Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Integridade e Confidencialidade dos Arquivos na Plataforma de Nuvem Federada BioNimbuZ

Rafael Scofield Sardenberg

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Aletéia Patricia Favacho de Araújo

Coorientador
Prof. MSc. João José Costa Gondim

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenador: Prof. Dr. Homero Luiz Piccolo

Banca examinadora composta por:

Prof.ª Dr.ª Aletéia Patricia Favacho de Araújo (Orientadora) — CIC/UnB
Prof. MSc. João José Costa Gondim — CIC/UnB
Prof.ª Dr.ª Maria Emília Machado Telles Walter — CIC/UnB
Prof.ª Dr.ª Maristela Terto de Holanda — CIC/UnB

CIP — Catalogação Internacional na Publicação

Sardenberg, Rafael Scofield.

Integridade e Confidencialidade dos Arquivos na Plataforma de Nuvem Federada BioNimbuZ / Rafael Scofield Sardenberg. Brasília : UnB, 2015.
129 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2015.

1. Computação em Nuvem, 2. Nuvem Federada, 3. Segurança,
4. Integridade, 5. Confidencialidade, 6. BioNimbuZ.

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Integridade e Confidencialidade dos Arquivos na Plataforma de Nuvem Federada BioNimbuZ

Rafael Scofield Sardenberg

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Aletéia Patricia Favacho de Araújo (Orientadora)
CIC/UnB

Prof. MSc. João José Costa Gondim
CIC/UnB

Prof.ª Dr.ª Maria Emília Machado Telles Walter
CIC/UnB

Prof.ª Dr.ª Maristela Terto de Holanda
CIC/UnB

Prof. Dr. Homero Luiz Piccolo
Coordenador do Bacharelado em Ciência da Computação

Brasília, 16 de dezembro de 2015

Dedicatória

Gostaria de dedicar este trabalho aos meus pais, Fernando e Eleanor, que sempre me ajudaram e me incentivaram durante todo o meu ciclo escolar. Dedico também aos meus avós maternos José e Eleanor por sempre estarem ao meu lado ao longo de toda a jornada da minha vida, desde a minha infância até os dias de hoje. Da mesma forma, dirijo esta dedicatória aos meus avós paternos José e Maria da Glória que, apesar de não estarem mais presentes entre nós, ainda continuam vivos em minha memória.

Agradecimentos

Agradeço, primeiramente, a meus pais, pelos inúmeros conselhos e críticas que me deram, pois sem esses eu jamais teria chegado onde cheguei. A minha irmã, pelos pelos momentos de conversa em que compartilhamos pensamentos e reflexões.

A minha namorada Raphaela, acima de tudo pelo companheirismo que tivemos desde o momento que nos conhecemos. Agradeço imensamente aos seus conselhos, foram extremamente construtivos pra mim.

Agradeço a minha professora e orientadora, Aletéia Patrícia, pela dedicação e paciência, sempre me guiando na condução deste trabalho. Obrigado por todos os feedbacks e por tornar este projeto uma realidade.

Obrigado ao meu amigo de BioNimbuZ, Breno Moura, por sempre estar disposto a me ajudar e sanar as minhas dúvidas mesmo em momentos difíceis.

A todos os meus amigos da UnB, pelos momentos de estudo e dedicação que tivemos que compartilhar juntos. Em especial: Hermano Leite, Lucas Fernandes, Thales Vinkler, Paulo Ricardo Giusti, Matheus Pimenta, Estêvão Aguiar, Renan Rheinboldt, Felipe Rodopoulos e Ícaro Dantas.

Agradeço também à Empresa Júnior de Computação - CJR, que me proporcionou um grandioso amadurecimento pelo período de três anos que tive o prazer de participar desta empresa.

Muito obrigado!

Resumo

Um ambiente de nuvens federadas é composto por nuvens interligadas entre si, que compartilham recursos utilizando uma interface transparente ao usuário. Nestes ambientes, garantir a integridade dos arquivos manipulados pelas nuvens desta federação não é uma tarefa fácil visto que os arquivos não estão mais fisicamente em posse do seu responsável. Além disso, em razão do usuário não deter mais fisicamente o seu arquivo, ele fica bastante limitado na tarefa de não permitir que entidades maliciosas acessem as informações dos seus arquivos. Este trabalho trata do problema da segurança em nuvens federadas, em especial na plataforma BioNimbuZ. Neste contexto, este trabalho propõe a utilização de uma política de integridade dos arquivos usando *hashes* e também uma política de confidencialidade dos arquivos utilizando algoritmos de criptografia. Os experimentos realizados ao final deste projeto mostraram a eficácia deste trabalho em verificar a corretude dos arquivos e em garantir que ninguém além dos usuários do BioNimbuZ tenha acesso às informações.

Palavras-chave: Computação em Nuvem, Nuvem Federada, Segurança, Integridade, Confidencialidade, BioNimbuZ.

Abstract

A federated cloud environment consists of interconnected clouds each other, which share resources using a transparent user interface. In these environments, ensure the integrity of files handled by peers this federation is not an easy task given that the files are no longer physically in possession of his charge. In addition, because the user does not hold more physically your file, it is quite limited in the task of not allowing malicious entities from accessing the information of your files. This work deals with the problem of security in federated clouds, especially in BioNimbuZ platform. In this context, this work proposes the use of a file integrity policy using hashes of the files and also a policy of confidentiality using encryption algorithms. Experiments carried out at the end of this project showed the effectiveness of this work to verify the correctness of files and ensure that no one but the BioNimbuZ users have access to information.

Keywords: Cloud Computing, Federated Cloud, Security, Integrity, Confidentiality, BioNimbuZ.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O mundo vem sofrendo diversas mudanças, e uma das grandes responsáveis por elas foi a Internet. Antes da Internet, a maior parte do processamento era realizado em servidores locais e *datacenters* proprietários, que mesmo em seus momentos ociosos geravam custos com manutenção e energia elétrica, e não utilizavam seu poder computacional por completo. Neste cenário, surgiu a ideia de um modelo de computação sob demanda, de forma escalável, no qual os usuários pagam apenas por aquilo que foi de fato utilizado. Este modelo recebeu o nome de computação em nuvem [32].

Em um ambiente de computação em nuvem, os recursos utilizados estão disponibilizados por meio da Internet, podendo serem acessados de qualquer lugar do mundo, motivo pelo qual ocorre a associação ao termo "nuvem". Nesse tipo de plataforma, a grande diferença é que os recursos são alocados de acordo com a da demanda. Do ponto de vista econômico, o usuário paga apenas pela utilização dos recursos em nuvem, sem a necessidade de utilizar os recursos de seu computador ou servidor local. A ideia é que não seja necessário adquirir softwares ou infraestruturas, pois é mais barato pagar apenas quando ocorre uma demanda para a utilização deles. Para empresas, isso significa uma grande redução nos custos, podendo atingir uma economia de milhões de dólares [32].

No cenário da computação em nuvem, ocorre uma alternativa na forma de utilização de serviços. Ao contrário do cenário no qual o usuário coloca serviços para serem feitos por um servidor local, ele contrata um provedor para que ele realize todos esses serviços. Estes serviços podem ser de armazenamento de dados, de utilização da memória, de poder de processamento da provedora para executar aplicações, e também de infraestrutura para hospedar serviços [32].

Assim sendo, o modelo de computação em nuvem é caracterizado por três pontos principais que o distingue de outros modelos de sistemas distribuídos, os quais são: o pagamento sob demanda, a computação elástica e a virtualização [51]. Para uma empresa, o pagamento sob demanda possibilita uma redução de custos em detrimento do consumidor somente contratar os recursos que realmente necessitam. O segundo ponto, a computação elástica, ocorre quando os recursos computacionais disponibilizados são reutilizáveis e realocados para diversos clientes, suportando, assim, alta escalabilidade. Por último, a virtualização, uma tecnologia de abstração do acoplamento entre hardware e sistema operacional, no qual os recursos são virtualizados, solucionando muitos problemas que podem ocorrer pela heterogeneidade computacional dos recursos. A virtualização permite que ambientes computacionais sejam dinamicamente criados, expandidos e removidos.

Entretanto, com o aumento das necessidades dos usuários por processamento e por escalabilidade, a utilização de nuvens isoladas passou a ser insuficiente para suprir a demanda das aplicações pelo consumo de recursos [33]. Para solucionar este impasse, emergiu o conceito de nuvem federada, isto é, um conjunto de nuvens interligadas, gerenciadas por uma interface que proporciona uma abstração para o usuário de modo que ele não perceba que são várias nuvens. Assim sendo, em um ambiente de nuvens federadas, várias nuvens podem compartilhar recursos entre si, possibilitando uma expansão maior do que aquela permitida apenas por uma única nuvem, e gerando uma melhor alocação de recursos.

O BioNimbuZ, uma arquitetura para federação de nuvens computacionais híbridas, permite a integração e o controle de diferentes provedores de infraestrutura com suas ferramentas de bioinformática oferecidas como serviço, de maneira transparente, flexível e tolerante a falhas, o que significa que provedores independentes, heterogêneos, privados ou públicos de nuvens computacionais podem oferecer seus serviços conjuntamente, mantendo suas características e políticas internas [53].

Neste contexto, o BioNimbuZ procura oferecer a ilusão de que os recursos computacionais disponíveis são ilimitados e que as demandas dos usuários sempre serão atendidas. Atualmente, federações de nuvens são implementadas por meio de um *middleware* que permite a interação entre diversas nuvens. A proposta do BioNimbuZ é justamente ser esse *middleware*, possibilitando a interação entre um ou mais serviços, hospedados em um ou mais provedores de nuvem.

Todavia, apesar do crescimento da utilização do BioNimbuZ e de todas as outras plataformas de computação em nuvem, ainda existe uma enorme preocupação com a segurança no uso destas plataformas. E esse é um motivo que prejudica consideravelmente a completa adoção deste paradigma de computação por organizações e empresas que possuem dados confidenciais [30].

Dessa forma, ao mesmo tempo em que cresce os adeptos à computação em nuvem, vários assuntos relacionados a segurança da informação em ambientes de nuvem aparecem como uma das principais barreiras para a adoção mais rápida e mais generalizada dessa tecnologia. Essa visão origina-se a partir de diferentes perspectivas de pesquisadores, tomadores de decisão da indústria e organizações governamentais. Para empresas cuja natureza de seus negócios são críticos, com aplicações extremamente sensíveis ao tempo e resposta, ou empresas do setor financeiro, a computação em nuvem parece desaconselhável, devido a questões como a disponibilidade dos serviços, a confidencialidade dos dados, a integridade das informações, entre outros [33].

Com base nessas informações, este trabalho se propõe a definir e implementar um padrão de integridade e de confidencialidade como parte do módulo de segurança da plataforma BioNimbuZ.

Assim, visando contribuir para a resolução do problema da segurança no ambiente de nuvem federada do BioNimbuZ, este trabalho propõe a verificação da integridade dos arquivos que estão sendo armazenados na plataforma. Além disso, para aumentar a confidencialidade dos arquivos, o trabalho propõe o uso de um algoritmo de criptografia desses dados.

1.1 Motivação

Dentro desse contexto, existe atualmente uma arquitetura de nuvens computacionais híbridas para a execução de *workflows* em bioinformática, conhecida como BioNimbuZ. Devido a grande quantidade de dados que precisam ser armazenados ou processados nessa plataforma, a motivação deste trabalho é definir e implementar uma política de integridade e de confidencialidade dos arquivos manipulados pelo BioNimbuZ.

1.2 Problema

Atualmente, não existe na plataforma BioNimbuZ uma política de integridade dos arquivos. Além disso, o usuário não possui a confiança de que o conteúdo dos arquivos não será acessado por usuários não habilitados. Assim sendo, os arquivos são transferidos para servidores externos e armazenados neles por tempo indeterminado, e o sistema não possui mais controle sobre os mesmos. Isso traz alguns pontos negativos para os usuários do BioNimbuZ, os principais são:

• Os usuários não possuem a certeza de que seus arquivos chegaram ao destino de forma fidedigna;

• Não existe uma confirmação de que os arquivos armazenados remotamente continuam idênticos aos arquivos originais de quando chegaram ao servidor;

• O conteúdo dos arquivos pode ser acessado por usuários que possuem acessos aos servidores externos.

1.3 Objetivos

1.3.1 Objetivo Geral

Este projeto visa, em um ambiente de nuvem federada, definir e a implementar uma política de integridade e de confidencialidade dos arquivos, tanto no momento que eles são transferidos para as nuvens externas, quanto a qualquer momento em que se desejar verificar se os arquivos continuam íntegros e secretos.

1.3.2 Objetivos Específicos

Para que o objetivo geral apresentado seja atingido, faz-se necessário cumprir os seguintes objetivos específicos:

• Definir uma arquitetura para a política de integridade para quando os arquivos forem transferidos;

• Definir uma arquitetura para a política de integridade para verificar se os arquivos foram modificados durante os seus respectivos armazenamentos;

• Definir uma política de confidencialidade que mantenha secretos os arquivos manipulados pelo BioNimbuZ;

• Implementar uma política de integridade para quando os arquivos forem transferidos e para verificar se os arquivos foram danificados durante os seus respectivos armazenamentos;

• Discutir os resultados obtidos, verificando as vantagens e as desvantagens da política de integridade que foi implementada.

O próximo capítulo descreve o contexto da Computação em Nuvem, ambiente em que este projeto foi desenvolvido, detalhando sua história, suas principais características e seus desafios futuros.

Capítulo 2

Computação em Nuvem

O objetivo deste capítulo é descrever os conceitos básicos relacionados a computação em nuvem. Serão descritos conceitos gerais sobre esse paradigma e também características específicas de cada serviço prestado por essa tecnologia. Em seguida, na Seção 2.1, serão detalhados os conceitos de computação em nuvem, o surgimento da ideia e um panorama geral sobre essa tecnologia. Após essa parte, tem-se a Seção 2.2, na qual são apresentadas as principais características que envolvem a computação em nuvem. Na Seção 2.3, são detalhados os modelos de serviços que são padrões arquiteturais para soluções em nuvem. A Seção 2.4 traz os conceitos de papéis dentro de um ambiente em nuvem. Após essa seção, a Seção 2.5 detalha os tipos de nuvem que hoje em dia existem no mercado. Por último, na Seção 2.6 são apresentados alguns conceitos relacionados a federação de nuvens.

2.1 Conceitos Básicos

Nos anos 60, a computação entrava na terceira geração de computadores, na qual os transistores foram substituídos por circuitos integrados, que possibilitaram um grande aumento de processamento de dados, diminuição do tamanho dos computadores e do consumo de energia elétrica. Com esse aumento no poder computacional e a diminuição no consumo de energia pelos computadores, surgiu a ideia de que a computação fosse vista como uma utilidade pública, como os serviços básicos de água, de energia, de gás e de telefone. Assim, o poder de processamento computacional e de armazenamento poderiam ser disponibilizados sob demanda [42].

O crescimento em poder computacional e em capacidade de armazenamento alcançados pelos computadores daquela época era cada vez maior, e a quantidade de problemas complexos que estava sendo tratado por cientistas, físicos, biólogos e engenheiros também estava crescendo cada vez mais, exigindo uma demanda computacional que muitas vezes não era disponibilizada por uma única máquina [42].

Essa realidade começou a ser transformada a partir da década de 70, quando surgiram os sistemas computacionais distribuídos, que evoluíram naturalmente, substituindo os sistemas centralizados (os chamados *mainframes*) [25].

Na década de 80, motivadas pelo alto custo de aquisição e manutenção de máquinas paralelas, assim como pela dependência estabelecida entre o usuário e o fabricante da máquina, as comunidades envolvidas com pesquisa em sistemas distribuídos e computação paralela propuseram a utilização de sistemas distribuídos como plataforma de execução

Figura 2.1: Visão Geral de Computação em Nuvem [36].

paralela. Isso concretizou uma melhor relação custo/benefício para a computação paralela, pois proporcionou menor custo de implantação e maior poder computacional a uma ampla variedade de aplicações [25].

Na década de 90, a Internet causou uma mudança drástica nos conceitos do mundo da computação, começando com o conceito de computação paralela, mudando para a computação distribuída, depois para a computação em grade e, recentemente, a computação em nuvem. Muitas empresas da área de tecnologia da informação entraram nesse novo paradigma realizando implementações na nuvem. A Amazon desempenhou um papel fundamental lançando o seu Amazon Web Services (AWS) [2] em 2006. Em seguida, a Google e a IBM também anunciaram planos, investimentos e começaram projetos de pesquisa em nuvem, buscando melhorar a utilização de recursos computacionais [41]. A Visão Geral de um ambiente em nuvem pode ser visto na Figura 2.1.

A infraestrutura do ambiente de computação em nuvem, normalmente, é composta por um grande número, centenas ou milhares, de máquinas físicas ou nós físicos, conectadas por meio de uma rede, como ilustra a Figura 2.2. Cada máquina física tem as mesmas configurações de software, mas pode ter variação na capacidade de hardware em termos de CPU, memória e armazenamento em disco. Dentro de cada máquina física existe um número variável de máquinas virtuais ou nós virtuais em execução, de acordo com a capacidade do hardware disponível na máquina física [56].

No cenário da computação em nuvem, ocorre uma alternativa na forma de utilização dos serviços. Ao contrário do cenário no qual o usuário coloca serviços para serem feitos por um servidor local, ele contrata um provedor para que seja realizado todos esses serviços. Estes serviços podem ser de armazenamento de dados, de utilização da memória e de poder de processamento da provedora para rodar aplicações, e também infraestrutura para hospedar serviços.

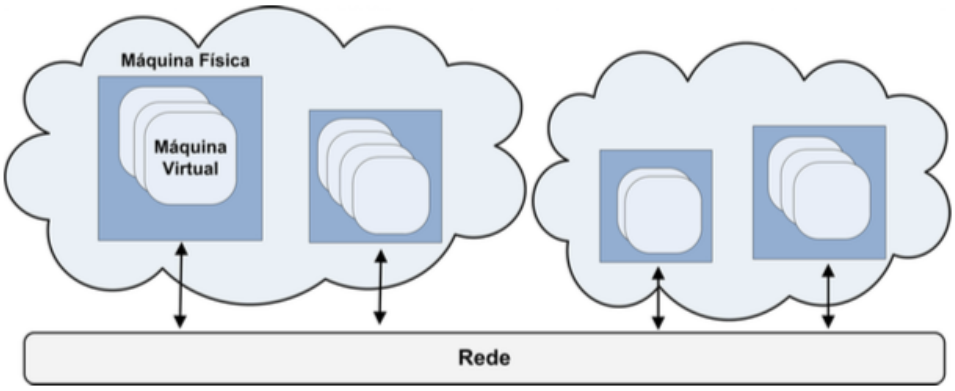

Figura 2.2: Ambiente em Nuvem [53].

2.2 Características Essenciais

As características essenciais são vantagens que as soluções de computação em nuvem oferecem. Algumas destas características, em conjunto, definem exclusivamente a computação em nuvem e fazem a distinção com relação a outros paradigmas [56]. Por exemplo, a elasticidade rápida de recursos, amplo acesso e medição de serviço são características básicas para compor uma solução de computação em nuvem. Assim, as principais características serão descritas a seguir:

- Serviço sob Demanda:

O usuário pode adquirir unilateralmente recurso computacional, como tempo de processamento no servidor ou armazenamento na rede, na medida em que necessite, e sem precisar de interação humana com os provedores de cada serviço. O hardware e o software dentro de uma nuvem podem ser automaticamente reconfigurados, orquestrados e estas modificações são apresentadas de forma transparente para os usuários, que possuem perfis diferentes, e assim podem personalizar os seus ambientes computacionais, por exemplo, instalação de software e configuração de rede para a definição de determinados privilégios.

- Acesso Amplo:

Recursos são disponibilizados por meio da rede e acessados através de mecanismos padronizados que possibilitam o uso por plataformas *thin* ou *thin client*, tais como celulares, laptops e PDAs [56]. A interface de acesso à nuvem não obriga os usuários a mudar suas condições e ambientes de trabalho, como por exemplo: linguagens de programação e sistema operacional. Já os sistemas de software clientes instalados localmente para o acesso à nuvem são leves, como um navegador de Internet.

- *Pooling* de Recursos:

Os recursos computacionais do provedor são organizados em um *pool* para servir múltiplos usuários usando um modelo *multi-tenant* ou multi-inquilino [40], com diferentes

recursos físicos e virtuais, dinamicamente, atribuídos e ajustados de acordo com a demanda dos usuários. Estes usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração, tais como o país, o estado ou o centro de dados.

• Elasticidade:

Recursos podem ser adquiridos de forma rápida e elástica, em alguns casos automaticamente, caso haja a necessidade de escalar com o aumento da demanda, e liberados na retração dessa demanda. Para os usuários, os recursos disponíveis para uso parecem ser ilimitados e podem ser adquiridos em qualquer quantidade e a qualquer momento. A virtualização auxilia a elasticidade rápida na computação em nuvem, criando várias instâncias de recursos requisitados por meio, por exemplo, de um único recurso real [22]. Além disso, a virtualização é uma maneira de abstrair características físicas de uma plataforma computacional dos usuários, exibindo outro hardware virtual e emulando um ou mais ambientes que podem ser independentes ou não.

• Serviço Medido:

Sistemas em nuvem automaticamente controlam e otimizam o uso de recursos por meio de uma capacidade de medição. A automação é realizada em algum nível de abstração apropriado para o tipo de serviço, tais como armazenamento, processamento, largura de banda e contas dos usuários ativos. O uso de recursos pode ser monitorado e controlado, possibilitando transparência para o provedor e para o usuário do serviço utilizado. Para garantir a qualidade do serviço ou *Quality of Service (QoS)*, pode-se utilizar a abordagem baseada em Acordo de Nível de Serviço ou *Services Level Agreement (SLA)*. O SLA fornece informações sobre os níveis de disponibilidade, de funcionalidade, de desempenho ou outros atributos do serviço, como o faturamento e até mesmo as penalidades em caso de violação destes níveis.

2.3 Modelos de Serviços

O ambiente de computação em nuvem é composto de três modelos de serviços. Estes modelos são importantes, pois eles definem um padrão arquitetural para soluções de computação em nuvem.

2.3.1 Software como um Serviço (SaaS)

O modelo de SaaS proporciona sistemas de software com propósitos específicos que estão disponíveis para os usuários através da Internet. Os sistemas de software são acessíveis a partir de vários dispositivos do usuário por meio de uma interface *thin client* como um navegador Web. No SaaS o usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais, armazenamento ou mesmo as características individuais da aplicação, exceto configurações específicas. Com isso, os desenvolvedores se concentram em inovação e não na infraestrutura, levando ao desenvolvimento rápido de sistemas de software [53].

Como o software está na Web, ele pode ser acessado pelos usuários de qualquer lugar e a qualquer momento, permitindo maior integração entre unidades de uma mesma empresa ou outros serviços de software. Assim, novos recursos podem ser incorporados automaticamente aos sistemas de software sem que os usuários percebam estas ações, tornando transparente a evolução e atualização dos sistemas. O SaaS reduz os custos, pois é dispensada a aquisição de licenças de sistemas de softwares.

2.3.2 Plataforma como um Serviço (PaaS)

O modelo PaaS oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem. O usuário não administra ou controla a infraestrutura subjacente, incluindo rede, servidores, sistemas operacionais ou armazenamento, mas tem controle sobre as aplicações implantadas e, possivelmente, as configurações das aplicações hospedadas nesta infraestrutura. O PaaS fornece sistema operacional, linguagens de programação e ambientes de desenvolvimento para as aplicações, auxiliando a implementação de sistemas de software, já que contém ferramentas de desenvolvimento e de colaboração entre os desenvolvedores [56].

Em geral, os desenvolvedores dispõem de ambientes escaláveis, mas eles têm que aceitar algumas restrições sobre o tipo de software que se pode desenvolver, desde limitações que o ambiente impõe na concepção das aplicações até a utilização de sistemas de gerenciamento de banco de dados (SGBDs) do tipo chave-valor, ao invés de SGBDs relacionais. Do ponto de vista do negócio, a PaaS permitirá aos usuários utilizarem serviços de terceiros, aumentando o uso do modelo de suporte no qual os usuários se inscrevem para solicitações de serviços de TI ou para resoluções de problemas pela Web. Com isso, pode-se melhorar o gerenciamento do trabalho e as responsabilidades das equipes de TI das empresas.

2.3.3 Infraestrutura como um Serviço (IaaS)

O IaaS é a parte responsável por prover toda a infraestrutura necessária para o PaaS e o SaaS. O principal objetivo do IaaS é tornar mais fácil e acessível o fornecimento de recursos, tais como servidores, rede, armazenamento e outros recursos de computação fundamentais para construir um ambiente sob demanda, que podem incluir sistemas operacionais e aplicativos. O IaaS possui algumas características, tais como uma interface única para administração da infraestrutura, *Application Programming Interface (API)* para interação com *hosts*, *switches*, balanceadores, roteadores e o suporte para a adição de novos equipamentos de forma simples e transparente. Em geral, o usuário não administra ou controla a infraestrutura da nuvem, mas tem controle sobre os sistemas operacionais, armazenamento e aplicativos implantados, e, eventualmente, seleciona componentes de rede, tais como *firewalls* [56].

O termo IaaS se refere a uma infraestrutura computacional baseada em técnicas de virtualização de recursos de computação. Esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. Do ponto de vista de economia e aproveitamento do legado, ao invés de comprar novos servidores e equipamentos de rede para a ampliação de serviços, pode-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente de forma dinâmica.

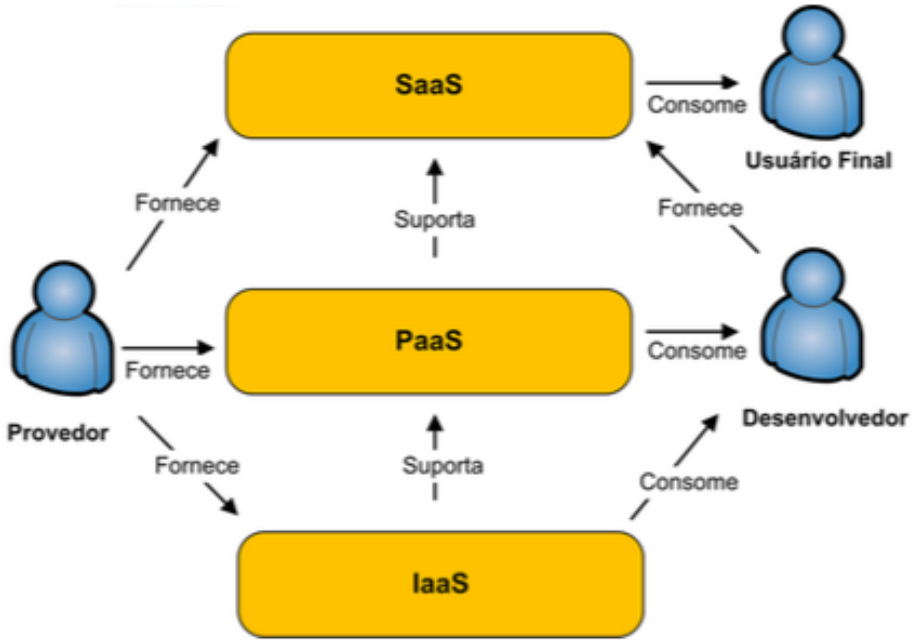

Figura 2.3: Papéis dentro de uma Nuvem [45].

2.4 Papéis na Computação em Nuvem

Os papéis são importantes para definir responsabilidades, acesso e perfil para os diferentes usuários que fazem parte, e estão envolvidos em uma solução de computação em nuvem [45]. Para entender melhor a computação em nuvem, pode-se classificar os atores dos modelos de acordo com os papéis desempenhados. A Figura 2.3 destaca estes papéis.

O provedor é responsável por disponibilizar, gerenciar e monitorar toda a estrutura para a solução de computação em nuvem, deixando o desenvolvedor e o usuário final sem esse tipo de responsabilidade, e fornecendo serviços nos três modelos de serviços. Os desenvolvedores utilizam os recursos fornecidos e disponibilizam serviços para os usuários finais. Esta organização em papéis ajuda a definir os atores e os seus diferentes interesses. Os atores podem assumir vários papéis ao mesmo tempo de acordo com os interesses, sendo que apenas o provedor fornece suporte a todos os modelos de serviços.

Do ponto de vista de interação entre os três modelos de serviços, a IaaS fornece recursos computacionais, seja de hardware ou de software, para o modelo PaaS, que por sua vez fornece recursos, tecnologias e ferramentas para o desenvolvimento e execução dos serviços implementados, a serem disponibilizados na visão de SaaS.

2.5 Tipos de Nuvem

As nuvens podem ser classificadas em quatro tipos de implementação, conforme demonstrado na Figura 2.4. A escolha de qual é o tipo mais adequado vai depender da necessidade da aplicação que será implementada. Os tipos de implementação de nuvens são públicas, privadas, híbridas e comunitárias [32]:

Figura 2.4: Tipos de Nuvem [45]

• Nuvem Pública: na nuvem pública todos os serviços podem ser acessados por qualquer usuário ou organização. As nuvens públicas possuem a vantagem de terem uma maior escalabilidade de recursos, evitando o inconveniente dos usuários terem que comprar equipamentos para a nuvem, caso ocorra uma necessidade temporária de um poder de processamento maior. Assim, geralmente, nuvens públicas são uma boa alternativa quando:

- Se deseja testar e desenvolver o código do aplicativo;
- Projetos colaborativos estão sendo desenvolvidos;
- Um aplicativo SaaS na nuvem tem uma estratégia de segurança bem implementada;
- Um aditivo do aplicativo é necessário por conta do horário (a capacidade de adicionar mais recursos à nuvem, em horários de pico, para não ocorrer problemas de indisponibilidade do serviço por conta do grande número de acessos).

• Nuvem Privada: é a nuvem que está dentro de um contexto empresarial e não está acessível a todas as pessoas, sendo restrita apenas aos funcionários e aos parceiros da empresa. Nesse caso, a infraestrutura utilizada pertence ao usuário, assim ele possui total controle sobre os aplicativos que são implementados. Geralmente, elas são construídas sobre *datacenters* privados, o que faz com que o usuário tenha que comprar e manter todo o software e infraestrutura da nuvem. É uma boa escolha quando o negócio em questão são os dados e as aplicações da empresa, e o controle e a segurança são fundamentais.

• Nuvem Híbrida: é a combinação de nuvens públicas com privadas. Uma vantagem é que uma nuvem privada pode ter os seus recursos ampliados a partir de recursos de uma nuvem pública. Apesar dessa vantagem, a complexidade para se determinar a maneira como as aplicações serão distribuídas não é irrelevante. Uma grande quantidade de dados para serem processados em uma nuvem pública pode não ser favorável, pois pode ser muito custoso passar essa quantidade de uma nuvem privada para uma nuvem pública.

• Nuvem Comunitária: infraestrutura que é compartilhada por organizações que mantém algum tipo de interesse em comum (jurisdição, segurança, economia), e pode ser administrada, gerenciada e operada por uma ou mais destas organizações;

Apesar de a computação em nuvem ser mais escalável que outros sistemas distribuídos, atualmente, ela já não é mais suficiente para suprir sozinha a demanda por recursos computacionais. A solução então foi integrar mais de uma nuvem computacional, o qual

foi chamado de federação de nuvens, que será descrita em detalhes na próxima seção.

2.6 Federação de Nuvens

Com o objetivo de suportar um grande número de consumidores de serviços de todo o mundo, os provedores de infraestrutura estabeleceram centros de dados em múltiplas localizações geográficas para fornecerem redundância e assegurar contabilidade em casos de falhas. A Amazon, por exemplo, possui centros de dados nos EUA e na Europa. No entanto, atualmente, eles esperam que seus clientes na nuvem (os provedores de SaaS) expressem uma preferência sobre o local que desejam hospedar os seus dados. Eles não fornecem mecanismos automáticos para escalar os seus serviços hospedados em vários centros de dados distribuídos geograficamente. Este tipo de abordagem apresenta algumas deficiências:

- É difícil para os clientes da nuvem determinarem com antecedência a melhor localização para hospedar os seus serviços;
- Os provedores de SaaS podem não ser capazes de atender as expectativas de qualidade de serviço dos consumidores que provenham de múltiplas localidades geográficas.

Segundo Buyya [31], isso exige a construção de mecanismos para uma federação de provedores de nuvens, com o objetivo de cumprir metas de Qualidade de Serviço (QoS). Além disso, nenhum fornecedor de infraestrutura em nuvem será capaz de estabelecer o seu *datacenter* em todos os locais do Mundo. Como os fornecedores de aplicativos nas nuvens terão dificuldades de cumprir as expectativas de Qualidade de Serviço (QoS) para todos os consumidores, ocorre a tentativa de realizar o melhor uso dos serviços de múltiplos provedores de infraestrutura em nuvem, para oferecer um suporte melhor para os usuários de acordo com a sua necessidade.

Nesse cenário, uma federação de nuvens é uma forma de interligar ambientes de computação em nuvem de dois ou mais provedores, com a finalidade de controlar e balancear a demanda computacional [31]. Para isso, é necessário um provedor para organizar e gerenciar recursos de uma nuvem para outra, esses podem ser temporários ou permanentes, dependendo do acordo realizado entre as nuvens federadas. Uma vantagem da federação é a utilização destes recursos que estariam ociosos de uma nuvem para a outra, pois não ocorreria nenhum desperdício.

A federação de nuvens passou por duas fases, sendo que existem mais duas que estão sendo trabalhadas para serem melhoradas e padronizadas. Na Figura 2.5 tem-se as quatro fases. Na primeira nota-se os *datacenters* sem a utilização da tecnologia de nuvem, ou seja, eles estão espalhados pelo mundo e realizam todo o trabalho sozinhos, contando apenas com a infraestrutura que dispõem localmente. Não ocorre cooperação com nenhum outro *datacenter*.

A segunda fase é a etapa na qual as nuvens passam a fornecer os seus serviços. Na fase três, as nuvens já conseguem trocar recursos entre si por meio de um provedor virtual que realiza todo o balanceamento das nuvens, de forma que nenhuma fique ociosa enquanto há outras que se encontram com a utilização máxima de seus recursos. Nesse caso, uma solicitação seria enviada para a nuvem com recursos ociosos, e esses seriam utilizados pela nuvem sobrecarregada.

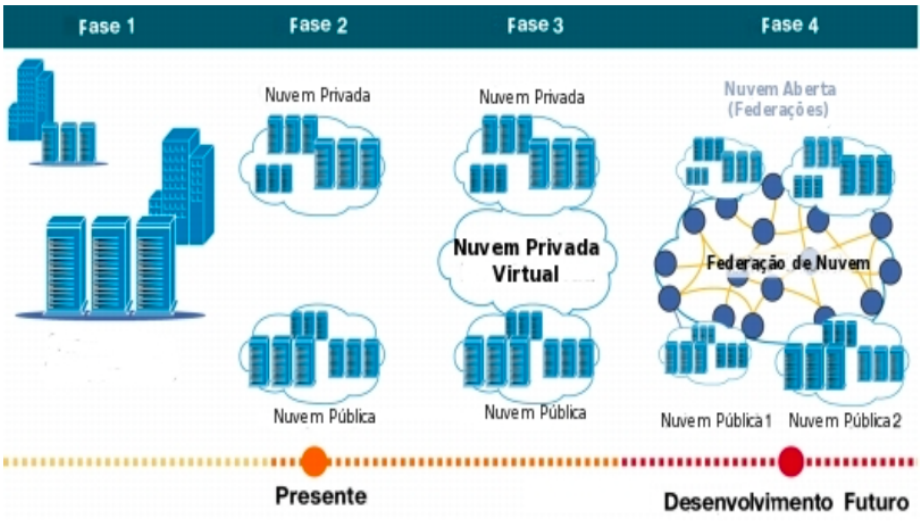

Figura 2.5: Fases da Computação em Nuvem [56].

Na última fase, encontram-se as várias federações espalhadas pelo mundo, e essas federações trabalhariam interoperando entre si, ou seja, dividindo os seus recursos da mesma forma que uma nuvem dentro de uma federação faz. Nesse último caso, formaria um cenário onde os recursos ociosos e os desperdiçados na nuvem seriam pequenos. Uma outra vantagem também seria o possível balanceamento na carga entre as nuvens e federações, fazendo com que os usuários tenham uma ótima experiência de uso das aplicações hospedadas nas nuvens.

Assim, tendo em vista que o tema deste projeto é trabalhar em um ambiente de nuvens federadas, o próximo capítulo visa explicar detalhamente como é que como funciona a plataforma alvo deste trabalho - BioNimbuZ.

Capítulo 3

BioNimbuZ

Este capítulo tem como objetivo apresentar a plataforma para federação de nuvens computacionais chamada BioNimbuz [53]. Serão detalhados seu funcionamento, sua arquitetura e os seus serviços e controladores, sendo eles: o Controlador de *Jobs*, o Controlador de SLA, o Serviço de Monitoramento, o Serviço de Descobrimento, o Serviço de Escalonamento, o Serviço de Armazenamento, o Serviço de Tarifação, o Serviço de Tolerância a Falhas e, por fim, o Serviço de Segurança.

3.1 Visão Geral

O BioNimbuZ é uma arquitetura para federação de nuvens híbridas, que foi proposta originalmente por Saldanha [53] e vem sendo aprimorada constantemente em outros trabalhos [45] [48] [35]. O BioNimbuZ foi desenvolvido para suprir a demanda pela federação de nuvens, visto que a utilização de nuvens de forma isolada já não estava atendendo mais às necessidades de processamento, de armazenamento, entre outros, na execução das aplicações de bioinformática, área que é o foco desta arquitetura.

O BioNimbuZ permite a integração entre nuvens de diversos tipos, tanto privadas quanto públicas, deixando com que cada provedor mantenha suas características e políticas internas, e oferendo ao usuário a transparência e a ilusão de se ter uma infinidade de recursos. Desta forma, o usuário pode usufruir de diversos serviços sem se preocupar com qual provedor está sendo de fato utilizado. Outra característica desta arquitetura é a flexibilidade na inclusão de novos provedores, pois são utilizados *plugins* de integração que se encarregam de mapear as requisições vindas da arquitetura para as requisões de cada provedor especificamente. Isso é fundamental para que alguns objetivos possam ser alcançados, principalmente em relação à escalabilidade e à flexibilidade.

A arquitetura original do BioNimbuZ pode ser vista na Figura 3.1. Ela representa a interação entre as três camadas principais: Aplicação, Núcleo e Infraestrutura que serão descritas com detalhes na Seção 3.2.

Originalmente, toda a comunicação existente entre as camadas e os serviços era realizada por meio de uma rede Peer-to-Peer (P2P) [58]. Porém, para alcançar os objetivos desejados de escalabilidade e de flexibilidade, percebeu-se a necessidade de alterar a forma de comunicação entre os componentes da arquitetura do BioNimbuZ, pois a utilização de uma rede de comunicação Peer-to-Peer (P2P) não estava mais suprindo as necessidades nestes dois quesitos.

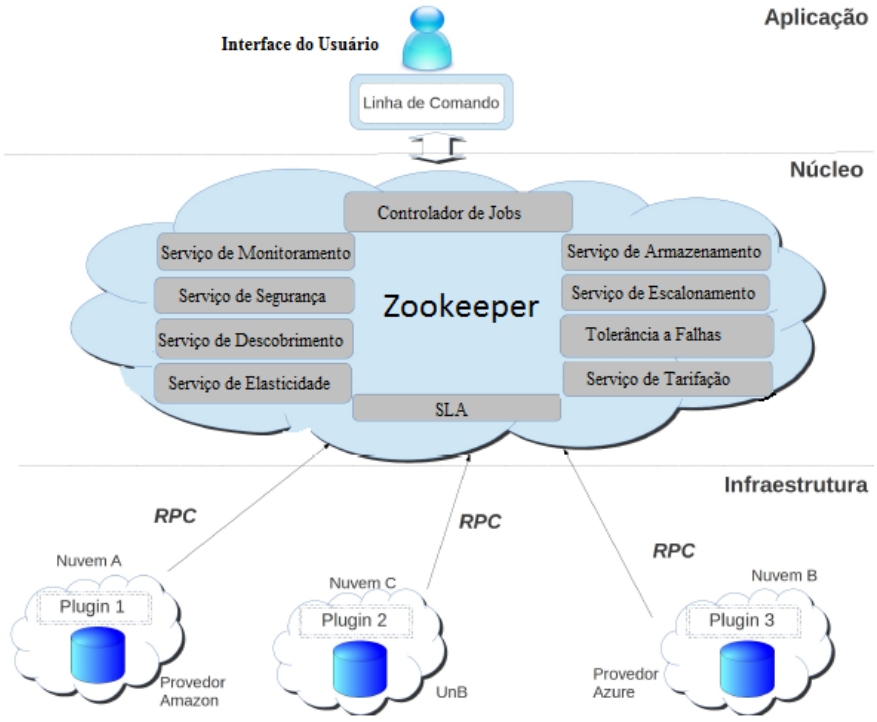

Figura 3.1: Arquitetura do BioNimbuz [45].

E para auxiliar na organização e na coordenação do BioNimbuZ, foi utilizado um serviço voltado à sistemas distribuídos chamado ZooKeeper [39], da Fundação Apache [3]. A seguir, será explicado de forma detalhada o funcionamento do Zookeeper e do Avro [5].

3.1.1 Apache Avro

O Apache Avro [5] é um sistema de *Remote Procedure Calls* (RPC) e de serialização de dados desenvolvido pela Fundação Apache. Algumas vantagens deste sistema são o fato de ser livre, a utilização de mais de um protocolo de transporte de dados em rede, e o suporte a mais de um formato de serialização de dados. Quanto ao formato dos dados, existe o suporte aos dados binários e aos dados no formato JSON [7]. E quanto aos protocolos, podem ser utilizados tanto o protocolo HTTP [6] como um próprio do Avro.

O Avro foi criado para ser utilizado com um grande volume de dados e possui algumas características definidas pela própria Fundação Apache, como uma rica estrutura de dados com tipos primitivos, um formato de dados compacto, rápido e binário, e integração de forma simples com diversas linguagens de programação.

O Avro foi escolhido como sistema de Chamada de Procedimento Remoto, no BioNimbuZ, por ser livre, ser flexível e possuir integração com várias linguagens.

3.1.2 Zookeeper Apache

O Zookeeper é um serviço de coordenação de sistemas distribuídos, criado pela Fundação Apache, para ser de fácil manuseio. Ele utiliza um modelo de dados que simula uma

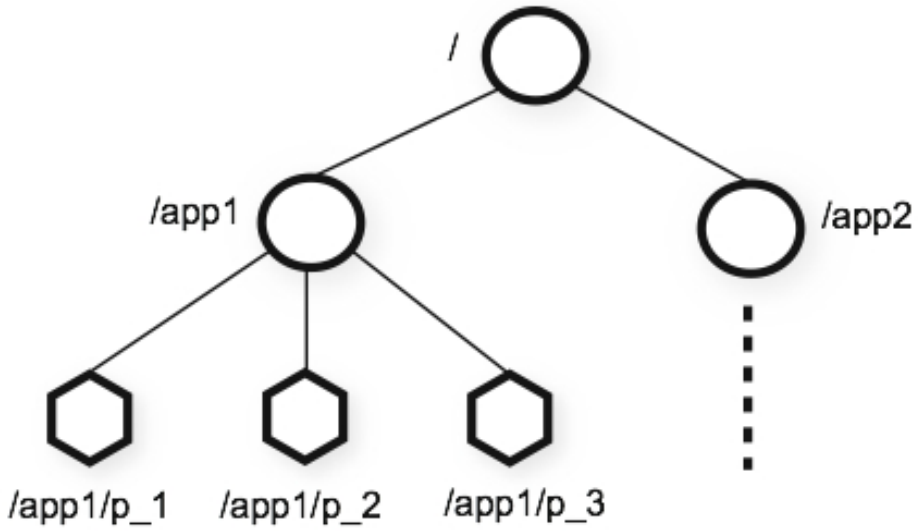

Figura 3.2: Estrutura Hierárquica dos *Znodes* [45].

estrutura de diretórios, e tem como finalidade facilitar a criação e a gestão de sistemas distribuídos, que podem ser de alta complexidade e de difícil coordenação e manutenção. São utilizados espaços de nomes chamados de *znodes*, que são organizados de forma hierárquica, assim como ocorre nos sistemas de arquivos. Cada *znode* tem a capacidade de armazenar no máximo 1 megabyte (MB) de informação, e é identificado pelo seu caminho na estrutura. Neles, podem ser armazenadas informações que facilitem o controle do sistema distribuído, como metadados, caminhos, dados de configuração e endereços. Existem dois tipos de *znodes*, os persistentes e os efêmeros. O primeiro continua a existir mesmo depois da queda de um provedor, sendo útil para armazenar informações a respeito dos *jobs* que foram executados e outros tipos de dados que não se deseja perder. Já os efêmeros, podem ser criados para cada novo partipante da federação, pois assim que este novo participante ficar indisponível, o *znode* efêmero referente a ele será eliminado e todos os outros componentes do sistema distribuído saberão que ele não se encontra disponível. Na Figura 3.2 pode ser visto um exemplo da estrutura hierárquica dos *znodes*. Nesse modelo, pode ser visto a árvore de diretórios padrão do ZooKeeper.

Além disso, o Zookeeper suporta o conceito de *watchers*, que funcionam como observadores de mudanças, e enviam alertas sobre alterações ocorridas em algum dos *znodes*. Este conceito é útil para sistemas distribuídos, pois permitem o monitoramento constante do sistema, deixando para que *watchers* avisem quando um provedor estiver fora do ar, por exemplo, ou então quando um novo recurso estiver disponível.

3.2 Arquitetura do BioNimbuZ

O BioNimbuZ faz uso de uma arquitetura hierárquica distribuída, conforme apresentada na Figura 3.1. Ela representa a interação entre as três camadas principais: Aplicação, Núcleo e Infraestrutura. A primeira camada - aplicação, permite a integração entre a aplicação do usuário e os serviços do núcleo da plataforma. A interface com o usuário pode ocorrer tanto através de linha de comando ou por uma interface gráfica, e é nela que

devem ser inseridos os *workflows* que serão executados nas diversas nuvens. A aplicação então se comunica com o núcleo que é o responsável por realizar toda a gerência, e por se comunicar com os *plugins* de cada provedor. Os *puglins* mapeiam as requisições e as enviam para cada provedor. A seguir serão descritos o funcionamento e as características de cada uma destas camadas.

3.2.1 Camada de Aplicação

A camada de aplicação é a responsável por fazer toda a comunicação com o usuário. É nela que os usuários devem inserir as ações que desejam executar na federação. A interação com o usuário pode ser realizada através de páginas web, linhas de comando, interface gráfica (GUI) ou *Workflows Management Systems (WfMS)*. Assim que a tarefa for concluída, é função desta camada mostrar o *feedback* da execução, sinalizando para o usuário se a tarefa foi completada com sucesso ou não. As principais ações que um um usário pode querer executar na federação são: *upload* de arquivos, listagem de arquivos, *download* de arquivos, submissão de *jobs*, consulta de *jobs*, cancelamento de *jobs*, listagem das faturas e visualização dos resultados dos *jobs* executados.

3.2.2 Camada de Núcleo

O núcleo é o responsável por toda a gerência da federação, tendo como atividades principais o descobrimento de provedores e recursos, o escalonamento, o acompanhamento de tarefas, o armazenamento de arquivos e o controle de acesso dos usuários, entre outras. Para cada uma destas atividades existe um serviço responsável, e novos serviços podem ser incorporados à medida em que forem necessários. A seguir serão descritos os serviços de monitoramento, de descobrimento, de escalonamento, de armazenamento, de tolerância a falhas, de elasticidade, de tarifação e de segurança, e mais dois controladores: o controlador de SLA (*Service Level Agreement*) e o controlador de *jobs*:

• Controlador de *Jobs*: é quem recebe a requisição que vem da aplicação e manda para o núcleo, verificando as credenciais dos usuários, com o serviço de segurança, para permitir ou não a execução do *workflow*. É responsável por gerenciar os vários pedidos e garantir que os resultados sejam entregues de forma correta para cada uma das requisições, e mantê-los para que possam ser consultados, posteriormente.

• Controlador de SLA: quando o usuário submete uma tarefa para a federação por meio da camada de aplicação, ele deve preencher um *template* de SLA, que representa, entre outras coisas, os parâmetros de QoS (*Quality-of-Service*) que ele deseja. Estes parâmetros podem descrever desde requisitos funcionais, como número de núcleos de CPU, tamanho de memória, tamanho de armazenamento, até requisitos não funcionais, como custo a pagar e taxa de transferência. O controlador de SLA é o responsável por implementar o ciclo de vida de SLA, que compreende seis atividades: descoberta de provedores de serviço, definição de SLA, estabelecimento do acordo, monitoramento de violação do acordo, término de acordo e aplicação de penalidades por violação. Para cada pedido de execução feito por um usuário, este controlador deve verificar junto à federação se ela pode suportar os parâmetros naquele momento, e isto é feito através dos *plugins* de cada

provedor. O pedido de execução junto com o *template* de SLA é repassado ao serviço de monitoramento que se encarrega de enviar as tarefas a serem executadas ao serviço de escalonamento e verificar se as mesmas foram executadas com sucesso ou não.

• Serviço de Monitoramento: é o serviço que recebe os pedidos de execução dos *jobs* provenientes do controlador de *jobs*. Para cada um destes pedidos de execução, o serviço de monitoramento verifica junto aos provedores de nuvem se aquele serviço está disponível e envia o pedido para o serviço de escalonamento. Após o envio, o serviço monitora o andamento das tarefas, para garantir que todas serão executas com sucesso, e ao fim da execução avisa ao controlador de *jobs* se houve êxito na realização das mesmas. Este serviço, portanto, está em constante contato com o controlador de *jobs*, recebendo pedidos de execução e respondendo sobre o estado dos mesmos. Este serviço está em contato também com o controlador de SLA, verificando se a federacão suporta os parâmetros definidos, e com o serviço de escalonamento, enviando os pedidos de execução das tarefas.

• Serviço de Descobrimento: é o serviço que identifica e mantém informações a respeito dos provedores de nuvem que estão na federação, como capacidade de armazenamento, processamento e latência de rede, e também mantém detalhes sobre parâmetros de execução e arquivos de entrada e de saída. Para obter estas informações a respeito dos provedores, o Zookeeper é consultado, pois todos os provedores presentes na federação possuem *znodes* que armazenam seus dados. Sempre que uma modificação é realizada, como a saída ou a entrada de algum provedor, os *watchers* alertam os outros serviços, mantendo assim todos os participantes da federação atualizados.

• Serviço de Escalonamento: é o serviço que recebe o pedido para a execução de alguma tarefa - aqui chamado de *job* - e as distribui dinamicamente entre os provedores disponíveis, divididas em instâncias menores - *tasks*. O serviço de escalonamento também é responsável por acompanhar toda a execução do *job*, e manter um registro das execuções já escalonadas. Para realizar a distribuição de tarefas na federação, algumas métricas são levadas em consideração, como latência, balanceamento de carga, tempo de espera e capacidade de processamento, entre outras, visando atender o que foi determinado no acordo de SLA. O serviço de escalonamento pode ser visto em mais detalhes no trabalho de Oliveira [48].

• Serviço de Armazenamento: responsável pela estratégia de armazenamento dos arquivos que são utilizados ou mantidos pelas aplicações. O armazenamento deve ocorrer de forma eficiente, para que as aplicações possam utilizar os arquivos com o menor custo possível. Este custo é calculado utilizando-se algumas métricas, tais como latência de rede, distância entre os provedores e capacidade de amarazenamento. Outra estratégia adotada no armazenamento dos arquivos é a replicação em mais de um provedor, para garantir que os arquivos estejam disponíveis quando as aplicações forem utilizá-los. A política de armazenamento que está implementada atualmente no BioNimbuZ foi introduzida no trabalho do Bacelar e Moura [45].

• Serviço de Tolerância a Falhas: tem como objetivo garantir que todos os principais serviços estejam sempre disponíveis, e em caso de falhas, possa ser feito uma recupera-

ção. Essa recuperação exige que todos os objetos que forem afetados pela falha voltem ao estado anterior. O serviço de tolerância a falhas possui atuação de forma distribuída na plataforma BioNimbuZ, e está presente em outros serviços. No Serviço de Armazenamento, quando um alerta de indisponibilidade de um recurso é lançado por um *watcher* , é iniciado uma rotina de recuperação para os arquivos que este recurso continha. Como os arquivos são armazenados de forma duplicada na federação, a recuperação ocorre de forma a identificar quais arquivos foram perdidos e realizar uma nova duplicação em outro servidor do BioNimbuZ. No Serviço de Escalonamento, a recuperação está presente quando é recebido um alerta de indisponiblidade de um determinado recurso, e todas as tarefas que foram escalonadas e ainda não haviam sido executadas, ou estavam em execução, são novamente escalonadas, agora para uma outra máquina na federação. Essas recuperações são possíveis devido aos *watchers*, que disparam alerta.

• Serviço de Elasticidade: é o serviço responsável por dinamicamente aumentar ou diminuir o número de instâncias de máquinas virtuais, ou então reconfigurar os parâmetros de utilização de CPU, de memória, de largura de banda, entre outros recursos que são disponibilizadas pelos provedores de nuvem, a fim de obter uma melhor utilização da infraestrutura disponível. A elasticidade pode ser vertical, quando há um redimensionamento dos atributos de CPU, de armazenamento, de rede ou de memória. Mas, a elasticidade também pode ser horizontal, quando o número de instâncias de máquinas virtuais é modificado para mais ou para menos.

• Serviço de Tarifação: é o serviço responsável por calcular o quanto os usuários devem pagar pela utilização dos serviços oferecidos na plataforma BioNimbuZ. Para que isto seja possível, este serviço se mantém em constante contato com o serviço de monitoramento, para obter informações, tais como o tempo de execução e a quantidade de máquinas virtuais alocadas para a tarefas que foram executadas.

• Serviço de Segurança: segurança em nuvens federadas é uma área de estudo em constante evolução e o serviço de segurança deve trabalhar em diversos pontos para fornecer um serviço seguro. No BioNimbuZ esse serviço já fornece a autenticação de usuários, ou seja, verificar se o usuário, que está tentando acessar algum recurso na federação é quem ele realmente diz ser. Além disso, o serviço fornece a autorização, que consiste em verificar se o usuário pode realizar as ações que deseja. Nesse contexto, o foco deste projeto é trabalhar a criptografia de mensagens para garantir a confidencialidade na troca de informações entre provedores, e também a verificação de integridade de arquivos, de modo que seja possível garantir que um arquivo não seja alterado por fatores externos à federação. Dessa forma, os serviços de integridade e confidencialidade propostos neste trabalho para a plataforma de nuvens federadas BioNimbuZ serão descritos em detalhes no capítulo 5.

3.2.3 Camada de Infraestrutura

A camada de infraestrutura consiste em todos os recursos que os provedores de nuvens colocam à disposição da federação, somados aos seus respectivos *plugins* de integração. O principal objetivo desta camada é prover uma interface de comunicação entre o BioNim-

buZ e os provedores de nuvens, utilizando para tal, os *plugins* que mapeiam as requisições provenientes da arquitetura, para comandos que cada provedor, individualmente, seja capaz de entender e executar. Esses *plugins* garantem que o BioNimbuZ crie uma plataforma de nuvens federadas com uma interface independente aos provedores de nuvens, pois nenhuma adaptação é necessária por parte de cada provedor.

Contudo, não adianta um sistema ser escalável, flexível e tolerante a falhas, se ele não garante a segurança das informações dos usuários. Um arquivo pode ser danificado durante o seu *upload* e também pode ser acessado por outra pessoa que não é autorizada para tal, expondo informações que podem ser secretas. Nesse contexto, o próximo capítulo abordará temas relevantes a segurança em ambientes em nuvem, tema principal deste projeto.

Capítulo 4

Segurança em Nuvem Federada

O objetivo deste capítulo é apresentar os principais problemas envolvendo segurança em nuvem federada, os quais estão relacionados à infraestrutura, à virtualização e também à aplicação. Assim, serão discutidos pontos que englobam a confiança, a integridade, a confidencialidade, a disponibilidade e as ameaças e riscos que envolvem essas arquiteturas. Além disso, este capítulo também apresenta os trabalhos relacionados à garantia de integridade e de confidencialidade em ambientes de nuvens federadas.

4.1 Visão Geral

Atualmente, as organizações estão cada vez mais analisando a computação em nuvem como uma nova tecnologia revolucionária que promete cortar custos de desenvolvimento e manutenção, e ainda conseguir serviços altamente confiáveis e elásticos. A tecnologia em nuvem é uma tendência crescente e que tem como objetivo proporcionar enormes benefícios de custo, agilidade e escalabilidade para vários tipos de negócio [57].

Os dados do negócio são armazenados em servidores, em locais remotos, conhecidos como centros de dados. Este ambientes, geralmente centralizados, permitem às empresas executarem aplicações mais rápidas, com mais gerenciabilidade e menos esforço de manutenção, e também facilita um redimensionamento dos recursos para atender às necessidades de negócios flutuantes. Assim, o fluxo de dados é constante entre as máquinas locais detentoras dos dados, e os provedores da nuvem responsáveis pelo armazenamento dos arquivos, cujo conteúdo é geralmente privado. Essa movimentação de dados e seu futuro armazenamento podem contribuir para afetar a privacidade e a segurança dos arquivos salvos nos servidores em nuvem. As próximas seções tratarão mais detalhadamente sobre os principais conceitos relacionados à segurança em nuvens federadas.

Para finalizar, também serão apresentados estudos sobre o uso de *hashes* para verificar a integridade dos arquivos além de estudos que demonstram a eficácia de algoritmos de criptografia para garantir a confidencialidade dos arquivos, os quais foram implementados na solução proposta para o BioNimbuZ.

4.2 Principais conceitos

Essencialmente, garantir a segurança de um sistema de informação envolve a identificação das ameaças e dos desafios que precisam ser abordados para implementar as contramedidas adequadas [29]. A computação em nuvem, devido às suas características, impõe uma série de benefícios de segurança, que incluem centralização de segurança, segmentação de processos e dados, redundância e elevada disponibilidade. Enquanto muitos riscos tradicionais são contrariados de forma eficaz numa plataforma em nuvem, uma vasta série de desafios de segurança são introduzidos [55]. Computação em nuvem tem "atributos únicos que exigem avaliação de riscos em áreas como disponibilidade e confiabilidade, integridade dos dados, recuperação, privacidade e auditoria"[43].

Dessa forma, a segurança em geral está relacionada aos aspectos importantes da confidencialidade, da integridade e da disponibilidade. Eles são os blocos fundamentais a serem utilizados na concepção de sistemas seguros. Estes aspectos importantes de segurança aplicam-se as três grandes categorias de ativos que são necessários de proteção: dados, software e hardware. Para entender mais detalhadamente a segurança de uma plataforma de nuvens federadas é preciso entender algumas propriedades como confiança, confidencialidade, integridade e disponibilidade, as quais serão descritas a seguir.

4.2.1 Confiança

A confiança não é um tema novo de pesquisa em Ciência da Computação, abrangendo áreas tão diversas como a segurança e o controle de acesso em redes de computadores, confiabilidade em sistemas distribuídos, teoria dos jogos e políticas para a tomada de decisões em situações de incerteza [27]. Asssim, talvez o mais notável exemplo seja o desenvolvimento do *Trusted Computer System Evaluation Criteria* (TCSEC) [50], no final dos anos 70 e início dos 80. Nesse caso, a confiança foi usada no processo de convencer observadores de que um sistema estava correto e seguro [47].

O conceito de confiança, ajustado para o caso de duas partes envolvidas em uma transação, pode ser descrito da seguinte forma: uma entidade A confia em outra entidade B, quando a entidade A acredita que a entidade B irá se comportar exatamente como o esperado e necessário". Assim, uma entidade pode ser considerada confiável, se as partes ou as pessoas envolvidas em transações com essa entidade contam com a sua credibilidade [47]. Em geral, o conceito descrito pode ser verbalmente representado pelo termo em inglês *reliability*, o qual refere-se à qualidade de uma pessoa ou entidade que é digno de confiança. Confiança na sociedade é construída por várias razões diferentes, com base em cálculos, em conhecimento ou em razões sociais. A noção de confiança dentro de uma organização pode ser definida como a certeza do cliente de que a organização é capaz de fornecer os serviços necessários com precisão. Uma certeza que também expressa a fé do cliente na integridade moral da organização, na solidez da operação, na eficácia dos seus mecanismos de segurança, nos seus conhecimentos e no seu rigor por cumprir todos os regulamentos e leis. A noção de segurança refere-se a uma determinada situação em que todos os possíveis riscos ou são eliminados ou estão em um nível mínimo [44].

Dessa forma, a confiança em um ambiente de nuvem depende muito do modelo selecionado de implantação. Em arquiteturas tradicionais, a confiança foi reforçada por uma eficiente política de segurança, que verificou as restrições sobre as funções e o fluxo entre

elas, as restrições ao acesso de sistemas externos e adversários, incluindo programas e acesso de dados por pessoas.

Em uma implantação de nuvem, essa percepção é totalmente obscura. No caso de nuvens públicas ou comunitárias, o controle é delegado para a organização proprietária da infraestrutura. Ao implantar uma nuvem pública, o controle é designado ao proprietário da infraestrutura que possui a responsabilidade de impor uma política de segurança suficiente que garanta que as atividades de segurança adequadas estejam sendo realizadas. Isto introduz uma série de riscos e ameaças, pois essencialmente a segurança está relacionada a como os processos estão confiando no proprietário da nuvem [55].

É crucial diferenciar modelos de implantação, como uma nuvem privada, onde a infraestrutura é operada e gerenciada por uma organização privada, não introduzindo riscos de segurança adicionais, já que a confiança permanece dentro da organização. Em tais situações, o proprietário permanece dono da infraestrutura e dos processos [55].

Em um ambiente de nuvem, a percepção de um perímetro de segurança não é muito clara. Perímetro de segurança é um conjunto de políticas de segurança físicas e de software que fornecem níveis de proteção em uma fronteira conceitual contra atividades remotas maliciosas [55]. Tradicionalmente, acredita-se que qualquer ligação de sistemas ou organizações de fora de uma organização fornece uma abertura para entidades não autorizadas, para ganhar acesso ou adulterar recursos de informação. Assim, sobre essa fronteira conceitual, os controles de segurança foram mobilizados para proteger o sistema de informação dentro dele.

Em um modelo de computação em nuvem, o perímetro torna-se difuso, enfraquecendo a eficácia deste. Do ponto de vista tradicional de perímetro de segurança, a nuvem aparece fora da fronteira de confiança e deve ser vista como suspeita, pois tornou-se complexo colocar um fosso virtual em torno das organizações, já que vários serviços foram terceirizados. A capacidade de identificar claramente, autenticar, autorizar e fiscalizar quem, ou o que está acessando os ativos de uma organização é essencial para protege-la de ameaças e vulnerabilidades. A separação é o ingrediente chave de qualquer sistema seguro, e baseia-se na capacidade de criar fronteiras entre as entidades que devem ser protegidas e aquelas que não podem ser confiáveis [55].

4.2.2 Confidencialidade e Privacidade

Confidencialidade refere-se somente às partes ou sistemas autorizados terem a capacidade de acessar dados protegidos. A ameaça aos dados em uma nuvem tende a crescer, devido ao aumento do número de partes, dispositivos e aplicativos envolvidos, que leva a um aumento do número de pontos de acesso. Uma série de preocupações emergem como questões de *multitenancy*, remanescência de dados, segurança de aplicativos e privacidade [24].

Multitenancy refere-se à característica da nuvem de partilhar recursos. Vários aspectos do sistema são compartilhados, incluindo, memória, programas, redes e dados. A computação em nuvem é baseada em um modelo de negócio em que os recursos são partilhados (isto é, vários usuários usam a mesma fonte), em nível de rede e em nível de aplicativo. Embora os usuários sejam isolados em um nível virtual, o hardware não é separado. Com uma arquitetura *multitenant*, um software é projetado para particionar virtualmente seus dados e configurações de modo que cada organização cliente (nó) trabalha com uma ins-

tância do aplicativo virtual personalizado. Dessa forma, esse particionamento acaba por abrir buracos de segurança já que os recursos são fisicamente compartilhados [43].

Uma outra característica vigente em ambientes em nuvem é a reutilização de objetos, e esses objetos reutilizáveis devem ser cuidadosamente controlados a fim de não criar sérias vulnerabilidades. Dessa forma, devido às separações virtuais e a falta de separação do hardware, a representação virtual de dados, que já foram de alguma forma apagados ou removidos, pode levar à divulgação involuntária de dados privados [43].

A confidencialidade dos dados na nuvem está correlacionada com a autenticação do usuário. Proteger a conta de um usuário de roubo é somente uma instância de um problema maior que é de controlar o acesso a objetos, incluindo a memória, os dispositivos, o software, etc. A autenticação eletrônica é o processo de estabelecer a confiança em identidades de usuários. Falta de autenticação pode levar ao acesso não autorizado de utilizadores na nuvem, levando a uma brecha na privacidade [43]. A confidencialidade do software é tão importante quanto a confidencialidade dos dados para a segurança geral do sistema. Confidencialidade do software refere-se a confiar que aplicativos ou processos específicos manterão os dados pessoais do usuário de uma maneira segura. Em um ambiente de nuvem, o usuário é obrigado a delegar confiança nas aplicações fornecidas pela organização que é proprietária da infraestrutura. Aplicativos que interagem com os dados do usuário devem ser certificados para não introduzir riscos de confidencialidade e privacidade adicionais. O acesso não autorizado pode tornar-se possível por meio da exploração de uma vulnerabilidade da aplicação ou falta de identificação forte, trazendo à tona questões de confidencialidade e de privacidade dos dados. Além disso, o provedor da nuvem é responsável por fornecer instâncias seguras da nuvem, o que deve garantir a privacidade dos usuários [37].

Por outro lado, o conceito de privacidade está relacionado com o desejo de uma pessoa em controlar a divulgação de informações pessoais. Organizações que lidam com dados pessoais são obrigadas a obedecer ao quadro jurídico de um país que garante privacidade e proteção as informações. A nuvem apresenta uma série de desafios legais em relação às questões de privacidade, pois os dados são armazenados em vários locais na nuvem. Em vez dos dados serem armazenados em servidores da empresa, os dados são armazenados em servidores do provedor de serviços, que pode estar localizado na Europa, na Ásia, ou em qualquer outro lugar. Este princípio de computação em nuvem entra em conflito com vários requisitos legais, tais como as leis europeias que requerem que uma organização saiba onde os dados pessoais, em sua posse, estão em todos os momentos.

4.2.3 Integridade

A integridade é uma propriedade que se refere a confiabilidade dos recursos (hardware, software e dados), e pode ser entendida como a prevenção contra ações indevidas ou não autorizadas que podem afetar estes recursos [43]. Essa propriedade pode ser classificada em [43]:

• Integridade de dados: está ligado à proteção contra modificações, ou exclusão de informações, e se refere diretamente ao conteúdo dos dados. Um usuário que armazenou seus dados em uma federação de nuvens espera que eles não sejam modificados por nin-

guém que não tenha autorização, e deve poder recuperar todas as informações da forma que estavam originalmente;

• Integridade de fonte: a origem da informação deve possuir credibilidade e ser autêntica para que todos os usuários tenham confiança naquele dado. É possível fazer um paralelo com um jornal que recebe uma indicação de matéria de uma fonte falsa a respeito de algo que não existiu, e mesmo assim publica o texto integral. Neste caso, houve integridade de dados, pois o texto foi publicado da forma que foi recebido, porém não houve integridade de fonte, pois não foi possível verificar a veracidade das informações;

• Integridade de software: refere-se à integridade dos serviços disponibilizados por um provedor de nuvem, que não devem ser modificados por pessoas não autorizadas [37]. Um determinado serviço deve funcionar da mesma forma sempre que for requisitado, a menos que o provedor decida ou autorize alguém a alterá-lo.

Os mecanismos de integridade podem ser divididos em duas classes, os de prevenção e os de detecção. Os mecanismos da primeira classe basicamente se encarregam de bloquear as tentativas não autorizadas de alterar um dado, por exemplo, um usuário que não possui autorização e quer modificar um arquivo, ou talvez até tenha autorização para algumas operações e não para outras. Sistemas de autenticação e controle de acesso são representantes dessa classe [37].

Já os mecanismos da segunda classe tem como objetivo informar quando os dados não estão mais confiáveis. Para fazer isto, podem analisar os eventos do sistemas (como ações dos usuários) ou verificar os próprios dados para procurar indícios de violação [37]. Esses mecanismos podem ainda aplicar algoritmos de correção para que os dados voltem a um estado confiável.

A integridade está intimimamente ligada a confidencialidade, porém é muito mais ampla, pois leva em consideração a origem dos dados (como e quando foram obtidos). Também está ligada ao armazenamento (é necessário saber se os dados foram armazenados corretamente) além de estar ligada a autorização e da autenticação (também presentes na confidencialidade) [37].

4.2.4 Disponibilidade

Disponibilidade refere-se a estabelecer que um sistema é acessível e utilizável sob demanda por uma entidade autorizada. Assim, a disponibilidade do sistema inclui a capacidade de realizar operações mesmo quando algumas autoridades se comportam mal. O sistema deve ter a capacidade de continuar as operações até mesmo quando ocorre uma violação de segurança. Disponibilidade refere-se a dados, software, mas também ao hardware estar disponível a usuários autorizados sob demanda. Ao deixar os usuários fora da infraestrutura de hardware, gera-se uma forte dependência da disponibilidade da rede. A rede agora está sobrecarregada com a recuperação e o processamento de dados.

O proprietário da nuvem necessita garantir que o processamento das informações e as próprias informações estejam disponíveis aos clientes sob demanda. Dessa forma, os serviços de computação em nuvem apresentam uma forte dependência de infraestruturas e recursos de rede em todos os momentos [43].

Compreender e documentar claramente os requisitos específicos de usuários é bastante necessário na concepção de uma solução que visa assegurar essas necessidades. Verificar

identidades que compartilham requisitos de segurança fundamentais, bem como a determinação das necessidades de proteção de dados pode ser um dos elementos mais complexos do projeto. Dessa forma, os objetivos de segurança dentro de um sistema distribuído são essencialmente [55]:

• Assegurar a disponibilidade das informações comunicadas entre os sistemas participantes;
• Manter a integridade das informações comunicadas entre os sistemas participantes, ou seja, prevenir a perda ou a alteração de informações devido a acessos não autorizados, falhas de componentes ou outros erros;
• Manter a integridade dos serviços prestados, ou seja, a confidencialidade e a operação correta;
• Fornecer o controle sobre o acesso aos serviços ou aos seus componentes para garantir que os utilizadores só possam utilizar serviços pelos quais são autorizados;
• Autenticar a identidade de parceiros comunicantes;
• Garantir a separação de dados e de processos em nível virtual da nuvem, garantindo nenhum vazamento de dados entre diferentes aplicações.
• Manter o mesmo nível de segurança ao adicionar ou remover recursos no nível físico.

Assim, tendo em vista que o tema deste projeto é garantir a segurança de uma arquitetura de nuvens federadas, a próxima seção visa explicar mais detalhadamente para a realidade da plataforma de nuvens federadas BioNimbuZ, algumas pesquisas já realizadas sobre a integridade e a confidencialidade, assuntos que são o foco deste projeto.

4.3 Técnicas de Integridade

A integridade dos dados é definida como a precisão e a consistência dos dados armazenados, na ausência de qualquer alteração aos dados entre duas atualizações de um arquivo ou registro [23]. Serviços em nuvem devem garantir a integridade dos dados e fornecer confiança para o usuário, e embora a terceirização de dados para a nuvem seja economicamente atraente, a falta de uma garantia forte da integridade e da confidencialidade de dados pode impedir sua ampla adoção.

Um provedor de serviços em nuvem pode acidentalmente ou deliberadamente alterar ou excluir algumas informações. Assim, um sistema que possua interfaces com nuvens externas deve ter algum tipo de mecanismo para garantir a integridade desses dados. O modelo corrente de segurança em nuvem baseia-se no pressuposto de que o utilizador e o cliente devem apenas confiar no provedor da nuvem, o que leva, às vezes, a extravios de informações. Esta confiança é tipicamente governada e baseada em um Acordo de Nível de Serviço (SLA) que, em geral, define as expectativas dos usuários e suas obrigações.

A fim de garantir a integridade dos dados em uma ambiente em nuvem, e fazer cumprir a qualidade do serviço de armazenamento em nuvem, alguns métodos eficazes que permitem a verificação da corretude dos dados *on-demand*, em nome dos usuários da nuvem, são concebidos.

Este trabalho foi baseado em uma combinação de teorias e temáticas que estão fragmentadas em diversos trabalhos científicos. Diversos métodos e práticas são difundidas dentre o meio acadêmico para assegurar a integridade dos arquivos em uma plataforma

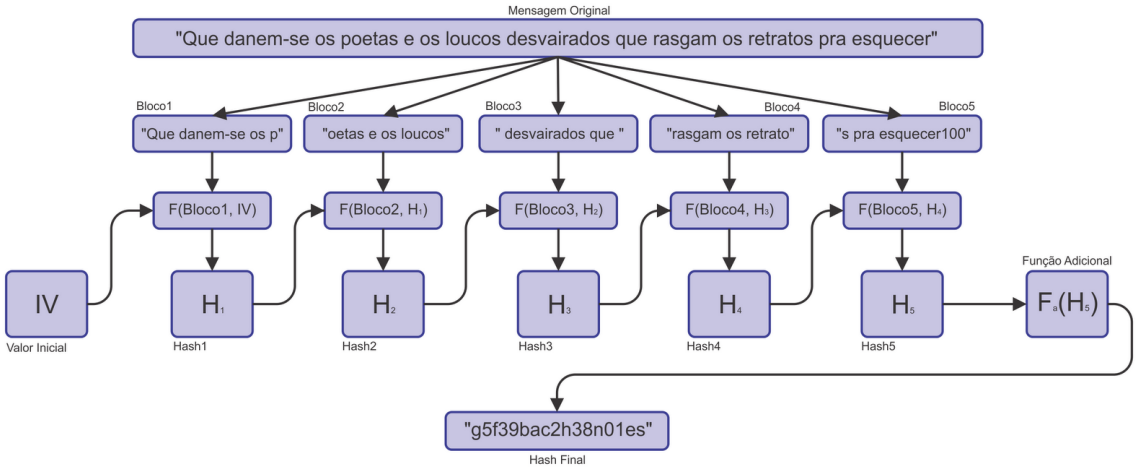

Figura 4.1: Funcionamento básico de um algoritmo de *hash* [13].

de computação em nuvem. Algumas das teorias encontradas são:

4.3.1 Usando *Hashes* Seguros para Verificar a Integridade dos Dados

Algoritmos de *hash* basicamente consistem de funções que recebem uma mensagem de tamanho variável como entrada e retornam um *hash* de tamanho fixo na saída, como demonstrado na Figura 4.1. Isto ocorre de tal forma que uma simples mudança na mensagem de entrada resulte em uma mudança no *hash* de saída. Um algoritmo seguro deve seguir as seguintes premissas:

- Dado um determinado *hash* h, deve ser impraticável obter uma mensagem m que satisfaça a condição $\text{hash}(m) = h$.
- Dada uma determinada mensagem $m1$, deve ser impraticável obter uma segunda mensagem $m2$ que satisfaça a condição $\text{hash}(m1) = \text{hash}(m2)$.
- Deve ser impraticável encontrar duas mensagens arbitrárias que retornem o mesmo *hash*.

Para o caso de se verificar a integridade de arquivos, o processo usa uma função matemática única que gera um bloco de dados de tamanho fixo. Esse resultado é chamado de digestão e pode ser pensado como uma impressão digital dos dados. Assim, os dados digeridos podem ser reproduzidos com o mesmo fluxo de dados, mas é praticamente impossível de se produzir um fluxo de dados diferente da que produz o mesmo resumo de dados. Este resumo de dados único pode ser usado para comparar a integridade dos dados antes e após a transferência [23]. Como o *hash* gerado antes da transferência é único, qualquer modificação no arquivo, por menor que ela seja, vai alterar consistentemente o *hash* final. Dessa forma, apesar do usuário não saber qual foi a alteração que o arquivo sofreu, ele sabe que o arquivo foi alterado e que aquela transferência necessita ser refeita.

Figura 4.2: Situação em que é gerado uma assinatura digital de um documento [4].

No entanto, este método não protege o receptor de um ataque se terceiros interceptarem mensagens do remetente, e substituí-las por uma nova mensagem e por um novo resumo.

Por fim, o *hash* resultante é armazenado junto com o seu arquivo correspondente na nuvem. Se o usuário pede dados a partir de uma nuvem, os dados e seu respectivo *hash* são enviados para o utilizador. Ele pode voltar a gerar um outro *hash*, caso necessário, para proteger contra alterações nos dados de qualquer lugar. Ataques externos podem interceptar mensagens do remetente e substituí-la por uma nova mensagem, mas ele não pode gerar um *hash* aceitável sem saber a chave secreta [54]. Vários algoritmos de *hash* podem ser usados para este fim, como: HMAC [12], *Message Digest* 5 (MD5) [14], SHA1, SHA2 e SHA3 [21].

4.3.2 Utilização de uma Assinatura Digital

Em criptografia, a assinatura ou firma digital é um método de autenticação de informação digital tipicamente tratada como análoga à assinatura física em papel. Embora existam analogias, existem diferenças importantes. O termo assinatura eletrônica, por vezes confundido, tem um significado diferente: refere-se a qualquer mecanismo, não necessariamente cifrado, para identificar o remetente de uma mensagem electrônica.

A utilização da assinatura ou firma digital providencia a prova inegável de que uma mensagem veio do emissor. Para verificar este requisito, uma assinatura digital deve ter as propriedades de autenticidade (o receptor deve poder confirmar que a assinatura foi feita pelo emissor), de integridade (qualquer alteração da mensagem faz com que a assinatura não corresponda mais ao documento) e de irretratabilidade ou não-repúdio (o emissor não pode negar a autenticidade da mensagem) [23].

Essas características fazem a assinatura digital ser fundamentalmente diferente da assinatura manuscrita.

A Figura 4.2 exemplifica essa situação, em que se necessita garantir que um determinado documento não seja alterado após assinado.

4.3.3 Usando Segurança das Infraestruturas

Para alcançar uma ampla segurança, o emissor e o receptor vão precisar usar vários tipos de mecanismos de segurança criptográficos [23]. Em particular, eles vão precisar distribuir chaves de criptografia simétrica para alcançar a integridade dos dados, conforme demonstrado na Figura 4.3. A distribuição de chaves simétricas pode ser feita de forma

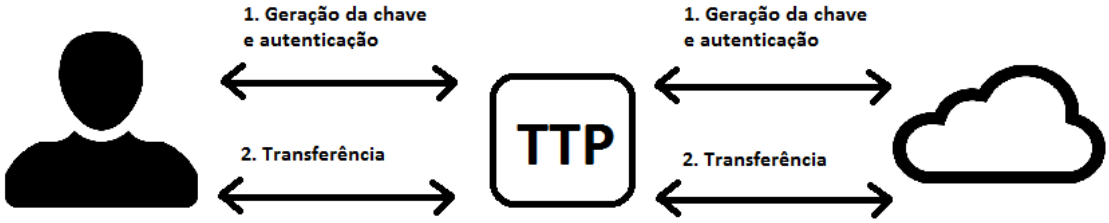

Figura 4.3: Algoritmo de Verificação de Integridade dos Dados por uma Terceira Parte Confiável (TTP).

eficaz usando o gerenciamento de chaves públicas por intermédio de uma terceira parte externa (TTP) que se tem a certeza que é confiável. Para atingir um nível elevado de segurança dentro da infraestrutura de uma organização, muitas TTPs interligadas serão necessárias. Assim, emissor e receptor devem ser capazes de obter um do outro, chaves públicas e autenticar a identidade da outra parte. Dessa forma, eles devem depender de um terceiro confiável para distribuir as chaves públicas e autenticar a identidade da parte associada com o par de chaves correspondente.

4.3.4 Usando *Tokenization* de Dados na Nuvem

Pode ser usado para manter uma cópia segura de dados sensíveis já que somente os *tokens* referentes aos dados são armazenados e processados na nuvem. Os *tokens* são gerados por meio de uma tabela de correspondência, e estes *tokens* podem ser revertidos de volta para seus valores originais utilizando uma tabela *look-up* que possui as correspondências aos valores originais. Essas tabelas são tipicamente mantidas em um banco de dados seguro localizado dentro do *firewall* da empresa para evitar acessos externos não autorizados [23].

4.3.5 Utilizando Decifragem dentro de um Processador

Neste modelo, os dados sensíveis são decifrados dentro de um processador que já possui conhecimento que os dados que são enviados a ele são baseados em uma tecnologia criptográfica. Esta tecnologia permitirá que as empresas tenham confiança na segurança da máquina virtual rodando na nuvem, enquanto ela estiver sendo executada pelo processador na plataforma confiável [23].

4.3.6 Aplicar Acesso Restrito a Informações Sensíveis

Apenas a não autorização de certos usuários a certos arquivos impõe uma barreira tecnológica a entidades maliciosas. Este método previne acessos não autorizados aos dados intencionalmente ou acidentalmente por meio do controle de usuários e níveis de acesso às funcionalidades do sistema [23].

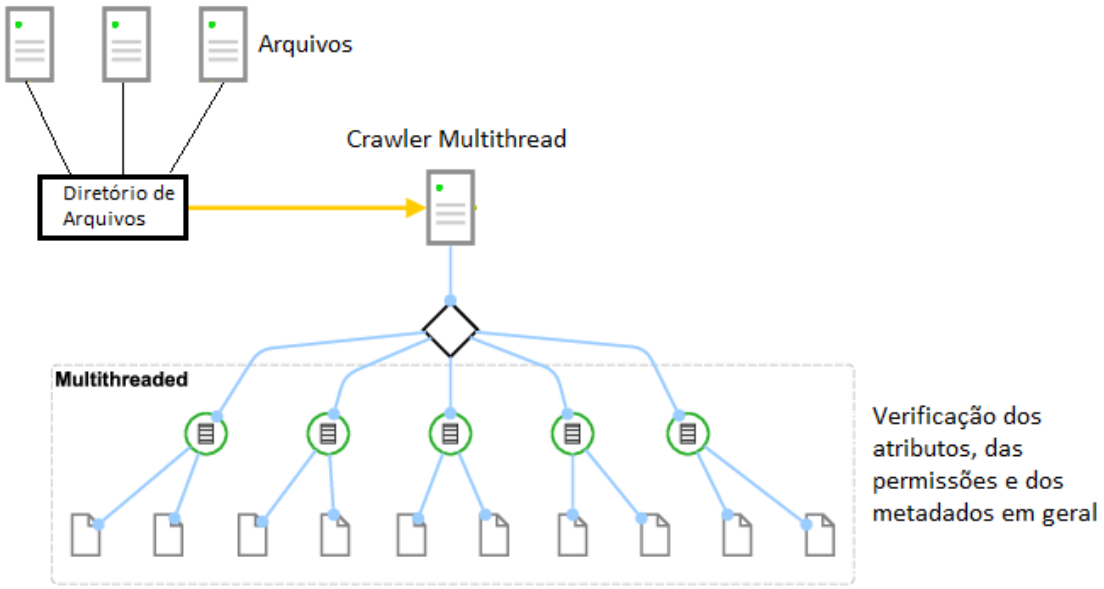

Figura 4.4: Uso de um *crawler* em um sistema de arquivos.

4.3.7 Aplicar Ferramentas de Monitoramento

Este modelo se baseia em um software, uma espécie de *crawler*. Usado bastante na internet, um *crawler* é um software desenvolvido para realizar uma varredura na páginas de internet de maneira sistemática através de informação vista como relevante à sua função. Eles capturam os textos das páginas e cadastram os *links* encontrados e assim possibilitam encontrar novas páginas. Similarmente, um *crawler* pode ser instalado em um sistema de arquivos. Neste caso, como mostrado na Figura 4.4, o objetivo é prover um monitoramento da integridade no nível da máquina virtual com o intuito de se verificar extensivamente os arquivos, incluindo atributos, permissões e monitoramento em nível de diretório [23].

4.3.8 *Backup* de Dados

É o meio mais importante e também o mais simples para evitar que os dados sejam perdidos devido a meios intencionais ou não intencionais. Para aumentar a eficácia do *backup* também é importante manter criptografados os conteúdos destes *backups* [23].

4.4 Técnicas de Confidencialidade

A computação em nuvem está propensa a sofrer de uma série de vulnerabilidades já conhecidas, permitindo vários ataques maliciosos que visam obter informações dos usuários da nuvem [52]. No mundo da computação, as questões de segurança e de privacidade são uma das principais preocupações, e a computação em nuvem não é uma exceção a estas questões.

Proteger os dados que estão em nuvens externas é, na verdade, mais caro do que as perdas associadas, levando em consideração que são usados, normalmente, mecanismos de

segurança bastante modernos [34]. Do ponto de vista do usuário da nuvem, utilizar uma nuvem é parecido com a confiança depositada em uma empresa de telefone contratada pelo usuário, ou também com o serviço de correios em que o usuário, apesar de estar pagando, não possui uma garantia física de que a sua correspondência, de fato, chegará ao destino.

Um problema com a computação em nuvem é que a gestão dos dados pode não ser totalmente confiável. O risco de ataques maliciosos na nuvem e a falha de serviços em nuvem têm recebido uma forte atenção por empresas, atualmente. Dessa forma, alguns tipos de usuários colocam informações confidenciais nas mãos das operadoras das nuvens e apenas confiam que suas informações não serão acessadas. Por outro lado, há uma classe de usuários que não iria usar o telefone sem levar antes precauções de segurança de que o seu serviço será executado exatamente como acordado previamente. Para a aquisição de serviços de armazenamento na nuvem, a mesma situação se aplica, de forma que a mensagem transmitida não pode ser visível aos servidores externos, ou seja, mensagens criptografadas é que devem ser enviadas para a nuvem [59].

A técnica utilizada neste trabalho foi baseada em várias pesquisas já realizadas e documentadas que demonstram e expõem quais são as melhores técnicas e os melhores algoritmos usados para garantir uma confidencialidade dos arquivos. Os trabalhos mais relevantes para este projeto estão descritos com mais detalhes nos próximos tópicos.

4.4.1 Comparação dos Algoritmos RSA, DES e AES

Rachna Arora e Anshu Parashar [26] propõem uma comparação entre diferentes algoritmos que visam eliminar preocupações referentes a perda de dados, segregação de dados e também da privacidade desses dados em um ambiente em nuvem. Algoritmos como RSA (*Rivest Shamir Adleman*) [20], DES (*Data Encryption Standard*) [11], AES (*Advanced Encryption Standard*) [1] e Blowfish [8] têm sido utilizados atualmente no mercado para garantir a segurança dos dados em nuvem.

DES, AES e Blowfish são algoritmos de chave simétrica, em que uma única chave é usada para a codificação e para a decodificação das mensagens. O DES foi desenvolvido no início dos anos 1970. Blowfish foi projetado em 1993, expressamente para ser usado em sistemas embarcados. AES foi projetado pela NIST em 2001. Já o RSA é um algoritmo de chave pública, projetado por Rivest, Shamir e Adleman [20] em 1978 e também denominado como algoritmo de chave assimétrica, já que ele utiliza chaves diferentes para fins de criptografia e descriptografia. Os tamanhos das chaves são distintos uns dos outros. O comprimento da chave do algoritmo DES é 56 bits. O tamanho da chave do algoritmo AES é de 128, 192 e 256 bits. O tamanho da chave do algoritmo de Blowfish é de 128 a 448 pedaços. E o tamanho da chave do algoritmo RSA é de 1024 bits.

- RSA

O algoritmo RSA é o algoritmo de chave pública mais utilizado atualmente [26]. O RSA é basicamente um algoritmo de chave assimétrica para criptografia e descriptografia. É assimétrica no sentido de que a chave pública, por meio da qual se cifra a mensagem, é distribuída a todos, porém a chave privada que é utilizada para a decodificação é mantida em segredo e não é compartilhada.

A criptografia dos dados tem o objetivo de proporcionar segurança e o objetivo da proteção de dados é proporcionar que somente os usuários autorizados possam acessar as informações. Depois da criptografia dos dados, o criptograma é armazenado na nuvem, de modo que quando for necessário, um pedido possa ser solicitado ao provedor da nuvem. O provedor da nuvem, assim, autentica o usuário e fornece os dados a ele.

Em um ambiente em nuvem, a chave pública é conhecida de todos, enquanto a chave privada é conhecida apenas pelo usuário que originalmente possui os dados. Assim, a criptografia é feita pelo provedor de serviços da nuvem e a descriptografia é feita pelo usuário ou consumidor da nuvem. Uma vez que os dados são criptografados com a chave pública, só será decifrada utilizando a correspondente chave privada única.

O algoritmo RSA é baseado na construção de chaves públicas e privadas, utilizando números primos. Inicialmente, devem ser escolhidos dois números primos quaisquer A e B. Quanto maior o número escolhido mais seguro será o algoritmo. A seguir são calculados dois novos números N e Z de acordo com os números A e B escolhidos, conforme apresentado nas Equações 4.1 e 4.2.

$$N = A * B \tag{4.1}$$

$$Z = (A - 1) * (B - 1) \tag{4.2}$$

Agora se define um número P que tenha a propriedade de ser primo em relação à Z. De posse desses números, começa o processo de criação das chaves públicas e privadas. É necessário encontrar um número S que satisfaça a seguinte propriedade, de acordo com a Equação 4.3.

$$(P * S) mod Z = 1 \tag{4.3}$$

Com esse processo se definem as chaves de cifragem e decifragem. Para codificar, ou seja, para gerar a chave pública, utiliza-se P e N. Já para decodificar, isto é, para gerar a chave privada, utiliza-se S e N. Mais detalhadamente, esses conceitos são apresentados nas Equações 4.4 e 4.5

$$textocriptografado = (textooriginal^{p}) mod N \tag{4.4}$$

$$textooriginal = (textocriptografado^{s}) mod N \tag{4.5}$$

- DES

DES é um tipo de cifra em bloco, ou seja, um algoritmo que toma uma *string* de tamanho fixo e a transforma, através de uma série de complicadas operações, em um texto cifrado de mesmo tamanho. No caso do DES, o tamanho do bloco é de 64 bits. DES também usa uma chave para personalizar a transformação, de modo que a descriptografia somente seria possível, teoricamente, por aqueles que conhecem a chave particular uti-

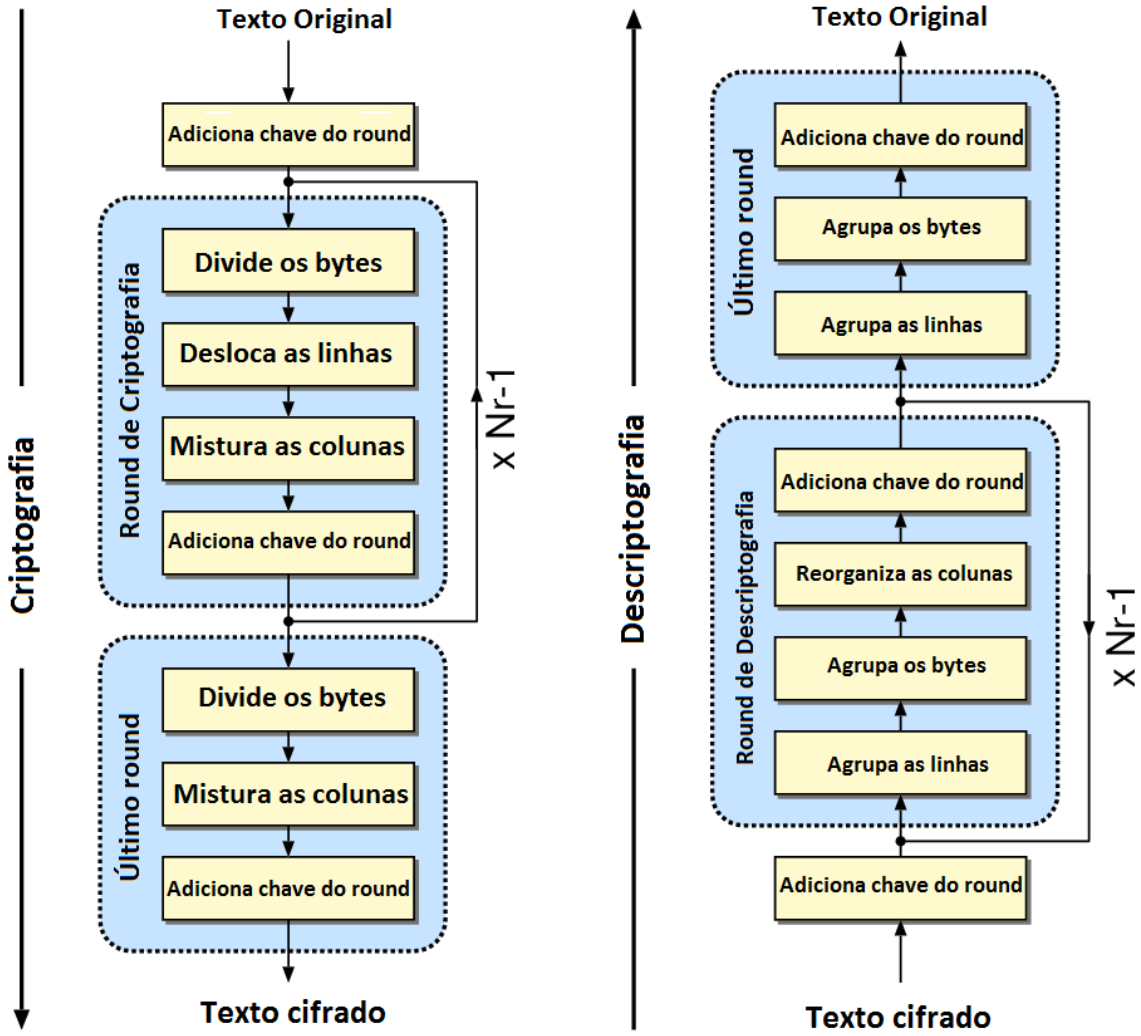

Figura 4.5: Algoritmo AES, adaptado de [10].

lizada para criptografar. A chave consiste nominalmente de 64 bits, porém somente 56 deles são realmente utilizados pelo algoritmo. Os oito bits restantes são utilizados para verificar a paridade e depois são descartados, portanto, o tamanho efetivo da chave é de 56 bits, e assim é citado o tamanho de sua chave. O DES é, atualmente, considerado inseguro para muitas aplicações. Isto se deve principalmente a pequena chave de 56-bit [26].

- AES

AES, como monstrado na Figura 4.5, também conhecido como Rijindael é usado para proteger as informações de usuários [26]. AES é um bloco de chave simétrica que tem sido estudado extensivamente e é amplamente usado nos dias atuais, principalmente, para ambientes em nuvem.

O nome é uma fusão de Vincent Rijmen e Joan Daemen, os dois belgas criadores do algoritmo. O algoritmo combina características de segurança, desempenho, facilidade de implementação e flexibilidade. O Rijndael apresenta alta resistência a ataques como *power attack* e *timing attack* e exige pouca memória, o que o torna adequado para operar

em ambientes restritos como *smart cards*, PDAs e telefones celulares [26].

A proposta do algoritmo é que, quando a migração dos dados da máquina local para o provedor de serviço da nuvem (CSP) aconteça e, no futuro, sempre que um aplicativo carregar qualquer dados para a nuvem, os dados serão criptografados utilizando primeiramente o algoritmo AES e, só depois, enviado para o fornecedor. Uma vez criptografados, os dados são carregados para a nuvem e qualquer pedido para ler os dados somente irão ocorrer após este ser decifrado. O arquivo original nunca é salvo na nuvem.

Esta solução de criptografia é transparente para o software e pode ser integrada de forma rápida e fácil sem quaisquer alterações na aplicação [26]. A chave, da mesma forma, nunca é armazenada próxima aos dados codificados, uma vez que isso pode comprometer a chave também. Esta cifragem protege os dados e as chaves, e garante que permaneçam sob controle do usuário e que nunca serão expostos ao armazenamento em nuvem. O algoritmo AES substituiu o DES como o padrão que é usado por uma vasta gama de aplicações.

Assim sendo, nota-se que os algoritmos de criptografia desempenham um importante papel na segurança de dados em nuvem e por comparação de diferentes parâmetros utilizados em algoritmos no estudo apresentado por Rachna Arora e Anshu Parashar [26], foi descoberto que o algoritmo AES usa menos tempo para executar dados em nuvem. O algoritmo DES é o que consome o menor tempo para ser criptografado. O RSA é o que consome mais mais memória e também é o que leva mais tempo para criptografar os dados. Os resultados mais detalhados encontram-se na Tabela 4.1.

4.4.2 Criptografia usando o Algoritmo AES

O trabalho proposto por Debajyoti Mukhopadhyay e Gitesh Sonawane [46] cita que as informações estarão menos expostas a ataques externos se forem criptografadas. A ideia é que o arquivo presente no dispositivo de um cliente seja criptografado com base no algoritmo AES.

As vantagens da AES são muitas, pois ele não é suscetível a qualquer ataque, mas somente aos ataques do tipo *Brute Force*. No entanto, os ataques *Brute Force* não são uma tarefa fácil, mesmo para um super computador. Isto se deve porque o tamanho da chave de criptografia usada pelo algoritmo AES é da ordem de 128, 192 ou 256 bits, o que resulta em milhares de milhões de permutações e combinações. O algoritmo AES também é muito mais rápido do que os algoritmos tradicionais, como RSA [46].

O trabalho proposto por Mukhopadhyay e Sonawane divide-se basicamente em dois passos principais: *Upload* e *Download* do arquivo.

- *Upload* do arquivo

Conforme ilustrado na Figura 4.6, o passo 1 é autenticar o usuário. A metodologia sugerida pelo trabalho visa prevenir qualquer possível ataque aos dados do usuário. O sistema irá aceitar o nome de usuário e a sua senha. Uma vez que os dois são verificados, o usuário tem acesso ao seus arquivos. Se o nome inserido e a senha não forem válidos, o sistema mostra um erro e rejeita o usuário. Assim, os próximos passos trabalham sob a condição de que o usuário é autenticado e uma conexão para trabalhar com a nuvem foi estabelecida.

Tabela 4.1: Planilha de Comparação entre os algoritmos AES, RSA e DES [26].

Algoritmo	AES	RSA	DES
Plataforma	Plataforma em nuvem	Plataforma em nuvem	Plataforma em nuvem
Tamanho da Chave	128, 192, 256 bits	1024 bits	56 bits
Chave utilizada	A mesma chave é utilizada para criptografar e descriptografar os dados	Uma chave pública é usada para criptografar os dados e uma chave privada é utilizada para descriptografar os dados	A mesma chave é utilizada para criptografar e descriptografar os dados
Escalabilidade	Escalável	Não escalável	Escalável
Tamanho do vetor inicial	128 bits	1024 bits	64 bits
Segurança	Seguro tanto para o usuário quanto para o provedor da nuvem	Seguro somente para o usuário	Seguro tanto para o usuário quanto para o provedor da nuvem
Capacidade da criptografia de dados	Utilizado para criptografar grandes quantidades de dados	Utilizado para criptografar pequenas quantidades de dados	Utilizado para criptografar quantidades médias de dados
Uso da memória	Pouco uso da memória	Uso alto da memória	Uso mediano da memória
Tempo de execução	O mais rápido	Mais lento que os outros dois	Tão rápido quanto o AES

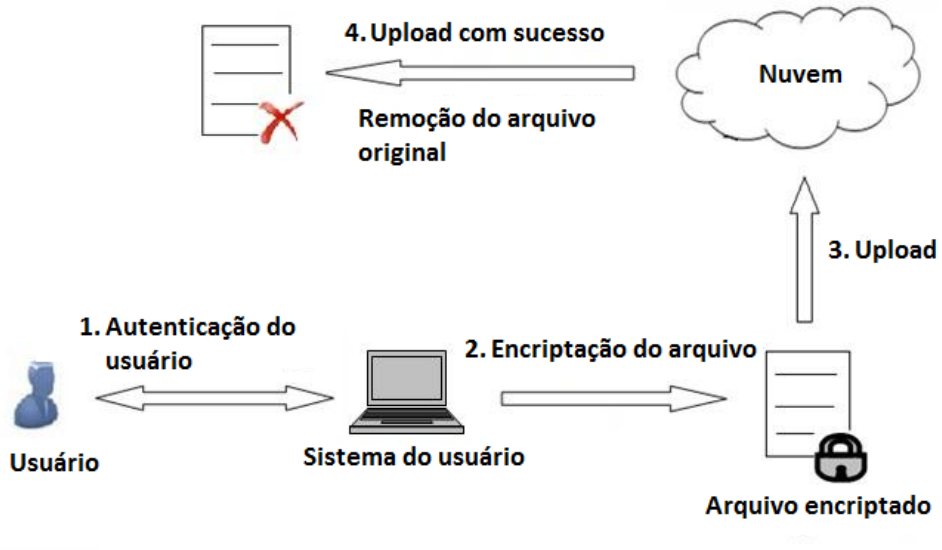

Figura 4.6: *Upload* de Arquivo [46].

A seleção do arquivo a ser carregado é feita na etapa 2. O usuário pode selecionar qualquer arquivo presente na memória da máquina que ele está usando atualmente. Uma vez que o arquivo que será carregado foi selecionado, o processo passa para a etapa 3. Esta etapa pede ao usuário uma senha para a criptografia do processo que será usada posteriormente para a geração de uma chave. Ao usuário, é recomendado o uso de frases-senhas longas como suas senhas. Os detalhes desta geração são descritos na próxima etapa.

O passo 4 é um passo muito importante para o sistema. Neste passo, uma chave para o processo de criptografia é gerada. O algoritmo AES é um processo que usa uma chave simétrica, ou seja, ele usa a mesma chave que é usada para criptografar e para descriptografar os dados também. Esta chave é gerada a partir da palavra-passe utilizando uma função-chave do gerador. O trabalho proposto por Mukhopadhyay e Sonawane [46] recomenda o uso de PBKDF2 [18]. O PBKDF2 usa iterações na ordem de milhares, o que faz com que seja mais complicado burlar o algoritmo AES. Este processo é chamado de *stretching*. É de notar que embora as chaves utilizadas no algoritmo AES não sejam suscetíveis a qualquer ataque conhecido, há uma possibilidade de a senha ser atacada por força bruta, como citado anteriormente. Portanto, o usuário é altamente recomendado a usar longas frases-chave para a geração da chave. Assim, neste passo, o sistema guarda a palavra-passe e gera uma chave aleatória para criptografia.

O passo 5 é o passo da cifragem, de fato. Neste passo, o algoritmo de criptografia, isto é, o algoritmo de AES é aplicado para o texto de entrada do usuário, o que gera o texto cifrado. Como mencionado antes, o algoritmo AES não é suscetível a quaisquer ataques conhecidos.

Os dados do usuário, dessa maneira, são duplamente assegurados porque uma pessoa pode acessar os dados somente se tiver inserido o nome de usuário válido e a senha válida. E mesmo se o *login* e senha do usuário forem burlados, o arquivo que for enviado será o criptografado. Assim, o ataque só terá acesso as informações reais se possuir a chave para decifrar o arquivo. Isso garante a confidencialidade. Além disso, uma vez que os dados

enviados são criptografados, nenhuma modificação pode ser feita para o texto cifrado. Isso garante a integridade dos dados também.

Uma vez que o texto cifrado for gerado, o próximo passo é fazer o *upload* do arquivo criptografado para a nuvem. Esta é a etapa 6 do processo.

A etapa 7 está relacionada com a eliminação do arquivo de texto puro original da memória da máquina do cliente. Uma vez que o arquivo de texto cifrado é carregado com êxito na nuvem e o usuário não tem mais arquivos para serem enviados, o processo prossegue para o *logout* da conta de usuário, e desconecta a conexão estabelecida com a nuvem.

- *Download* do arquivo

Como mostrado na Figura 4.7, a primeira tarefa é a mesma do passo 1 na operação de *upload* de arquivos. A identidade do usuário é autenticada nesta etapa. No passo 2, a lista de arquivos que o usuário tem é carregado pela nuvem e é exibida ao cliente. O usuário agora é solicitado a selecionar um dos arquivos da lista.

Na etapa 3, o usuário é solicitado a digitar a senha que ele tinha entrado durante a criptografia do arquivo. A validação desta senha digitada é feita na etapa 4. Dessa forma, o arquivo de texto cifrado carregado pelo usuário só será decifrado e transferido se a senha digitada for a mesma que a senha digitada durante a criptografia dos arquivos. Esta é a razão pela qual a senha é salva durante o processo de criptografia, ou seja, a senha armazenada é usada para validar a senha introduzida. Como mencionado anteriormente, o processo se baseia no algoritmo AES de chave simétrica, ou seja, é necessária a mesma chave para criptografar e descriptografar os dados. Isso só será possível se a mesma senha for inserida na função de gerador de chaves para gerar a chave.

Uma vez que a senha digitada for validada, usa-se a senha para gerar a chave de decifração. No passo 5, usando a chave gerada, usa-se o algoritmo AES para descriptografar o texto cifrado carregado.

A operação para salvar o texto original descriptografado na máquina do usuário é feita na etapa 6. E por fim, na etapa 7, o arquivo de texto cifrado é removido da nuvem.

Baseado nos trabalhos e pesquisas acima citados, o próximo capítulo irá detalhar quais foram os métodos escolhidos para aprimorar a segurança da plataforma BioNimbuZ.

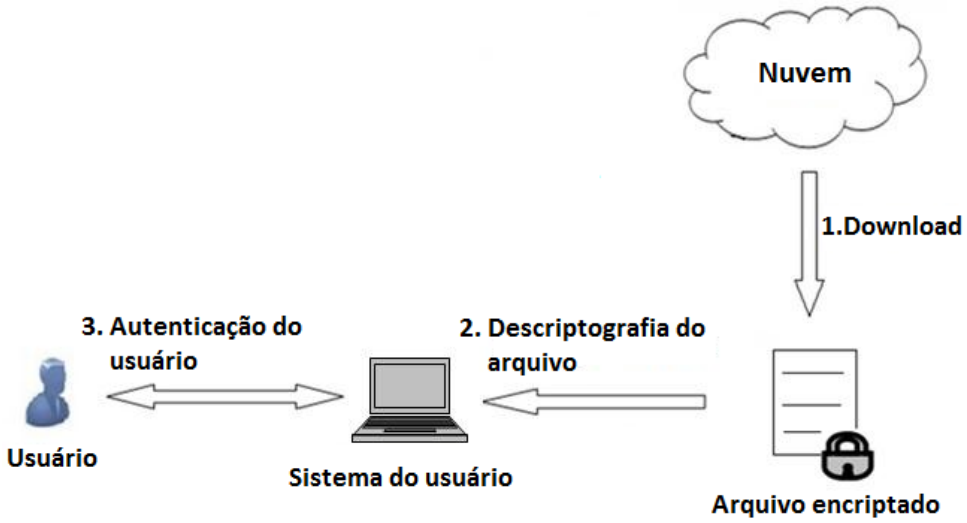

Figura 4.7: *Download* de arquivo [46].

Capítulo 5

Integridade e Confidencialidade no BioNimbuZ

O objetivo deste projeto, tendo em vista que já foi implementado um controle de autenticação e autorização dos usuários [35] na plataforma de nuvens federadas BioNimbuZ, é aprimorar dois outros aspectos de segurança: Integridade e Confidencialidade. Assim, este capítulo visa apresentar a política definida para verificar a integridade dos arquivos por meio do uso de *hashes* e também, apresentar a política de confidencialidade dos arquivos por meio da criptografia dos dados. Além disso, este capítulo apresenta quais modificações foram efetuadas no código original do BioNimbuZ para que a integração pudesse ser feita de maneira eficiente. A seção 5.1 descreve com detalhes como é a política de integridade dos arquivos e seus impactos no BioNimbuZ. E a seção 5.2 detalha a política definida para garantir a confidencialidade dos arquivos na plataforma de nuvens federadas.

5.1 Política de Integridade

Um arquivo dentro da plataforma de nuvens federadas BioNimbuZ tem o objetivo de servir como entrada ou como configuração para a realização de tarefas de bioinformática. Assim, todos esses arquivos devem primeiramente ser transferidos ao BioNimbuZ para só depois o *job* poder ser executado. Para que essa transferência ocorra, o usuário deve utilizar somente a opção de *Upload* e selecionar o arquivo que deseja inserir na plataforma, não devendo adicionar manualmente o arquivo ou por outras formas de transferência.

Somente após todos os arquivos estarem disponíveis na plataforma é que o *workflow* de bioinformática poderá ser executado. Assim, esse *job* poderá ser executado logo após os arquivos serem transferidos ou a qualquer momento que o usuário desejar. Assim sendo, a política de integridade proposta para a plataforma BioNimbuZ teve sua ação claramente definida em dois momentos mais críticos, em que o arquivo pode ser danificado ou corrompido. O primeiro é o momento em que o arquivo é transferido da máquina do usuário para o BioNimbuZ. O segundo é durante o tempo em que o arquivo passa armazenado esperando o *job* ao qual ele está relacionado. Ambos momentos estão descritos com mais detalhes nas próximas seções.

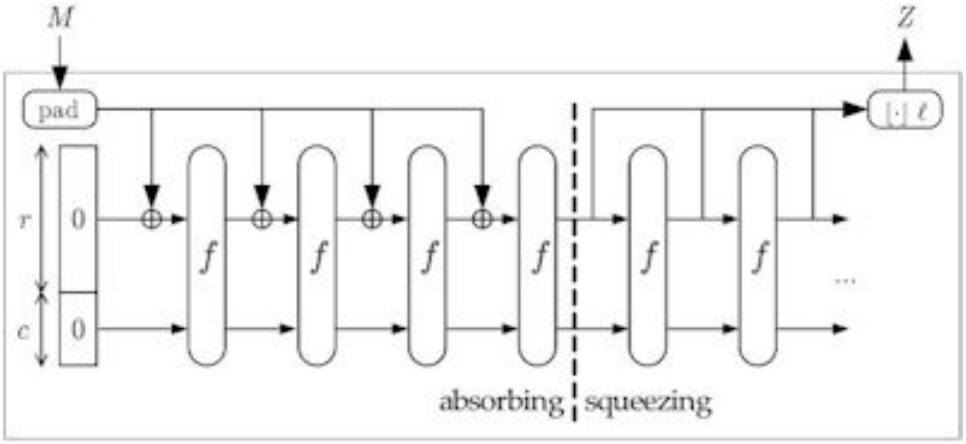

Figura 5.1: Construção Esponja [49].

5.1.1 Verificação da Integridade no Momento da Transferência

A utilização de *hashes* foi o método escolhido para verificar se o arquivo recebido pelo sistema é idêntico ao que foi salvo no provedor da nuvem. Mais especificamente o algoritmo SHA3 [21], descrito a seguir.

Várias funções utilizadas como funções de resumo sofreram quebras, como MD5 e SHA-1, resultando assim a família de algoritmos SHA-2. Dessa forma, sendo essa a única função sem vulnerabilidades graves conhecidas, passa a ser fortemente estudada e alvo de tentativas de quebras. Com sua estrutura sendo questionada e estudada, apontou-se que o uso prolongado desse algoritmo deve ser cauteloso. Para isso, o NIST [17] promoveu um concurso para eleger outro algoritmo e trazer mais uma alternativa confiável de implementação de funções de resumo. Após 5 anos de concurso, a proposta vencedora Keccak [21] passou a ser o novo padrão SHA-3.

Esse algoritmo faz uso do paradigma esponja, composto por duas fases de processamento, conforme mostrado na Figura 5.1. A primeira delas divide a mensagem em blocos e os absorve em estados internos. Esses estados são originados a partir de um estado sendo inicializado com zeros e, em seguida, passa a ser iterado com rodadas que possuem cinco mapeamentos, que fazem a difusão e a distribuição dos elementos nos estados. Depois de finalizados, a função passa para a fase de esmagamento, que por sua vez intercala a aplicação das funções de mapeamento, até que se tenha o número de bits que atende ao tamanho da saída no nível de segurança escolhido [49].

Dessa forma, o processo para verificar a integridade do arquivo durante a sua transferência para o BioNimbuZ segue algumas etapas, como mostrado na Figura 5.2. O passo 1 é selecionar a opção de *Upload* e escolher o arquivo a ser transferido. O usuário só deve transferir os arquivos à plataforma desta forma, pois este é o único caminho que faz o arquivo ser encriptado e também o único caminho que gera o seu *hash* ou assinatura.

O passo 2 é o recebimento do arquivo por parte do BioNimbuZ. Nesta etapa, o sistema analisa o arquivo e recupera os metadados referentes a ele, como nome, extensão e tamanho. Além disso, é durante o passo 2 que o arquivo é encriptado usando o algoritmo AES [1], que será explicado posteriormente na seção de Política de Confidencialidade.

Após isso, o passo 3 consiste na geração do *hash* propriamente dito do arquivo. Como comentado anteriormente, o algoritmo escolhido foi o SHA-3 [21], e para a geração deste

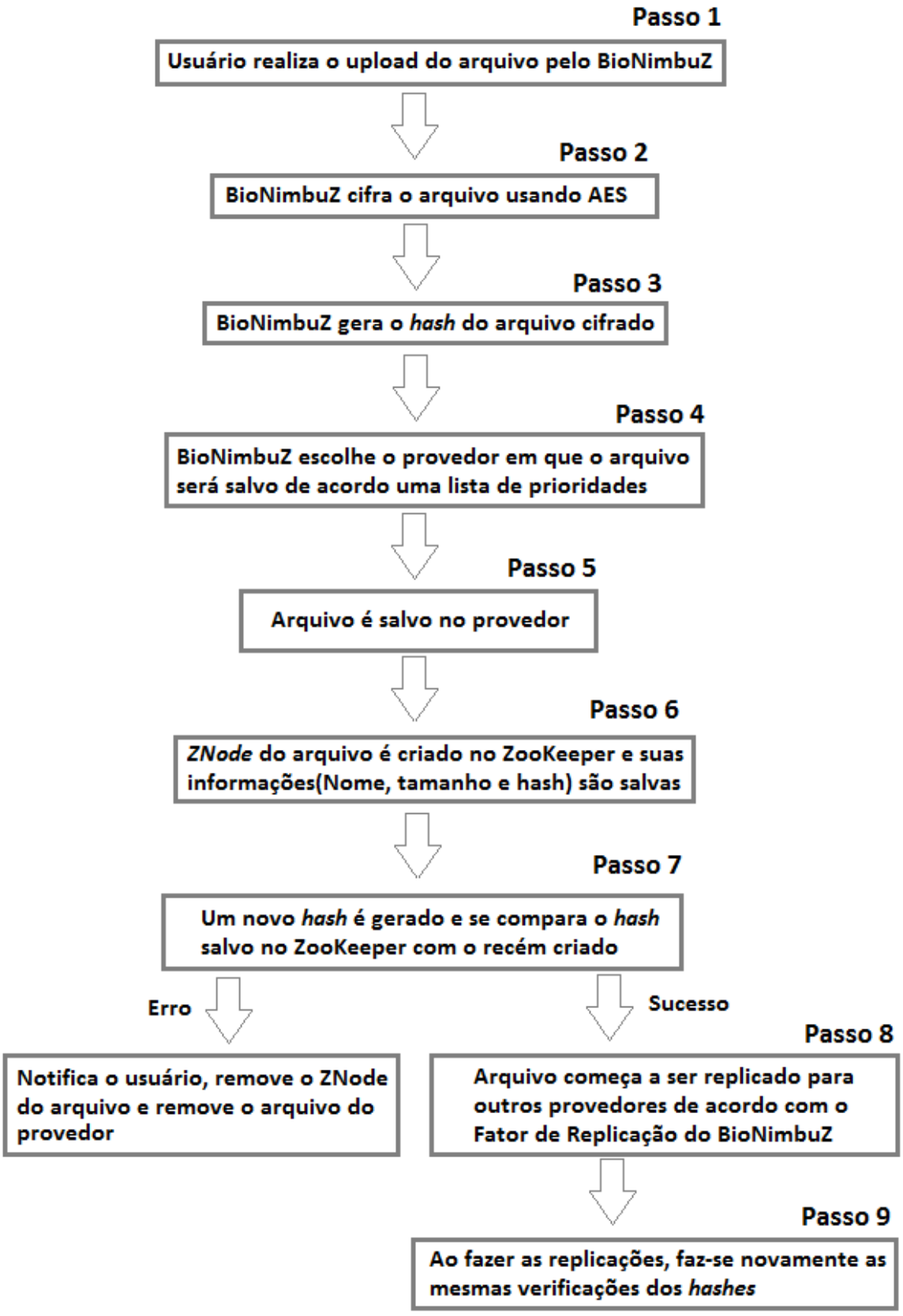

Figura 5.2: Processo de Verificação da Integridade dos Arquivos no BioNimbuZ.

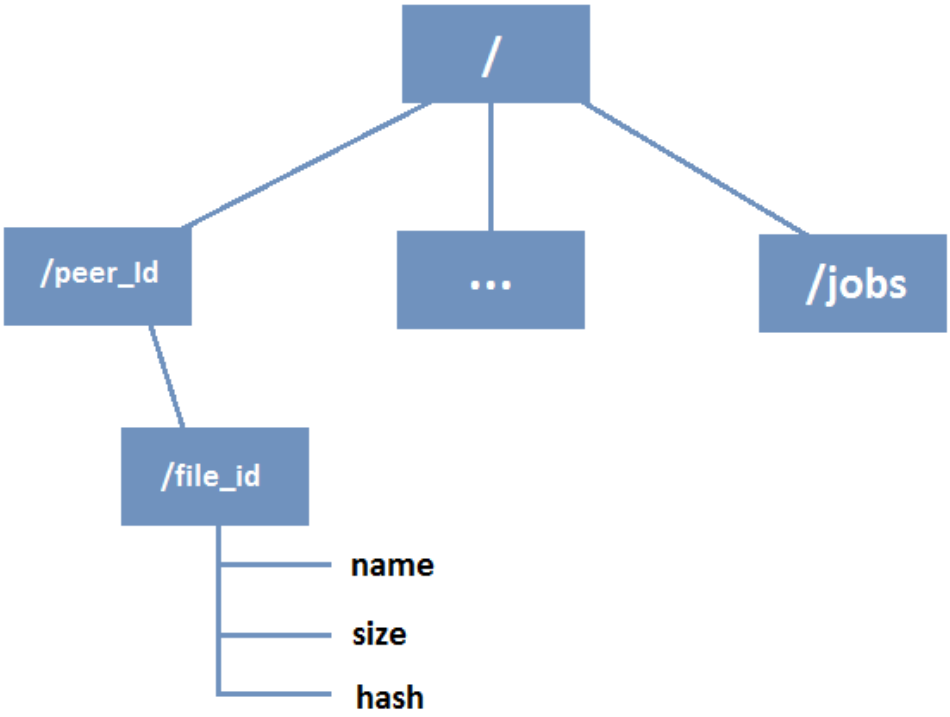

Figura 5.3: Nova Estrutura dos *ZNodes*.

hash foi utilizada uma biblioteca da Bouncy Castle [9], que é um grupo de estudos focado em criptografia que desenvolve bibliotecas e APIs para diversas linguagens. Assim, a função utilizada recebe como parâmetro o caminho do arquivo e retorna um sequência de 256 bits, ou seja, o *hash* do arquivo.

Com os dados do arquivo já gerados, o BioNimbuZ passa à etapa 4 que é a fase de escolher em qual provedor o arquivo será salvo, de acordo com uma lista de vários critérios como preço, velocidade e espaço disponível. Essa decisão é tomada pelo serviço de armazenamento do BioNimbuZ [45] [38] [28].

O passo 5 é a transferência do arquivo para o provedor que foi escolhido anteriormente. Dessa forma, após o arquivo ser inserido no provedor, o passo 6 consiste na inserção no Zookeeper dos metadados do arquivo juntamente com o seu *hash*. Para tal, é criado um novo nó abaixo do nó referente ao provedor, de forma que todos os arquivos daquele provedor terão um nó específico no Zookeeper, conforme mostrado na Figura 5.3. É importante salientar o motivo de se ter escolhido o Zookeeper para armazenar o *hash* e não um banco de dados como MySQL [16] ou PostgreSQL [19].

Na implementação atual do BioNimbuZ, o Zookeeper já está completamente integrado às funcionalidades da plataforma. Este serviço é de fácil manuseio e todas as funções atuais do BioNimbuZ possuem acesso aos seus *ZNodes*. Dessa forma, como a informação que necessita ser inserida é pequena, um *hash* de 256 bits, não há a necessidade de criar um banco de dados específico para isso, haja vista que um *ZNode* pode armazenar até 1 megabyte (MB) de informação.

A partir do momento em que os dados estão salvos no Zookeeper, o processo passa à etapa 7. Nesta fase, o BioNimbuZ, por intermédio do Avro, conecta-se ao provedor e busca nas suas pastas internas o arquivo que foi recentemente transferido. Ao encontrar

esse arquivo, gera-se novamente o *hash* SHA3, e faz-se a comparação do *hash* que foi salvo na etapa anterior com o que foi gerado no momento.

Havendo divergência nos *hashes*, o BioNimbuZ notifica o usuário, remove o *ZNode* referente ao arquivo e também remove o arquivo salvo no provedor. Dessa forma, o usuário pode tentar novamente fazer essa transferência.

Caso a comparação dos *hashes* não encontre nenhuma divergência, o BioNimbuZ passa para a última fase, a etapa 8. Neste momento, o BioNimbuZ começa o processo de replicação desse arquivo para outros servidores de acordo com o Fator de Replicação atual do serviço de armazenamento do BioNimbuZ. Ao replicar esses arquivos, a mesma verificação de *hashes* é feita, e caso encontre algum erro na transferência, o sistema envia o arquivo para outros servidores.

5.1.2 Verificação da Integridade dos Arquivos Armazenados

Assim como os dados podem ser danificados ou corrompidos durante a transferência da máquina do usuário para a nuvem, esses dados podem também sofrer danos durante o período que passam armazenados nos servidores externos à espera do usuário solicitar a execução do *job* de bioinformática.

Dessa forma, a verificação da integridade dos arquivos que já estão armazenados nos servidores funciona da mesma forma como na primeira verificação detalhada acima. Esta verificação dos arquivos que já estão, há algum tempo, armazenados na nuvem é feita usando *hashes*. Na verdade é usado o mesmo processo que é feito quando um arquivo é transferido. A diferença é que neste processo todos os arquivos de todos os servidores são verificados.

Para realizar essa varredura completa, o usuário dispõe agora de uma nova funcionalidade no *Simple Shell* da aplicação chamada *Integrity*. Quando esta função é chamada, o BioNimbuZ primeiramente faz uma varredura pelo ZooKeeper com o objetivo de retornar uma lista com todos os servidores encontrados na federação no momento.

Para cada servidor o sistema faz uma nova varredura para listar os arquivos armazenados naquela máquina. Assim, via Avro, o sistema conecta-se àquele ambiente e procura nos seus diretórios o arquivo, mais especificamente dentro da pasta *data-folder* do projeto. Ao encontrar esse arquivo, gera-se um *hash* SHA-3, e faz-se a comparação do *hash* que está armazenado na estrutura do ZooKeeper com o que foi gerado recentemente.

Havendo divergência nos *hashes*, o BioNimbuZ notifica o usuário e solicita que ele faça novamente o *upload* desse arquivo para a federação com o risco dos seus *jobs* não saírem com o resultado esperado.

5.2 Política de Confidencialidade

A confidencialidade, como detalhada anteriormente, é a garantia do resguardo das informações e proteção contra a sua revelação não autorizada. Dessa forma, dentro da realidade do BioNimbuZ, a partir do momento em que o arquivo é transferido aos servidores externos, o cliente e também o próprio BioNimbuZ, não possuem mais controle sobre esse arquivo. Assim, qualquer entidade que tenha acesso aos dados armazenados nesses servidores poderá verificar o conteúdo desses dados.

Mesmo havendo garantias jurídicas por parte do provedor da nuvem que as informações não serão violadas, há ainda a necessidade de se ter a certeza de que os dados do arquivo não serão visualizados por entidades não autorizadas. E é visando alcançar essa certeza que este trabalho propõe uma forma de garantir que somente requisições do próprio sistema, ao executar tarefas ou executar o *Download*, terão acesso às informações reais do arquivo.

Esses arquivos serão salvos dentro de diretórios de máquinas virtuais hospedadas em diversos servidores externos. Assim, de qualquer modo, caso exista alguma entidade maliciosa dentro do ambiente desse servidor, existirá sempre a possibilidade de se conseguir acessar o conteúdo deste arquivo. Entretanto, o ponto proposto por este trabalho é de que o conteúdo seja criptografado para que, caso ele seja acessado, as informações úteis não consigam ser extraídas.

Dessa maneira, o algoritmo escolhido para criptografar as informações foi o AES [1]. Este algoritmo de criptografia possui um tempo de resposta bastante baixo comparado à outros padrões criptográficos, ou seja, ele consegue criptografar um arquivo de tamanho grande em um relativo curto período de tempo. Um motivo dessa velocidade é o seu processo de encriptação que não emprega operações aritméticas, isto é, não exige muito poder de processamento. Além disso, o processo do algoritmo AES não usa elementos já previamente processados, por exemplo, a etapa 9 do processo de encriptação não necessita de nenhum elemento das etapas anteriores, a não ser única e exclusivamente do estado da etapa 8.

Assim, a política de confidencialidade dos arquivos no BioNimbuZ começa no momento em que o sistema recebe o arquivo do usuário. O modelo proposto, mostrado na Figura 5.4, prevê que o arquivo seja criptografado usando o algoritmo AES, e que só depois desse processo criptográfico, ele possa ser armazenado nos servidores externos. Dessa forma, antes do *hash* para verificar a integridade ser gerado, o arquivo é criptografado. A fórmula desse *hash* é mostrada a seguir na Equação 5.1.

$$Hash = SHA3(AES(arquivoOriginal)) \tag{5.1}$$

Por motivos de segurança, foi definido que o BioNimbuZ não terá acesso livre aos conteúdos originais dos arquivos, somente na situação em que o *job* referente à esse arquivo seja executado, caso em que ocorre a desencriptação deste, e a posterior remoção deste arquivo da nuvem. Assim, o *hash* teve que ser gerado apenas em cima do arquivo encriptado em decorrência da política de confidencialidade estipulada para o BioNimbuZ.

A chave única do algoritmo que será usada no processo de encriptação de todos os arquivos transferidos ao BioNimbuZ é armazenada no código fonte do projeto, o que faz com que somente os envolvidos diretos no projeto do BioNimbuZ tenham acesso à ela.

Além disso, antes do arquivo ser requerido como entrada de uma tarefa, ele é descriptografado utilizando a mesma chave única. Dessa forma, o conteúdo original é usado quando é requerido pelo sistema e o conteúdo criptografado é visto por entidades maliciosas.

O próximo capítulo traz os resultados e testes feitos para comprovar a eficácia e a eficiência das políticas de integridade e de confidencialidade propostas neste trabalho para a plataforma de nuvens federadas BioNimbuZ.

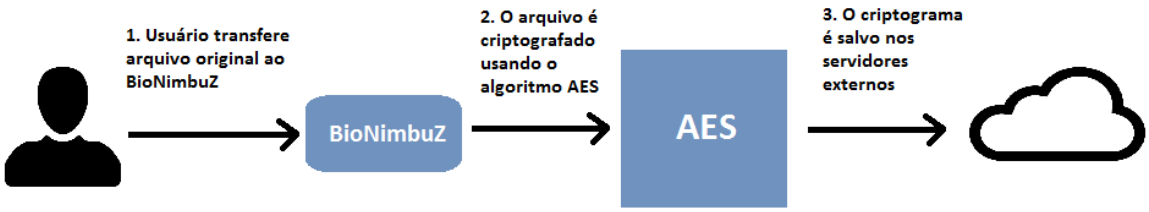

Figura 5.4: Processo do Arquivo ao ser Feito o *Upload* ao BioNimbuZ.

Capítulo 6

Análise dos Resultados

Neste capítulo serão apresentados os resultados obtidos nos testes de performance e nos testes de corretude das políticas propostas neste trabalho. Para isso, serão mostrados em quais ambientes os testes foram executados, quais arquivos foram utilizados e demais características dos testes realizados.

6.1 Experimentos

Os testes realizados neste trabalho tinham dois objetivos. O primeiro objetivo foi verificar o tempo em que os arquivos levavam para serem criptografados, além de verificar quanto tempo era necessário para gerar um *hash* deste arquivo. E o segundo objetivo foi garantir que a criptografia dos arquivos não teria impacto na realização das tarefas de bioinformática executadas na plataforma de nuvens federadas BioNimbuZ. Para isso, os testes foram divididos em três grupos, os quais são apresentados nas Subseções 6.1.2, 6.1.3 e 6.1.4.

6.1.1 Ambiente de Execução

Para a realização dos testes foram utilizadas duas nuvens na federação, as quais tinham as seguintes configurações:

• Uma nuvem privada, localizada na Universidade de Brasília (UnB), composta por 2 computadores com a configuração de 8 GB de memória primária, processador Intel(R) Core(TM) i7-3770 com frequência de 1.6 GHz, com oito núcleos e 2 TB de memória secundária;
• Uma nuvem pública na Azure [15], com duas máquinas com processador A1, com um núcleo, memória de 1,75 GB e 40 GB de memória secundária.

Nestas nuvens estavam executando o BioNimbuZ e o servidor Zookeeper, sendo a primeira nuvem o *host* do Zookeeper. Assim, dado este ambiente, os testes foram realizados a fim de garantir a efetividade das políticas implementadas. Para verificar a política de integridade, foi analisado o tempo em que um arquivo levava para ser criptografado, e depois o tempo necessário para gerar o *hash* deste arquivo encriptado.

6.1.2 Teste de Duração da Criptografia e da Geração dos *Hashes*

Para este teste foram utilizadas duas versões do BioNimbuZ, uma antes das inserções das políticas de integridade e de confidencialidade e uma mais atualizada, já com as políticas propostas. Assim, a primeira versão não possuía as funcionalidades de criptografar com o algoritmo AES esses arquivos, e também não possuía a geração do *hash* SHA-3 destes arquivos.

Visando atingir o objetivo de verificar se a inserção das políticas não acarretaria um *overhead* muito grande nos *uploads* dos arquivos à plataforma de nuvens federadas, foram utilizados arquivos reais que normalmente são utilizados por usuários do BioNimbuZ. Assim, foram utilizados quatro arquivos de sequências genômicas e um arquivo texto de configuração para este fim. As características dos arquivos estão descritas a seguir:

- Arquivo 1 - Tamanho: 252,8 MB - Extensão: .fa
- Arquivo 2 - Tamanho: 246,7 MB - Extensão: .fa
- Arquivo 3 - Tamanho: 200,9 MB - Extensão: .fa
- Arquivo 4 - Tamanho: 114,5 MB - Extensão: .fa
- Arquivo 5 - Tamanho: 448 Bytes - Extensão: .txt (configuração)

O resultado deste teste é mostrado na Figura 6.1. Este gráfico mostra as comparações do tempo que um arquivo leva para ser transferido ao BioNimbuZ passando por um processo criptográfico e por uma geração de *hashes* e do tempo que esse mesmo arquivo leva para ser transferido sem passar por esses processos. Como pode ser visualizado nessa figura, a criptografia dos arquivos e a geração dos seus respectivos *hashes* não impactou um *overhead* muito grande nos *uploads* dos arquivos. Tomando como exemplo o arquivo 2, o tempo que o arquivo demorou para ser criptografado com o algoritmo AES somado com o tempo que o seu *hash* SHA-3 levou para ser gerado foi 19,65% maior que o tempo sem criptografia e sem geração dos *hashes*. O acréscimo na duração do *upload* foi em média 15% do tempo anterior, o que não demonstra algo preocupante já que a quantidade de vezes que um arquivo é carregado na nuvem não é muito alta.

Dessa forma, as políticas propostas neste trabalho demonstraram ser eficientes e eficazes na proteção dos dados armazenados na nuvem federada.

Além disso, a política de integridade inserida neste trabalho contém uma funcionalidade que permite ao usuário verificar se os seus arquivos foram modificados durante o seu armazenamento nas nuvens. O teste a seguir verifica a corretude desta funcionalidade.

6.1.3 Teste de Corretude da Função de *Integrity*

A função *integrity* visa verificar se algum dos arquivos que está armazenado nos servidores da federação sofreu alguma modificação indesejada, pois qualquer modificação em um arquivo de sequência genômica pode vir a gerar um resultado totalmente diferente do esperado.

Dessa forma, foi desenvolvido um teste em que foram feitos primeiramente os *uploads* de 5 arquivos à federação. Após isso, a função *Integrity* do *Simple Shell* foi chamada para verificar se os 5 arquivos continuavam contendo exatamente as mesmas informações dos arquivos que estavam na máquina do usuário. A função, como mostra a Figura 6.2, retornou que os arquivos continuavam íntegros.

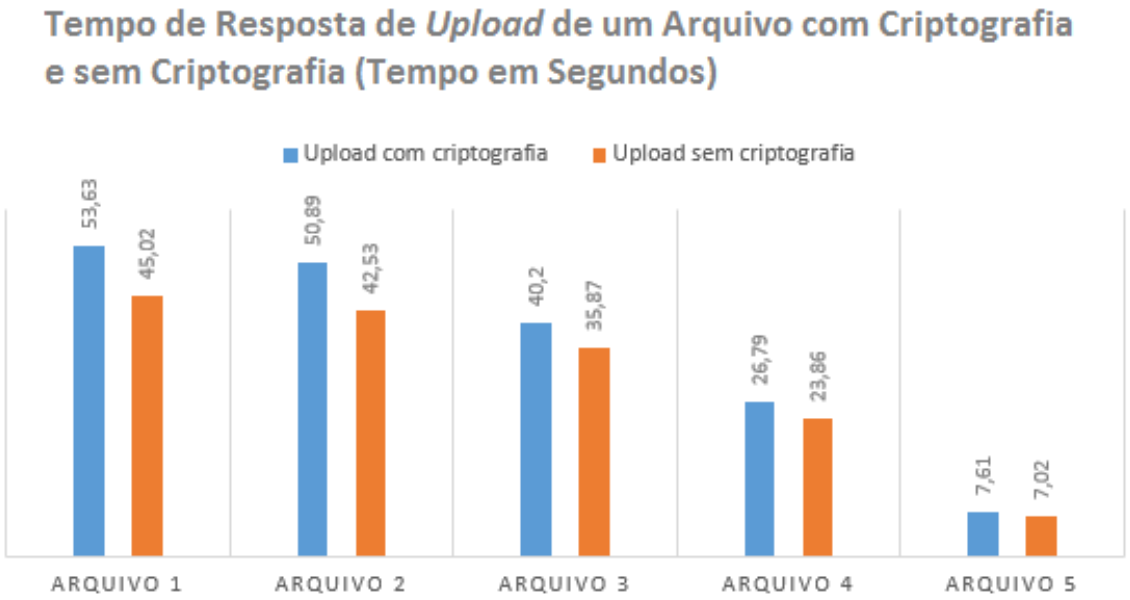

Figura 6.1: Gráfico de Comparação entre os Tempos de *Upload* de um Arquivo Encriptado e de um não Encriptado.

```
2015-11-30 18:33:37,030 DEBUG [main] (AvroClient.java:39) - HTTP client built
File integrity verified: File uploaded correctly.
[@bionimbus]$ integrity
2015-11-30 18:34:17,280 DEBUG [main] (AvroClient.java:39) - HTTP client built
2015-11-30 18:34:17,287 DEBUG [main] (AvroClient.java:39) - HTTP client built
2015-11-30 18:34:17,301 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo s1.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:34:20,734 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo pipelineSample1.txt do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:34:20,736 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr3.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:34:26,840 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr2.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:34:34,507 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr1.fa do peer 164.41.209.88 armazenado corretamente

Verificacao da integridade dos arquivos finalizada:
5 arquivos verificados
0 arquivos com erro
```

Figura 6.2: Verificação da Integridade dos Arquivos - Arquivos não Alterados.

```
[@bionimbus]$ integrity
2015-11-30 18:51:19,455 DEBUG [main] (AvroClient.java:39) - HTTP client built
2015-11-30 18:51:19,462 DEBUG [main] (AvroClient.java:39) - HTTP client built
2015-11-30 18:51:19,471 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr2.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:51:27,299 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr1.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:51:35,263 DEBUG [main] (AvroClient.java:39) - HTTP client built
Erro no armazenamento do arquivo: pipelineSample1.txt do peer 164.41.209.88
2015-11-30 18:51:35,265 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo hs_ref_GRCh37.p10_chr3.fa do peer 164.41.209.88 armazenado corretamente
2015-11-30 18:51:41,449 DEBUG [main] (AvroClient.java:39) - HTTP client built
Arquivo s1.fa do peer 164.41.209.88 armazenado corretamente

Verificacao da integridade dos arquivos finalizada:
5 arquivos verificados
1 arquivos com erro
```

Figura 6.3: Verificação da Integridade dos Arquivos - Arquivos Alterados (Sinalização de Erro).

Entretanto, para forçar um erro e verificar se a função estava funcionando, foi feita uma modificação manual de apenas um caracter em um dos arquivos. Após isso, a função *Integrity* foi chamada novamente e o resultado foi como era esperado, mostrado na Figura 6.3, o retorno apontou um erro de integridade em um dos arquivos.

6.1.4 Teste de Corretude dos *Jobs* utilizando Arquivos Encriptados

Por fim, para garantir que a criptografia dos arquivos não estava alterando o funcionamento da execução dos *jobs*, foi feito um teste para verificar a saída da tarefa utilizando arquivos encriptados. Para isso, foram feitos os *uploads* dos arquivos de entrada e de configuração necessários para a execucação da tarefa.

Após isso, a função *PipelineTestGenerator*, que faz a leitura desses arquivos e executa o *job* de bioinformática, foi chamada. O resultado mostrou que a tarefa foi executada com sucesso, demonstrando que, apesar dos arquivos se encontrarem criptografados na federação, o BioNimbuZ conseguiu descriptografar os arquivos e executar os *jobs* como se os arquivos estivessem armazenados em seus estados naturais na nuvem. Esse teste pode ser verificado por meio da Figura 6.4.

6.1.5 Considerações Finais

Neste capítulo, foram apresentados os testes realizados no serviço de segurança da plataforma BioNimbuZ, que utiliza a política de integridade e a de confidencialidade dos arquivos, propostas neste trabalho.

Os testes tinham dois objetivos, sendo o primeiro o de verificar se a criptografia e a geração do *hash* geravam um *overhead* grande ao BioNimbuZ, e o segundo objetivava verificar se a execução dos *jobs* estava gerando resultados corretos, mesmo recebendo como entrada arquivos criptografados.

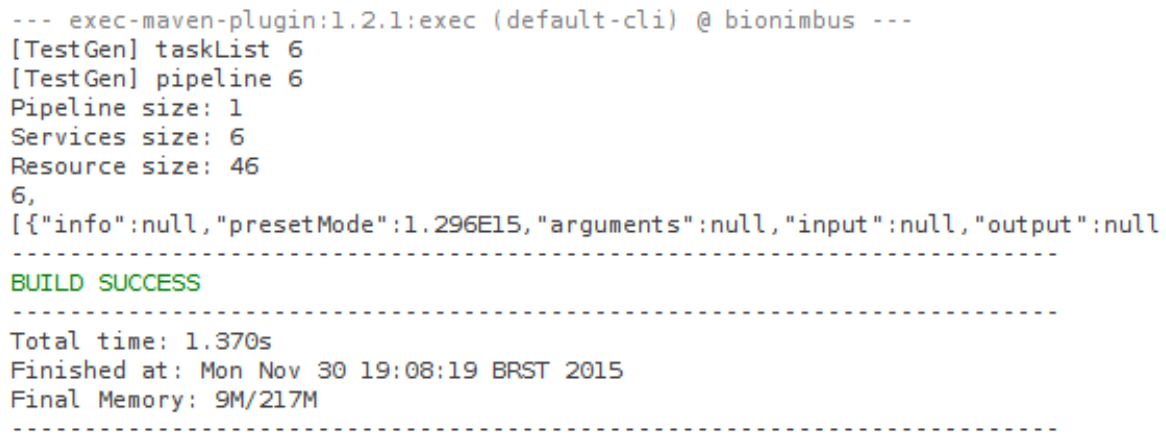

Figura 6.4: Resultado Correto de um *Job* que Utilizou Arquivos Encriptados como Entradas da Tarefa.

Assim sendo, os dois objetivos foram atingidos neste trabalho, pois foi mostrado que o *overhead* para criptografar e gerar os *hashes* foi de menos de 15%, e ainda os *jobs* conseguiam executar suas tarefas com arquivos inicialmente criptografados.

Capítulo 7

Conclusão e Trabalhos Futuros

A computação em nuvem é uma forte tendência atualmente e que tem como objetivo proporcionar enormes benefícios de custo, agilidade e escalabilidade para vários tipos de negócio, porém trouxe também algumas novas preocupações concernentes à segurança dos dados. Este trabalho, dessa maneira, priorizou duas propriedades essenciais da segurança de uma federação de nuvens, a integridade e a confidencialidade dos seus arquivos.

Neste trabalho foram propostos políticas de integridade e de confidencialidade dos arquivos na plataforma de nuvens federadas BioNimbuZ. Este trabalho foi o segundo projeto que englobou o tema de segurança no BioNimbuZ, sendo o primeiro projeto responsável por verificar o controle de acesso dos usuários na plataforma de nuvens federadas [35].

O modelo proposto por este projeto se baseia na verificação de integridade dos arquivos por meio do uso de *hashes* SHA-3 [21] tanto no momento em que o arquivo é transferido ao provedor de armazenamento na nuvem quanto no período em que ele passa armazenado nesses servidores. Além disso, o modelo deste trabalho inclui uma política de confidencialidade dos arquivos por meio da criptografia dos dados utilizando o algoritmo AES [1].

Os testes realizados demonstraram que o *overhead* gerado pela criptografia e pela geração dos *hashes* foi de menos de 15% além de demonstrarem que os *workflows* de bioinformática continuaram a funcionar normalmente mesmo com os arquivos encriptados.

Antes da implementação dessas políticas na federação, os usuários não sabiam se a transferência que eles haviam feito tinha sido correta, e também se os seus arquivos estavam armazenados corretamente nos servidores. Além disso, os usuários não possuíam uma garantia real de que seus arquivos e seus respectivos dados não estavam sendo acessados por entidades maliciosas. Assim, depois deste projeto, o usuário tem a certeza do *status* do seu arquivo logo após a transferência para federação, e também a qualquer momento que desejar fazer essa verificação.

Como trabalhos futuros, propõe-se uma troca automática dos arquivos caso se encontre algum erro de integridade nesses arquivos. Dessa forma, caso as funções implementadas neste trabalho encontrassem divergência nos *hashes*, ao invés de comunicar diretamente o usuário, o BioNimbuZ irá fazer uma varredura e encontrar em qual outro servidor o arquivo está replicado e assim fazer a substituição do arquivo corrompido pelo correto.

Além disso, o usuário do BioNimbuZ poderia possuir a opção de escolher a chave que desejaria encriptar o seu arquivo, de forma que somente o dono do arquivo saberia a chave usada para descriptografar o arquivo ao executar os *workflows* de bioinformática.

Por fim, propõe-se ainda a inserção de um controle de acesso dos usuários mais voltado à realidade da plataforma BioNimbuZ. Seria interessante que os usuários e suas respectivas permissões estivessem armazenados no ZooKeeper e que esse controle estivesse integrado com a interface web do projeto.

Referências

[1] Aes. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard. Accessed: 2015-09-20. 31, 40, 44, 51

[2] Amazon ws. https://aws.amazon.com/pt/?nc2=h_lg. Accessed: 2015-09-30. 6

[3] Apache foundation. http://http://www.apache.org/. Accessed: 2015-09-30. 15

[4] Assinatura digital. http://esaj.tjce.jus.br. Accessed: 2015-11-19. vii, 28

[5] Avro apache. https://avro.apache.org/. Accessed: 2015-11-21. 15

[6] Avro http. http://tools.ietf.org/pdf/rfc2616.pdf. Accessed: 2015-11-21. 15

[7] Avro json. http://tools.ietf.org/pdf/rfc4627.pdf. Accessed: 2015-11-21. 15

[8] Blowfish. https://en.wikipedia.org/wiki/Blowfish_(28cipher). Accessed: 2015-09-30. 31

[9] Bouncy castle. https://www.bouncycastle.org/about.html. Accessed: 2015-11-21. 42

[10] Crypto stack exchange. http://http://crypto.stackexchange.com/questions/2711/does-the-mixcolumns-step-come-before-or-after-addroundkey-in-aes-decryption. Accessed: 2015-10-27. vii, 33

[11] Des. https://en.wikipedia.org/wiki/Data_Encryption_Standard. Accessed: 2015-09-30. 31

[12] Hmac. https://www.openssl.org/docs/manmaster/crypto/hmac.html. Accessed: 2015-11-21. 28

[13] Jonathan hepp. http://www.jonathanhepp.com.br/2012/10/algoritmos-de-hash-unidirecionais-para_16.html#.VmDX19CLfIU. Accessed: 2015-09-30. vii, 27

[14] Md5. https://tools.ietf.org/html/rfc1321. Accessed: 2015-11-21. 28

[15] Microsoft azure. https://azure.microsoft.com/. Accessed: 2015-11-10. 46

[16] Mysql. https://www.mysql.com/. Accessed: 2015-12-02. 42

[17] Nist. http://www.nist.gov/. Accessed: 2015-11-21. 40

[18] Pbkdf2. https://en.wikipedia.org/wiki/PBKDF2. Accessed: 2015-11-21. 36

[19] Postgresql. http://www.postgresql.org/. Accessed: 2015-12-02. 42

[20] Rsa. http://searchsecurity.techtarget.com/definition/RSA. Accessed: 2015-10-16. 31

[21] Sha-1. https://www.openssl.org/docs/manmaster/crypto/sha.html. Accessed: 2015-11-21. 28, 40, 51

[22] Ashraf Aboulnaga, Kenneth Salem, Ahmed A Soror, Umar Farooq Minhas, Peter Kokosielis, and Sunil Kamath. Deploying database appliances in the cloud. *IEEE Data Eng. Bull.*, 32(1):13–20, 2009. 8

[23] NEDHAL A AL-SAIYD and NADA SAIL. Data integrity in cloud computing security. *Journal of Theoretical & Applied Information Technology*, 58(3), 2013. 26, 27, 28, 29, 30

[24] Coud Sercurity Alliance. Top threats to cloud computing v1. 0. *White Paper*, 2010. 23

[25] Aletéia Patrıcia Favacho Araújo. *Paralelizaç ao Autonômica de Metaheurısticas em Ambientes de Grid*. PhD thesis, PhD thesis, Departamento de Informática, Pontificia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2008. 5, 6

[26] Rachna Arora, Anshu Parashar, and Cloud Computing Is Transforming. Secure user data in cloud computing using encryption algorithms. *International Journal of Engineering Research and Applications*, 3(4):1922–1926, 2013. viii, 31, 33, 34, 35

[27] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, 2007. 22

[28] Diego Rodrigues Azevedo, Breno Rodrigues Moura, and Aletéia Patrıcia Favacho de Araújo. Nova polıtica de armazenamento em nuvens federadas para a plataforma bionimbuz. 42

[29] William C Barker. Guide for mapping types of information and information systems to security categories. *Network Security*, 2003. 22

[30] Karin Bernsmed, Martin Gilje Jaatun, Per Hakon Meland, and Astrid Undheim. Thunder in the clouds: Security challenges and solutions for federated clouds. In *Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on*, pages 113–120. IEEE, 2012. 2

[31] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010. 12

[32] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009. 1, 10

[33] Antonio Celesti, Francesco Tusa, Massimo Villari, and Antonio Puliafito. How to enhance cloud architectures to enable cross-federation. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, pages 337–345. IEEE, 2010. 2

[34] Yao Chen and Radu Sion. On securing untrusted clouds with cryptography. In *Proceedings of the 9th annual ACM workshop on Privacy in the electronic society*, pages 109–114. ACM, 2010. 31

[35] Heitor Henrique de Paula Moraes Costa. *Controle de Acesso na Plataforma de Nuvem Federada BioNimbuZ*. PhD thesis, Universidade de Brasília, 2015. 14, 39, 51

[36] C Dinesh. Data integrity and dynamic storage way in cloud computing. *arXiv preprint arXiv:1111.2418*, 2011. vii, 6

[37] Fabricio Dyck. Computaç ao em nuvem e segurança. 24, 25

[38] Ricardo Gallon, Maristela Holanda, Aletéia Araújo, and Maria E Walter. Storage policy for genomic data in hybrid federated clouds. In *Advances in Bioinformatics and Computational Biology*, pages 107–114. Springer, 2014. 42

[39] Patrick Hunt, Mahadev Konar, Flavio Paiva Junqueira, and Benjamin Reed. Zookeeper: Wait-free coordination for internet-scale systems. In *USENIX Annual Technical Conference*, volume 8, page 9, 2010. 15

[40] Dean Jacobs, Stefan Aulbach, et al. Ruminations on multi-tenant databases. In *BTW*, volume 103, pages 514–521, 2007. 7

[41] Yashpalsinh Jadeja and Kirit Modi. Cloud computing-concepts, architecture and challenges. In *Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on*, pages 877–880. IEEE, 2012. 6

[42] MT Jones. Cloud computing with linux cloud pomputing platforms and applications [eb/ol], 2008. 5

[43] Arlindo Marcon Jr, Marcos Laureano, Altair Santin, and Carlos Maziero. Aspectos de segurança e privacidade em ambientes de computação em nuvem. 22, 24, 25

[44] Dimitrios Lekkas. Establishing and managing trust within the public key infrastructure. *Computer Communications*, 26(16):1815–1825, 2003. 22

[45] Breno Rodrigues Moura and Deric Lima Bacelar. Política para armazenamento de arquivos no zoonimbus. 2013. vii, 10, 11, 14, 15, 16, 18, 42

[46] Debajyoti Mukhopadhyay, Gitesh Sonawane, Parth Sarthi Gupta, Sagar Bhavsar, and Vibha Mittal. Enhanced security for cloud storage using file encryption. *arXiv preprint arXiv:1303.7075*, 2013. vii, 34, 36, 38

[47] Aarthi Nagarajan and Vijay Varadharajan. Dynamic trust enhanced security model for trusted platform based services. *Future Generation Computer Systems*, 27(5):564–573, 2011. 22

[48] Gabriel Silva Souza de Oliveira. Acosched: um escalonador para o ambiente de nuvem federada zoonimbus. 2013. 14, 18

[49] Gracielle Forechi Olivier. Estudo e implementação do algoritmo de resumo criptográfico sha-3. 2013. vii, 40

[50] Lili Qiu, Yin Zhang, Feng Wang, Mi Kyung, and Han Ratul Mahajan. Trusted computer system evaluation criteria. In *National Computer Security Center*. Citeseer, 1985. 22

[51] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*, pages 44–51. Ieee, 2009. 1

[52] Abha Sachdev and Mohit Bhansali. Enhancing cloud computing security using aes algorithm. *International Journal of Computer Applications*, 67(9):19–23, 2013. 30

[53] Hugo Saldanha, Aletéia Araújo, Carlos Borges, Edward Ribeiro, João Carlos Setubal, Maria Emília Walter, Maristela Holanda, Ricardo Gallon, and Roberto Togawa. *Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows*. INTECH Open Access Publisher, 2012. vii, 2, 7, 8, 14

[54] Walter J Scheirer, William Bishop, and Terrance E Boult. Beyond pki: The biocryptographic key infrastructure. In *Security and Privacy in Biometrics*, pages 45–68. Springer, 2013. 28

[55] Robin L Sherman. Distributed systems security. *Computers & Security*, 11(1):24–28, 1992. 22, 23, 26

[56] Flávio RC Sousa, Leonardo O Moreira, and Javam C Machado. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. *II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)*, pages 150–175, 2009. vii, 6, 7, 9, 13

[57] Prashant Srivastava, Satyam Singh, Ashwin Alfred Pinto, Shvetank Verma, Vijay K Chaurasiya, and Rahul Gupta. An architecture based on proactive model for security in cloud computing. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on*, pages 661–666. IEEE, 2011. 21

[58] Ion Stoica, Robert Morris, David Liben-Nowell, David R Karger, M Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *Networking, IEEE/ACM Transactions on*, 11(1):17–32, 2003. 14

[59] David Talbot. How secure is cloud computing. *Technology Review (Nov 2009), http://www. technologyreview. com/computing/23951*, 2009. 31