.3(b)). This method retrieves, harmonizes and stores data from multiple sources into a local, central storage. A client need to perform only one query to the data warehouse to receive an harmonized and de-duplicated answer. This system does not rely on the network to access data during a query, since it previously fetched all data to the local storage (dashed arrows marked with an R). Furthermore, it avoids network bottlenecks, slow response times and occasional unavailability of remote data sources. Queries can easily be optimized in execution time since there is only one local data repository. It also allows users to filter, validate, modify...
and annotate data obtained from multiple sources. The main problems of this model are the cost of maintaining such system and the possible loss of freshness when accessing out-of-date information stored locally. Due to the second problem, warehouses need to regularly check the underlying sources for new and updated data to reflect them on the local copy.

The second data integration model analyzed in this study is the **mediator-based** [72, 127, 90] (Figure 3.3(c)). This method considers the existence of a mediator that maps each client query into a set of specific queries for underlying sources at runtime and aggregate the replies in a single answer to the client. The main advantages of this model are that it does not need a large centralized storage system (lower maintenance cost than warehouses) and the data is always up-to-date (do not need the synchronization step as in previous model). The disadvantages are that it relies on the network to access data on demand (overall performance is equal to the slowest source) and it is difficult to optimize queries for all external data sources. The federated database model [90, 100] is a smooth case of mediator-based model, since data sources collaborate among themselves.

The **navigational integration** [72, 127, 100] (Figure 3.3(d)) method provides interactions between users and pages similarly to a point-and-click web navigation. It is also known as link-based integration and relies in cross-references between services to allow users to navigate from one page to another in a different service. Workflows running over this method redirects the output of one service to the input of the service responsible for the subsequent workflow step. User queries are translated to path expressions that results in reaching pages containing...
the desired information and is only reachable through this particular path. Each data source is defined as a set of pages with interconnections among them and specific entry-points. The main disadvantages of this method are the vulnerability to ambiguity, the problem of broken links and that the onus of integration and interpretation is on researcher side [127].

The loosely coupled method [90, 100] (Figure 3.3(e)) integrates data from multiple sources using the minimal amount of knowledge as possible about each of them. This method is also known as integration by peer management systems and normally uses flexible file formats such as XML and JSON. Each network contains a minimal mediated schema that represents the semantic knowledge of this network. New sources entering in the network are mapped to provide information using this mediated schema. Internal modifications on each source are reflected in this mapping, making them still comply with the mediated schema. Loosely coupled systems normally use a minimalistic approach to integrate diverse databases by mapping to only basic data types and using only modest adaptations of existing web resources. A client issues requests to any network member, which are forwarded to other peers that will aggregate information to create the final answers.

The integration method based on semantic web [124, 100, 101, 93] (Figure 3.3(f)) allows the integration of semantically related information, regardless of distribution and heterogeneity. Clients issue queries to end points (e.g. a SPARQL engine) that perform queries to databases through languages for data representation (e.g. RDF – Resource Description Framework). These end points can be centralized or distributed, allowing a wide range of ways to integrate data sources. The usage of languages for data representation allows the integration of very different data sources, for instance a web-server, a relational database and a file server. RDF describes data in triplets containing: the subject (resource identifier), the predicate (an attribute name), and the object (attribute value). The answer to each request from clients can contain diverse types of data representing the object, from simple numbers or ontology terms, to unstructured data files. Linking data received from queries is one of the most important steps when integrating data through semantic web, because it is when complementary data are correlated to make sense for several applications.

3.3 Biobank Integration in the Real World

In this section we discuss the approaches in use to integrate biobanks and bioinformatics data repositories. We start by presenting one of the first integration initiatives for public data, and follow the discussion with initiatives to integrate biobanks from a country and a continent. The last biobank-related case is about integrating biobanks with similar missions, and we finalize the presentation of initiatives by analysing cloud-based solutions.

3.3.1 INSDC: Integrating Public Sequences

Our analysis of real cases starts with the International Nucleotide Sequence Database Collaboration (INSDC) [16], a joint effort to collect and provide a globally comprehensive collection of public domain nucleotide sequences and associated meta-data [46]. There are three members leading this initiative for more than three decades: the DNA Data Bank of Japan (DDBJ) [6], the National Center for Biotechnology Information (NCBI) [13] and the European Bioinformatics Institute (EBI) [28].

When submitting a sequence to this system, a researcher must send the sequence to only one of the INSDC partners. Such partner will be the authority for this sequence, reserving an
accession number for it in the shared accessioning system. The three partners synchronize their databases daily by forwarding to others all new, modified or removed entries, as presented in Figure 3.4. As a consequence, INSDC can be seen as a geographically replicated repository deployed in three federated databases, since the data model and relationships among entities are exactly the same in all partners.

![Figure 3.4: The INSDC distribution and synchronization.](image)

Each partner is responsible for propagating those records for which it is authority to the other partners. Under a pull model, each of the two other partners pulls this authority dataset and integrates the new records in its own system. In the case of an updated record, this simply overwrites the previous record under the same accession number. There are checks and list exchanges at the end of this daily exchange protocol to be sure data is synchronized across the entire system. Each partner, then, holds the complete set, but it is authority only for the portion of the records that has been submitted to the system through itself. Extra data (non-shared) are added across the complete set at each site - e.g., cross-references are added to entries at EBI. There is no requirement for any specific shared database or file system, it simply relies on the universal accessioning namespace and a mutually understood presentation format for each type of data.

The concurrency control is done through the shared accessioning system, since it allocates a sequence identifier when a create request is issued. However, there is no protection against duplicated sequences if the same sequence is submitted to more than one partner. The INSDC provides versions of the entire database to be downloaded through the Sequence Version Archive service [10] and some scripts for updating local instances daily. Finally, the infrastructure allows an entry to be removed from their database, however it does not provide guarantees that the entry will disappear from everywhere, since other people can already have downloaded local copies from the database.

The main limitation of INSDC is that it does not deal with private sequences: everything submitted and manipulated by the system is considered to be public.
3.3.2 UK Biobank: Integrating Biobanks from a Country

The UK Biobank is a major UK collaborative research project to recruit and follow longitudinally the health of 500,000 volunteers aged between 40–69 years from this country [111]. Collected physical samples are transported to a central site in Manchester for processing and they are stored on two geographically separate cryopreservation facilities. This infrastructure also stores the datasets associated with such physical samples. The data in UK Biobank is divided into three main categories: protected, managed and open resources. The first contains participants health data (including DNA) and medical records, the second one contains non-sensitive material that still needs to be protected for scientific and ethical reasons, and the latter can be freely available. DNA is extracted from stored blood samples by a company called Affymetrix and returned to the UK Biobank.

Integration between entities occurs in the sample level since samples are sent to a central biobank right after the collection. There is also a central data repository that stores all data sets locally. Additionally, they employ a data warehouse model considering the integration of local data with other national health systems. Researchers present proposals of studies to obtain access to data, which must be approved by a council. There are costs involving this access, from proposal analysis fees to separated values for accessing only data or data and physical samples.

3.3.3 BBMRI and ELIXIR: Integrating Biobanks from a Continent

The BBMRI (Biobanking and Biomolecular Research Infrastructure) [3] is a pan-European research infrastructure aiming to improve the accessibility and interoperability of the existing collections of biological samples from different populations of Europe. The preparatory phase of this project finished in 2011 with a joint catalogue comprised of 311 biobanks distributed in Europe and more than 1.8 million DNA samples.

The next phase is the BBMRI-ERIC (European Research Infrastructure Consortium) [4, 143] that will integrate all these resources into a hub-and-spoke [105] network properly embedded into European scientific, ethical, legal and societal frameworks. The hub-and-spoke model consists basically in creating or choosing a central resource and connecting all others to it instead of creating one connection between each resource pair. BBMRI-ERIC intends to create a network of hub-and-spoke instances, by choosing major nodes as hubs in behalf of a region or country, and local biobanks acting as end nodes (spokes). A researcher sends queries to one of global portals that is connected to all hubs, and consequently to all spokes. Each hub receives the query and forwards it to the end nodes of its responsibility. Each spoke provides the local biobank service that returns the result of those issued queries over the local database. Hubs need to aggregate these answers and present a final, integrated answer in the global portal. The integration prototype from BBMRI-PP is already implemented in test-instances of existing biobanks.

ELIXIR [9] is an European project that aims to builds and operate a sustainable infrastructure for biological information in Europe. It intends to support life science research and its translation to medicine and environment, bio-industries and society [50]. Information gathered on data access and user requirements was central in designing ELIXIR as a distributed infrastructure with a central hub. The hub will be connected to ELIXIR nodes (spokes) hosted at centers of excellence in universities and institutions across Europe. ELIXIR is expected to integrate access and data from 12 research infrastructures being funded by European Commission, including the BBMRI.
3.3.4 GenomEUtwin: Integrating Biobanks with Similar Missions

GenomEUtwin [88, 105] is an international collaboration between eight registries providing information about more than 600,000 human twins pairs. They propose the TwinNET, a federation of local data warehouses, combined with a global mediator that provides transparent access to them through database instances and using DiscoveryLink. The DiscoveryLink is a database middleware created by IBM that extracts data from multiple sources in response to a single query [70]. This system architecture is also a hub-and-spoke, where the hub is the integration node and spokes are data-providing centers. Connections between hub and spokes are made using VPN tunnels, which are initiated from spokes. Each spoke (data-providing centre) contains a local data warehouse, which is fed with harmonized data from local production databases and LIMS. Data is translated and transferred into this data warehouse (called TwinMart) located in a demilitarized zone within each spoke of TwinNET. Each subject receives an unique GenomEUtwin identifier and twins share portions of it. All databases and data sets maintained under the TwinNET are anonymous, where the only allowed identifier is the GenomEUtwin identifier.

Some advantages of this model are the opportunity for query optimization and the transparent access to data. One major weakness of this prospective cohort approach is the enormous amount of money that must be invested before information can be retrieved. Other problem is that partners must increase the maintenance cost of their local infrastructures since they need to control one more component, the TwinMart (a local data warehouse).

3.3.5 COMMIT: Integrating Classical Biobanks in Virtual Biobanks

A virtual biobank is a repository that provides data obtained by means of characterization and sequencing from samples stored in classical biobanks. The COMMIT project [5] is a virtual tissue biobank that provides access to digital information about physical samples (harder to manipulate). In the specific case of COMMIT the digital information are data and image sets obtained from mass spectrometry and tissue microarray experiments. These data sets are an important input to proteomics workloads, for instance, the discovery of amino acids composing a protein and the validation of protein folding predictions. Additionally, the physical samples are breast cancer sections collected from medical partners integrating COMMIT’s partners. The fact of this samples being extracted from specific cases or diseases characterizes this initiative also as an example of integration of biobanks with similar missions. The infrastructure is comprised of a central repository for physical and digital samples in one project partner. Data is accessible from exterior through a web portal, after researchers being authorized through formal bureaucrat agreement protocols. The COMMIT project provide a workflow based management system and distributed processing resources. We can consider the goal of this project similar to the idea of storing and providing FASTQ files (the basic input for many genetic workflows) instead of sequencing the entire genome each time one wants to execute an experiment.

3.3.6 Cloud Initiatives: DNAnexus, BaseSpace and Galaxy

This section analyze three cloud-based solutions for storing and analyzing biological sequences, namely: DNAnexus [7], BaseSpace [2] and Galaxy [29]. We group these systems together because they are quite similar: they are implemented in (mostly public) cloud-infrastructures and assume users can create a virtual storage and processing infrastructure for managing its data. Notice that there is a change of paradigm here. Instead of downloading datasets, working
on them locally and producing new datasets to be inserted in shared infrastructures (e.g., UK Biobank or INSDC), these cloud infrastructures promote storing the data in the public cloud, and processing also in the cloud, close to the data. This is done through the use of high-level tools (e.g., operated by web interfaces) or through the usual tools deployed on cloud virtual machines.

**DNAnexus** [7] is an American startup company that provides a cloud-based platform for genomic enterprises to expand their local infrastructures. It is an API-based infrastructure and a workflow-based tool. The entire solution is deployed over Amazon Web Services (AWS). DNAnexus addresses some security concerns during data transfer and storage [54], namely:

- Network communication are encrypted and secured with SSHv2 and HTTPS (SSL/TLS).
- Access is controlled with strong passwords and two-factor authentication.
- All access is logged for accountability.
- All uploaded files are virus scanned and stored encrypted (using AES-256) on disk.
- The system is compliant with several privacy regulations, such as, HIPAA, 21 CFR Part 493, 21 CRF Part 58, 21 CRF Part 11, GCP, CLIA, EU Directive 95/46/EC and ISO 27002.

**BaseSpace** [2] is a product from Illumina that allows their Next Generation Sequencing (NGS) clients to directly connect their sequencing machines with the cloud. The idea is to help Illumina clients avoiding the need of a local IT infrastructure to store and analyze genomic data. One important idea of BaseSpace is to provide an ecosystem in which third-party developers can create new tools for BaseSpace users, in a similar way to what is provided by Amazon in its market place. The BaseSpace addresses some security concerns during data transfer and storage [75], namely:

- Illumina instruments provide on-board software for encryption (AES-256 standard), transmission and rescheduling for transferring data to BaseSpace.
- All traffic is sent over SSL channels.
- All service requests require a signature with the API key for authentication.
- Requests are monitored and they employ blacklist methods.
- All data is stored in S3 and no files are publicly accessible.
- AWS transparently provides replication and integrity checks.
- The system is compliant with several privacy regulations, such as, SOC 1/SSAE 16/ISAE 3402, FISMA moderate, PCI DSS Level 1, ISO 27001, FIPS-140-2 (due to AWS certifications).

**Galaxy** [29] is an open-source software package that provides three possible usages: a free web-based service, a cloud-based deployment, and a local private deployment. Such flexibility is a consequence from the fact that Galaxy is not a company or a cloud-based service, like previous solutions, but instead a software package to be deployed in physical or virtual machines. In the public cloud-based deployment, Galaxy provides a wizard for installing it on Amazon
EC2 from AWS. A possible security problem is that users must provide their AWS credentials to deploy a cluster through the wizard. An advantage of creating your own cluster is that you can increase or reduce the number of computing instances running your installation, as well as persistently terminate and relaunch the cluster. This solution is based on an Infrastructure-as-a-Service scenario since users can determine how many resources will be allocated to run their workloads.

Additionally, anyone can customize Galaxy to create its own instantiation. One example is the e-BioGrid project [8], which supports life-science research through the preparation and maintenance of computing environments running over the BigGrid. The BigGrid is a national computing infrastructure created to support the execution of research experiments from Netherlands. Other countries also have similar infrastructures, for instance, the Grid’5000 [15] from France and the WestGrid [31] from Canada. The main contribution of e-BioGrid project is that they provide specific computing environments that are functional by default for predefined experiments and studies. The custom instance of Galaxy Project running over BigGrid is one example of these environments.

**Comparison.** Table 3.3.6 presents a comparison among the three cloud-based systems discussed in this section. The focus of these services is to provide a complete platform for managing and analyzing sequencing data rather than data integration. All three solutions are deployed in Amazon AWS, which means that they are not immune to vendor lock-in issues due to pricing raising or change of policies. The table also includes a column about Amazon AWS for the sake of comparison with a public cloud provider.

<table>
<thead>
<tr>
<th></th>
<th>BaseSpace</th>
<th>DNAnexus</th>
<th>Galaxy</th>
<th>Amazon AWS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>Cloud-based platform</td>
<td>Cloud-based platform</td>
<td>Cloud-based platform</td>
<td>Cloud provider</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Product by Illumina</td>
<td>Startup company</td>
<td>Academic project</td>
<td>Product</td>
</tr>
<tr>
<td><strong>Philosophy</strong></td>
<td>Proprietary</td>
<td>Proprietary</td>
<td>Open-source</td>
<td>Hybrid</td>
</tr>
<tr>
<td><strong>Public cloud provider</strong></td>
<td>Amazon AWS</td>
<td>Amazon AWS</td>
<td>Amazon AWS</td>
<td>-</td>
</tr>
<tr>
<td><strong>Storage cost</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 TB</td>
<td>Free</td>
<td>$143/month</td>
<td>Same as AWS</td>
<td>$97/month</td>
</tr>
<tr>
<td>2 TB</td>
<td>$250/month</td>
<td>$286/month</td>
<td>Same as AWS</td>
<td>$160/month</td>
</tr>
<tr>
<td>10 TB</td>
<td>$1500/month</td>
<td>$1433/month</td>
<td>Same as AWS</td>
<td>$820/month</td>
</tr>
<tr>
<td><strong>Processing cost (min-max)</strong></td>
<td>Depends on the tool</td>
<td>$0.20 – $2.80/hour</td>
<td>Same as AWS</td>
<td>$0.06 – $4.60/hour</td>
</tr>
<tr>
<td><strong>Predefined workloads</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td><strong>Custom workloads</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Custom scripts</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Share workloads</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Share data</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of the three cloud initiatives for storing and analyzing biological sequences when deployed in Amazon AWS.

By analyzing this table, we can infer that using BaseSpace and DNAnexus have a cost increase (e.g., 56% to 82% for storage) when compared with deploying bioinformatics tools directly on Amazon AWS or using Galaxy. However, it is important to understand that most end-users of sequence analysis are biologists and biochemists, which normally do not have enough expertise to build such infrastructure directly.

The table also shows that all infrastructures are quite flexible in supporting custom workloads, scripts and data sharing. In this sense, these systems are leading the migration of computing to where data is [98] and avoiding expensive local IT infrastructures. One of the main advantages of cloud-based solutions is that there is no need for a manual and time-consuming data-transfer step each time a workload is executed. After uploading data to the cloud once, it is already up in there, accessible from anywhere.
3.4 Final Remarks

In this chapter we provided an overview of the state of the art in integration of biobanks and sequence repositories. Table 3.2 presents an overview of the main characteristics of the initiatives we discussed. More specifically, we compare the integration method employed in these systems (see Section 3.2.2), how the stored data can be accessed, if the managed datasets are public, private or controlled and if dependability measurements are employed in these systems.

<table>
<thead>
<tr>
<th>Initiative</th>
<th>Integration Method</th>
<th>Accessibility</th>
<th>Datasets</th>
<th>Dependability</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSDC</td>
<td>Data warehouse</td>
<td>web</td>
<td>public(^1)</td>
<td>3 replicas</td>
</tr>
<tr>
<td>UKBiobank</td>
<td>Data warehouse</td>
<td>web</td>
<td>private</td>
<td>in-site</td>
</tr>
<tr>
<td>BBMRI-ELIXIR</td>
<td>Mediator-based</td>
<td>web</td>
<td>public</td>
<td>in-site</td>
</tr>
<tr>
<td>GenoEUtwin</td>
<td>Mediator-based</td>
<td>web</td>
<td>private</td>
<td>in-site</td>
</tr>
<tr>
<td>COMMIT</td>
<td>Data warehouse</td>
<td>web</td>
<td>private</td>
<td>in-site</td>
</tr>
<tr>
<td>Cloud-based</td>
<td>Data warehouse</td>
<td>web + API</td>
<td>controlled(^2)</td>
<td>AWS-based</td>
</tr>
</tbody>
</table>

\(^1\) Datasets can be kept private temporarily until a paper publication.
\(^2\) Even private datasets are accessible to the cloud provider (i.e., AWS).

Table 3.2: Comparison of the biobank integration initiatives discussed in this chapter.

As can be seen in the table, all initiatives employ either the data warehouse approach – a centralized infrastructure where the information is stored – or the mediator approach – where a central entity (e.g., a hub) is used as an intermediate to access distributed databases.

All in all, through an analysis of these systems it is possible to observe some interesting trends:

- An integrated accessioning number system is very important, but at this point it appears there is no evolved protection against duplicate entries, sequences or individuals;

- All systems make their datasets available through web portals, that can be either freely accessible (e.g., if the data is public) or implement authentication and access control mechanisms to give access to certain datasets only to authorized users.

- All mediator-based systems employ dependability mechanisms only in the end points (in-site). INSDC replicates all data in three globally distributed replicas, while UKBiobank uses two facilities to store only physical samples. Cloud-based solutions rely on transparent replication and recovery mechanisms offered by AWS.

- All cloud-based solutions focus in approaching computation to where data is located. This can bring dramatic improvements in terms of bioinformatics workflow performance since data sets upload, download and even normalization can be avoided.

- It is interesting to observe that most practical repositories do not provide any kind of semantic integration. It appears such kind of integration is usually employed only locally, on downloaded datasets.

In the next chapter we provide an overview of our design for the OVERBANK, the data integration infrastructure being developed in the BiobankCloud project.
Chapter 4

OVERBANK Storage Architecture

Chapter Authors:
Alysson Bessani, Ricardo Mendes, Tiago Oliveira, Ricardo Fonseca, Fernando Ramos, Nuno Neves (FFCUL).

4.1 Introduction

As discussed in Chapter 3, there is a lot of work in syntactic, structural and even semantic integration of bioinformatics, including biobank, information. However, the current infrastructures used for sharing such data have some limitations when compared with current solutions. First, while we (as end-users) share information using simple directories in personal cloud storage services such as Dropbox [55], most infrastructures for sharing bioinformatics datasets are either based on uploading and downloading of such data from public repositories, or require the acquisition of privacy-controlled data directly from their owners. Furthermore, the present solutions are either centralized repositories for public datasets supporting published work or completely closed systems storing private non-shareable information. Neither of these solutions are adequate for integrating biobank clouds, since biobank-managed data can be either public, controlled (i.e., can be shared with selected parties) or private.

In the BiobankCloud project we aim to solve this problem through the development of a novel infrastructure for sharing data with different privacy requirements and criticality levels between BiobankCloud platform deployments. Such infrastructure will be based on a distributed file system abstraction, which is adequate for this problem for several reasons:

1. The fact we are dealing with raw files, and thus implementing integration at system level, allows the reuse of any already existing solution for syntactic, structural or semantic interoperability [124], which will operate on files managed by the system. In particular, the MIABIS data model [109] (described in Section 3.2.1) can be implemented over the shared infrastructure.

2. Most tools employed in bioinformatics workflows work with files following de-facto standard formats (e.g., fasta for aligned sequences). Integrating biobanks at file level allows one to reuse these tools, without requiring further integration with our infrastructure.

3. The popularity of Dropbox and other personal cloud storage services has shown that a shared file system namespace (i.e., a directory tree) is indeed the most simple and intu-
itive way to share information among a set of users. We envision our infrastructure as a Dropbox-like service for life sciences.

4. Under the file system framework it is possible to implement several access control and location policies taking into account different user roles, sharing groups or even the private or public nature of some datasets. All these requirements can be easily integrated with directories with different access permissions.

5. Similarly, a file system abstraction makes possible to transparently store files in local repositories, private or public, in the repositories of multiple peer biobanks or even in public clouds. Furthermore, data can be moved between these locations without the user having to worry about them or even noticing the data transfers.

Figure 4.1 presents an overview of the proposed infrastructure, called OVERBANK. As a starting point, this infrastructure considers two types of organizations: biobanks running the BiobankCloud platform (called biobank clouds) and public storage clouds. The idea of having public storage clouds as first-class participants of the infrastructure is to allow it to scale up to meet the demanding storage requirements of bioinformatics data. The implementation of OVERBANK requires three main features, which corresponds to the bulk of work being done in workpackage 4. First, a secure, reliable and highly efficient communication backbone needs to be devised for exchanging the large volumes of information between the federation of biobanks running instances of the BiobankCloud PaaS. The second feature is related to the use of public clouds for storing biobank data. Given the non-negligible number of service outages [44] and security incidents [51] affecting cloud providers, we want to store data in multiple cloud providers, implementing a cloud-of-clouds storage for biobank information [34]. The last required feature, not represented in the figure, is the shared file system abstraction enveloping the two previous features, making such complex system as simple to use as Dropbox.

In the Figure 4.2 we illustrate the two main subsystems implementing OVERBANK: the RANC communication infrastructure and the Charon cloud-backed file system. The former provides an efficient communication backbone for integrating federated biobanks while the latter implements both the file system abstraction and the cloud and cloud-of-clouds storage protocols.

A key issue on implementing the file system abstraction for OVERBANK is where to store the file system namespace with the associated files’ metadata. More specifically, any file system
needs to store and manage both the file contents (the file data) and the metadata associated with the file, i.e., the name, location, parent directory, time of creation and permissions associated with the file. Furthermore, a tree describing the shared directories that store such files needs also to be shared among the clients mounting the file system. Figure 4.2 also illustrates how data and metadata are managed by our infrastructure. In the figure it is possible to see a namespace tree with six nodes: directories d1 and d2 and files A, B and C and D. Charon maintains all the namespace tree, together with the files’ metadata, in the cloud-of-clouds storage. The rationale for this is to keep the file system structure secure and available by exploiting (1) the extremely high availability of the cloud-of-clouds (which is expected to be much higher than the availability of any biobank or individual public cloud) and (2) the secure protocols for cloud-of-clouds storage developed in the TClouds project [34]. The idea is to maintain only soft state\(^1\) in Charon clients and store all information about the system organization on the clouds.

Although the metadata is fully stored in public clouds, different file contents may need to be stored in different locations to comply with security and privacy requirements affecting sequence and clinical data (e.g., [113, 40]). This is true even considering the inherent security of storing data in the cloud-of-clouds (see Section 4.2.3), since the main reason why private data should not be stored in such environment is not technical, but legal: the current European legislation does not allow privacy-related information to be stored in countries that do not comply with certain restrictions.

In order to operate under these constraints the OVERBANK must be able to store file contents transparently in different locations. Figure 4.2 shows that file B is stored in the cloud-of-clouds (whether or not it is shared) while file A, which cannot leave biobank 2, is stored locally in this biobank. File C, on the other hand, is private, but shared by biobanks 1 and 2, thus being stored\(^1\)That can be reconstructed after a crash by fetching data from other sources.
in these two biobanks, but not in the cloud-of-clouds. This scenario may be useful, for example, for datasets that need to comply with certain regulations, but can nevertheless be shared between biobanks that are compliant with such regulations (e.g., in the same country). Finally, Charon also supports the storage of non-critical and non-privacy-sensitive files in a single public cloud, as done with file D in the figure.

In the following sections we describe a preliminary architecture of Charon and RANC together with some state-of-the-art techniques we aim to explore to implement these subsystems efficiently.

### 4.2 Charon Architecture

#### 4.2.1 Background

A file system is responsible for organizing the users’ data and to store/retrieve such data to/from storage devices (e.g., hard disks, file servers and storage clouds).

As explained before, there is a number of reasons for choosing a file system abstraction to make biobanks share. Besides these reasons explained, there is another motivation for this choice. In the TClouds FP7 project [27] it was show that it is possible to build a secure and performant cloud-of-clouds file system. We are bringing this experience to the BiobankCloud through the evolution of SCFS, a Shared Cloud-backed File System for Clouds and Cloud-of-Clouds [35]. This system, which was started in TCLOUDs and is actively being evolved in BiobankCloud, implements some techniques to improve the management of data stored in the clouds. However, SCFS does not cover all the needs of OVERBANK, namely:

- SCFS is not optimized for dealing with big files. Given that the files used by biobanks are relatively large (in the order of a few GBs) and given that SCFS has been developed to deal with small files (less than 50MBs), there is a need to develop new strategies to support big files efficiently.

- One of the main concerns of SCFS is the version consistency in shared files. To achieve strong consistency [85], SCFS utilizes a consistency anchor, a coordination service deployed in computing clouds. This coordination service is used to store the file system’s metadata and to coordinate concurrent writes over the same file by different clients. The use of a consistency anchor allows SCFS to provide controlled file sharing and strong consistency over the stored data, even if the underlying clouds only guarantee eventual consistency [134]. However, the coordination service deployment increases the costs associated with the usage of this kind of systems. Since consistency is not a main concern of this project, as shared files are not usually changed, we need to redesign the way file system’s metadata is maintained and the way data writes access data in order to decrease usage costs.

- SCFS stores all file system data in the clouds. Since we envision the existence of data that should never be stored in clouds, we need to investigate mechanisms that transparently support the sharing of file data both in local storage devices and in the cloud.

The Charon file system uses some principles employed in SCFS and extend them to address these limitations.
4.2.2 Charon Overview

Charon is a file system that allows users to share data in a controlled and secure way. We use FUSE [12] (Filesystem in Userspace) to help developing Charon. FUSE is a library that allows the development of a file system in user space, without requiring modifications on a operating system kernel. As can be seen in Figure 4.3 (bottom), we specify a local empty folder in which the shared namespace will be mounted. After that, the VFS (Virtual File System) layer of the OS kernel receives all operations made in that folder and redirects them to the FUSE module, which finally delivers it to the Charon application.

Figure 4.3: Charon architecture.

Charon uses the public storage clouds to save its metadata. Charon adopts the cloud-of-cloud paradigm when storing its metadata in the public storage clouds in order to ensure its privacy, integrity and availability. It is also capable of saving file data in public clouds – either in a “single” public cloud or in a cloud-of-clouds – depending on the criticality of the data. However, files that cannot be stored in the clouds are transferred directly between the respective Charon users via RANC channels (see Section 4.3). Those files are maintained in a Local Repository, which can be located in the machine’s local disk or in a more complex and reliable storage system attached to the client (i.e., biobank), such as a secure private cloud or a trusted file server. RANC channels provide a highly reliable and efficient mechanism to transfer data among users. We also implement various levels of cache to increase the system performance and, in some cases, to decrease the cloud usage costs. These caches are used for maintained both data and metadata, as will be explained in Section 4.2.4. Finally, in order to coordinate concurrent accesses to the files’ metadata and shared files we employ coarse-grained locks.

4.2.3 Cloud-of-Clouds Data Store

This section describes how our system guarantees availability, integrity and confidentiality using public clouds. Availability ensures that, under certain conditions (disk failure or even cloud
failure), data can always be retrieved; integrity ensures that the data retrieved from the cloud will be the same that was written before (e.g., unauthorized changes on a file will be detected); and confidentiality ensures that no unauthorized party – including the cloud storage provider – can access the contents of the stored data.

Availability is achieved by storing all data in a set of public clouds instead of only in one cloud, as most of other cloud-backed storage systems do \[135, 25, 26, 126\]. Since we want to minimize trust on cloud storage providers, we replicate data using Byzantine quorum protocols \[96\]. These protocols require a set of \( n = 3f + 1 \) clouds, in which \( f \) can be faulty. Additionally, the protocols use quorums of any subset of \( n - f \) clouds to replicate the data. To avoid the storage cost of having \( n \) copies of the data (e.g., an overhead of 300\% for \( f = 1 \)), our solution also employs RAID-like techniques, such as erasure codes (see Appendix A) to store only portions of a file on each cloud (e.g., which reduces the storage overhead to 50\% for \( f = 1 \)). To ensure integrity we store a cryptographic hash together with file data. Confidentially is obtained by encrypting the data before storing it in the cloud-of-clouds. This approach is the same introduced in DepSky to store data securely in a cloud-of-clouds \[34\] (also inherited from the TClouds project).

Figure 4.4 presents the process undergone by a file when it is stored in four clouds. First a cryptographic key is generated (step 1), and then the data is encrypted using it (2). Afterwards, the encrypted data is erasure coded, i.e., four different blocks are generated from it (3). The cryptographic key goes through a secret sharing scheme \[120\] to allow sharing data without a key distribution mechanism (4). This process generates four key shares in such a way that at least two of them are needed to reconstruct the key. The quorum protocol is used to store these shares and file blocks in different clouds. To reconstruct the original data two correct clouds need to be accessed for obtaining the key shares and the encoded blocks.

![Figure 4.4: Storing data securely in the DepSky cloud-of-clouds.](image)

It is important to stress that, as already mentioned, the price of maintaining a system like this (using a cloud-of-clouds with four different clouds) does not imply paying four times more storage space. In fact, to maintain the cloud-of-clouds-backed storage system explained above, the price increases only 50\% when compared with a single cloud scenario. This happens due to the use of erasure codes: each encoded block has half of the size of the original data and we only need to store blocks in a quorum of (three) clouds. We consider such overhead affordable for critical sequence data, specially when consider the high compression rates achieved by modern algorithms for such kind of data (e.g., \[136\]).
4.2.4 Caching

Maintaining a local cache is very important to obtain a good performance in cloud-backed file systems, specially when the purpose is to deal with big files [35]. Cloud accesses usually imply huge latencies, making the usability of any cloud-backed file system an important concern. In this section we explain how Charon deals with the high latencies associated with remote data access.

We opt for caching as much data and metadata as possible to make the file system operates over cached files most of the time. All data and metadata modifications are enqueued to be sent in background to the clouds after the close of an updated file. This design choice allows us to hide the latency of data uploads, thereby improving the user-perceived performance. On the other hand, our system loses consistency as the client cannot know the exact moment that the file reaches the cloud storage. To validate the cache, i.e., to know whether a cached file has the latest version or not, we use a hash that is also stored in the clouds. Basically, if we fetch from the clouds a hash that does not match the cached file, the system knows that there is a new version of the file available in the clouds. Thus, if sufficient storage is available locally for caching, we only require to download from the clouds when a new version is available. With shared files this only happens if they are modified by another client. With private files – maintained in the biobank local repository – this only happens if there are two mounted file systems with the same credentials. Note that regarding private files (the files that are not shared), it is expected to have all files in the cache. This implies that the performance of Charon for this kind of file should be similar to the performance of local file system.

We also maintain a small main memory cache to improve subsequent data accesses. This cache’s aim is to keep open files’ data. First, the data is read from disk (or from clouds if a new version is available) and pushed to memory when a file is opened. Therefore, subsequent writes and reads will operate in main memory. Finally, all modifications are saved in disk when the file is closed or synchronized. Additionally the data is deleted from main memory. Both main memory cache and disk cache implement LRU policies, where in the event of a cache being full the oldest data is eliminated to create space for new data.

Figure 4.5 illustrates the data flow between different cache locations. As mentioned before, clients only perform reads and writes in main memory. Memory data is written to disk, and later, it is written to the clouds. As the figure shows, it is not allowed for data to be written directly from memory to clouds. This choice makes possible the recovery of data from disk if a failure occurs during the upload to the clouds. Likewise, when data is obtained from the clouds, it is first stored in disk and only afterwards in the memory cache.

![Figure 4.5: Charon’s cache hierarchy.](image)

The three data locations explained above (memory, disk, and clouds) have different levels of durability. Table 4.1 shows the fault tolerance provided by each one of them, together with the expected storage latency and examples of POSIX calls that cause data to be stored at the different levels. As can be seen, the data is written in main memory using the write system call.
In this case, the access to the data is very fast, however, it does not provide any kind of fault tolerance. The fsync system call can be used to flush the data (if modified) to the local disk. At this level, process and system crashes are tolerated. At the cloud-of-clouds, the storage latency increases significantly, however, the system can survive not only disk failures, but provider failures as well.

This cache not only brings an important performance increase, but it is also very interesting in terms of costs. When the files are stored locally, it is necessary to fetch their data from the clouds only if they are modified by another client. Moreover, any modification on data or metadata represents an upload. Since the download of data from the clouds is much more expensive than upload (which is usually free in most clouds) [1, 14, 32, 24], the operational cost of using such system will be affected only by to the number of updates and the amount of stored data.

### 4.2.5 Sharing with Charon

Charon uses multiple clouds to store the file system’s metadata. This decision comes from the need of this information to be always available and accessible to all Charon’s clients. Furthermore, Charon can also store file contents in the clouds to allow clients to share it.

Charon must be able to manage different sharing groups. It ensures that each member of a group is able to access all files shared with it, but is not able to access files from other groups. For example, if Alice shares one folder with Bob and another with Caroline, then neither Bob nor Caroline should be able to access each other’s files (in the absence of a group shared between the two). To make this possible we organize group data and metadata by namespaces, and store them in the cloud-of-clouds. There are two kinds of namespaces: personal namespaces (PNS) and shared namespaces (SNS). Each client has its own PNS that stores all of its private files’ data and metadata. For each sharing group, there is one SNS that stores all metadata and data of all the files shared between the members of that group.

Figure 4.6 depicts how information is organized and stored in the clouds both for a PNS and a SNS. In the figure it is possible to see that there are three kinds of information stored in the clouds: the files’ metadata, the files’ data and a hash file. As can be seen, each client has its own PNS, where the metadata of private files is stored (e.g. files, folders and links) together with their data. We can also observe the organization of one SNS (in the figure, shared by client A and client B). In SNS1 we can see the same kind of information stored in the PNSs (file data and metadata) plus a hash file. This file stores the cryptographic hash of the last version of each file in the SNS1. This is not needed in a PNS since all files are private and only their owner updates them. Consequently, the cryptographic hashes can be put together with the files’ metadata in the PNS, while the metadata of files in SNS points to the hash file storing its cryptographic hash.

It is important to note that in the case of a PNS, the metadata of all private files is only downloaded from the clouds when Charon is mounted by the client. As mentioned before, in the case of file data stored in the clouds, such download only occurs if the local copy of some
file data was erased or if the file was updated in other client (e.g., a user mounts Charon on her machine at home and creates a file, and later updates the file on a different machine at work).

On the other hand, in the case of SNSs, the metadata is periodically downloaded from the clouds. This allows users to view updates on the SNS such as the creation, deletion or changes in files’ metadata. The hash file is only downloaded when a client opens some file (either for read or write) and the previously read corresponding hash version has already expired.

We use locks to avoid concurrent updates over file system data and metadata made by different clients. Therefore, when a client wants to update some shared file it must first obtain a lock over the SNS that contains the file. As an initial solution, Charon uses the lock algorithm associated with DepSky [34]. However, we expect to devise a more efficient protocol in future versions of the system. All locks have an expiration time in order to ensure the liveness of the system despite client failures. In this way, when some client obtains a lock over some SNS, it is able to perform locally all operations the user wants during the lock’s validity period, while the updates are uploaded asynchronously to the clouds before the lock expires. This approach allows Charon to achieve good performance even when it is operating over shared files. When a client has no lock over the SNS, it can still perform read-only operations over all files stored in that SNS.

With all this in mind, when an open is performed over a file, Charon performs these three main steps: (i) read the file’s metadata, (ii) optionally create a lock if the file is opened for writing, and (iii) read the file data into the local cache. The read and write operations only need to interact with the local storage. Writing to a file requires updating the memory-cached file and the associated metadata cache entry (e.g., the size and the last-modified timestamp). Reading only causes the data to be fetched from the main memory cache (as it was copied there when the file was opened). When a file is closed, it involves the synchronization of cached data and metadata with the clouds. First, the updated file data is copied to the local disk and, after that, to the storage cloud. Then, if the cached metadata was modified, it is pushed to the clouds as well. Lastly, the file is unlocked if it was originally opened for writing.

Figure 4.7 shows how Charon references all information stored in the cloud-of-clouds. As
can be seen, each Charon’s client knows the location of its PNS’s metadata, which stores the location of all SNSes the client has access to. Each SNS’s metadata maintains the locations of all files data in the clouds, as well as the hash file that contains the cryptographic hashes of the latest version of each file’s data.

![Figure 4.7: Relationship between a PNS and a SNS in Charon.](image)

Notice that, if there are a lot of files in a SNS, the hash file could grow indefinitely. To solve this problem, for each SNS there may exist multiple hash files. The location of the hash file containing the hash of the last version of a given file is stored together with this file’s metadata.

### 4.2.6 Garbage Collector

During normal operation, Charon saves new versions of file data without deleting the previous ones, and files removed by the user are only marked as deleted in the associated metadata. These two features enable the recovery of old versions of the files, which is useful for some applications. However, this feature increases the monetary cost of running the system, and therefore, Charon includes a flexible garbage collector (similar to the one used in SCFS [35]) to enable various policies for reclaiming space.

Garbage collection runs in isolation at each Charon client, and the decision about reclaiming space is based on the preferences (and budgets) of individual users. By default, its activation is guided by two parameters defined upon the mounting of the file system: number of written bytes $W$ and number of versions to keep $V$. Every time a Charon uploads more than $W$ bytes to the clouds, it starts the garbage collector as a separated thread that runs in parallel with the rest of the system (other policies are possible). This thread fetches the list of files owned by the user and gets the associated metadata. Next, it issues commands to delete old file data versions from the cloud storage, such that only the last $V$ versions are kept (refined policies that keep one version per day or week are also possible). Additionally, it also eliminates the data versions of the files removed by the user. Later on, the corresponding metadata entries are also erased.

### 4.2.7 Dealing with Big Files

As explained before, one of the key limitations of SCFS is that it does not work well with big files. Consequently, one of the challenges that Charon needs to address is the provision of acceptable performance when dealing with big files.

In a nutshell, maintaining big files in a cloud-backed file systems like Charon brings two main difficulties. First, it is impractical to read (resp. write) whole files (as is done in SCFS [35])
from (resp. to) the cloud since the time required for downloading (resp. uploading) such big files is prohibitively higher. Second, cloud-backed file systems strongly rely on cache for ensuring usable performance [35, 135], and it is difficult to keep these files in cache due to its size. This last problem is specially relevant when considering main memory cache.

We address these issues by splitting (big) files into blocks of at most 16 MB each (the optimal size is still being studied, and will be a system parameter). This technique is already used by other cloud-backed file systems (e.g., [135]). A block of this size is at the same time (1) relatively fast to load from disc to memory, (2) can be transferred to/from the clouds in a reasonable time and (3) is still small enough to be stored in memory for absorbing bursts of sequential accesses (more specifically, 4096 accesses since Linux fetches blocks of 4 kB by default). This feature is also interesting in terms of costs because, after a file is updated, only the modified data blocks need to be downloaded from the clouds.

As an additional measure to improve performance, Charon can run a background thread that is responsible for pre-fetching file blocks that could be used in the future by the application to memory. This “pre-fetching” will increase the performance of the system for sequential workloads, which are quite common with omics data.

We maintain the identifiers of the file’s blocks in its metadata. In the case of hash files, we cannot simply store all file block’s hashes as we do with single-block files since the hash file can grow indefinitely, making it larger and prohibitively costly to read. To solve this issue, we defined a maximum number of hashes stored in each hash file. Therefore, when a file reaches a certain number of blocks, all hashes of these blocks are migrated to a separated hash file. Note that the hashes of the blocks of some file will never be spread over more than one hash file, even if the number of hashes to store is bigger than the hash file’s limit. For example, let the maximum number of hashes per hash file be 50 and suppose we have 40 one-block files. At some point, we add a file that has 30 blocks, resulting in 70 hashes to store. In this case, all the hashes of the new file are saved in a new hash file. However, if this new file grows and reaches 60 blocks, the hash file will kept all these 60 hashes (10 more than the limit). The reason for storing all the blocks’ hashes in the same hash file is to avoid fetching several hash files when opening a big file (when these hashes are verified), which would bring performance issues.

### 4.3 RANC Architecture

The Resilient and Adaptive Network Communication (RANC) channel is an interconnected set of processes that work cooperatively to provide a robust, fault-tolerant and secure communication to applications. The RANC is a follow-up to the overlay network Resilient Event Bus (REB), which was developed within the context of the MASSIF FP7 project [18]. The RANC main features are:

- Peers in the RANC, denominated as RANC nodes, receive data from applications in the form of a stream of bytes, and transmit those bytes via point-to-point communication.

- RANC nodes authenticate themselves before and during transmission of data in order to provide authenticity and integrity of data; confidentiality is also provided on an optional basis.

- RANC nodes transmit data over an underlying UDP/IP network, and can use multiple paths in the physical and overlay networks in order to increase spatial redundancy for communication resiliency (overlay paths have at most one intermediary node).
Erasure coding is applied over the transmitted data in order to reduce the cost of replication by senders and also increase the recovery efficiency by receivers.

Communication paths are periodically probed by source nodes in order to select the fastest and most reliable links during data transmission.

Figure 4.8: Data transmission over the RANC.

Figure 4.8 depicts two facilities that use the RANC for data transmission. Each facility runs a Charon application that directly interacts with a RANC node, potentially in the same computer process, in order to transmit data to another facility over an insecure network (such as the Internet). Two paths are used to deliver data from one RANC node to another, a direct path and an intermediary path that utilizes one extra RANC node as a forwarder (which can be part of an independent facility). By exploring spatial redundancy in the network, transmissions become more resilient to faults and attacks on physical network nodes (e.g., congestion or a DoS attack on a network router) or even on intermediary RANC nodes.

Machines hosting RANC nodes may contain an arbitrary number of Network Interface Cards (NIC), which can be used by the RANC to increase the total number of paths available to a given source node. This form of communication is called multihoming, and its benefit of increasing spatial redundancy works especially well if each NIC in a source machine is connected to a different transmission technology (e.g., wired and wireless), or even connected to a different Internet service provider. In Figure 4.8, the source node has four available paths for transmission (despite only two of those paths being depicted).

RANC nodes can also recover from packet drops or corruptions by employing erasure coding over the transmitted data (see Appendix A.3). Instead of simply replicating the whole data through each path, a sender RANC node encodes it into a number of blocks with total length only slightly larger than that of the original data, and transmits those blocks over the selected paths. The receiver RANC node only requires a subset of the transmitted blocks in order to reconstruct the original data. This coding procedure reduces the size of transmitted data by senders, compared with simple packet duplication over all selected paths, while still providing an excellent recovery capability to receivers in the presence of block erasures, compared with delayed retransmission of lost packets. Figure 4.8 depicts the loss of two blocks during transmission and a nonetheless successful recovery of the original data by the receiver.
Ongoing work is being done to improve the capabilities of the RANC, for instance adding support for multicast, and including the latest generation of erasure codes, the RaptorQ codes [91], for optimal performance. We also envision a change in the fundamental architecture of the RANC in order to support the capabilities of a Software Defined Network (see the next section and Appendix B for a discussion on this new type of network architecture). RANC nodes should become less complex and shift their adaptive mechanisms (route probing and selection of best paths) to Traffic Engineering applications running on top of SDN/OpenFlow controllers. The design of these applications should take into account the measurement of quality of service of multiple paths, and the selection of the fastest and less congested ones for traffic diversion.

4.3.1 Integrating the RANC with Charon

One of the benefits of using the RANC in the communication among biobanks is providing a reliable direct communication channel that does not depend on clouds. For privacy reasons, a biobank may require to confidentially share a file with another biobank, without risking data leakage from the cloud-of-clouds. Alternatively, a direct communication channel should also be able to deliver a large amount of volatile data more economically than using clouds as an intermediary (saving on data transfer costs enforced by cloud providers).

In order to continue using the Charon file system to handle remote files, we plan to use Charon on top of the RANC for direct file transmissions. Those transmissions may have multiple properties, such as having multiple receiver parties (e.g., a file share between multiple users) or being done asynchronously. To keep file handling semantics the same as in Charon, we plan to develop a File Transmission Module to aid in the integration between Charon and the RANC. Figure 4.9 shows a conceptual architecture of this design. The File Transmission Module will abstract the sharing of files among multiple biobanks in the absence of the cloud-of-clouds, receiving file sharing instructions from Charon and interacting with the RANC to send/receive the file data to/from other nodes. To save on transmission costs, the module should also cache frequent requests in local persistent storage.

4.4 An SDN architecture for RANC

The main objective of the RANC channels is to provide a highly reliable, secure, and efficient mechanism to transfer data among users. Sophisticated routing and probing mechanisms are used to achieve these goals. For example, it uses multiple paths and multihoming techniques
for communication resiliency. In addition, communication paths are periodically probed by source nodes in order to select the best network links for data transmission. RANC nodes are therefore complex network elements that need to simultaneously examine network state, make decisions (based on that state) and act (based on the decisions made).

The complexity of network nodes – this need to simultaneously probe, decide and act – is a common problem in traditional computer networks. Routers and switches run complex, distributed control software that, after assessing network state, decide on the best routing paths. Simultaneously, they need to forward traffic. Furthermore, networks are dynamic, and network operators have little mechanisms to automatically respond to network events. It is therefore difficult to enforce the required network policies (routing, traffic engineering, security, and so on) in such a continually changing environment. Software defined networking (SDN) is an emerging paradigm in computer networking that promises to alleviate these problems.

The key enabler for an SDN is the separation of the network’s control logic (the control plane, where decisions of how to handle network traffic are made) from the underlying physical routers and switches that forward traffic (the data plane, which forwards traffic according to the decisions made by the control plane). This separation of the control plane and the data plane can be realized by means of a well-defined programming interface between the switch and the SDN controller The most notable example of such API is OpenFlow [103, 22]. The core component of an SDN is the controller, but the actual logic that implements the desired network policies is defined by the SDN applications that run on top of it. Several applications have been proposed recently to achieve various purposes: server load balancing, traffic engineering, security (access control), network virtualization, switching and routing, among others. This separation of concerns – applications make decisions, controllers implement decisions (in switches’ forwarding tables), and switches forward packets – is fundamental to simplify network management and to maintain the high-level policies in place in such dynamic environment. This ability to program the network in order to control the underlying data plane is the crucial value proposition of SDN. In Appendix B the reader will find a more thorough introduction to Software Defined Networking.

### 4.4.1 Rationale for an SDN infrastructure

The motivation to use an SDN/Openflow architecture to build a secure, reliable and highly efficient communication backbone for the OVERBANK is rooted in its fundamental advantage: the possibility to programmatically control the network. This ability to control relatively simple (and cheap) Openflow switches to control the paths used to deliver traffic is an important means to guarantee the levels of performance and reliability required, at reasonable cost. In an SDN it is possible to deploy routing and traffic engineering protocols – as SDN applications – customized to our unique requirements. RANC nodes can thus become less complex and shift their adaptive mechanisms to Traffic Engineering applications running on top of SDN controllers. This flexibility and customizability will be crucial assets to enable the exchange of large volumes of information between the federation of biobanks running the BiobankCloud platform in an efficient manner.

More than merely integrating the federated biobanks, the RANC communication infrastructure will need to be secure and reliable. A secure and dependable SDN controller [83, 37] can be pivotal in this respect. The deployment of security network applications on top of such SDN controller may allow, for instance, biobanks to define network-wide access control policies, and enforce them directly. An SDN architecture also allows fine-grain monitoring and measurement to generate and impose dynamic security policies. For instance, the ability to classify packet
and perform flexible routing can easily direct a chosen subset of traffic to be dropped or sent through a set of specific middleboxes for specific processing and analysis.

4.5 Final Remarks

In this chapter we introduced the main design choices and subsystems of the OVERBANK storage infrastructure. This infrastructure will be used to interconnect biobank clouds in a secure and dependable way through a Dropbox-like shared file system abstraction. The technology described in this chapter is under development and will be fully described and made available in the 2nd and 3rd year WP4 deliverables, respectively.
Chapter 5

A Privacy-assuring Disclosure Filter for Genetic Information

Chapter Authors:
Vinicius Cogo, Alysson Bessani (FFCUL).

5.1 Introduction

The decreasing cost of sequencing genomes is increasing exponentially the amount of data to be stored and analysed [140], whereas the IT capacity of biobanks and researchers is still increasing linearly. Cloud computing is an economical alternative to local IT infrastructures, since cloud providers have virtually infinite resources that are charged at low prices in a pay-as-you-go model. However, using public clouds increases the attack surface and strengthens challenges on privacy of genomic data and efficiency of bioinformatics workloads.

The main privacy risk regarding genetic information is the disclosure of private information about individuals. A human genome can uniquely identify its owner and reveal information about him and his relatives, even for several generations into the future [49]. In addition, small portions of DNA can drastically reduce the set of candidates to such ownership, compromising the donor privacy.

In this chapter, we present some initial efforts to create a privacy-preserving disclosure filter for classifying DNA segments in public and private. The key idea is, giving a segment of predefined size $s$, such filter will be able to say if there is a chance of this segment to contain some privacy-sensitive information or not. In a nutshell, this work use the available information about how one can be identified using its DNA to build a black list with all possible signatures of such data. Although conceptually simple, the realization of this idea is subject to several challenges:

- How to classify DNA segments in public and private?
- How to implement an efficient filtering solution that supports the high-throughput of modern sequencing machines?

Using this filter, it would be possible to exploit the use of public clouds or shared infrastructures for both storage and processing of DNA information without endangering the privacy.
of its donor. The idea is to have few local resources to deal with the (expectedly) small number of privacy-sensitive sequences and use public or shared computing and storage resources to deal with the non-privacy-sensitive parts. As a starting point we consider the case of hybrid sequence alignment algorithms and the storage of raw and aligned DNA information. For the best of our knowledge, this is the first work that presents a concrete proposal on how to classify and filter DNA sequences considering their privacy sensitivity. Additionally, it contains the first non-cryptographic solution to protect biobanks against referred attacks [92, 112].

This chapter is organized as follows. Section 5.2 presents some background and preliminary work on DNA privacy and privacy-preserving algorithms. Section 5.3 describes the methods we use to build the black list of segments in the filter. Sections 5.4 and 5.5 contain a brief discussion of how the filter is implemented and some results of its experimental evaluation, respectively. The chapter concludes with a discussion about the usage scenarios for the filter within the OVERBANK framework (Section 5.6) and some final remarks (Section 5.7).

5.2 Related Work

Transferring genomic data in plain text is a naive procedure that disregards any privacy concern by exposing genomes and their identification, which still happens in some bioinformatics workloads. Obviously, an adversary only needs to eavesdrop the network between sender and receiver to intercept these identified data in order to compromise the privacy of the individuals in question. Shuffling multiple genomes or inserting noise in communications are basic approaches employed to delude the eavesdropper. Nonetheless, an adversary still obtains all data in clear and might eventually realize which portions are noises and which are valid data about each individual. Biobanks usually employ de-identification of data by removing explicit identifiers (for instance, participant name and any identification document number). However, approaches such as de-identification, de-nominalization, trusted and semitrusted third parties do not guarantee the privacy of DNA owners [95].

Encryption is one of the best known methods to protect data privacy, since data is meaningless without the decryption key, even in a disclosure case. It implies maintaining the encryption and decryption keys securely and increased computational cost in comparison to plain text solutions. Despite advances in the area, cryptographic solutions from today may become insecure in the future. An encryption scheme considered strong today might gradually weaken in the long term, whereas genome sensitivity does not dissipate over time [49]. This is especially true when considering the inception and popularization of disruptive computation models such as quantum computing [125]. Additionally, implementations of cryptographic protocols are not immune to software vulnerabilities [67], and the leakage of cryptographic keys still is possible. A key feature of the disclosure filter proposed in this work is to not rely to cryptography for ensuring privacy. We advocate the approach of taking less risks, processing all privacy-sensitive sequences locally (see next section).

The use of hybrid clouds (private and public) is becoming a common practice in data mining with selective privacy requirements. There are some researches [141, 79] that follow this approach employing public clouds to securely execute part of their workloads. For instance, sequence alignment algorithms map small sequence reads in their most probable positions in a reference genome using the seed-and-extend approach [87]. Chen et al. [141] proposes an algorithm that hashes all reads to perform the seed step in public clouds and the extend step in private clouds. The Sedic system [79], by the other hand, proposes mapping private sensitive sequences entirely in private cloud and common sequences in public clouds. This system is
implemented on a modified Apache Hadoop MapReduce framework [62]. The main limitation of this solution is the fact that they require users to previously mark data as private or public before starting a job. Additionally, they recognize that classifying genomic data is a specific, complex task to which they do not propose a concrete solution. The filter we propose in this work can be used to do this classification automatically.

In the next section, we present our approach to address these challenges through a disclosure filter service and two methods to classify DNA sequences as private or public.

5.3 Disclosure Filter

Modern sequencing machines are able to filter genetic data based on block quality and size, or coding and non-coding regions. We propose a disclosure filter for genetic data that is based on knowledge about privacy sensitivity of small sequences. Figure 5.1 contains a global view of a hybrid infrastructure with the disclosure filter placed between a data set (for instance, sequences coming from NGS reads) and both public and private infrastructures or clouds. We consider that the disclosure filter has input channels that receive DNA sequences and decides which sequences should be forwarded, either to the local private infrastructure or to the public one. The decision is made considering a black list database of sequences defined by statistical heuristics and previous knowledge about a context or population. If a DNA sequence is present in this list, then it is considered as private and must never be send to the public domain.

![Figure 5.1: Global view of a system using the disclosure filter component.](image)

In the remaining of this section we present methods for obtaining data to be added to the filter and discuss some limitations of these methods.

5.3.1 Adversary Model

In this work, we consider that data disclosure can be accidentally or maliciously caused by software bugs, service clients, researchers, legitimate cloud users, external attackers or em-
ployees of cloud providers [52]. Further, adversaries have two main objectives: re-identify to whom a DNA belongs [92], and detect if an identified DNA is member of a study or a set of genomes [106]. In both cases, the access to certain portions of the victim’s DNA is fundamental for performing the attacks.

5.3.2 Populating the Filter: Algorithms

Populating the disclosure filter with DNA sequences is a similar procedure as filling a black list of strings. We initially implemented two populating methods based on (1) genetic genealogy profiling and (2) rare variants present on individuals. The former aims to protect the dataset from re-identification while the latter aims to disallow the verification of a known DNA is part of some dataset. This covers the objectives from our adversary model present in the previous subsection.

A Method Based on Genetic Genealogy Profiling

The first method for obtaining entries for our black list of sequences is based on genetic genealogy profiling, which is also employed in forensic identification, paternity tests, missing persons investigations, among others. A genealogical profile from paternal lineage is a set of counters of how many times each selected short tandem repeat from chromosome Y (Y-STR) appears in an individual. Short tandem repeats (STR) are small strings comprised of A, C, G and T characters, for instance, the Y-STR called DYS392 is represented by [TAT]ₙ and an individual who contains in his DNA a sequence like cgacTATTATTATTATtgca has in his profile the number 4 for DYS392.

There are registers of several known STR available in public databases such as the STR-Base [108, 115] and the TRDB [63]. In the United States, a core set of 13 STR markers are being used to generate a nationwide DNA database [42] called FBI Combined DNA Index System (CODIS). Other countries and organization such as EU, UK, DE and Interpol also selected a set of core STR markers to identify individuals. Public databases have thousands of registered STRs, much more than these core STR. STRs can also suffer from mutations and the responsible databases store all variations found until today.

The Y chromosome is important because fathers pass their exactly copy of this chromosome to their male sons. It is used to preform genealogical analysis since an individual with a specific profile, has exactly the same profile as his father, which is used to perform paternity tests.

In early 2013, Gymrek et al. published an article [92] describing an attack that re-identifies participants of the 1000 Genomes Project [47]. It is based in two facts: surnames are paternally inherited in most human societies, as well as Y-STR in male individuals [81]. The attack has two goals: (1) obtaining the surname of individuals and (2) triangulating their identity. For surname inference, they profiled Y-STR of individuals, queried these profiles in recreational genetic genealogy databases and obtained a list of possible surnames for the profiles in question. Each Y-STR profile contains registers about 30 known Y-STR in this case. Authors queried the Y-STR profiles in the YSearch [64] and the SMGF [61] databases, and recovered the correct surname in 12% of cases (with 82% of confidence).

For triangulating identities, authors combined the obtained surnames with age and state, which are considered public information that do not need to be suppressed in anonymization process. A query on U.S. census by year of birth and state results in 60,000 U.S. males in 50% of cases. Aggregating the surname to the query shrinks the result to only 12 males. Each surname inference breached the privacy of nearly 16 individuals. From the number of entries
existent in those databases, one can identify millions of U.S. males. Although the result of 12% appears to be not impressive, it means that from the 1092 participants of 1000 Genomes project, 131 of them will never recover their privacy and nearly 2100 participant’s relatives had their privacy breached by this attack [92].

In our approach, we can protect biobanks against this attack by registering information about all known Y-STR in a black list. It avoids the disclosure of privacy-sensitive sequences to the public cloud and allows their local analysis and storage. We need to aggregate the following information for each known Y-STR:

1. The Y-STR repeat regular expression. E.g.: \[TAT\]ₙ for DYS392.
2. Minimum and maximum number of repetitions already observed. E.g.: 6-17 (for the above STR).
3. All known mutations of the STR [58, 69].
4. All observed left and right flanking sequences, which are commonly found either before or after the STR. E.g.: 5’-TAGAGGCAGTCATCGCAGTG-3’ is a primer sequence observed before DYS392 and 5’-AAGGAATGGGATTGGTAGGTC-3’ after DYS392.

As a STR is a repetition of a small string, the number of different possible combinations is initially equal to the size of the repeated pattern, \(PS\). A read can start with each different letter from the string, which in the case of DYS392 creates three possibilities \((PS = 3)\) for base sequences, with strings starting with TAT, ATT or TTA. Supposing the reads contain 10 base pairs \((s = 10)\), any read with this size matching entirely this entry should be only TATTATTATT, ATTATTATTA or TTATTATTAT. \(PS\) new base sequences are created for each known mutation (the number of mutations is represented by \(M\)), which increases the number of base sequences to \((M + 1) \times PS\). Left and right flanking sequences are concatenated with each base sequence creating all possible combinations, which we call long sequences. Each long sequence is composed by a left flanking sequence, a base sequence and a right flanking sequence. There are a total of \(FL \times ((M + 1) \times PS) \times FR\) long sequences (represented by \(TLS\)) for each STR analyzed, where \(FL\) and \(FR\) are the number of known left and right flanking sequences, respectively. For each long sequence with size \(LS\), we create \(LS - 1\) filter sequences with \(S\) nucleotides each.

Figure 5.2 contains an illustration of the entire process described previously. One specific implementation detail is that we ignore portions of flanking regions that could lead to the creation of a sequence entry to the filter composed only by N characters. It is important for two reasons: (1) a sequence block composed only by N characters will never disclosure private information about individuals, and (2) almost 10% of our genome still contains blocks with this letter because sequencing and alignment solutions can’t determine precisely which nucleotides should be placed in that positions. Note that the item 2 would lead to increasing the amount of private sequences in almost 10% unnecessarily.

A Method Based on Rare Variants Present on Individuals

The second method for creating entries to our black list of sequences is based on rare variants present on individuals. Humans are 99.5% genetically similar one to another, however small portions of the remaining 0.5% can uniquely identify to whom a DNA belongs [49]. There are studies called allele-frequency analysis (e.g., [65, 48]) that roughly identify how common or rare the sequence variants of an individual are in comparison to a specific population. Genome-wide association studies (GWAS) correlates several traits with these genetic variants that are
common in a population (e.g., [138, 74, 73]). In 2009, Wang et al. published an article [112] proving that it is possible to acquire knowledge about known individuals from statistical results publicly released by GWAS studies. More precisely, the attacker is assumed to have a blood sample of the victim and genotyped as few as a couple hundreds of his/her SNPs. Then, the attacker wants to determine the victim’s presence in the GWAS group, which indicate his/her contraction of a disease. This article is based on another work published one year before by Homer [106], which shows a similar attack to other common techniques employed in genetic studies, for example microarrays.

To avoid any possibility of such attacks to succeed, one needs to define what is a rare variant within a population. The 1000 Genomes Project classifies variants (mutations) in four groups [48]: very-rare (<0.1%, i.e., a sequence that appears in less than 0.1% of the population), rare (0.1 - 0.5%), low-frequency (0.5 - 5%) and common (>5%). Another article [118] classifies sequences in two categories: rare (<1%) and common (>1%).

Table 5.1 shows (among other things) some results of an allele-frequency analysis on all SNPs of 1092 genomes available from 1000 Genomes Project (v3.20101123).

We consider as dangerous any genome position that contains an allele frequency smaller than the threshold specified by the classification from 1000 Genome Project about how rare a sequence is. Based on this analysis, we can state that if we filter all very-rare SNPs, then the filter will have to deal with approximately 12.5 million dangerous positions, and this number increases as we consider more common variants. For example, considering to filter any common or rarer SNPs, the disclosure filter will have to consider 37.7 millions positions.

The evolution of this distribution face to the total of SNPs from 1000 Genome Project is
### Table 5.1: Allele frequency and number of dangerous positions in genome variants in the 1000 Genomes Project dataset [48], together with the estimated number of entries on the black list for different categories and sequence/read sizes. Legend: $M = 10^6$ and $B = 10^9$.

<table>
<thead>
<tr>
<th>Category</th>
<th>Allele Frequency</th>
<th>Dangerous positions</th>
<th>Entries ($s = 30$)</th>
<th>Entries ($s = 500$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very-rare</td>
<td>$\leq 0.1%$</td>
<td>12.5M</td>
<td>377M</td>
<td>6.3B</td>
</tr>
<tr>
<td>Rare</td>
<td>$\leq 0.5%$</td>
<td>22.5M</td>
<td>674M</td>
<td>11.2B</td>
</tr>
<tr>
<td>Low-frequency</td>
<td>$\leq 5%$</td>
<td>31.6M</td>
<td>949M</td>
<td>15.8B</td>
</tr>
<tr>
<td>Common</td>
<td>$\leq 50%$</td>
<td>37.7M</td>
<td>1.1B</td>
<td>18.8B</td>
</tr>
</tbody>
</table>

Presented in Figure 5.3. This graph shows the cumulative nature of SNP categories, where the Common category includes its cases together with those from the Low-Frequency category, which by its turn includes those from the Rare category, and so on.

![Figure 5.3: Cumulative histogram of all SNPs from 1092 genomes [48] based on their allele-frequency. The x-axis is in logarithmic scale.](image)

After obtaining the desired level of filtering, which can vary for each scenario, we must to convert a dangerous position in dangerous sequences. It is done by joining dangerous positions to all $s - 1$ neighbors placing this character in all position, where $s$ is again the size of sequence blocks to be added in the filter. The size of the filter will be the number of dangerous positions $N$ multiplied by the block size $(N \times s)$. It will cause the filter to receive the expected number of entries presented in Table 5.1.

### 5.3.3 Discussion

A key limitation of our disclosure filter is that entries for the black list are based on what is currently known about DNA sequences and its capability of identifying their owners. In some sense this problem is quite similar to the building of anti-virus software malware signature lists or pattern-based intrusion detection. It is possible to avoid attacks that you know about, but not
others. In the following we consider specific limitations of each method we use for creating entries to the disclosure filter.

The method based on genetic genealogy profiling has the same problem as any solution based on black lists, since all unknown cases will not be caught during the filtering, causing a false negative. A possible way to avoid such problems, specially when considering the number of repetitions of a STR, would be to use a pattern matching algorithm instead of having all possible repetitions in a black list. However, this would make the classification of sequences in public or private much more complicated than a simple lookup in such list. Furthermore, it does not solve the problem of new STR sequences being discovered.

The method based on rare variants within a population is as complete as the size of the population considered in the frequency study used to generate entries and the relationship of the genetic sequences with such population. Note that the probability of an attack based on membership detection [106, 112] to succeed is directly proportional to the number of SNPs obtained from the victim by the attacker [133, 116, 76]. At the same time, it is inversely proportional to the number of individuals composing the study dataset. Authors of scientific studies can reduce the risk of membership detection by limiting the number of allele frequencies available in publicly disclosed information. Obviously, if all SNP information is removed, then attacks must never succeed (because the number of SNPs obtained by attacker will be zero), however it would not be interesting for data sharing in science. We still have to better understand the limits of membership detection attacks when facing our disclosure filter, which do not remove all information but reduces it in a controlled way.

As the population on the filter increases, the frequency of certain variants can increase or decrease. If the frequency increases, a dangerous position may become non-dangerous since its number of cases increases to a certain balanced point compared to other alleles in that position. If it decreases, a non-dangerous position may become dangerous since the number of cases from other alleles increases while the number of cases from the allele in question is maintained. If new variants are discovered they can be added to the allele frequency analysis, in order to create a new data set as input to the filter. There is the need of creating a new filter with the new data set in any of previous adaptation cases.

Notice that, in the same way as anti-virus software, the disclosure filter can be updated with new entries created using the same heuristics discussed in this section if new privacy-sensitive information about the DNA is made available. Furthermore, other methods for generating entries can be implemented in the future.

Another limitation of our methods for building the filter is that the number of entries is dependent on the size $s$ of the sequences we are considering. As a compromise solution, we consider that the use of small sequences of 30 base pairs is a good decision because (1) it is adequate for classifying reads from high-throughput machines that generate reads of this size and (2) can be used with larger sequences if they are broken in sub-sequences of 30 bps. The big question is related with the size and performance of a practical filter. In the next sections we try to investigate this question with an implementation of the disclosure filter.

5.4 Implementation

After building the black list of sequence to use on the disclosure filter, a second challenge that needs to be addressed is how to implement this filter in an efficient (in terms of space and performance) and effective way. In terms of performance, we intent to achieve high throughput (i.e., high number of classifications per second) through employing good engineering techniques.
from distributed and parallel computing areas. Such metric is important because our goal is to support the throughput from multiple existent next-generation sequencing (NGS) machines or even at least one of future, faster solutions. Since NGS machines are really expensive, their owners aim to maintain them always working on sequencing data, which leads to provide the filter as a service that support pipelined workloads.

The spatial axis is also important to our scenario since the amount of sequenced genetic data is growing exponentially. We employ the Bloom filter data structure [36] as the selected solution to our filter component. Bloom filters are probabilistic space-efficient data structures that maps entries of any size to a reduced number of bits resulting from multiple hash algorithms. Checking if an entry is member of a set, using this solution, does not cause false negatives and has a configurable rate of false positives. It roughly returns false if an entry definitely does not belong to the set or true if it probably belongs to the set. The expected amount of false positives tolerated has a direct consequence on the size of the filter. Bigger filters have less probability of returning false positives, while smaller filters have more probability of returning wrong information. Notice that false positives does not affect the privacy guarantees of the filter, only its effectiveness. More precisely, a false positive implies in a public sequence being classified as private, and thus it needs to be dealt as privacy-sensitive (e.g., stored or processed locally instead of using a shared infrastructure).

In the next section we study the tradeoffs considering different configurations of several disclosure filters.

**Bloom filter details.** An interesting aspect of our implementation is that, due to the large number of entries in the black list, we will be using an unconventionally bigger Bloom filter, with several gigabytes of size. We choose a Bloom filter implementation called JavaLongFastBloomFilter [17], which is a bigger and faster solution than most Bloom filter implementations. Bigger because it uses numbers of long type (64-bits) to index the bit set of Bloom filters, while others still use numbers of int type (32-bits). Faster for two reasons. One, it uses a 64-bit Murmur Hash, which is one of the fastest non-cryptographic hash functions with good random distribution of regular keys. Two, it has an algorithmic optimization that allows reducing (by configuration) the number of hash keys needed to index an entry by increasing the Bloom filter size until a configurable size (by default, 20%).

### 5.5 Experimental Evaluation

In this section we present some experimental results considering the implementation of the disclosure filter based on Bloom filters discussed in the previous section.

#### 5.5.1 Experimental Environment

Our experimental environment is one physical machine that runs all disclosure filter components. This machine is a Dell PowerEdge R410 server, equipped with two Intel Xeon E5520 (quad-core, HT, 2.27Ghz), 32 GB of RAM, two Gigabit Ethernet and a hard disk with 146 GB (15k RPM). The operating system is an Ubuntu Server Lucid Lynx (10.04 LTS, 64-bits), running with a kernel 2.6.32-21-server, and the Java version is the 1.7.0_25 (64-bits).
5.5.2 Disclosure Filter Input Data

The input data, as explained in Section 5.3, is a set of small sequence entries comprised of 30 base pairs each. Each method, henceforward refereed as STR- and SNP-based, generates different sets of sequences that will be used as input to our tests. The STR-based method contains two data sets, the Y-STR and All-STR, which respectively contain short tandem repeats from chromosome Y and from all chromosomes. The SNP-based method contains four data sets, the Very-Rare, the Rare, the Low-Frequency and the Common, which depend on how rare their single nucleotide polymorphisms are. Table 5.2 contains the input size (in millions of entries) of each data set previously refereed and of combinations among them. We are particularly interested in the two distinguished setups (in bold) from this table, the Y-STR data set (Y-None) and the combination of All-STR with Common SNPs (All-Common). The first is the minimal dataset needed to protect biobanks from the attack performed against donor’s privacy through genetic genealogical profiling [92]. The second uses the largest dataset available and is arguably the most powerful filter to protect biobanks from the attacks we are aware of [92, 112].

<table>
<thead>
<tr>
<th>STR</th>
<th>SNP</th>
<th>None</th>
<th>Very-Rare</th>
<th>Rare</th>
<th>Low-Frequency</th>
<th>Common</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>None</td>
<td>0.5</td>
<td>377.6</td>
<td>673.8</td>
<td>928.5</td>
<td>1092.5</td>
</tr>
<tr>
<td>All</td>
<td>Y</td>
<td>22.5</td>
<td>378.1</td>
<td>674.3</td>
<td>929</td>
<td>1093</td>
</tr>
<tr>
<td>All</td>
<td>All</td>
<td>22.5</td>
<td>400.1</td>
<td>696.3</td>
<td>951</td>
<td>1115</td>
</tr>
</tbody>
</table>

Table 5.2: Number of entries on the disclosure filter in millions ($10^6$) considering different amounts of SNP and STR sequences.

The number of entries in Table 5.2 can be directly translated to the amount of storage space need for such entries. For example, if each base pair requires one byte to be stored, the Y-None dataset would require $1 \times 30 \times 0.5 \times 10^6$ bytes, or 15MB. The All-Common, by the other hand, would require $1 \times 30 \times 1115 \times 10^6$, or 33.45GB.

5.5.3 Bloom Filter Size

This first analysis contains a discussion on how big the disclosure filter is considering each separate data set from Table 5.2 and those two combinations of interest. Since we use a Bloom filter, theoretically, the filter size depends only on the expected number of entries and the expected false positive probability [36]. The former is static for each case and follows the values present in Table 5.2. The later is dynamic and refers to limitations on server’s memory capacity to run the entire disclosure filter in memory. Figure 5.4 contains the resulting filter size based on the size of input data and the expected false positive rate of the Bloom filter.

The filter size using input from STR-based method is 10- to 40-fold smaller than the size if using the SNP-based method, as expected considering the values from Table 5.2. The biggest filter size from this graph is observed when using the biggest data set (All-Common) and a false positive probability of $10^{-6}$ (1 in 1 million), which leads to a filter as big as 4.3GB. Notice this is still around 8× smaller than the size of the entries stored in this filter, and can easily be kept in main memory for fast lookups. Additionally, all input data sets generate filters smaller than 1GB of space in memory when using a false positive probability of 10%, which may or may not be acceptable for some applications.
Figure 5.4: Bloom filter size for different false positive rates and input data (see Table 5.2). The x-axis is in logarithmic scale.

5.5.4 Local Performance

Our second analysis contains a discussion on how many classifications per second our filter is able to perform using only a single core in our test machine. Figure 5.5 shows these throughput numbers.

As expected, the smaller the filter, the higher the throughput is. This happens because small filters require less hash operations to test if a sequence belongs to the set or not. Notice that even our biggest and most powerful filter (All-Common with a probability of false positive of $10^{-6}$) is still able to classify more than 1.3 million 30-bp sequences per second with a single core. This value suggest that our disclosure filter will not be the bottleneck of any bioinformatics workload.
5.5.5 Sensitive Reads

Our last analysis considers how much of an entire genome is considered private for each combination of the input data set and false positive rate. Once more, we consider each separate data set from Table 5.2 and those two combinations of interest (All-Common and Y-None). We randomly picked one entire genome from the 1000 Genomes Project to perform the test, and the selected donor was the one identified as HG01140. This genome was split in 97 millions sequences with 30 bps each, which means a test input of 2.9GB. Figure 5.6 contains the resulting percentages of sensitive entries for this genome, which in average took 1 minute to finish each execution.

![Figure 5.6: Percentage of privacy-sensitive sequences for different false positive rates and input data. X and y axes are in logarithmic scale.](image)

Note that there is a minimal percentage of sensitive reads that is independent of false positive probability, which is present in the similar results from probabilities $10^{-6}$, $10^{-5}$, $10^{-4}$ and $10^{-3}$. It means that a biobank needs to maintain in private boundaries at least 0.12% (3.6 MB\(^1\)) of HG01140’s genome if it is using the Y-None filter, and 7.98% (240 MB\(^1\)) if All-Common is used.

5.6 Use Cases

In this section we propose two use cases for the privacy filter within the context of the BiobankCloud project.

5.6.1 Archival of Raw Data

A first obvious use case for the disclosure filter would be for raw data archival. This workflow is quite simple and directly matches what is presented in Figure 5.7: a sequence machine creates a

\(^{1}\)Values in MB without compression.
number of reads (i.e., sequences) that are classified in privacy-sensitive (private) or not (public). Private sequences are stored locally while public sequences are stored in the public storage clouds. In terms of performance, the use of the filter to classify such reads is not expected to create any bottleneck since NGS machines 300000 bp/sec [89], and our filter can deal with more than two orders of magnitude this value.

Notice this workflow can be easily implemented with the OVERBANK infrastructure described in Chapter 4: we can have two directories in Charon, one (A) corresponding to a local or trusted and regulation-compliant remote storage while the other (B) maps to the cloud-of-clouds. For each read sequence, a simple shell script can invoke the filter to verify if it should be written in files stored in A or B. The association between the public and private files will be made either through an additional metadata file or through the names of the created files.

![Figure 5.7: Basic workload for archival of raw data.](image)

5.6.2 Hybrid Sequence Alignment

Aligning sequences is one of the most important steps in many bioinformatics workloads, since it recognizes where any sequence should be aligned to a common reference genome. There are several algorithms for aligning DNA sequences [87] and they are divided mainly in two categories: the seed-and-extend and the prefix-trees. Some of them already where ported to cloud computing (e.g., [130, 107, 80, 53, 99]) and some were developed to this computation model (e.g., [117]). The algorithm to be chosen will depend on your requirements, for example: the amount of resources available to aligners, the throughput desired, the alignment latency, etc.

Figure 5.8 presents a complete sequence alignment using the disclosure filter to guarantee the privacy of sequences. The first component is the sequencing machine that receive human samples to be sequenced. As soon as NGS sequences a DNA read, it sends this read to the disclosure filter. The disclosure filter classify the read as private or public and forward it to the proper next workflow step. If the sequence is considered public, then it is compressed and sent to the cloud. Once the compressed sequence arrived in the public cloud, it is decompressed and used as input to the alignment algorithm. The alignment algorithm will determine the best mapping position to that sequence and will send the result to a private resource, called Integrator. If the sequence was considered private, then it is aligned locally. The local alignment also is sent to the Integrator component, which will be responsible for gathering all aligned reads and
mounting a contiguous genome. This final genome is compressed using referential compression and stored in the private infrastructure.

We intend to create solutions that surpass the throughput of modern NGS machines [89] by employing dynamic resources from public cloud providers. In order to achieve such objective, the disclosure filter must classify at least $10000$ small sequences (e.g., of 30 base pairs) per second nowadays. Subsequently, the compression and decompression algorithms should deal with a throughput of at least $10000 \times PNS$ sequences per second, where $PNS$ is the percentage of non-sensitive reads. The alignment component placed in the public cloud can scale until where it is needed by using more machines. The local aligner probably will receive much less sequences to align then the one place in public cloud, which leads that it does not need so high throughput, but still it needs to surpass the mark of $10000 \times PS$ sequences per second, where $PS$ is the percentage of sensitive reads. The integrator receives input from both local and external alignment components and must to deal with $10000$ sequences per second without creating bottlenecks. Finally, the referential compression algorithm must finish the compression before the integrator needs to send other contiguous sequence. The final result must be a small portion of the size needed by an entire genome without compression.

5.7 Final Remarks

In this chapter, we presented the rationale, design and implementation of a disclosure filter that is based on privacy-sensitivity of genomics data. Such component aims to support a black list of fixed-size sequences that may identify a sequence donor. We propose to build this black list based on genetic genealogy profiling and rare variants present on individuals. Such black list can be used to filter sequences to defend donors privacy from two recently identified attacks [92, 112]. Our implementation shows that the disclosure filter can be implemented efficiently (both in terms of performance and memory-usage) using Bloom filters.
Chapter 6

Conclusions and Future Work

Chapter Authors:
Alysson Bessani (FFCUL).

This deliverable presented an analysis of the state of the art in biobank integration and a preliminary architecture for the integration of biobank clouds. This integration considers each biobank cloud as a computing infrastructure running the BiobankCloud PaaS (described in deliverables 2.1 and 5.1). Furthermore, these clouds will be interconnected using a Dropbox-like shared infrastructure called OVERBANK, in which files can be stored and shared. This infrastructure would support several storage options, from local trusted repositories (inside the biobank clouds) to public storage clouds (e.g., Amazon S3) and even secure and dependable clouds-of-clouds [34].

The OVERBANK will be used as a substrate for implementing the BiobankCloud data model (as described in deliverable 1.1 and 1.2) and the ethical and legal constraints for privacy-sensitive data (as defined in deliverable 1.5).

As the title of this deliverable suggests, the work described here is by no means complete. Our focus here was on presenting some design principles and a preliminary architecture for the main mechanisms and components we are devising for realizing the OVERBANK concept. In particular, there are some loose ends that need to be addressed in the remaining of the project:

1. The Charon file system is currently under development and we expect a first proof-of-concept prototype to be ready in few months. One of the main issues we are currently addressing is how to hide the huge latencies of accessing big files in the cloud. Another open issue to be addressed in the future is if compression should be integrated on the file system or done entirely by the applications. Finally, a fundamental work still to be done is the integration of Charon with the BiobankCloud PaaS file system (described in D2.1) and the security toolset of the project (described in D3.2). This last issue is planned to be addressed in the 3rd year of the project, but we are closely following the developments in WPs 2, 3 and 5 to guide our design towards such integration.

2. The RANC communication system is still in its early stages of development, and we are still defining how to use state-of-the-art networking technologies for efficient communication between biobanks. More specifically, our current efforts focus on the integration of multihoming, network coding and software-defined networking in the system. The final
objective is to have an incrementally deployable communication infrastructure using the same kind of technology employed by big internet service providers (e.g., Google).

3. Another contribution presented in this deliverable is a disclosure filter for genetic information. Although we presented the rationale, design, implementation and even some evaluation, there are still some work to do regarding the limitations of our approach. In particular, we want to answer the following question: “what is the coverage of the black list of sequences in the filter?” We are currently working on such question and we expect to have an answer in the next weeks. After that, we expect to start using the filter for the anonymization of public datasets and in the use cases discussed in Section 5.6.

4. Last but not least, in Chapter 2 we briefly discussed the possibility of supporting certified computations in private data maintained in remote biobanks (see Figure 2.6(a)). Although this deliverable does not further discuss how such feature can be implemented in our architecture, we plan investigate this feature in the 2nd year of the project.

Our aim is to make a first demonstration of the OVERBANK in the 2nd project review, at the beginning of 2015.
Appendix A

Erasure and Network Codes

*Chapter Authors:*
Ricardo Fonseca, Nuno Neves (FFCUL).

A.1 Introduction

Forward Error Correction (FEC), in general, is a technique for recovering information from errors or erasures in data that passes through a noisy channel. This is done at the expense of introducing extra (redundant) information, although the redundancy cost can be made significantly less than when applying simple data replication mechanisms. Errors typically occur when a bit transmitted over some channel is flipped in transit and arrives at the receiver with the opposite value. However, errors may also occur at the level of codewords with variable length, for example deletions or insertions of bits in individual packets. These errors may, for example, be caused arbitrarily by adversaries that attempt to confuse receivers or exploit vulnerabilities. Erasures, on the other hand, give a richer information to receivers in respect to the contents of the original data. Erasures are defined by the theoretical model named Binary Erasure Channel (BEC), in which either a transmitted bit is delivered to a receiver (without errors), or the receiver is notified of the fact that the bit was “erased”.

We are particularly interested in erasure recovery capabilities here and their applications for a) distributed data storage in cloud-of-clouds and b) resilient packet transmission in multipath communication. The focus on erasures instead of errors comes from the fact that data is authenticated and integrity-protected in both scenarios a) and b), and thus bit/packet-level errors can be detected (and the incorrect data discarded). This provides an ideal environment for dealing with erasures, and we can therefore exploit the extra information they convey for a more optimal recovery capability.

A.2 Distributed data storage in cloud-of-clouds

Storing data in a cloud-of-clouds has the advantage of increasing the availability of the information (see Section 4.2.3). In order to receive the benefits of such a property, it is required to carefully design the cloud allocation strategy. Replicating the whole data over all clouds becomes prohibitively costly as the number of clouds increases. On the other hand, simply splitting the data over all the clouds is not ideal, since the availability would then depend on all the clouds and defeat the purpose. We encode the data using erasure codes before transmitting...
it to the clouds, thus being able to reduce the individual per-cloud cost, while requiring only a subset of all clouds to retrieve the stored data.

We use a class of FEC codes, the Reed-Solomon codes [84], for the splitting and encoding of data before transmission to the clouds. Data is split in $k$ blocks, and from those, $p$ parity blocks are generated. In total, we have $N = k + p$ blocks, and we transmit those $N$ blocks, one to each cloud. When used as erasure codes, the Reed-Solomon codes can correct up to $p$ block erasures, which means that we only need any $(N - p)$ or $k$ blocks to recover the original data. We exploit the nature of the erasure codes to only require a subset of the clouds to be used during retrieval of data.

Reed-Solomon codes, however, require a fixed code rate that must be defined a priori. The code rate $R$ of an erasure code corresponds to the proportion of the total useful (non-redundant) data, and is given by $R = k/N$. Let $C$ be the number of clouds we use in the cloud-of-clouds. The size of the clouds subset to be used during retrieval of data is given by $\lceil C \times R \rceil$.

### A.3 Resilient data transmission in multipath communication

Multipath communication, such as one performed in the Resilient and Adaptive Network Communication (RANC) channels, is done by a sender that uses a few of the available routes to the destination (see Section 4.3). However, data transmission through multiple routes per se is not sufficient to achieve robustness in the communications. In fact, even a single route behaving erroneously (e.g., losing packets) is enough to prevent the original data from being reconstructed. Therefore, using multiple concurrent routes can actually degrade the reliability of the whole communication. Two possible approaches to recover from losses are (1) the retransmission of the packets at a later time, or (2) the concurrent transmission of several copies of the packets over different paths. The first solution has the advantage of minimizing the amount of data that is sent, at the cost of delaying the delivery of the packets (since retransmissions occur after a timeout). The second approach has the opposite characteristics. The RANC uses erasure codes to both decrease the amount of transmitted data and minimize the delays in case of losses.

The transmission process begins by splitting data into a number $k$ of source blocks and then producing, from those, a number $N$ of encoded blocks (with the same size as the former) that are sent over the network. Recovery of the data is possible by receiving only a subset $K$ of the $N$ encoded blocks. Generally, $k \leq K \leq N$. The reception efficiency of an erasure code is given by the number $K$ of encoded blocks required by a receiver in order to successfully recover the original $k$ source blocks. As $K$ approaches $k$, the reception efficiency increases and reaches an optimal value when $K = k$. However, an optimal reception efficiency comes at the expense of good memory and time performances, and as such the RANC uses a class of near-optimal erasure codes instead that provide near-linear encoding and decoding complexity. Using such codes, receivers can attain the successful recovery of $k$ source blocks by receiving $(1 + \epsilon) \times k$ encoded blocks, where $\epsilon \geq 0$. In practice, though, the value of $\epsilon$ can be made small (e.g. $\epsilon = 0.05$) while granting a very high probability of successful recovery.

The near-optimal erasure codes used by the RANC are the Fountain Codes [94], which have the additional property of being rateless. This allows senders to adjust to an increase of the data loss rate by simply transmitting more encoded blocks (i.e. by increasing $N$). This works well in the presence of multiple routes to transmit data, since these routes have typically independent loss rates.
Appendix B

Software-Defined Networks

Chapter Authors:
Fernando Ramos (FFCUL).

B.1 An introduction to Software Defined Networking

Conventional networks are very hard to manage. To express the required high-level network policies, network operators need to configure each individual network device separately — from a heterogeneous collection of switches, routers, middleboxes, etc. — using vendor-specific and low-level commands. Routers and switches run complex, distributed control software that is typically closed and proprietary. In addition to configuration complexity, networks are dynamic, and operators have little or no mechanisms to automatically respond to network events. It is therefore difficult to enforce the required policies in such a continually changing environment. This complex mode of operation has hindered innovation and inflated the capital and operational costs of running a network.

Software defined networking [102, 121] (SDN) is an emerging paradigm in computer networking that promises to change the current undesirable state of affairs. The key enabler for an SDN is the separation of the network’s control logic (the control plane, where decisions of how to handle network traffic are made) from the underlying physical routers and switches that forward traffic (the data plane, which forwards traffic according to the decisions made by the control plane). With the separation of the control plane from the data plane that lays the ground to the Software Defined Networking paradigm, network switches become simple forwarding devices and the control logic is implemented in a logically centralized controller.

The separation of the control plane and the data plane can be realized by means of a well-defined programming interface between the switch and the SDN controller. The controller exercises direct control over the state in the data-plane elements via this well-defined API. The most notable example of such API is OpenFlow [103, 22]. An OpenFlow switch has one or more tables of packet-handling rules. Each rule matches a subset of traffic and performs certain actions (dropping, forwarding, modifying, etc.) on the traffic that matches a rule. Depending on the rules installed by a controller application, an OpenFlow switch can thus behave like a router, switch, firewall, or perform other less conventional roles.

The logical centralization of the control logic in a software module that runs in a standard server offers several benefits. First, it is simpler and less error-prone to modify network policies through software, than via low-level device configurations. Second, a control program can
automatically react to spurious changes of the network state and thus maintain the high-level policies in place. Third, the centralization of the control logic in a controller with global knowledge of the network state simplifies the development of more sophisticated network functions. This ability to program the network in order to control the underlying data plane is therefore the crucial value proposition of SDN.

OpenFlow (and SDN) started as an academic experience [103] but has gained significant traction in industry over the past few years. Most vendors of commercial switches now include support of the OpenFlow API in their equipment. As the initial concerns with SDN scalability were understood — in particular the myth that logical centralization implied a physically centralized controller, an issue we will return later — SDN ideas have matured and shifted SDN from an academic artefact to becoming a commercial success. Google, for example, has deployed an SDN architecture to connect its datacenters across the planet. This production network has been in deployment for 3 years, with success, helping the company to improve operational efficiency and significantly reduce costs [77]. VMware’s network virtualization platform, NSX [30], is another example. NSX is a commercial solution that delivers a fully functional network in software provisioned independently of the underlying network hardware, entirely based around SDN principles. As a final example, the world’s largest IT companies (from carriers and equipment manufacturers to cloud providers and financial-services companies) have recently joined SDN consortia like the Open Networking Foundation [22] and the Open Daylight initiative [23], which is another clear sign of the importance ascribed to SDN by the industry.

**B.2 SDN Controllers and Programming Languages**

An operating system (OS) is software that manages computer hardware resources, abstracting the hardware and providing common services for computer programs. The work on OpenFlow and SDN led to the notion of a network operating system [68] (commonly referred in SDN as the controller). Similar to a typical OS, a network operating system is software that abstracts the installation of state in network switches from the logic and applications that control the behavior of the network. This idea led to a conceptual decomposition of an SDN into three layers, as in Figure B.1.

1. The data plane layer with an open interface to the control plane. This layer consists of the network switches that forward traffic. In the figure, the open interface is Openflow.

2. The control plane layer that is responsible for maintaining a consistent view of network state. It is the SDN controller (network operating system) that provides this network abstraction to the applications.

3. The management plane layer that consists of the control logic that performs operations depending on its view of network state. We include in this layer the programming languages for SDN that have recently been proposed as a tool to ease network application development.

Many different SDN controller platforms have emerged in the past few years. The first OpenFlow/SDN controller was NOX [68]. NOX is a platform that provides a C++ OpenFlow API for building network control applications. For Python developers NOX has a younger sibling, POX [19]. Other popular platforms include Beacon [57] and Floodlight [11], open
source controllers developed in Java. An interesting particularity of Floodlight — which is the core of a commercial product from Big Switch Networks — is its OpenStack cloud orchestration platform. This makes it an attractive alternative to cloud environments. The first SDN control platform to address the challenges of large scale networks was Onix [82]. Onix is a closed-source, distributed control platform, designed with scalability, reliability, and performance from the outset. This platform is the basis of the already mentioned VMware NSX solution and is one of the core components of Google’s SDN infrastructure.

To the best of our knowledge, no open-source, distributed control platform (Onix-like) exists at the moment. In this regard it is worth mentioning two projects where this goal is being pursued. The first is OpenDaylight [23], a community-led, open, industry-supported framework created with the aim to accelerate the adoption of Software-Defined Networking. As a collaborative project under The Linux Foundation, OpenDaylight is structured using open source development best practices, and is comprised of the leading organizations in the technology industry (Cisco, Microsoft, Red Hat, among others). Another project where an open-source distributed SDN controller is being developed is ONOS, from the Open Networking Lab (ON.LAB) [21].

Current SDN controllers offer low-level APIs that mimic the underlying switch hardware, forcing programmers to grapple with low-level details. Supporting multiple tasks at the same time (say, routing and access control, for instance) is extremely difficult, since the application must ultimately install a single set of rules on the underlying switches. Recent work [59] has started to identify higher-level abstractions for creating and composing network applications. These abstractions are provided by network programming languages (Frenetic [60] being the prime example) which enable programmers to specify policies in a higher-level language, and have these policies automatically compiled to Openflow rules.

### B.3 SDN Applications

The core component of an SDN is the controller, but the actual logic that implements the desired network policies is defined by the SDN applications that run on top of it. Several applications have been proposed recently to achieve various purposes: server load balancing, traffic engineering, security (access control), network virtualization, switching and routing, among others. In this section we present a sample of proposed SDN applications.
Ethane [43], one of the earliest SDN applications, allows managers to define a single network-wide fine-grain access control policy, and then enforce it directly. This application was, to the best of our knowledge, the first where it was made clear the advantage of using simple and “dumb” flow-based Ethernet switches with a centralized controller in easing innovation.

The means for performing traffic engineering (TE is the ability to control the paths used to deliver traffic) using conventional routing protocols is very primitive (to say the least), whereas with SDN it is possible to deploy routing and traffic engineering protocols as SDN applications customized to our unique requirements. One of the selling points of SDN has therefore been the ease to perform traffic engineering in the network. This was Google’s main motivation for opting to build an SDN Wide Area Network. Another example of a traffic engineering application is ElasticTree [71], and SDN application that allows the system to actively monitor the data plane load and optimize the network’s energy consumption. The authors propose a set of optimization algorithms to meet the network goals (in terms of latency, performance, and fault tolerance), while simultaneously reducing power consumption.

Another relevant advantage of SDN applications is their potential to replace network middleboxes [122]. Middleboxes are network appliances that augment the network data plane to provide all sorts of functionality: firewalls, load balancing, intrusion detection, etc. These appliances are proliferating in networks and as a result the once thought best-effort, “simple” dataplane, is becoming increasingly complex. SDN applications can mimic the functionality of different types of middleboxes. Load balancing is a good example: an SDN application can perform the role of traditional load balancers — expensive devices with limited customizability —, cheaply and with added flexibility [137]. Such low-cost solution can be further enhanced with the ability to schedule load not only between servers, but also between paths, something that is not possible with traditional load balancing middleboxes.

The applications introduced so far are examples of unconventional, yet useful, routing methods. An SDN can naturally also run conventional routing protocols, such as OSPF or BGP (the most common intra- and inter-domain routing protocols used in current networks, respectively). An application running on top of the controller, such as Routeflow [114], can offer conventional routing services in an SDN environment. Google’s SDN has a similar service to uses as a fault recovery mechanism. If there is any critical issue in traffic engineering, the network has a “red button” to fall back to conventional shortest path routing.

Network virtualization has been considered by many as the probable “killer app” for SDNs [122]. By presenting an abstraction that represents a logical network that is independent from the underlying physical network hardware, network virtualization allows multiple virtual networks to run over a shared infrastructure. For example, a Virtual Local Area Network (VLAN) provides the illusion of a single LAN spanning multiple physical subnets, and multiple VLANs can run over the same collection of switches/routers. Flowvisor [123] was the first instance of a network virtualization application for SDN. Its original aim was to serve as a research platform to allow multiple network experiments to run side-by-side with production traffic. For that purpose it proposed a novel approach to Openflow switch virtualization. The already mentioned commercial offering from VMWare (NSX) is yet another example of network virtualization using SDN as an enabling technology.

### B.4 SDN Main Challenges

A fundamental abstraction introduced by Software Defined Networking is the concept of *logical centralization*. In an SDN, network control applications can be designed and operated on a
global, centralized network view. This global view enables simplified programming models and facilitates network applications design and development. This perception that control in SDN is centralized has lead to concerns about SDN scalability, resiliency, and security.

Indeed, a physically centralized controller does not scale as the network grows. However, a logically centralized programmatic model does not postulate a centralized system. In fact, the need to guarantee adequate levels of performance, scalability, and reliability preclude a fully centralized solution. Instead, production-level SDN network designs resort to physically distributed control planes [82]. The distribution of the control plane is, nevertheless, a complicated problem to solve. For instance, the designers of these systems have to face the fundamental trade-offs between the different consistency models and the need to have a highly available system (even in the presence of network partitions) [39], while simultaneously guaranteeing adequate performance. Recently, we demonstrated that the use of a consistent and fault-tolerant control plane does imply necessarily in bad performance [37], and we expect to exploit this result within the BiobankCloud project. Notwithstanding, the scalability concerns have been gradually deconstructed in the past few years and it has been convincingly argued that they are neither caused by nor are fundamentally unique to SDN [142]. Current SDN deployments clearly support this argument [77].

SDN enables the introduction of sophisticated network policies, such as security and dependability. However, the security and dependability of the SDN itself has been a relatively neglected topic up to now. The main causes of concern lie in SDN’s main benefits: network programmability and control logic centralization. These capabilities actually introduce new fault and attack planes, which open the doors for new threats that did not exist before or were harder to exploit. Traditional networks have “natural protections” against what would be common vulnerabilities in regular IT systems. This diversity is comparatively smaller in SDNs. These properties do not postulate that software-defined networks are inherently less secure when compared to current networks, but they pose threats of a different nature that need therefore to be dealt with differently [83].
Bibliography


[140] K. Wetterstrand. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). Available at: www.genome.gov/sequencingcosts.

