UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA BASEADA EM CONFIANÇA PARA A VERIFICAÇÃO DA INTEGRIDADE DE ARQUIVOS EM NUVENS COMPUTACIONAIS

ALEXANDRE PINHEIRO

ORIENTADOR: RAFAEL TIMÓTEO DE SOUSA JÚNIOR

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM – 634/2016
BRASÍLIA/DF: JULHO – 2016

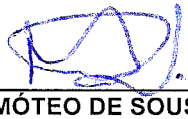
UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ARQUITETURA BASEADA EM CONFIANÇA PARA A VERIFICAÇÃO DA INTEGRIDADE DE ARQUIVOS EM NUVENS COMPUTACIONAIS

ALEXANDRE PINHEIRO

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADA POR:



RAFAEL TIMÓTEO DE SOUSA JR, Dr., ENE/UNB
(ORIENTADOR)

GEORGES DANIEL AMVANE NZE, Dr., ENE/UNB
(EXAMINADOR INTERNO)

Aldri Luiz dos Santos

ALDRI LUIZ DOS SANTOS, Dr., UFPR
(EXAMINADOR EXTERNO)

Brasília, 11 de julho de 2016.

FICHA CATALOGRÁFICA

PINHEIRO, ALEXANDRE

Arquitetura Baseada em Confiança para a Verificação da Integridade de Arquivos em Nuvens Computacionais [Distrito Federal] 2016.

xiii, 109p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2016).

Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.

Departamento de Engenharia Elétrica.

1.Monitoramento 2.Integridade

3.Nuvens Computacionais 4.Confiança

5. Criptografia

I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PINHEIRO, ALEXANDRE (2016). Arquitetura Baseada em Confiança para a Verificação da Integridade de Arquivos em Nuvens Computacionais. Dissertação de Mestrado em Engenharia Elétrica, Publicação PPGENE.DM-634/2016, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 122p.

CESSÃO DE DIREITOS

AUTOR: Alexandre Pinheiro.

TÍTULO: Arquitetura Baseada em Confiança para a Verificação da Integridade de Arquivos em Nuvens Computacionais.

GRAU: Mestre ANO: 2016

Alexandre Pinheiro
SQN 111 – Bloco B – Apto 601, Asa Norte.
70.754-020 Brasília – DF – Brasil.

RESUMO

ARQUITETURA BASEADA EM CONFIANÇA PARA A VERIFICAÇÃO DA INTEGRIDADE DE ARQUIVOS EM NUVENS COMPUTACIONAIS

As vantagens de usar nuvens computacionais incluem a escalabilidade, disponibilidade e uma "ilimitada" capacidade de armazenamento. Entretanto, é um desafio disponibilizar serviços de armazenamento que sejam, ao mesmo tempo, seguros do ponto de vista do cliente e possam ser executados em infraestruturas de nuvem públicas gerenciadas por provedores de serviço considerados não totalmente confiáveis. Proprietários de grandes quantidades de dados necessitam armazená-los na nuvem por longos períodos de tempo sem a necessidade de manter cópias dos dados originais ou acessá-los. Nestes casos, questões como integridade, disponibilidade, privacidade e confiança ainda são desafios a serem superados para a adoção de serviços de armazenamento na nuvem e garantir a segurança, especialmente quando a perda ou vazamento de informações possam levar a significantes prejuízos sejam por razões legais ou de negócio. Com essa preocupação em mente, este trabalho apresenta uma arquitetura computacional que permite monitorar periodicamente as informações armazenadas na nuvem e o comportamento dos serviços de armazenamento contratados. A arquitetura baseia-se na proposta de um protocolo que aplica conceitos de confiança e criptografia para garantir a integridade dos dados armazenados na nuvem sem comprometer a confidencialidade ou sobrecarregar os serviços de armazenamento. Para corroborar a aplicabilidade da arquitetura, após a sua implementação, foram realizados testes e simulações por meio dos quais foi possível determinar características funcionais da arquitetura. Além disso, por intermédio da análise dos resultados foi possível confirmar a eficácia das aplicações desenvolvidas e confirmar características importantes do protocolo do proposto.

ABSTRACT

TRUST BASED ARCHITECTURE FOR CHECKING OF FILE INTEGRITY IN COMPUTER CLOUDS

The advantages of using cloud computing include scalability, availability and an 'unlimited' storage capacity. However, it is a challenge to build storage services that they are at the same time safe from the client point-of-view, and that they run in public cloud infrastructures managed by service providers that can not be fully considered trustworthy. Owners of large amounts of data have to keep their data in cloud for a long period of time without the need to keep copies of the original data or to access it. In such cases, questions of integrity, availability, privacy and trust are still challenges in the adoption of Cloud Storage Services to ensure security, especially when losing or leaking information can bring significant damage, be it legal or business-related. With such concerns in mind, this paper proposes a architecture for periodically monitoring both the information stored in the cloud and the behaviour of contracted storage services. The architecture is based on the proposal for a protocol that applies trust and encryption concepts to ensure the integrity of data stored in the cloud without compromising confidentiality or overload storage services. To corroborate the applicability of the architecture, after its implementation, testing and simulations were performed, by means of which, it was possible to determine functional characteristics of the architecture. Furthermore, through the analysis of the results it was possible to confirm both the efficacy of the developed applications and important characteristics of the proposed protocol.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

3DES	*Triple Data Encryption Standard*
AES	*Advanced Encryption Standard*
Amazon S3	*Amazon Simple Storage Service*
CBC	*Cipher-Block Chaining*
CN	Computação em Nuvem
DES	*Data Encryption Standard*
DHT	Tabela *Hash* Distribuída
ER	Entidades e Relacionamentos
GED	Gestão Eletrônica de Documentos
GFS	*Google File System*
HDFS	*Hadoop Distributed File System*
HMAC	*Hash-based Message Authentication Code*
IaaS	Infraestrutura como Serviço
LP	Linguagem de Programação
MD5	*Message-Digest 5*
MHT	*Merkle Hash Tree*
NISTI	*National Institute of Standards and Technology*
NTFS	Sistema de Arquivos de Nova Tecnologia
PAA	Processo de Armazenamento do Arquivo
PaaS	Plataforma como Serviço
PBKDF2	*Password based Key Derivative Function*
PV	Processo de Verificação
SaaS	Software como Serviço
SADN	Serviço de Armazenamento de Dados na Nuvem
SGBD	Sistema Gerenciador de Banco de Dados
SHA	*Secure Hash Algorithms*
SHS	*Secure Hash Standard*
SVI	Serviço de Verificação de Integridade
TabInfor	Tabela de Informações
TI	Tecnologia da Informação
TPA	*Third Party Auditor*

URL	*Uniform Resource Locator*
VC	Valor da Confiança
VD	Valor do Decremento
VI	Valor do Incremento
VM	Máquina Virtual

1 - INTRODUÇÃO

A incrível velocidade com a qual evoluiu a Tecnologia da Informação (TI) nas últimas décadas proporcionou uma grande mudança no cotidiano das pessoas e em praticamente todas as áreas do conhecimento. A diversificação dos meios de acesso com a utilização de dispositivos móveis, a evolução de tecnologias como a virtualização, somada a uma crescente exigência dos usuários por novos sistemas e serviços adaptados a essas novas tendências de mercado, foram o combustível para o surgimento de um novo paradigma, a Computação em Nuvem (CN).

As "nuvens" são a evolução natural dos tradicionais *datacenters,* dos quais se distingue por oferecer recursos computacionais como armazenamento, processamento e aplicações gerenciadas por meio de *web services* padronizados e por seguir um modelo comercial onde o cliente é cobrado pela utilização destes recursos. Os provedores de CN oferecem aos seus clientes, na maioria dos casos, de forma rápida e sem burocracia, através da simples assinatura de um contrato de serviços, acesso à infraestrutura computacional, plataformas de software e aplicações (Buyya, Broberg, Goscinski 2011).

Dentre os serviços disponibilizados pelos provedores de nuvem, um dos mais comuns é o serviço de armazenamento de dados. O *Amazon Simple Storage Service (Amazon* S3) (Amazon Web Services 2016), o *Microsoft Azure Storage* (Microsoft 2016) e o *Google Cloud Storage* (Google 2016) são exemplos de serviços de armazenamento pertencentes ao portfólio de serviços comercializados pelas plataformas de CN de grandes corporações internacionais.

Serviços de Armazenamento de Dados na Nuvem (SADNs) são rápidos, baratos e extensivamente escaláveis. No entanto, existem problemas relacionados a confiabilidade dos dados, uma vez que mesmo os melhores provedores de serviços às vezes falham (Tandel, Shah, Hiranwal 2013). Para muitas organizações, a segurança e a privacidade são os principais obstáculos à contratação de serviços na nuvem (Dabas, Wadhwa 2014). Desta forma, é importante que os SADNs sejam capazes de fornecer mecanismos para confirmar a integridade dos dados armazenados e ainda garantir a sua privacidade.

Apesar da flexibilidade oferecida pela CN, eximindo os usuários de se preocupar com a complexidade de gerenciamento da infraestrutura de armazenamento, manter dados em ambientes terceirizados nem sempre é confiável, pois torna os seus usuários dependentes da disponibilidade e da integridade fornecida pelos provedores do serviço. Nessa situação, falhas estruturais ou humanas, que possam corromper ou permitir o vazamento dos dados armazenados, assim como, no pior dos casos, a exclusão deliberada destes dados, podem ser omitidas dos seus proprietários por longos períodos de tempo, principalmente quando os dados em questão são raramente acessados.

Considerando o exposto, torna-se necessária a adoção de medidas que garantam, do ponto do vista do cliente, a integridade e a confidencialidade dos arquivos armazenados na nuvem. A confidencialidade pode ser obtida através da aplicação de soluções criptográficas conhecidas como o algoritmo criptográfico *Advanced Encryption Standard* (AES) (NIST FIPS PUB 197). Entretanto, garantir a integridade ainda é um desafio.

Existem vários trabalhos de pesquisa que tratam da verificação da integridade de arquivos, tais como George e Sabitha (2013), Kavuri, Kancherla e Bobba (2014), Al-Jaberi e Zainal (2014), Kay et al. (2013) e Wang et al. (2009). Como resultado, estas pesquisas propõem diferentes abordagens para permitir que o cliente verifique a integridade de seus arquivos, dentre as quais pode-se destacar a aplicação de criptografia assimétrica homomórfica, uso de algoritmos criptográficos de *hashes,* auditoria realizada por terceiros e o uso de desafios.

As propostas analisadas apesar de apontarem diferentes mecanismos para a verificação da integridade de arquivos, elas apresentam características que dificultam a sua utilização fora do domínio para o qual foram concebidas. Para utilização das soluções propostas no monitoramento periódico de integridade de grandes arquivos destaca-se os seguintes óbices: o alto consumo de recursos computacionais devido a utilização da criptografia assimétrica (George, Sabitha 2013) (Kay et al. 2013) (Wang et al. 2009); e o alto consumo de banda de rede gerado pela necessidade de recuperar todo o conteúdo do arquivo antes de realizar a verificação (Kavuri, Kancherla, Bobba 2014) (Al-Jaberi, Zainal 2014).

Além disso, nenhuma das soluções propostas apresentam qualquer mecanismo que tenha como objetivo responder proativamente a falhas identificadas em determinado provedor de

serviço de forma a agilizar o processo de verificação do restante dos arquivos nele armazenados. Outro ponto não abordado nos trabalhos analisados corresponde ao desperdício de recursos computacionais utilizados na verificação de arquivos armazenados em provedores com que apresentam um excelente histórico, principalmente nos quesitos integridade e disponibilidade.

Neste contexto, a arquitetura proposta neste trabalho, composta de um protocolo e de sua implementação, apresenta uma solução completa para monitoramento permanente da integridade de arquivos armazenados na nuvem, utilizando exclusivamente criptografia simétrica e gerando um baixo e previsível consumo de banda de rede. A solução proposta ainda utiliza conceitos de confiança para realizar um balanceamento da carga de verificações de integridade, priorizando a verificação dos arquivos armazenados nos SADNs que já apresentaram falhas e poupando recursos computacionais daqueles SADNs que historicamente proveem um serviço de armazenamento de qualidade.

1.1 - MOTIVAÇÃO

Com o crescimento da aplicação da Gestão Eletrônica de Documentos (GED) nos órgãos integrantes da administração pública, assim como na iniciativa privada, a necessidade de manter a integridade das informações tornou-se ainda mais necessária. Com a utilização da GED, substituiu-se os documentos em meio físico (papel) por documentos exclusivamente eletrônicos, aos quais a validade, a temporalidade e a sua classificação em relação ao sigilo estão amparadas na legislação vigente. A temporalidade dos documentos eletrônicos é o prazo para a sua guarda e conservação definido na legislação, que varia na maioria dos casos entre 5 e 30 anos, de acordo com a natureza e o conteúdo do documento.

Tendo em vista que após a tramitação e o arquivamento de um documento, apesar de a legislação exigir a sua guarda, é muito rara a necessidade de acessá-lo, principalmente quando já decorridos mais de um ano do seu arquivamento. A depender da quantidade e do conteúdo dos documentos tramitados na organização, o tamanho das cópias de segurança podem facilmente ultrapassar uma dezena de gigabytes.

Assim, o uso de SADNs para o seu armazenamento pode ser uma alternativa viável, tendo em vista a possibilidade de redução da necessidade de investimento na infraestrutura de TI

da organização. No entanto, para a utilização de SADNs, principalmente em razão das exigências legais, é necessário garantir que a integridade dos arquivos armazenados possam ser permanentemente monitoradas.

Os trabalhos atuais abordando a verificação da integridade de arquivos armazenados na nuvem, apesar de apresentarem soluções interessantes para os domínios para os quais foram desenvolvidos, quando analisados individualmente, deixam a desejar em alguns aspectos como: alto custo computacional seja pelo uso de criptografia assimétrica homomórfica ou pela necessidade de recuperar o arquivo como um todo para realizar a verificação; exigir confiança plena no Serviço de Verificação de Integridade (SVI) devido ao compartilhamento das chaves criptográficas necessárias ao acesso ao conteúdo do arquivo durante a verificação; e falta de redundância.

Considerando os problemas ainda existentes e expostos acima, torna-se necessário um estudo sobre as tecnologias disponíveis e o desenvolvimento de uma proposta de um protocolo, que permita aos proprietários de arquivos armazenados na nuvem monitorar permanentemente a integridade destes por meio de um serviço especializado em verificação de integridade fornecido por terceiros, de baixo custo computacional e passível de redundância.

1.2 - OBJETIVOS DO TRABALHO

O objetivo geral deste trabalho é viabilizar a oferta de Serviços de Verificação de Integridade (SVI) providos por terceiros por meio dos quais clientes de Serviços de Armazenamento de Dados na Nuvem (SADNs) possam monitorar continuamente a integridade de seus arquivos.

Os objetivos específicos deste trabalho são:

- prover um mecanismo que permita a verificação periódica da integridade de grandes arquivos, durante longos períodos de tempo, sem que os SADNs necessitem acessar o conteúdo total do arquivo a cada verificação.

- garantir a recuperabilidade de um arquivo armazenado na nuvem por meio de redundância, conforme a criticidade do mesmo, permitindo que o cliente não precise manter cópia do arquivo original;

- impedir que os SADNs obtenham o acesso ao conteúdo dos arquivos armazenados por meio da utilização de algoritmos criptográficos e senhas fortes o suficiente para resistir a ataques de força bruta durante o período previsto de armazenamento;

- possibilitar que o SVI realize uma verificação fracionada da integridade dos arquivos armazenados na nuvem, sem necessitar, no entanto, ter acesso a qualquer parte dos conteúdos do arquivo original, do arquivo cifrado armazenado ou de sua chave criptográfica;

- evitar que os SADNs possam cobrar por serviços de armazenamento que na prática não estão sendo prestados;

- prover um protocolo que tenha um baixo custo computacional, utilizando para isso única e exclusivamente técnicas baseadas em criptografia simétrica;

- possibilitar que o SVI realize uma classificação contínua dos SADNs, atribuindo-lhes um nível de confiança de acordo com os resultados das verificações nos arquivos armazenados pelos mesmos;

- prover um mecanismo que possibilite que o SVI realize o balanceamento da carga diária de verificações dos arquivos priorizando aqueles armazenados em SADNs que já apresentaram falhas e reduzindo o consumo de recursos computacionais nos SADNs que, por não apresentarem histórico de falhas, são considerados mais confiáveis.

- implementar e disponibilizar uma arquitetura computacional de acordo com o proposto no protocolo;

- comprovar a eficiência e a eficácia da arquitetura implementada por meio de simulações e testes.

1.3 - METODOLOGIA DE PESQUISA

Este trabalho se caracteriza como uma pesquisa de natureza aplicada e abordagem qualitativa, com objetivo exploratório onde a análise bibliográfica, a seleção das tecnologias disponíveis no mercado, a preparação de uma proposta de um protocolo que possibilite a verificação periódica da integridade de arquivos armazenados na nuvem, a implementação da arquitetura o desenvolvimento de implementação de referência para o protocolo proposto e a realização de testes e simulações foram os principais processos técnicos aplicados.

Inicialmente foi realizada uma fundamentação teórica, abordando o estado da arte das tecnologias, protocolos, algoritmos criptográficos e algoritmos de *hash*, passíveis de serem utilizados para verificar a integridade de arquivos. As principais características avaliadas foram a capacidade de realizar a verificação parcial de um arquivo, custo computacional/velocidade de execução, possibilidade de utilização na nuvem, capacidade de redundância, resistência a ataques e o tipo de criptografia utilizada (simétrica/assimétrica). Nesta etapa, objetivando um trabalho produtivo e dinâmico, os artigos e documentos selecionados foram analisados e registrados.

A seguir, considerando as soluções relevantes identificadas durante a fundamentação teórica e os objetivos do trabalho descritos na Seção 1.2, foi preparada a proposta do protocolo. Após a definição do mecanismo de funcionamento do protocolo, foram realizadas simulações matemáticas de forma a calibrar os valores utilizados e definir as regras de funcionamento do protocolo.

Na sequência foram as implementadas as aplicações que compõem a arquitetura contemplando os três papéis previstos no protocolo. Cada papel representa as atividades a serem executadas em separado por cada um dos entes envolvidos no processo de armazenamento e monitoramento de arquivos na nuvem. Os papéis representam o cliente, o SVI e o SADN. No início desta etapa, foram escolhidas as ferramentas a serem utilizadas

no desenvolvimento das aplicações que iriam compor a arquitetura, como Linguagem de Programação (LP) e o Sistema Gerenciador de Banco de Dados (SGBD).

Para concluir, foi montado, em laboratório, um ambiente de testes de forma que cada papel previsto no protocolo e implementado como uma das aplicações da arquitetura, fosse representado por uma máquina virtual diferente. Utilizando este ambiente foram realizados testes com arquivos de diversos tamanhos e períodos de tempo de armazenamento. Foram ainda, realizadas simulações nas quais os arquivos eram corrompidos propositalmente, através da injeção de falhas, de forma a atestar a eficiência e a eficácia da arquitetura e, por consequência, do protocolo. Nesta etapa, a cada teste/simulação foram colhidas e registradas informações como o custo computacional/tempo de execução de cada processo e o consumo de banda exigido.

Encerrando o trabalho, foram realizadas as análises e comparações entre os resultados obtidos. Com base nestes resultados foi preparada uma conclusão, na qual foram identificadas as principais contribuições geradas a partir deste trabalho.

1.4 - CONTRIBUIÇÕES DO TRABALHO

Buscam-se com este trabalho as seguintes contribuições:

- Apresentação do estado da arte da segurança e da computação em nuvem, levando em consideração aspectos que são importantes para a garantia da integridade dos arquivos;

- Apresentação dos trabalhos recentes que envolvem a integridade de arquivos armazenados na nuvem;

- Apresentação de uma proposta de um protocolo que permite o monitoramento constante da integridade de arquivos armazenados na nuvem;

- Disponibilizar a implementação de uma arquitetura computacional composta de aplicações que representam cada um dos papéis propostos no protocolo;

- Realização de testes utilizando a arquitetura implementada, simulando diversos cenários de forma a demonstrar a sua eficiência e eficácia.

1.5 - ORGANIZAÇÃO DO TRABALHO

Para um melhor entendimento deste trabalho, a sua organização é descrita a seguir.

O Capítulo 2 oferece uma revisão dos principais conceitos abordados, incluindo principalmente a computação em nuvem, confiança, segurança e criptografia. Além disso, são apresentados os trabalhos correlatos, bem como alguns dos problemas em aberto.

No Capítulo 3 é apresentado a proposta de um protocolo que possibilita o monitoramento constante da integridade de arquivos armazenados na nuvem baseado em confiança e a implementação da arquitetura composta pelas aplicações correspondentes a cada um dos papéis previstos no protocolo.

O Capítulo 4 apresenta as simulações realizadas e os resultados obtidos.

O Capítulo 5 conclui este trabalho, sinalizando algumas perspectivas possíveis, o fechamento dos resultados obtidos e os caminhos futuros que poderão ser seguidos na sequência deste trabalho.

2 - REVISÃO DA LITERATURA

Este capítulo apresenta os conceitos necessários para o entendimento do trabalho proposto. Na Seção 2.1 serão apresentados os conceitos a cerca da computação em nuvem. Na Seção 2.2 serão apresentados os conceitos relacionados a confiança. Os conceitos de Criptografia, Criptografia simétrica, Criptografia assimétrica e *Hash* serão apresentados na Seção 2.3. A Seção 2.4 apresenta os trabalhos correlatos e os problemas em abertos são apresentados na Seção 2.5.

2.1 - COMPUTAÇÃO EM NUVEM

A Computação em Nuvem (CN) é um novo paradigma para a provisão de Infraestrutura Computacional, para o qual podem ser encontradas diversas definições. Um dos motivos para essa situação deve-se ao fato da CN ser uma área relativamente nova e os autores apresentarem abordagens distintas ou diretamente relacionadas com as características de serviços oferecidos. Neste trabalho serão citadas apenas algumas destas definições, mas que contemplam as principais características encontradas nas demais definições.

De acordo com Mell e Grance (2011), a CN é um modelo que permite o acesso através da rede, de forma conveniente e sob demanda, a um conjunto compartilhado de recursos computacionais configuráveis que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação do prestador de serviços.

Já para Armbrust et al. (2009), pode-se entender a CN como um conjunto de serviços computacionais virtualizados, compartilhados por meio da Internet onde, as principais características são: agilidade, baixo custo, localização, confiabilidade, alta escalabilidade, segurança e sustentabilidade.

Segundo Vaquero et al. (2009), a CN representa um grande grupo de recursos computacionais virtualizados, acessíveis e facilmente utilizáveis, como hardware, plataformas de desenvolvimento e serviços, dinamicamente configuráveis e ajustáveis a carga de trabalho de forma a permitir a máxima utilização destes recursos, os quais são explorados economicamente através do modelo “pague pelo que usar”.

A CN agrupa diversas tecnologias como virtualização¸ computação distribuída, computação em grade, arquitetura orientada a serviços entre outros, não se tratando apenas de um paradigma computacional, mas também de um modelo de negócios. O modelo representa a convergência de duas tendências na TI, a eficiência e a agilidade (Marston et al. 2011).

A eficiência é obtida pela utilização da capacidade de processamento dos computadores modernos de forma mais eficiente através de recursos de *hardware* e *software* altamente escaláveis. Além disso, abrange ideias da computação verde, pois possibilita que os computadores possam ser fisicamente locados em regiões onde a energia é mais barata e acessadas remotamente pela internet a partir de qualquer localização geográfica.

A agilidade se dá por meio da possibilidade de rápida implantação, do processamento em paralelo, do uso de computação intensiva para análises e do uso de aplicações móveis interativas que respondem em tempo real as solicitações do usuário. No longo prazo, a agilidade também é obtida pela capacidade de fazer uso de ferramentas computacionais que podem ser implantadas e escaladas rapidamente reduzindo simultaneamente a necessidade de enormes investimentos iniciais que caracterizam as configurações de TI atuais.

Segundo Armbrust et al. (2009), a instalação e operação de *datacenters* de larguíssima escala, em locais de baixo custo operacional, foi o fator essencial para a CN. Nesses ambientes identificou-se uma redução entre cinco e sete vezes nos custos com energia, banda de rede, operações, *hardware* e *software.*

2.1.1 - Principais Características da Computação em Nuvem

Segundo Mell e Grace (2011), características como entrega de serviços sob demanda, flexibilidade de acesso, compartilhamento de recursos, elasticidade e a medição dos serviços consumidos são essenciais na CN. A Tabela 2.1 apresenta a descrição de cada uma destas características.

Tabela 2.1 - Características essenciais da computação em nuvem
(Mell, Grace 2011, adaptado)

Característica	Descrição
Entrega de serviços sob demanda	A seleção e uso de serviços deve ser realizada sob demanda e sem a interferência humana.
Flexibilidade de acesso	Os recursos devem estar disponíveis na rede e serem acessados por mecanismos padronizados disponíveis aos clientes em todas as plataformas.
Compartilhamento de recursos	Recursos como armazenamento, processamento, largura de banda e memória devem ser agrupados e compartilhadas entre diversos clientes, sendo atribuídos virtualmente a estes de acordo com a demanda.
Elasticidade	O provisionamento e a liberação de recursos devem ser realizados rapidamente, seja de forma automática conforme a demanda ou a pedido do cliente, o qual deve poder se apropriar de qualquer quantidade do recurso a qualquer momento.
Medição dos serviços consumidos	O consumo dos recursos deve ser monitorado, controlado e reportado por meio de um mecanismo de medição, de forma que haja transparência entre o provedor e o consumidor do serviço.

Segundo Armbrust et al. (2009), a CN apresenta três novos aspectos do ponto de vista do *hardware*. Os aspectos e as vantagens oriundas dos mesmos são apresentadas na Tabela 2.2.

Tabela 2.2 - Novos aspectos advindos da computação em nuvem
(Armbrust et al. 2009, adaptado)

Aspectos	Vantagens
Ilusão sobre disponibilidade de recursos computacionais infinitos disponíveis sobre demanda.	Os usuários de CN não tem a obrigação de planejar previamente a sua necessidade futura de recursos computacionais.
O usuário de CN não tem necessidade de realizar de um compromisso inicial de consumo de recursos com o provedor do serviço.	As organizações podem iniciar pequenas e ir aumentando a quantidade de recursos de *hardware* apenas quando existir um aumento de suas necessidades.
A capacidade de pagar pela utilização de recursos computacionais por um curto espaço de tempo e conforme o necessário (processamento por hora, armazenamento por dia etc.)	Economizar com a liberação de recursos como processamento e armazenamento sempre que estes não estão sendo necessários.

2.1.2 - Arquitetura da Computação em Nuvem

Muitas das tecnologias que oferecem suporte a CN como a virtualização e a computação em grade não são necessariamente novas. No entanto, a sua confluência em um ambiente onde as informações podem ser acessadas independentemente do dispositivo utilizado pelo cliente e de sua localização geográfica, representa uma grande mudança em relação a computação tradicional (Marston et al. 2011).

A infraestrutura da nuvem é uma coleção de hardware e software distribuídas em duas camadas, a física e a de abstração. A camada física consiste dos recursos de hardware que são necessários para prover os serviços de nuvem. Já a camada de abstração consiste do software implantado sobre a camada física por meio do qual a nuvem manifesta as suas características (Mell, Grance 2011).

Os recursos de hardware que compõem a camada física como servidores, *storages* e dispositivos de rede são normalmente locados em *datacenters,* os quais*,* por sua vez*,* podem estar instalados em diferentes localizações geográficas. A camada de abstração é composta pela combinação das camadas de virtualização e de gerenciamento (Zissis, Lekkas 2012).

A camada de virtualização é o elemento crítico da implementação da nuvem pois provê as características essências da CN como independência de localização, compartilhamento de recursos e a rápida provisão ou liberação destes. A camada de gerenciamento é a responsável por monitorar o tráfego e responder aos picos ou quedas no nível de utilização dos recursos com a criação de novos servidores ou a destruição daqueles que não são mais necessários.

2.1.3 - Modelos de Entrega de Serviços

Com a CN não é necessário que o cliente tenha conhecimento sobre a quantidade de servidores utilizados, as configurações do hardware e do software, como é realizado o escalonamento dos processos ou ainda, sobre qual é a localização geográfica do

datacenter. O que importa é que o serviço esteja disponível a qualquer hora, em qualquer dispositivo e em qualquer lugar.

Os serviços na nuvem são classificados conforme o seu modelo de entrega em: Software como Serviço (SaaS), Plataforma como Serviço (PaaS) e Infraestrutura como Serviço (IaaS). A Tabela 2.3 apresenta os modelos dos serviços disponibilizados na nuvem.

Tabela 2.3 - Modelos de serviços em nuvens computacionais
(Mell, Grance 2011, adaptado)

Modelo	Características
Software como Serviço (SaaS)	Nesse modelo é disponibilizado para o cliente uma aplicação, a qual, por sua vez, é executada sobre a infraestrutura da nuvem. O acesso a aplicação é realizada pelos dispositivos do cliente através dos navegadores *web* ou por uma interface própria da aplicação. O cliente não tem qualquer controle sobre os elementos que compõem a infraestrutura que da suporte a aplicação como a rede, sistema operacional, armazenamento etc., nem sobre a aplicação.
Plataforma como Serviço (PaaS)	Nesse modelo é disponibilizado para o cliente um ambiente que possibilita a instalação uma aplicação que utilize apenas recursos linguagem de programação, bibliotecas e ferramentas suportadas pelo provedor da nuvem. O cliente não tem qualquer controle sobre os elementos que compõem a infraestrutura que da suporte a aplicação como a rede, sistema operacional, armazenamento etc., no entanto, tem o controle sobre a aplicação instalada e as suas configurações.
Infraestrutura como Serviço (IaaS)	Nesse modelo é disponibilizado para o cliente uma capacidade de processamento, armazenamento, recursos de rede e outros recursos computacionais sobre o quais o cliente pode instalar e executar qualquer *software*, desde o sistema operacional até a aplicação final. O cliente não tem controle sobre a infraestrutura, mas tem controle sobre o sistema operacional, o armazenamento, as aplicações instaladas e uma limitada capacidade de seleção e controle dos componentes de rede.

2.1.4 - Modelos de Implantação

A arquitetura de uma solução em nuvem depende do modelo escolhido para sua implantação, que pode ser privada, comunitária, pública, híbrida ou federada. A escolha do

modelo baseia-se principalmente nas características da carga de trabalho computacional e não necessariamente no tamanho da organização (Kajeepeta 2010). A Tabela 2.4 descreve as características dos modelos de implantação (Mell, Grance 2011) (Buyya, Ranjan, Calheiros 2010) (Kertész et al. 2012).

Tabela 2.4 - Modelos de implantação de nuvens computacionais

Modelo	Características
Nuvens Privadas (*Private Cloud*)	A infraestrutura da nuvem é provida para uso exclusivo de uma organização, no entanto pode ser administrada, operada e até de propriedade de terceiros, podendo ainda estar locada dentro ou fora das instalações da organização.
Nuvens Comunitárias (*Community Cloud*)	A infraestrutura da nuvem é provida para uso exclusivo de um conjunto de organizações que possuem preocupações em comuns como missão, requisitos de segurança, política etc. Pode ser administrada, operada e até de propriedade de uma ou mais organizações da comunidade ou por terceiros, podendo ainda estar locada dentro ou fora das instalações destas organizações.
Nuvem Pública (*Public Cloud*)	A infraestrutura da nuvem é disponibilizada para o público em geral ou para um grande grupo de industriais, sendo de propriedade de uma organização especializada em vender os serviços da nuvem.
Nuvens Híbridas (*Hybrid Cloud*)	A infraestrutura da nuvem é a composição de duas ou mais infraestruturas de nuvens distintas, sejam elas privadas, comunitárias ou públicas. Estas estruturas permanecem como entidades independentes, mas são vinculadas entre si por intermédio de uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações.
Nuvens Federadas (*Intercloud, Cross-Cloud, Cloud-of-Clouds*)	Um conjunto de provedores de nuvens computacionais, públicos e privados, conectados por meio da internet, cujo objetivo é dar suporte aos usuários que necessitam de maior poder computacional, provendo uma única interface na qual eles podem transparentemente manipular diferentes provedores de nuvens como eles fariam com um único provedor.

2.1.5 - Ameaças

A CN é vista atualmente como uma das tecnologias mais promissoras na TI, pois devido às suas características únicas, possui em sua essência a capacidade de abordar uma série de deficiências identificadas nas arquiteturas tradicionais. Entretanto, a adoção de uma

arquitetura inovadora pode introduzir também uma série de novas ameaças (Zissis, Lekkas 2012). A Tabela 2.5 apresenta as principais ameças presentes na CN de acordo com o modelo de serviço utilizado.

Tabela 2.5 - Principais ameaças presentes na computação em nuvem
(Zissis, Lekkas 2012, adaptado)

Camada	Serviço	Ameaças
Abstração	SaaS	• Interceptação • Modificação dos dados armazenados ou em trânsito • Exclusão dos dados • Violação de privacidade • Personificação • Roubo de seção • Análise de tráfego • Exposição na rede
	PaaS	• Falhas de programação • Modificação no software • Exclusão do software • Personificação • *Deface*
	IaaS	• Roubo de seção • Análise de tráfego • Exposição na rede • Inundação de conexões • Ataques de negação de serviço distribuídos • Personificação • Interrupção das comunicações
Física	*Datacenter*	• Ataques de rede • Inundação de Conexões • Ataques de negação de serviço distribuídos • Interrupção do funcionamento do hardware • Roubo do hardware • Modificação no hardware • Mal uso da infraestrutura • Desastres naturais

2.1.6 - Armazenamento de Dados na Nuvem

O armazenamento de dados é um dos serviços mais comumente ofertados pelos provedores de serviços na nuvem. Estes serviços, com o objetivo de garantir a integridade dos dados de seus clientes, fazem uso de diferentes modelos de armazenamento de dados,

os quais utilizam abordagens específicas para persistir os dados (Wang et al. 2009). Nas seções seguintes é apresentada uma visão geral dos sistemas de arquivos *Google File System, Amazon S3, Microsoft Azure e Hadoop*.

2.1.6.1 - Google File System

O *Google File System* (GFS) é um sistema de arquivos distribuídos desenvolvido para uso em grandes *clusters* de servidores, com o objetivo de fornecer acesso a grandes quantidades de dados de forma eficiente, confiável e com alto desempenho, mesmo em máquinas não confiáveis. O GFS foi projetado e otimizado para ser executado em *datacenters* e fornecer alta vazão, baixa latência e tolerância a falhas individuais de servidores (Ghemawat, Gobioff, Leung 2003).

Para gerenciar de forma eficiente o armazenamento de arquivos que podem variar de 100 megabytes e vários gigabytes, o GFS organiza os dados em blocos denominados "*chunks*" com 64 megabytes, os quais raramente são sobrescritos ou comprimidos. Cada *chunk* recebe um identificador único de 64 bits definidos no momento de sua criação.

Como o GPS foi projetado para ser executado em *clusters* com inúmeros nós, onde a possibilidade de falha é uma constante. Um *cluster* GPS é formado por um servidor *master* e múltiplos servidores de *chunk* denominados "*chunkservers*", e é acessado por inúmeros clientes. Para contornar os possíveis problemas oriundos das falhas nos nós, cada "*chunk*" é armazenado em pelo menos 03 *chunkservers*, podendo esse número ser maior de acordo com as configurações do cliente. A arquitetura do GFS é apresentada na Figura 2.1.

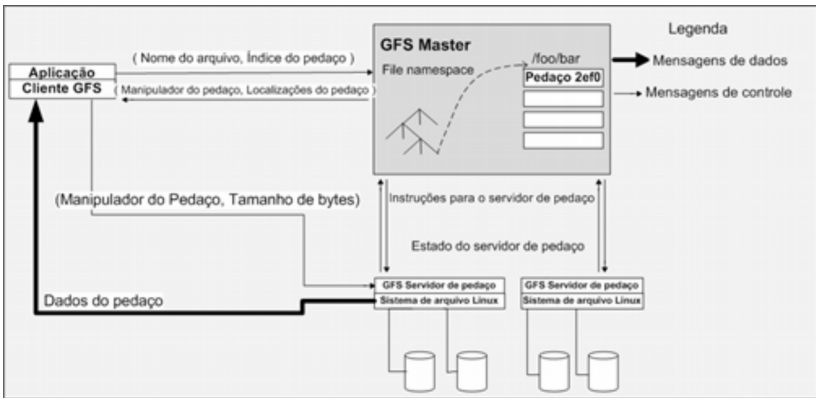


Figura 2.1 - Arquitetura do *Google File System*
(Ghemawat, Gobioff, Leung 2003, adaptado)

O servidor *master* não armazena blocos de dados (*chunks*), ele armazena apenas metadados dos arquivos como espaço de nomes, informações para controle de acesso e o mapeamento dos seus blocos, assim como a localização de cada um dos seus *chunks* armazenados nos *chunkservers*. A atualização destes metadados são realizadas pelo próprio servidor *master*, o qual periodicamente troca mensagens denominadas "*Heartbeat messages"* com cada um dos seus *chunkservers* com o objetivo de coletar informações sobre o seu estado e dar-lhe instruções.

2.1.6.2 - *Amazon Simple Storage Service*

O *Amazon Simple Storage Service* (*Amazon S3*) é um serviço de armazenamento de dados na nuvem altamente seguro, durável e escalável. O serviço é ofertado através de uma interface *web service* simples que possibilita, a partir de qualquer parte da web, armazenar e recuperar qualquer volume de dados (Amazon Web Services 2016).

O *Amazon S3* é um sistema de armazenamento distribuído desenvolvido com base no *Dynamo*. O *Dynamo* é um sistema que utiliza uma Tabela *Hash* Distribuída (DHT) para armazenar dados com base em estrutura formada por chave e valor. O particionamento e a replicação dos dados utilizando objetos versionados garante disponibilidade e escalabilidade. A técnica denominada *quorum-like* e um protocolo de sincronização de réplicas descentralizado, mantém a consistência entre as réplicas durante as atualizações (DeCandia et al. 2007).

No *Dynamo*, o armazenamento é realizado como um objeto binário identificado com uma chave única, utilizada para as operações de leitura e escrita. Características como atomicidade e isolamento são garantidos pela escrita de um único objeto, já a durabilidade é obtida por meio de escrita replicada.

O *Dynamo* provê uma interface simples que permite a adição de nós, mesmo com diferentes e características, e a distribuição das requisições, garantindo assim, alta escalabilidade e o balanceamento de carga. Para evitar a existência de pontos únicos de falhas todos os nós são simétricos em relação as suas responsabilidades.

A estratégia de virtualização utilizada pelo *Dynamo* possibilita que a capacidade de processamento e armazenamento de máquinas mais robustas sejam divididas em diversos nós virtuais distribuídos aleatoriamente. Esta estratégia contorna o problema de balanceamento de carga, gerado pela disponibilidade de um número reduzido de nós ou quando os mesmos são heterogêneos. A fragmentação e a replicação são providas pela técnica de *hashing* consistente com o uso de objetos versionados (DeCandia et al. 2007).

O conceito de objeto utilizado pelo *Amazon S3* consiste de um identificador único, seguido de dados e metadados. A organização dos objetos é realizada através do agrupamento de objetos ou espaço de nomes em *buckets*. Em caso de necessidade, a localização geográfica de um *bucket* pode ser especificada.

Para tornar o serviço tolerante a falhas, após um armazenamento inicial, os dados são propagados entre os *datacenters*. O *Amazon S3* possui ainda as seguintes características: o sistema não fornece mecanismos de bloqueio; e realiza as operações de escrita de forma atômica, podendo entretanto serem executadas sobre múltiplas chaves.

2.1.6.3 - *Microsoft Azure*

Microsoft SQL Azure é formato por um conjunto de serviços voltados para o armazenamento e o processamento de dados em nuvens computacionais. O *SQL Azure* em conjunto com o *Windows Azure Storage* compõem a solução de gerenciamento de dados em nuvem comercializada pela *Microsoft* (Microsoft 2016).

O *Windows Azure Storage* oferece armazenamento escalável, durável, com alta disponibilidade e pagamento sob demanda. Através da oferta de uma interface simples, possibilita o fácil acesso aos dados disponíveis remotamente em seus *datacenters*. Os quatro níveis de abstração oferecidos pelos serviços de armazenamento do *Windows Azure Storage* são descritos na Tabela 2.6.

Tabela 2.6 - Níveis de abstração do *Windows Azure Storage*
(*Microsoft* 2016, adaptado)

Nível de abstração	Descrição
BLOB	Fornece uma interface simples para armazenamento de grandes itens de dados. Um BLOB é um par (nome, objeto) que permite armazenar objetos com tamanho de até 50 GB.
Tabela	Fornece um conjunto de entidades, que contêm um conjunto de propriedades. Um aplicativo pode manipular as entidades e consultar qualquer uma das propriedades armazenadas em uma tabela. São diferentes das tabelas relacionais pois são compostas de entidades. Elas não são acessadas usando a linguagem SQL, mas por meio de serviços de dados.
Fila	Fornece armazenamento confiável para a entrega de mensagens, propiciando expedição assíncrona de trabalhos para habilitar a comunicação entre os serviços de diferentes partes (papéis) de sua aplicação. Sua função principal é fornecer um serviço de troca de mensagens persistentes e confiável.
Drive	Fornece volumes NTFS (Sistema de Arquivos de Nova Tecnologia) duráveis para aplicações.

Para usar o serviço de armazenamento do *Windows Azure*, o usuário precisa criar uma conta de armazenamento, para a qual receberá uma chave de 256 bits. Esta chave deverá ser utilizada toda vez que o usuário desejar enviar ou buscar os dados a partir da nuvem. Para cada requisição do usuário, será gerada uma assinatura a partir da sua chave secreta, que por sua vez será utilizada para autenticá-lo junto ao servidor.

No *Windows Azure Storage,* a verificação da integridade dos arquivos é realizada utilizando o algoritmo de *hash Message-Digest* 5 (MD5) desenvolvido pela empresa RSA *Data Security.* Por intermédio deste algoritmo, é verificado se todos os bits do arquivo obtido na origem mantiveram-se totalmente íntegros no destino (Buyya, Ranjan, Calheiros 2011).

2.1.6.4 - *Hadoop* / *Hadoop Distributed File System*

O *Hadoop* é uma implementação de um *framework* em código livre baseado no *framework* proprietário desenvolvido pelo *Google* denominado "*MapReduce",* sendo utilizado para a análise e transformação de conjuntos extremamente grandes de dados (Shvachko et al.

2010). O *Hadoop Distributed File System* (HDFS), um dos componentes do *Hadoop,* tem por finalidade armazenar grande quantidade de dados de forma distribuída em múltiplos nós. O HDFS tem como característica a alta tolerância a falhas e a capacidade de ser implementado em *hardwares* de baixo custo (Borthakur 2007).

A replicação é o mecanismo utilizado pelo HDFS para garantir a tolerância a falhas. Para isso, os dados são fracionados em blocos com 64 MB e replicados em três nós, sendo dois pertencentes ao mesmo *rack* e outro em um *rack* diferente. Dessa forma, o *framework* prioriza o tráfego de dados em nós pertencentes ao mesmo *rack*. Esta estratégia é denominada "*rack-awareness"*.

A arquitetura do HDFS é composta de um nó mestre denominado "*NameNode*" e de múltiplos nós escravos denominados "*DataNodes*". O *NameNode* é responsável pelo gerenciamento da estrutura hierárquica de diretórios, por controlar o acesso dos clientes aos arquivos e gerenciar a localização física dos diversos blocos de dados. Os *DataNodes* são os responsáveis por armazenar os blocos de dados no sistema de arquivos, atender as requisições de leitura e escrita oriundas do cliente e realizar a replicação dos blocos de dados de acordo com as instruções do *NameNode* (Borthakur 2007).

Outra característica do HDFS é a priorização da execução de aplicações o mais próximo possível da máquina de origem dos dados processados. Assim, quando os nós responsáveis pela execução da aplicação encontram-se distantes dos *DataNodes*, os dados são transferidos pela rede para um local mais próximo.

Assim como no GFS, o HDFS particiona os arquivos em grandes blocos e os armazena em nós geograficamente distribuídos, replicando-os para lidar com as possíveis falhas de *hardware*. Diferentemente de outros sistemas de arquivos distribuídos, no HDFS o armazenamento e o processamento são realizados diretamente em cada nó do sistema.

2.2 - CONFIANÇA

A confiança é reconhecida como um importante aspecto a ser considerado nas tomadas de decisões em sistemas distribuídos e auto-organizados (Marsh 1994) (Beth, Borcherding,

Klien 1994). Além disso, vem sendo estudada a sua aplicação na solução de problemas relacionados a segurança da informação nos mais diversos ambientes computacionais.

Reconhecer a confiança é uma tarefa simples devido a nossa experiência e principalmente, a nossa dependência dela para agirmos no nosso dia a dia. Entretanto, defini-la é uma tarefa bem mais complexa, pois esta se manifesta de diferentes formas. Nesse sentido, a confiança pode ser definida tanto da perspectiva humana, quanto da perspectiva computacional.

2.2.1 - Perspectiva Humana da Confiança

Da perspectiva humana, com base em sentimentos como conforto, certeza, tranquilidade e segurança, o homem diariamente toma decisões baseadas na confiança que possui sobre determinado assunto, coisa ou pessoa dentro de determinado contexto. Tais decisões têm por objetivo a satisfação de uma expectativa de obter soluções para problemas que acredita-se serem solucionáveis. Neste contexto, a partir da análise da literatura foram encontradas diversas maneiras de conceituar confiança, dos quais foram selecionados os conceitos a seguir.

Confiança é um fenômeno comum a partir do qual os seres humanos são capazes de enfrentar as complexidades do mundo e raciocinar de forma sensata sobre as possibilidades da vida cotidiana. A confiança está fortemente ligada na fé em alguma coisa, o que implica em certo grau de incerteza e otimismo. O comportamento de um terceiro no decorrer do tempo é o que determina como agir em uma determinada situação (Marsh 1994).

A confiança pode ser descrita também como a probabilidade pela qual um indivíduo “A”, espera que outro indivíduo “B”, execute uma determinada ação da qual o bem-estar de A dependa. Esta definição inclui o conceito de dependência na parte confiada e de credibilidade da mesma, do ponto de vista da parte confiante (Jøsang, Ismail, Boyd 2007).

Segundo Gambeta (2000), sem confiança em outros seres humanos e/ou organizações, não existe cooperação e por consequência não haveria sociedade. A confiança pode ainda ser

definida como o conceito social mais importante que auxilia os seres humanos a cooperar em seu ambiente social e está presente em todas as iterações humanas.

2.2.2 - Perspectiva Computacional da Confiança

Considerando a perspectiva computacional, a confiança é um nível de probabilidade subjetiva, na qual um agente acredita que outro realizará uma determinada ação, em um contexto em que ela afeta a sua própria ação (Gambetta 2000). Assim, se um agente é confiável, significa que há uma probabilidade alta o suficiente de que este executará uma ação considerada benéfica a tal ponto que seja considerada a cooperação com o mesmo. Da mesma forma, a desconfiança ocorre quando acredita-se que há uma probabilidade baixa o suficiente para que a cooperação seja evitada.

Segundo Marsh (1994), a confiança pode ser implementada em um sistema computacional, pois seu escopo pode ser representado matematicamente. O conceito de confiança varia do valor -1, que representa a desconfiança total, até o valor 1, que representa a confiança total. Desse modo, o fato da existência da possibilidade de desconfiança é o que da relevância a confiança.

2.2.3 - Características da Confiança

A visão da confiança é modular, pois o seu cálculo deve considerar, além das observações momentâneas do agente, as situações passadas, obtidas em um contexto social através de organizações e relacionamentos (Patel 2007). Baseando-se em um modelo, o cálculo da confiança pode ser realizado a partir do resultado da coleta de opiniões sobre o comportamento de uma determinada entidade.

Segundo Abdul-Rahman e Hailes (1997), a confiança possui as seguintes características: a subjetividade, a probabilidade esperada e a relevância. A subjetividade está relacionada ao fato de que sempre há fatores ocultos (intencionalmente ou inconscientemente) por trás de uma decisão sobre confiar ou não em determinada entidade. A probabilidade esperada está relacionada ao grau de certeza sobre a ocorrência de determinado evento,

independentemente da sua capacidade de monitorá-lo. Já a relevância está relacionada ao nível em que as ações de um agente podem influenciar nas suas próprias ações.

Já Albuquerque (2008), define que a confiança possui um conjunto de características básicas. Estas características são apresentadas na Tabela 2.7.

Tabela 2.7 - Características básicas da confiança
(Albuquerque 2008)

Características	Exemplo
A confiança é relativa a um determinado contexto ou situação.	**A** pode confiar em **B** para lhe oferecer uma carona. Entretanto, **A** não confia em **B** o suficiente para dirigir seu carro.
A confiança tem um aspecto direcional.	**A** pode confiar em **B**, entretanto **B** pode não confiar em **A**.
A confiança é passível de ser mensurada.	**A** pode confiar mais em **B** do que confia em **C**.
A confiança possui aspecto temporal e evolutivo.	A confiança que **A** tem em **B** pode aumentar ou diminuir conforme **A** e **B** interagem.
A confiança pode ser influenciada por uma recomendação.	**A** que já confia em **B**, passa a ter certa confiança em **C**, que por sua vez lhe foi apresentado por **B**.
A confiança não é transitiva.	Se **A** confia em **B** e **B** confia em **A** e **C**, não significa dizer que **A** confia em **C**.

Segundo Albuquerque (2008), em determinados casos pode ocorrer a transitividade da confiança, ou seja, o compartilhamento entre entidades como A e B de algum indicativo ou de um valor de confiança, de forma que A passe a acreditar na entidade C em relação a um determinado propósito por meio de uma confiança derivada. No entanto, para que a transitividade da confiança seja válida, existem algumas restrições semânticas que devem ser consideradas.

Segundo Abdul-Rahman e Hailes (1997), as condições que habilitam a transitividade são:

- a entidade B comunicar sua confiança na entidade C para a entidade A como uma "recomendação";

- a entidade A confiar na entidade B como um recomendador;

- a entidade A estar habilitada a fazer julgamentos sobre a "qualidade" das recomendações da entidade B, conforme as políticas de A.

- a confiança não ser absoluta, ou seja, a entidade A pode confiar menos na entidade C do que a entidade B confia, baseada na recomendação de B.

Segundo Jøsang e Pope (2005), para a validade da transitividade é necessário que o propósito da confiança seja correspondente e semanticamente consistente ao longo de todo o caminho até chegar a entidade que fará uso da confiança derivada. É necessário ainda, que somente as experiências observadas pela própria entidade, sejam utilizadas para gerar a informação sobre confiança que será compartilhada com as demais entidades.

2.3 - CRIPTOGRAFIA

A criptografia é o estudo de técnicas matemáticas utilizadas para transformar informações legíveis em dados ilegíveis. A criptografia permite o armazenamento ou transmissão de informações sensíveis através de redes inseguras como a internet, de forma que não possa ser lida por qualquer um, exceto pelo próprio destinatário. Enquanto que a criptografia é a ciência que estuda como garantir a segurança dos dados, a criptoanalise é a ciência que estuda como quebrar a essa segurança (Mao, W. 2003).

No contexto da criptografia, o conteúdo de uma mensagem que pode ser lida e entendida sem a utilização nenhum recurso especial denomina-se "texto claro". O método que transforma o texto claro de forma a ocultar o seu conteúdo denomina-se "cifragem". O conteúdo não legível obtido pela transformação do "texto claro" realizado pelo processo de cifragem denomina-se "texto cifrado". O processo de transformação do texto cifrado em texto claro denomina-se "decifragem".

A criptografia pode ser considerada forte ou fraca, e a medida de sua força é obtida de acordo com o tempo e os recursos necessários para recuperar o texto claro de acordo com o poder computacional atualmente disponível. No entanto, não há como provar que uma criptografia considerada forte hoje resistirá ao crescimento do poder computacional no futuro (Network Associates 2000).

Criptografia não é um meio único para prover a segurança da informação, mas sim um conjunto de técnicas destinadas a este fim (Menezes, Oorschot, Vanstone 1996). Os principais objetivos da criptografia são apresentados na Tabela 2.8.

Tabela 2.8 - Objetivos da criptografia
(Menezes, Oorschot, Vanstone 1996, adaptado)

Objetivos	Descrição
Confidencialidade	Impedir o acesso ao conteúdo dos dados exceto para para aqueles que são autorizados a acessá-lo.
Integridade	Impedir ou possibilitar a identificação de alterações não autorizadas aos dados.
Autenticação	Identificar tudo o que envolve o processo de comunicação, ou seja, confirmar a identidade das entidades envolvidas, assim como a origem dos dados.
Não-repúdio	Prevenir que uma entidade possa negar ações realizadas ou compromissos acordados anteriormente.

Os processos de cifragem e decifragem são realizados por intermédio de algoritmos criptográficos baseados em funções matemáticas. O algoritmo criptográfico trabalha em conjunto com uma chave, que pode ser uma palavra, um número ou até uma frase. O mesmo texto claro ao ser cifrado usando o mesmo algoritmo criptográfico mas utilizando chaves distintas, gera de diferentes textos cifrados (Network Associates 2000).

Diferentemente do ocorria no passado onde a criptografia era utilizada basicamente a nível governamental e o desconhecimento do algoritmo criptográfico fazia parte do mecanismo de segurança do processo, atualmente a segurança do texto cifrado é totalmente dependente da força do algoritmo e da manutenção em segredo da chave utilizada.

Os algoritmos criptográficos podem ser divididos em dois grandes grupos, os algoritmos simétricos e algoritmos assimétricos. Nos algoritmos simétricos, uma única chave é usada para cifrar e decifrar a mensagem, enquanto que nos algoritmos assimétricos, a mensagem é cifrada com uma chave, denominada “chave pública” e decifrada por outra, denominada “chave privada” (Menezes, Oorschot, Vanstone 1996).

2.3.1 - Criptografia Simétrica

O uso da criptografia simétrica ou convencional tem como principal benefício a altíssima velocidade quando comparada a criptografia assimétrica. No entanto, não é indicada seu uso para transmissão segura de dados devido à dificuldade de se realizar o compartilhamento da chave sem que alguém possa interceptá-la (Network Associates 2000).

Os algoritmos simétricos podem ser divididos em "algoritmos simétricos de fluxo" e "algoritmos simétricos de bloco". Os algoritmos de fluxo cifram os bits da mensagem um a um, enquanto que os algoritmos de bloco pegam um determinado número de bits e os cifram como uma única unidade (Stallings 2005).

Dentre os algoritmos criptográficos de bloco, o *Data Encryption Standard* (DES), desenvolvido pela IBM, foi um dos primeiros padrões criados e por consequência, um dos mais conhecidos. Atualmente o DES é considerado inseguro, porém outros algoritmos mais seguros e modernos como o *Triple Data Encryption Standard* (3DES) e o *Advanced Encryption Standard* (AES) o substituíram.

2.3.2 - Criptografia Assimétrica

Com a descoberta dos conceitos da criptografia assimétrica também conhecida como criptografia de chave pública na década de 70, permitiu-se que cada usuário pudesse ter duas chaves matematicamente relacionadas, uma denominada "privada" e que deveria ser mantida e segredo e outra, denominada "pública" que poderia ser livremente publicada. Esta descoberta resolveu o problema relacionado ao compartilhamento da chave existente na criptografia simétrica (Network Associates 2000).

Considerando o poder computacional atual, é computacionalmente inviável deduzir a chave privada a partir da chave pública, de forma que qualquer um pode criptografar um texto claro usando uma chave pública, no entanto, somente o detentor da chave privada correspondente poderá decifrá-lo (Stallings 2005).

O principal óbice em relação a criptografia assimétrica é a complexidade empregada no desenvolvimento dos algoritmos e o alto consumo de processamento computacional necessário para sua execução, fato este que os torna lentos quando comparados aos algoritmos simétricos. Para inviabilizar a realização de ataques de força bruta é necessário que as chaves (pública e privada) utilizadas sejam bastante grandes, o que faz com que os algoritmos se tornem ainda menos eficientes. A Tabela 2.9 apresenta os principais algoritmos assimétricos.

Tabela 2.9 - Principais algoritmos assimétricos
(Stallings 2005, adaptado)

Algoritmo	Descrição
RSA	Baseado na facilidade de se multiplicar dois números primos e na dificuldade de se recuperar os dois primos originais a partir resultado. Segurança baseada na dificuldade de fatorar números grandes. O primeiro algoritmo a possibilitar criptografia e assinatura digital. Amplamente utilizado.
Diffie-Hellman	Segurança é oriunda da dificuldade de calcular logaritmos discretos. Usado para possibilitar que entidades A e B possam compartilhar (gerar um acordo sobre) um segredo através de um canal público.
ElGammal	O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Segurança é oriunda da dificuldade de calcular logaritmos discretos de corpos finitos. Usado em padrões de assinatura digital e segurança de e-mail.
Curvas Elípticas	Sistemas criptográficos de curvas elípticas consistem em adaptar outros sistemas para usar o domínio das curvas elípticas, em vez do domínio dos corpos finitos. Provê segurança similar ao RSA, mas com chaves menores. Usado para troca de chaves e assinatura digital.

2.3.3 - *Hash*

O *hash* é o resultado da aplicação de um algoritmo criptográfico de dispersão, que toma como fonte de dados uma cadeia de caracteres (ou conjunto de bytes) de qualquer tamanho e retorna como resultado uma sequência de bits de comprimento fixo e muito pequena. Qualquer mudança na sequência de origem, por mais simples que seja, altera drasticamente o *hash* resultante (Bose 2008).

Os algoritmos criptográficos de *hash* são componentes muito importantes nas aplicações destinadas a segurança da informação como a geração e a verificação de assinaturas digitais, a derivação de chaves e a geração pseudorrandômica de bits (NIST FIPS PUB 202).

A segurança do algoritmo criptográfico de *hash* está diretamente relacionado a sua capacidade de evitar colisões intencionais, já que são projetados para que seja praticamente "impossível" de deduzir, a partir de um *hash* resultante, a mensagem que lhe deu origem ou mesmo qualquer outro conjunto de bytes que representem uma colisão. As colisões ocorrem quando ao aplicar um algoritmo criptográfico de *hash* sobre duas mensagens de conteúdos diferentes, obtenha-se o mesmo *hash* como resultado, situação que é matematicamente impossível de ser evitada, tendo em vista que o domínio de entrada normalmente é maior que o contradomínio de saída.

Ao longo dos anos muitos algoritmos criptográficos foram desenvolvidos, dos quais pode-se destacar o MD5 e os algoritmos da família SHA, devido ampla utilização dos mesmos nos mais diversos softwares voltados para a segurança da informação. O MD5, a família SHA e o Blake2, este último por ter sido escolhido como o algoritmo de *hash* a ser utilizado nas soluções propostas neste trabalho, serão rapidamente descritos na Seção 2.3.3.1.

2.3.3.1 - Algoritmos Criptográficos de *Hash*

O MD5 é um algoritmo criptográfico que recebe como entrada uma mensagem de tamanho aleatório e produz como saída um *hash* com tamanho igual a 128 bits. A função foi desenvolvida originalmente para ser utilizada em aplicações de assinatura digital. O MD5 foi projetado para ser muito rápido em máquinas com arquitetura 32bits e devido ao seu algoritmo não exigir grandes tabelas de substituição pode ser codificada de forma bastante compacta (Rivest 1992).

A família de algoritmos criptográficos denominados "*Secure Hash Algorithms"* (SHA) foram definidos pelo padrão *Secure Hash Standard* (SHS), o qual especifica os algoritmos SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 e SHA-512/256 (NIST FIPS PUB 180-4). A principal diferença entre estes algoritmos é o nível de segurança

provido pelos mesmos, além disso existem algumas diferenças em suas propriedades em termos de tamanho dos blocos, do tamanho das palavras utilizadas no processo geração do *hash* e do tamanho do *hash* gerado. A Tabela 2.10 apresenta as propriedades básicas desses algoritmos.

Tabela 2.10 - Propriedades básicas do algoritmos de *hash* da família SHA (NIST FIPS PUB 180.4, adaptado)

Algoritmo	**Tamanho do bloco**	**Tamanho da palavra**	**Tamanho do *hash***
SHA-1	512	32	160
SHA-224	512	32	224
SHA-256	512	32	256
SHA-384	1024	64	384
SHA-512	1024	64	512
SHA-512/224	1024	64	224
SHA-512/256	1024	64	256

O algoritmo SHA-3 foi escolhido pelo *National Institute of Standards and Technology* (NISTI) através de uma competição pública que teve por objetivo substituir os algoritmos anteriores da família SHA. Todas as funções disponíveis no SHA-3 são baseadas no algoritmo denominado "*Keccak*", vencedor da competição (NIST FIPS PUB 202).

A família SHA-3 consiste de 4 funções criptográficas de *hash* denominadas SHA3-224, SHA3-256, SHA3-384, e SHA3-512, e duas funções para as quais o tamanho do *hash* resultante pode ter seu tamanho estendido, a SHAKE128 e a SHAKE256. Os valores 128 e 256 indicam a equivalência em relação ao nível de segurança que o algoritmo da suporte. As funções SHA-3 podem ser implementadas em *software*, *firmware*, *hardware* ou qualquer combinação destes.

O algoritmo *Blake2* é uma versão melhorada do algoritmo criptográfico de *hash* denominado "*Blake*", finalista do concurso para seleção do SHA-3, otimizado para aplicação em software. O *Blake2* possui duas versões de acordo com a plataforma para a qual foi otimizada, o *Blake2b* para plataformas de 64 bits que produz *hashes* com tamanho entre 8 e 512 bits e Blake2s para plataformas 32 bits que produz *hashes* com tamanho entre 8 e 256 bits (Aumasson et al. 2013).

As principais melhorias apresentadas na função Blake2 em relação a função que lhe deu origem, a função *Blake*, são: redução do consumo de memória em 32%; apresentar velocidade de processamento superior ao MD5 em plataformas 64 bits; inserção de bytes adicionais no último bloco de dados (*padding*) somente quando necessário; suporte direto a paralelismo sem *overhead* e geração de *hash* muito mais rápida em processadores *multicore*. A Figura 2.2 apresenta um gráfico comparativo entre as performances das principais funções de *hash* em um processador *Intel Sandy Bridge* 3.1 GHz.

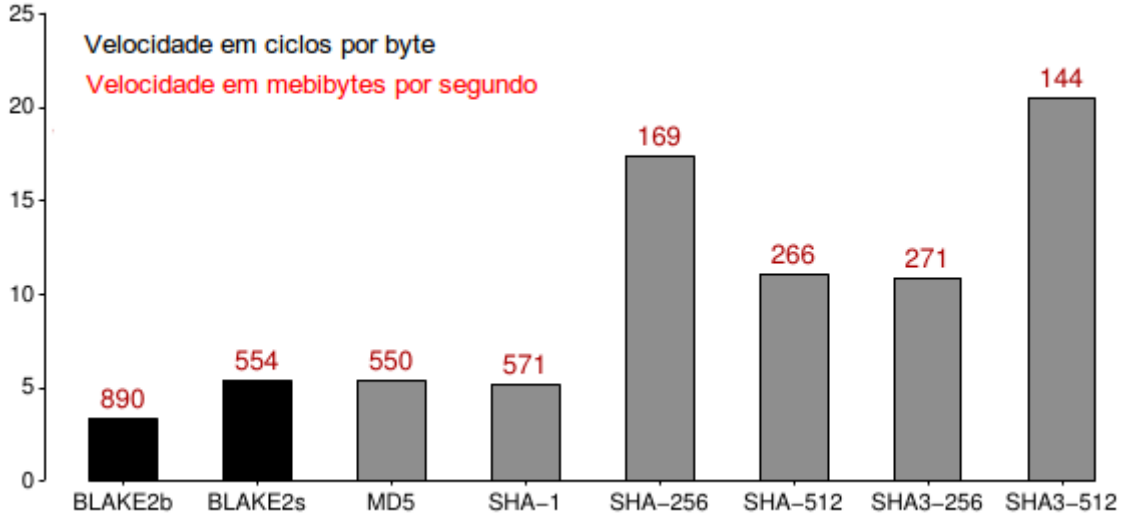


Figura 2.2 - Comparativo de performance entre as principais funções de *hash*. (Aumasson et al. 2013, adaptado)

Em relação a segurança, assim como o SHA-3, o Blake2 não possui nenhuma vulnerabilidade conhecida. Já as funções SHA-1, MD5, SHA-256 e SHA-512 são vulneráveis a ataques de extensão de comprimento, por meio da qual é possível adicionar informação extra ao *hash* sem o conhecimento da chave. São vulneráveis ainda, o SHA-1 ao ataque de colisão e o MD5 aos ataques de colisão e de colisão com prefixo escolhido (Aumasson et al. 2013).

Neste trabalho será utilizado o algoritmo Blake2 pelo fato de o mesmo não possuir vulnerabilidades conhecidas e ainda, de acordo com literatura, apresentar uma performance superior aos demais algoritmos.

2.4 - TRABALHOS CORRELATOS

Esta Seção apresenta os trabalhos estudados e revisados, relacionados a aplicação de confiança computacional, a garantia da privacidade e a verificação da integridade de dados voltados ou aplicáveis a ambientes de computação em nuvem.

2.4.1 - Aplicação da Confiança Computacional

A confiança, dependendo da abordagem utilizada, pode ser mensurada pelo resultado direto das próprias experiências ou através do uso de recomendações de terceiros. Em Tahta, Sen e Can (2015), foi proposto um modelo de confiança para sistemas *Peer-to-Peer* (P2P), que faz uso de algoritmos genéticos para reconhecer diversos tipos de ataques e ajudar um nó com bom comportamento a encontrar outros nós confiáveis. Em sistemas P2P mensurar a confiança é um problema desafiador, principalmente porque na maioria das vezes cada nó precisa interagir com outros nós totalmente desconhecidos.

O modelo de confiança proposto, denominado GenTrust, utiliza características extraídas (número de interações, número de interações com sucesso, média de tamanho dos arquivos baixados, média de tempo entre duas interações etc.) das interações do próprio nó ou, quando não possui informações suficientes, utiliza recomendações recebidas de outros nós. O algoritmo genético seleciona quais características quando avaliadas em conjunto e em determinado contexto, apresentam o melhor resultado na identificação dos nós mais confiáveis.

Outra abordagem é apresentada por Gholami e Arani (2015), que propõe um modelo de confiança voltado para ajudar clientes a encontrar serviços na nuvem que possam atendê-los com base em requisitos de qualidade de serviço. O modelo denominado *Turnaround_Trust l*eva em consideração critérios de qualidade de serviço como custo, tempo de resposta, largura de banda e velocidade do processador para selecionar a fonte do serviço mais confiável dentre as disponíveis na nuvem, tendo como critério de maior peso a velocidade.

Em Canedo (2012), o modelo de confiança proposto baseia-se em conceitos como confiança direta, recomendação de confiança, confiança indireta, confiança situacional e reputação para permitir a seleção de nós para a troca confiável de arquivos em uma nuvem privada. Para o cálculo da confiança são adotadas como métricas a capacidade de processamento do nó, a sua capacidade de armazenamento, o sistema operacional adotado e a capacidade do enlace. Já para o cálculo da reputação são analisadas as experiências satisfatórias e não satisfatórias realizadas com este nó pelos demais.

O modelo proposto calcula a confiança e a reputação do nó analisado com base em informações previamente colhidas, informações solicitadas a outros nós da rede ou informações solicitadas diretamente ao referido nó. A escolha do nó mais confiável é efetuada levando em consideração a sua disponibilidade.

2.4.2 - Verificação da Integridade e Garantia da Privacidade

Apesar do sucesso inicial, da popularidade conquistada pelo paradigma da computação em nuvem e da grande disponibilidade de provedores e ferramentas, existe um número significativo de desafios e riscos que são inerentes a esse novo modelo de computação, que devem ser considerados pelos provedores, desenvolvedores e usuários finais. Dentre os problemas a serem enfrentados, pode-se destacar a privacidade do usuário, a segurança dos dados, o bloqueio de dados, a disponibilidade do serviço, a recuperação de desastres, a performance, a escalabilidade, a eficiência energética e a programabilidade (Buyya, Broberg, Goscinski 2011).

Outro problema a ser a considerado é a questão legal. Os dados quando armazenados em uma nuvem podem ser locados em qualquer lugar do planeta conforme escolha ou necessidade do provedor do serviço. Assim, a localização física do *datacenter* utilizado determinará o conjunto de leis que podem ser aplicadas aos dados por eles armazenados. Essa situação poderia permitir que um governo obtenha acesso, por intermédio de medidas judiciais, a dados privados hospedados em seu território.

Outra grande preocupação é a verificação da integridade dos dados armazenados na nuvem sobre a administração de terceiros não confiáveis. Nesses casos, poderiam ocorrer falhas na infraestrutura de armazenamento, corrompendo os dados armazenados e o provedor do

serviço ocultar essa informação do proprietário dos dados. Outra situação, ainda mais grave, com o objetivo de economizar recursos financeiros e liberar espaço de armazenamento, o provedor poderia deliberadamente excluir arquivos raramente acessados (Wang et al. 2009).

A necessidade de garantir a integridade dos dados armazenados na nuvem foi objeto de estudo em diversos trabalhos de pesquisa. Após o levantamento e análise dos resultados obtidos em algumas dessas pesquisas, foram extraídas as ideias que contribuíram para o desenvolvimento da proposta de solução para o problema da verificação da integridade apresentado nesse trabalho. Os trabalhos de pesquisa analisados são descritos a seguir.

Em George e Sabitha (2013), é apresentado uma proposta para melhorar a privacidade e a integridade de dados armazenados em bancos de dados hospedados na nuvem. A solução foi projetada para ser usada nas tabelas e foi dividida em duas partes. A primeira parte, denominada "*Anonymization*", é um processo usado para localizar os campos da tabela que poderiam ser utilizados para identificar os proprietários dos respectivos dados. A *Anonymization* usa técnicas de como generalização, supressão, ofuscação e adição de registros anônimos para aumentar a privacidade dos dados. A segunda parte, denominada "*Integrity Checking*", utiliza técnicas de criptografia de chave pública e privada para gerar uma etiqueta para cada registro na tabela. Ambas as partes são executadas com a ajuda de um terceiro confiável denominado "*enclave*", que armazena todos os dados gerados necessários para posteriormente desfazer o processo de anonimização e realizar a verificação da integridade.

Outro método para verificação da integridade de dados cifrados é proposto em Kavuri, Kancherla e Bobba (2014). O método utiliza um novo algoritmo de *hash* denominado "*Dynamic User Policy Based Hash Algorithm*". Os *hashes* sobre os dados são calculados para cada usuário autorizado. Os dados cifrados e o seu *hash* são salvos separadamente no serviço de armazenamento na nuvem. A integridade dos dados pode ser verificada apenas por um usuário autorizado, sendo que para isso, é necessário recuperar todo o dado cifrado e o seu *hash*.

Outra proposta para obter simultaneamente a verificação da integridade dos dados armazenados na nuvem e ainda preservar a privacidade dos mesmos foi apresentada por

Al-Jaberi e Zainal (2014). Para isso, foi proposto o uso de dois algoritmos criptográficos para cada transação de *upload* ou *download* de dados. O algoritmo *Advanced Encryption Standard* (AES) é usado para cifrar dados do cliente, que serão salvos no serviço de armazenamento na nuvem, e uma técnica de criptografia homomórfica parcial, baseada no algoritmo RSA, é usada para cifrar as chaves criptográficas AES que serão salvas em um terceiro confiável, juntamente com um *hash* do arquivo. A integridade dos dados é verificada apenas quando o cliente realizar o download completo do seu arquivo.

Um protocolo de auditoria da integridade de dados proposto por Kay et al. (2013) torna possível a identificação rápida de dados corrompidos usando a verificação de textos homomorficamente cifrados. Devido a metodologia adotada tanto o tempo total de auditoria como o custo de comunicação poderiam ser reduzidos. A verificação da integridade dos dados é executada periodicamente por um terceiro, confiável ou não denominado "*Third Party Auditor*" (TPA).

Em Wang et al. (2009), é proposto um modelo que permite de forma segura a verificação pública da integridade de dados armazenados na nuvem. Além disso, a proposta também prevê a possibilidade de manter o conteúdo armazenado dinâmico, ou seja, os dados armazenados podem sofrer operações como alteração, inclusão e exclusão. O modelo garante que os blocos de dados dos arquivos que são verificados por intermédio de desafios não necessitam ser recuperados pelo verificador e que nenhuma informação de estado deverá ser armazenada no verificador entre as auditorias. A *Merkle Hash Tree* (MHT) é empregada para salvar os *hashes* dos valores autênticos dos dados, sendo que, tanto os valores quanto as posições dos blocos de dados são autenticados pelo verificador.

2.5 - PROBLEMAS EM ABERTO

Embora as pesquisas descritas na Seção 2.4 apresentem alternativas viáveis que respondem ao problema da verificação da integridade dos arquivos armazenados na nuvem, não há em nenhum dos trabalhos avaliados, uma solução única e completa que possibilite:

- que a integridade dos arquivos armazenados na nuvem sejam permanentemente monitorados;

- que o processo de monitoramento possa ser realizado por terceiros;

- que o mecanismo utilizado para monitoramento não exija o acesso, mesmo que em parte, ao conteúdo do arquivo armazenado;

- que o custo computacional para realização do monitoramento seja baixo e previsível;

- que o consumo de recursos computacionais do processo de monitoramento seja balanceado conforme a qualidade do serviço prestado pelo provedor da nuvem.

2.6 - SÍNTESE DO CAPÍTULO

Neste capítulo foi realizada uma revisão dos principais conceitos de computação em nuvem, confiança e criptografia. Além disso, foi realizado um mapeamento das principais questões que envolvem o paradigma da computação em nuvem e uma análise das razões que levam a necessidade de manter um monitoramento contínuo da integridade dos arquivos armazenados na nuvem. Na sequência foram apresentados os trabalhos correlatos, onde foram citados os autores e as principais características das soluções propostas. Por fim, foram apresentados alguns problemas em aberto.

3 - PROPOSTA DA ARQUITETURA

Este capítulo apresenta a proposta de uma arquitetura e de um protocolo. A arquitetura define os papéis que operam em conjunto para possibilitar o monitoramento periódico de arquivos armazenados na nuvem. Já o protocolo descreve passo a passo as ações a serem implementadas por cada papel definido na arquitetura para a execução do processo de monitoramento.

3.1 - ARQUITETURA PROPOSTA

Este trabalho propõe uma arquitetura formada por três papéis: i) Cliente; ii) Serviço de Armazenamento de Dados na Nuvem (SADN) e, iii) Serviço de Verificação de Integridade (SVI). O Cliente representa o proprietário do arquivo a ser armazenado pelo provedor da nuvem e tem como responsabilidade a geração das informações necessárias para o seu armazenamento e monitoramento. O SADN representa o responsável por receber e armazenar o arquivo do cliente, além de receber e responder a desafios que atestem a integridade dos arquivos, durante o período em que estes permaneçam sob a responsabilidade do provedor. O SVI representa o elo entre o Cliente e o SADN, sendo o responsável por manter as informações sobre o arquivo e realizar o monitoramento constante da integridade do mesmo através da submissão de desafios ao SADN e validação de suas respostas.

3.2 - PROTOCOLO PROPOSTO

O protocolo define os processos para o monitoramento de arquivos armazenados na nuvem por meio de ações agrupadas em etapas a serem executadas de forma integrada por cada um dos papéis previstos na arquitetura. As ações do protocolo iniciam no Cliente, com a cifragem do arquivo original e o seu envio para o SADN, seguido de sua divisão em 4096 frações, as quais são agrupadas de forma aleatória, formando blocos de dados com 16 frações cada. Para cada bloco de dados é gerado um *hash* que, agrupado com os endereços de suas frações, formam os dados necessários para o monitoramento do arquivo. Estes dados são enviados ao SVI para serem armazenados em sua base de dados. Em seguida o SVI utiliza cada conjunto de endereços de frações para enviar um único desafio ao SADN.

Ao receber o desafio, o SADN efetua a leitura das frações no arquivo, monta o bloco de dados, gera o seu *hash* e o envia ao SVI como resposta ao desafio. A integridade do bloco de dados verificada é confirmada pelo SVI quando os *hashes* original e de resposta forem iguais.

O fluxo de funcionamento do protocolo é composto de dois processos principais. O primeiro, denominado Processo de Armazenamento do Arquivo (PAA), é executado sob demanda e tem como ponto de partida o Cliente. O segundo, denominado Processo de Verificação (PV), é instanciado pelo SVI e executado de forma contínua por meio de instâncias paralelas para cada SADN monitorado.

O PAA é o processo responsável pela preparação do arquivo a ser armazenado (cifragem e cálculo de um *hash* de identificação), pela geração das informações necessárias para o seu monitoramento (fracionamento do arquivo, montagem de blocos a partir das frações, geração do *hash* de cada bloco); pelo envio do arquivo para um ou mais SADNs; e pelo envio das informações necessárias ao monitoramento para o SVI.

O PV é o processo responsável por gerar desafios para cada arquivo, submetê-lo ao SADN; monitorar e receber as respostas de cada desafio; validar as respostas; identificar o arquivo como corrompido quando for o caso; atualizar o nível de confiança atribuído pelo SVI ao SADN; e informar ao Cliente quando identificar alguma falha. A arquitetura do protocolo proposto é apresentada na Figura 3.1.

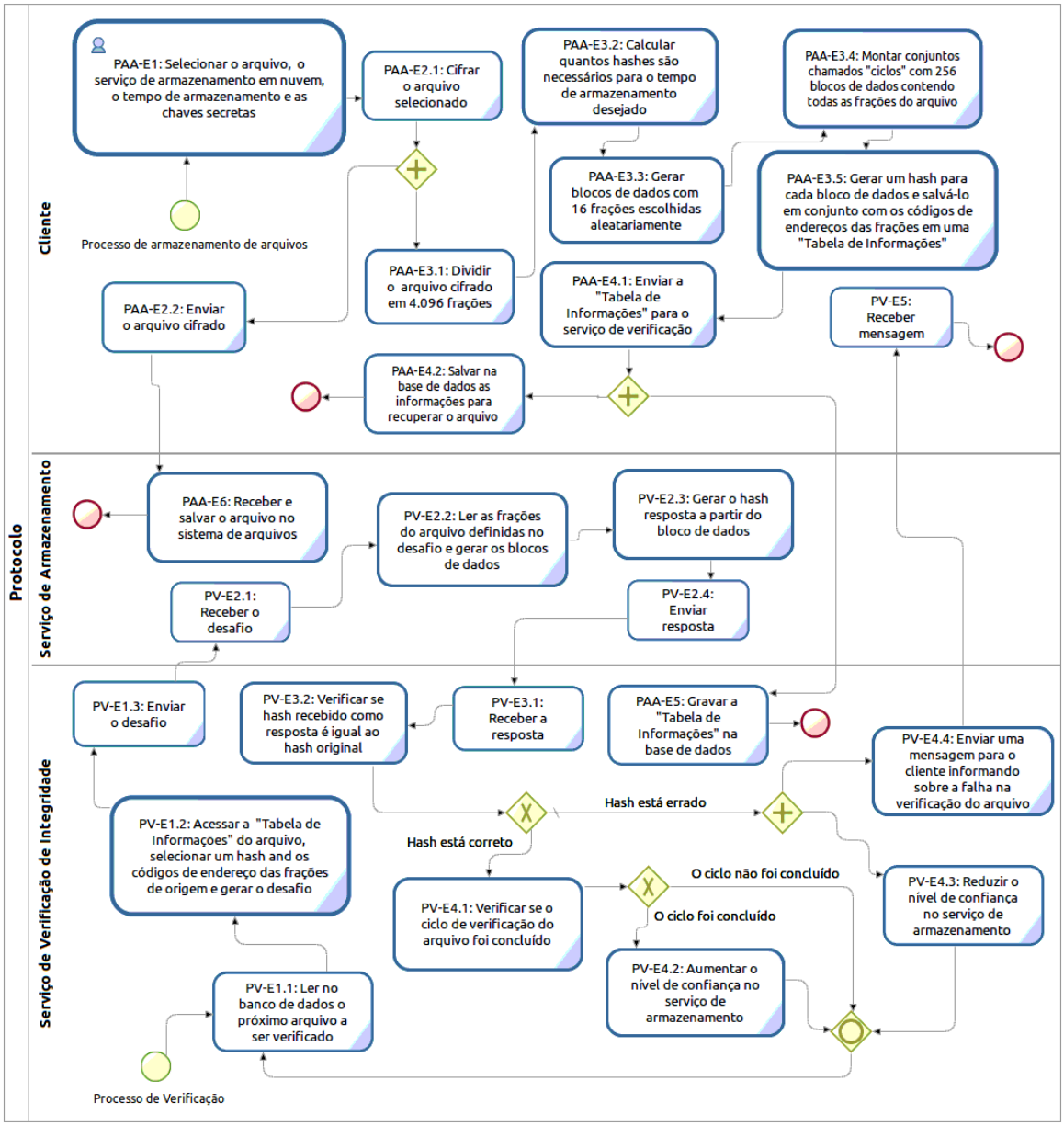


Figura 3.1 - Protocolo proposto.

Nas Seções 3.2.1, 3.2.2 e 3.2.3 será apresentado o detalhamento dos papéis do Cliente, do SVI e do SADN.

3.2.1 - Cliente

O papel do Cliente concentra as ações previstas no protocolo para serem executadas pelo proprietário do arquivo a ser armazenado no provedor de nuvem. Estas ações foram agrupadas em estágios nomeados de acordo com o processo do qual fazem parte. Os

estágios nomeados com o prefixo “PAA-E” compõem as ações oriundas do PAA e, da mesma forma, aqueles com o prefixo “PV-E” compõem as ações oriundas do PV.

Nas Seções 3.2.1.1, 3.2.1.2, 3.2.1.3 e 3.2.1.4 serão descritas as ações de cada estágio de funcionamento do protocolo apresentado na Figura 3.1.

3.2.1.1 - Seleções Iniciais

Este é o primeiro estágio do PAA, denominado PAA-E1. O estágio consiste das seguintes seleções: i) do arquivo a ser armazenado no provedor da nuvem; ii) do tempo de armazenamento (número de anos); iii) de um SVI; iv) de um ou mais SADN; v) de uma senha a ser utilizada para a cifragem do arquivo; e vi) de um valor numérico denominado "semente".

Para cada execução do PAA deverá ser selecionado um único arquivo que poderá ter qualquer tamanho, dependendo exclusivamente das limitações impostas pelos SADNs escolhidos. O tempo de armazenamento deverá ser informado em número de anos, que somado a data de execução do PAA, determinará a data limite de armazenamento do arquivo. O número de anos informado também será utilizado para determinar o número de desafios necessários para monitorar o arquivo durante o período de armazenamento sem repetir o desafio.

O SVI deve ser selecionado entre aqueles previamente contratados pelo proprietário do arquivo. O contrato com o SVI deverá definir quais SADNs este tem autorização para monitorar. A seleção de um ou mais SADNs será realizada entre aqueles previamente autorizados junto ao SVI selecionado.

A possibilidade de seleção de mais de um SADN leva em consideração a necessidade de prover redundância ao arquivo armazenado, o que é importante, pois um dos objetivos específicos deste trabalho é permitir que o cliente não mantenha cópias do arquivo original. Este procedimento visa garantir a recuperabilidade do arquivo armazenado, mesmo que algum dos SADNs selecionados venha a corrompê-lo.

As chaves secretas a serem informadas são compostas por uma senha e por uma semente. A senha deve ser composta por 32 caracteres, preferencialmente alternando letras, números e carácteres especiais. A senha é utilizada para cifrar o arquivo e a confidencialidade do conteúdo do mesmo depende diretamente da dificuldade de descobrir a senha por meio de ataques de força bruta e da qualidade do algoritmo criptográfico utilizado na implementação do protocolo.

A semente é um valor numérico inteiro e será utilizada para aumentar a entropia do mecanismo de geração das informações do arquivo a serem remetidas para o SVI. O uso da semente tem por finalidade impedir que os SADNs possam prever os desafios e gerar previamente as respostas.

3.2.1.2 - Preparação e Envio

Neste estágio, denominado PAA-E2, é realizada a preparação do arquivo selecionado e o seu envio para o provedor de armazenamento na nuvem. A preparação tem por objetivo garantir a confidencialidade do arquivo. O estágio é descrito em três fases de acordo com a sequência em que as suas ações são iniciadas, sendo que as duas primeiras são executadas pelo papel Cliente e a última pelo papel SADN.

Na primeira fase (PAA-E2.1) é realizado o processo responsável por garantir a confidencialidade dos dados armazenados. Para isso, é aplicado sobre o arquivo um algoritmo criptográfico, que deve ser forte o suficiente para garantir o sigilo do conteúdo do arquivo durante todo o período previsto para o seu armazenamento. O algoritmo realiza a cifragem do arquivo utilizando uma senha secreta informada pelo proprietário do arquivo.

Após a conclusão da cifragem do arquivo, a segunda fase deste estágio e a primeira fase do estágio PAA-E3 são iniciadas simultaneamente, de forma a aumentar a eficiência da utilização dos recursos computacionais envolvidos e reduzir o tempo necessário para a conclusão do PAA. A segunda fase (PAA-E2.2) é a responsável por enviar o arquivo cifrado para os SADNs selecionados. Para cada SADN é iniciada simultaneamente uma instância do processo de envio do arquivo, sendo estas instâncias executadas paralelamente até a sua conclusão. A última fase (PAA-E6) é a responsável por realizar a recepção do

arquivo no SADN e salvá-lo em seu sistema de arquivos. Esta fase será descrita na Seção 3.2.3.1.

3.2.1.3 - Preparação da Tabela de Informações do Arquivo

Neste estágio, denominado PAA-E3, é realizado o processo de preparação das informações necessárias para o monitoramento do arquivo. O estágio é dividido em cinco fases executadas sequencialmente.

A primeira fase (PAA-E3.1) consiste da divisão do arquivo em frações de tamanhos iguais. O tamanho de cada fração é obtido pela divisão do tamanho total em bytes do arquivo por 4096. Caso a divisão não seja exata, o valor resultante é arredondado para o próximo inteiro. Neste caso, os bytes faltantes para completar a última fração serão obtidos do início do arquivo. Cada fração do arquivo é identificada por um código de endereço que varia entre 0 e 4095.

Na segunda fase (PAA-E3.2) é calculada a quantidade de *hashes* e blocos de dados necessários para verificar o arquivo no período de tempo previsto para o seu armazenamento (número de anos informado na Seção 3.2.1.1). Como cada *hash* e seu respectivo bloco de dados só pode ser utilizado uma única e exclusiva vez, a quantidade total de *hashes*/blocos necessários é diretamente dependente do número de blocos de dados a serem verificados a cada dia.

Um percentual mínimo de blocos de dados, calculados a partir de um percentual mínimo dos arquivos armazenados em cada SADN estará sujeita a verificação diária, de acordo com o nível de confiança atribuído ao SADN. Os níveis de confiança baseados em (Marsh 1994) são apresentados na Tabela 3.1.

Tabela 3.1 - Classificação da confiança nos SADNs
(Marsh 1994, adaptado)

Nível de Confiança	Intervalo	Verificados por dia		
		% dos arquivos	% do arquivo	Blocos
Confiança altíssima	]0,9, 1[	15%	~ 0,4%	1
Confiança alta	]0,75, 0,9]	16%	~ 0,8%	2
Confiança meio alta	]0,5, 0,75]	17%	~ 1,2%	3
Confiança meio baixa	]0,25, 0,5]	18%	~ 1,6%	4
Confiança baixa	]0, 0,25]	19%	~ 2,0%	5
Desconfiança baixa	]-0,25, 0]	20%	~ 2,4%	6
Desconfiança meio baixa	]−0,5, −0,25]	25%	~ 3,2%	8
Desconfiança meio alta	]−0,75, −0,5]	30%	~ 4,0%	10
Desconfiança alta	]−0,9, −0,75]	35%	~ 4,8%	12
Desconfiança altíssima	] −1, −0,9]	50%	~ 5,6%	14

Como não é possível prever as variações do nível de confiança no decorrer do tempo, para este cálculo é considerado o pior caso. O pior caso é a situação onde o SADN permanece durante todo o período de armazenamento do arquivo avaliado no mais baixo nível de confiança, ou seja, no nível "Desconfiança altíssima".

Considerando que os blocos de dados são formados por 16 frações diferentes e que cada arquivo é divido em 4096 frações, são necessários formar 256 blocos de dados para que todos os bytes do arquivo façam parte de pelo menos um bloco de dados. Cada conjunto com 256 blocos de dados que representam o arquivo inteiro é denominado "Ciclo".

De acordo com o apresentado na Tabela 3.1, em um SADN classificado como "Desconfiança altíssima" é necessário verificar por dia no mínimo 14 (quatorze) blocos de dados de cada arquivo armazenado pelo provedor. A geração dos blocos de dados é realizada por Ciclo. Assim, para determinar o número total de blocos de dados a ser gerado, é necessário primeiro calcular o número total de ciclos. A Equação (3.1) define o cálculo do número de ciclos e a Equação (3.2) o cálculo do número total dos blocos de dados necessários.

$$Número\,de\,Ciclos = ARREDONDAR\left(\frac{(14*366)*Anos}{256}, 0\right) \quad \text{Equação (3.1)}$$

$$Número\,de\,Blocos\,de\,Dados = Número\,de\,Ciclos * 256 \qquad \text{Equação (3.2)}$$

Na terceira fase (PAA-E3.3) são gerados os blocos de dados a partir da seleção aleatória de 16 frações do arquivo. A definição dos códigos de endereços das frações que compõem cada bloco de dados é obtida através da execução de um gerador pseudorrandômico de endereços. Para adicionar uma camada extra de entropia ao gerador de endereços é utilizada a semente numérica informada na Seção 3.2.1.1.

Na quarta fase (PAA-E3.4) os blocos de dados gerados são identificados como pertencentes a um Ciclo. O Ciclo é identificado por meio de um número inteiro iniciando em 1 e incrementado a cada 256 blocos gerados.

Concluindo o estágio, na quinta fase (PAA-E3.5) são gerados os *hashes* dos blocos de dados, o *hash* do arquivo cifrado e a Tabela de Informações (TabInfor) do arquivo. A TabInfor é formada por um cabeçalho constituído pelo *hash* que representa o arquivo, o tamanho em bytes de cada fração, a quantidade de registros e o total de ciclos. Cada registro no corpo da TabInfor representa um bloco de dados e contém: o seu *hash*; os códigos de endereço de cada uma das 16 frações do arquivo que lhe deram origem; e o número do ciclo a que pertence.

Cada bloco de dados é submetido a uma função criptográfica resultando em um *hash* de 256 bits. A função de *hash* escolhida deve ser segura o suficiente para que o *hash* gerado não forneça informações sobre o conteúdo que lhe deu origem.

Utilizando a mesma função é gerado um *hash* sobre o conteúdo do arquivo cifrado. Esse *hash* será utilizado como o identificador do arquivo. Além disso, o *hash* identificador também será utilizado para a verificação da integridade do arquivo pelo SADN após receber a sua submissão e pelo Cliente após realizar o seu *download*. Concluída a geração de todos os *hashes* é montada a TabInfor com os dados necessários para o monitoramento do arquivo pelo SVI.

3.2.1.4 - Submissão da Tabela de Informações ao Serviço de Verificação de Integridade

Neste estágio, denominado PAA-E4, a TabInfor é enviada ao SVI e armazenada pelo mesmo em sua base de dados. O processo é dividido em duas fases, a primeira fase (PAA-E4.1) é a responsável por realizar a conexão com o SVI selecionado conforme descrito na Seção 3.2.1.1 e enviar a TabInfor. A recepção da TabInfor pelo SVI é realizada no estágio (PAA-E5) e será descrita na Seção 3.2.2.1.

Após concluído o envio com sucesso da TabInfor para o SVI é iniciada a segunda e última fase (PAA-E4.2) deste estágio. Nesta fase são registradas na base de dados do Cliente as informações necessárias para restauração do arquivo pelo seu proprietário. As informações armazenadas são: o *hash* de identificação do arquivo; o nome do arquivo; o local de onde o arquivo original foi obtido; as chaves criptográficas necessárias para decifrar o arquivo; a data final do período de armazenamento e monitoramento; e os identificadores do SVI e dos SADNs utilizados.

3.2.1.5 - Recepção de Mensagens oriundas do Serviço de Verificação de Integridade

No estágio denominado PV-E5, a mensagem enviada pelo SVI, informando que foi identificada uma falha em um arquivo de propriedade do cliente é recebida. Após receber a mensagem, a base de dados que contém as informações sobre os arquivos armazenados na nuvem é atualizada. O registro referente ao arquivo cuja falha foi identificada, recebe a sinalização de que o mesmo encontra-se corrompido.

3.2.2 - Serviço de Verificação de Integridade

O Serviço de Verificação de Integridade (SVI) é o papel que concentra as ações necessárias para a realização do monitoramento periódico dos arquivos hospedados em um ou mais SADNs. Assim como no papel Cliente, as ações do SVI estão representados na Figura 3.1 nos estágios PAA-E5, PV-E1, PV-E3 e PV-E4. Nas Seções 3.2.2.1, 4.1.1.2, 4.1.1.3, 4.1.1.4, 4.1.1.5 e 4.1.1.6 serão descritas as ações executadas em cada estágio.

3.2.2.1 - Recebimento da Tabela de Informações

Neste estágio, denominado PAA-E5, a TabInfor é recebida do Cliente e armazenada na base de dados do SVI. Para isso, o SVI deve manter um serviço continuamente disponível a aceitar as conexões dos clientes. O Cliente deve manter um acordo prévio junto ao SVI determinando com quais SADNs ele mantém contrato.

3.2.2.2 - Geração e Envio de Desafios

Este é o primeiro estágio do Processo de Verificação (PV) e é denominado PV-E1. O processo destina-se a realizar periodicamente a verificação da integridade dos arquivos armazenados nos SADNs e em consequência classificar estes serviços em níveis de confiança. Neste estágio são gerados os desafios e enviados para SADN, sendo suas ações distribuídas em três fases executadas sequencialmente.

O processamento do estágio se dá por meio de um serviço de agendamento, que diariamente inicia simultaneamente uma instância de execução do estágio para cada SADN que o SVI possui arquivos a monitorar. A execução das diversas instâncias são realizadas em paralelo de forma a aproveitar o máximo da eficiência computacional disponível no SVI. Em cada instância e de forma independente, as fases do estágio são executadas ciclicamente até que o percentual de arquivos verificados no dia seja igual ou maior que o percentual previsto na Tabela 3.1, conforme o nível de confiança atribuído pelo SVI ao SADN.

A primeira fase (PV-E1.1) tem como finalidade identificar o próximo arquivo a ser verificado. A partir da base de dados do SVI é localizado o último arquivo verificado, e a partir deste, é selecionado o primeiro arquivo na sequência, seguindo a ordem cronológica da recepção e registro de sua TabInfor pelo SVI. Caso seja o primeiro arquivo a ser verificado no SADN, o arquivo representado pela primeira TabInfor recebida é selecionado.

Após selecionado o arquivo, inicia-se a segunda fase (PV-E1.2) onde é recuperada, a partir da base de dados do SVI, a sua TabInfor. Conforme o nível de confiança no SADN e o

número de blocos de dados a ser verificado por arquivo a cada dia, conforme previsto na Tabela 3.1, é realizada uma seleção aleatória dos *hashes* registrados na TabInfor e que ainda não tenham sido utilizados anteriormente em nenhuma verificação.

Os endereços das frações dos blocos de dados que deram origem a cada *hash* selecionado, em conjunto com o identificador do arquivo, o identificador do cliente e um identificador do desafio formam a estrutura de dados dos desafios a serem enviados para o SADN. O SVI atualiza os registros dos *hashes* selecionados na TabInfor marcando-os como já utilizados na verificação do arquivo. Além disso, é inserido um novo registro em uma base de dados com os desafios aguardando respostas para cada *hash* selecionado.

Por fim, na terceira fase (PV-E1.3), o desafio é enviado para o SADN. Para isso o SVI realiza uma conexão com um serviço de recepção de desafios disponível no SADN. Caso o SADN esteja indisponível no momento da geração do desafio, o SVI deve manter um serviço que monitore o SADN e que, assim que o mesmo se torne disponível, envie todos os desafios pendentes. O estágio PV-E2 responsável pelas ações de recepção, processamento e geração da resposta ao desafio, todas executadas pelo SADN, é descrito na Seção 3.2.3.2.

3.2.2.3 - Recebimento e Verificação de Respostas

Este estágio, denominado PV-E3, é responsável por realizar o recebimento das respostas aos desafios submetidos ao SADN e verificar se a resposta recebida confirma a integridade dos blocos de dados verificados pelo desafio. Para isso, o SVI deve manter continuamente um serviço que atenda as conexões do SADN e receba as respostas dos desafios submetidos.

A estrutura de dados das respostas aos desafios é formada pelo *hash* da resposta e por um identificador do desafio que lhe deu origem. O *hash* resposta gerado pelo SADN representa o conteúdo do bloco de dados montado pelo mesmo a partir do arquivo armazenado utilizando os endereços das frações constantes do desafio.

As ações deste estágio compõem duas fases, na primeira fase (PV-E3.1) o SVI recebe a resposta do SADN e localiza em sua base de dados o registro do desafio pendente através do seu identificador, registrando o recebimento da resposta. Na segunda fase (PV-E3.2) é realizada a comparação do *hash* resposta com o *hash* original obtido no registro da TabInfor que deu origem ao desafio. O resultado dessa comparação é armazenado no registro de origem na TabInfor. Caso os *hashes* comparados sejam iguais, pode-se afirmar que o bloco de dados verificado no arquivo armazenado no SADN permanece íntegro. Caso contrário, a integridade do arquivo foi corrompida.

3.2.2.4 - Atualização do Nível de Confiança

Neste estágio, denominado PV-E4, é realizada a atualização do nível de confiança atribuído ao SADN. O estágio é divido em quatro fases distintas e independentes. Se o bloco de dados verificado na fase anterior for considerado íntegro, serão executadas a primeira fase e a segunda fase, caso contrário serão executadas apenas a terceira fase e a quarta fase.

A primeira fase (PV-E4.1) verifica se ainda existem blocos de dados não verificados no mesmo ciclo do bloco recém-verificado. Caso todos os blocos de dados de um determinado ciclo já tenham sido verificados e considerados íntegros, isso significa que o ciclo foi concluído e todos os bytes do arquivo armazenado no SADN foram verificados e nenhum problema foi identificado.

A conclusão bem-sucedida de um ciclo habilita o SADN a receber um incremento no seu nível de confiança, de acordo com o processo descrito na segunda fase. Caso o ciclo não tenha sido concluído, a segunda fase não será executada e um novo PV será iniciado conforme descrito na Seção 3.2.2.2.

Na segunda fase (PV-E4.2), o Valor da Confiança (VC) no SADN é incrementado. Conforme a Tabela 3.1 o VC é um valor real pertencente ao intervalo aberto entre -1 e 1. O Valor do Incremento (VI) varia de acordo com o VC atual. Se o VC é menor que zero, o VI é igual a 2,5% do módulo da diferença entre -1 e o VC. Se o VC é igual a zero, o VI é igual a 0,1. Se o VC é maior que zero e menor que 0,5, o VI é igual a 2,5% do VC. Caso o VC seja maior que 0,5, o VI é igual 0,5% da diferença entre 1 e o VC.

Realizado o cálculo, o VI é adicionado ao VC, que por sua vez é persistido no cadastro do SADN na base de dados do SVI. Concluída esta fase, um novo PV será iniciado conforme descrito na Seção 3.2.2.2.

A terceira fase (PV-E4.3) decrementa o VC. Esta fase é executada quando no estágio descrito na Seção 3.2.2.3 for identificada uma falha. O Valor do Decremento (VD) varia de acordo com o VC atual. Se o VC é maior que zero, o VD recebe o valor do VC. Se o VC é igual a zero, o VD é igual 0,1. Se o VC é menor que zero e maior ou igual a -0,5, o VD é igual ao módulo de 15% do VC. Caso o VC seja menor que -0,5, o VD é igual ao modulo de 2,5% da diferença entre -1 e o VC.

Realizado o cálculo, o VD é subtraído do VC, que por sua vez é persistido no cadastro do SADN na base de dados do SVI. Concluída esta fase, um novo PV será iniciado conforme descrito na Seção 3.2.2.2.

A quarta fase (PV-E4.4) é iniciada simultaneamente com a terceira fase, sendo executada em paralelo. Nesta fase é enviada uma mensagem para o Cliente, proprietário do arquivo verificado, informado-o sobre a falha identificada. A recepção da mensagem é realiza pelo Cliente no estágio PV-E5, conforme descrito na Seção 3.2.1.5.

3.2.3 - Serviço de Armazenamento de Dados na Nuvem

O Serviço de Armazenamento de Dados na Nuvem (SADN) é o papel que concentra as ações relativas ao armazenamento do arquivo na nuvem e o processamento e resposta a desafios de verificação de integridade. As ações do SADN estão representados na Figura 3.1 nos estágios PAA-E6 e PV-E2. Nas Seções 3.2.3.1, 3.2.3.2 serão descritas as ações executadas em cada estágio.

3.2.3.1 - Recebimento do Arquivo Cifrado

Neste estágio, denominado PAA-E6, o arquivo cifrado pelo Cliente é recebido pelo SADN e armazenado em sua estrutura de arquivos. Para isso, o SADN deve manter um serviço continuamente disponível a aceitar as conexões dos clientes. O Cliente deve manter um

acordo prévio junto ao SADN para que este aceite a sua conexão. A identificação do cliente deve ser realizada por meio de um certificado digital válido. Após concluída a recepção do arquivo, o SADN valida a sua integridade por meio do *hash* identificador e responde ao Cliente com o valor booleano "verdadeiro" se o processo de recebimento do arquivo foi realizado com sucesso, ou o com o valor "falso" se ocorrer algum problema.

3.2.3.2 - Recepção, Processamento e Resposta aos Desafios

Neste estágio, denominado PV-E2, são realizadas a recepção, o processamento e a resposta ao desafio gerado pelo SVI. O estágio é composto de quatro fases, sendo que cada fase pode possuir múltiplas instâncias sendo executadas simultaneamente.

Na primeira fase (PV-E2.1) é realizado o recebimento dos desafios e o armazenamento dos mesmos na base de dados de desafios pendentes. Para isso o SADN deve manter um serviço continuamente ativo que atende as conexões do SVI e verifica se o mesmo está autorizado pelo proprietário do arquivo a qual se refere os desafios recebidos. Concluído o recebimento dos desafios, imediatamente é disparada uma chamada assíncrona de um processo que verifica os desafios pendentes e inicia paralelamente a execução de uma instância da segunda fase para cada um dos desafios recebidos.

Na segunda fase (PV-E2.2) são extraídos do desafio os endereços das frações que compõem o bloco de dados, o identificador do arquivo e o tamanho de cada fração. De forma simultânea, são instanciados os processos de leitura de cada uma das frações no arquivo armazenado. Concluída a leitura de todas as frações, o bloco de dados é montado, seguindo a ordem das frações recebida no desafio.

Na terceira fase (PV-E2.3) é gerado o *hash* de 256 bits sobre o bloco de dados montado na segunda fase. O bloco de dados deve ser submetido a mesma função criptográfica de *hash* utilizada pelo Cliente na geração da TabInfor. O *hash* gerado é armazenado em uma base de dados de respostas pendentes.

Na quarta fase (PV-E2.4), finalizando o estágio, são enviadas ao SVI as respostas aos desafios. Para isso, um processo deve ficar constantemente verificando a existência de

respostas pendentes de envio. Havendo respostas pendentes, o SADN realiza uma conexão com o SVI e envia-lhe as respostas. Cada resposta é formada pelo *hash* gerado e o identificador do desafio que lhe deu origem.

4 - IMPLEMENTAÇÃO DA ARQUITETURA

A implementação da arquitetura foi dividida em três fases. Na primeira fase foram implementados todos os processos sob responsabilidade do cliente. Na segunda fase foram implementados todos os processos sob responsabilidade do SVI, e finalmente, na terceira fase, foram implementados todos os processos sob responsabilidade do SADN.

Para cada fase foi desenvolvida uma aplicação, todas elas utilizando componentes da Tecnologia Java EE como JPA, EJB, CDI e JAX-WS (Jendrock et al. 2014). Para o cliente foi desenvolvida uma aplicação *desktop* enquanto que para o SVI e para o SADN foram desenvolvidas aplicações compostas por *web services.* O servidor de aplicações utilizado foi o *Glassfish* 4 (Oracle 2016b) e o *PostgreSQL* (The PostgreSQL Global Development Group) foi o Sistema Gerenciador de Banco de Dados (SGBD) escolhido.

O processo de seleção das tecnologias utilizadas levou em consideração as características distribuídas do protocolo e a necessidade de comunicação contínua e assíncrona entre os papéis previstos na arquitetura. Assim, a escolha da Tecnologia Java EE baseou-se na facilidade de implementação de recursos como *web services*, agendamento de tarefas, monitoramento de eventos e chamadas assíncronas. Já o servidor de aplicação *Glassfish* e o SGBD *PostgreSQL* foram escolhidos por serem ambas aplicações *open source* e atenderem plenamente as necessidades de aplicação desenvolvida.

4.1.1 - Aplicação Cliente

As principais tarefas da aplicação cliente são: cifrar o arquivo e enviá-lo para um ou mais SADN, dividir e selecionar as frações, montar os blocos de dados, gerar seus *hashes*, agrupá-los em ciclos, gerar a TabInfor e enviá-la para o SVI. A aplicação cliente ainda possibilita controlar o inventário dos arquivos mantidos no provedor da nuvem, armazenar as chaves criptográficas, consultar a informações sobre o processo de verificação e a integridade de cada arquivo em cada um dos SADN, recuperar um arquivo de um SADN, confirmar sua integridade e decifrá-lo.

4.1.1.1 - Seleção do Arquivo, do SVI e dos SADNs

O processo de envio de um arquivo para seu armazenamento seguro em um SADN de forma que este possa ser verificado continuamente por um SVI inicia pela seleção dos mesmos. Para isso, na aplicação cliente foi implementada uma interface gráfica por meio da qual o cliente seleciona um arquivo em seu sistema de arquivos, um SVI e um ou mais SADNs. Na mesma interface, o cliente informa a senha a ser utilizada no processo de cifragem do arquivo e, para atender os requisitos do protocolo, informa o número de anos em que o arquivo deverá ser mantido armazenado no SADN; e uma semente numérica, utilizada para acrescer entropia ao processo de escolha das frações que comporão cada bloco de dados, conforme apresentado na Figura 4.1.

A lista de SVIs disponíveis para seleção é obtida a partir de um cadastro prévio, onde o cliente registra os SVIs com os quais o mesmo mantém um acordo/contrato de nível de serviço. Após a seleção do SVI, a aplicação cliente obtém uma lista dos SADNs para os quais o cliente mantém contratos de verificação de arquivos, por meio da chamada ao método "*getListContract*" do *web service* disponibilizado pelo SVI, conforme descrito na Seção 4.1.2.6. Para cada SADN é apresentado ao cliente o nível de confiança atual atribuído pelo SVI. A Figura 4.1 apresenta a interface da funcionalidade "*Upload File*" da aplicação cliente.

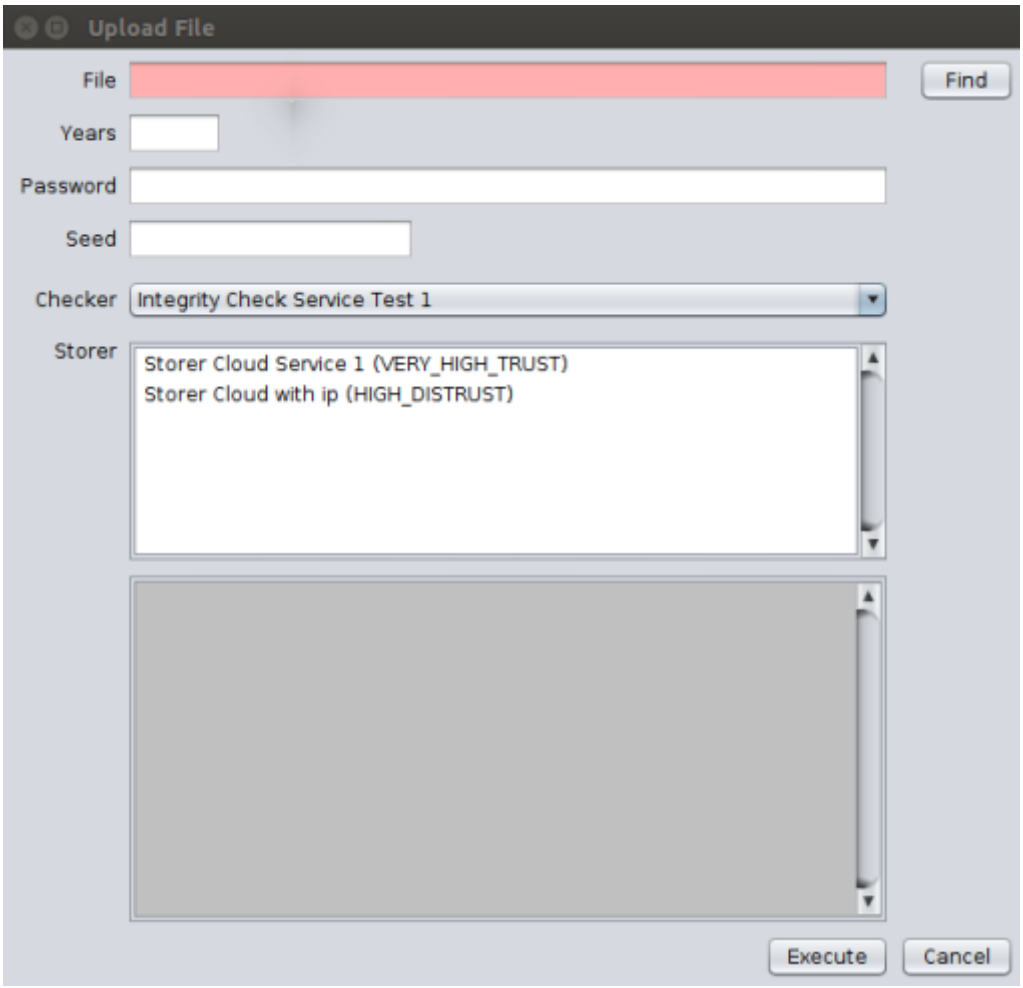


Figura 4.1 - Interface da funcionalidade "*Upload File*" na aplicação cliente.

4.1.1.2 - Processo de Cifragem

O processo de cifragem do arquivo é realizado por meio dos recursos disponibilizados pelo pacote "javax.crypto" (Oracle 2016), utilizando o algoritmo criptográfico o AES (NIST FIPS PUB 197) , uma chave com tamanho igual a 256 bits e o modo de operação *Cipher-Block Chaining* (CBC) (Bellare, Kilian, Rogway 1994).

O principal fator que levou a escolha do algoritmo AES usando uma chave de 256 bits é a necessidade de garantir a confidencialidade do conteúdo do arquivo, no mínimo pelo tempo definido para o seu armazenamento pelo SADN, conforme prevê os objetivos deste trabalho na Seção 1.2. Como fator negativo dessa escolha, destaca-se o tempo de ciframento/deciframento que pode aumentar em até 40% (McGrew 2011). No entanto, como os processos de ciframento e deciframento, de acordo com o que prevê o protocolo,

são realizados uma única vez cada um, no *upload* e *download* do arquivo, esse aumento não chega a influenciar na eficiência da função principal da arquitetura que é a verificação da integridade.

A senha informada, antes de ser aplicada no processo de cifragem, passa por um processo de preparação da chave criptográfica por meio de um algoritmo que a reforça, denominado Password based Key Derivative Function (PBKDF2) (Josefsson 2011), o qual aplica um algoritmo de *hash* com chave construído a partir da função de *hash* SHA-256, usada para gerar um código de autenticação de mensagens do tipo "*Hash-based Message Authentication Code*" (HMAC) (Krawczyk, Bellare, Canetti 1997).

4.1.1.3 - Envio do Arquivo para o Serviço de Armazenamento de Dados na Nuvem

A etapa de envio do arquivo para um ou mais SADNs foi implementada utilizando *Java Threads.* Este recurso possibilita iniciar de forma simultânea, assim que concluída a cifragem do arquivo, o envio (*upload*) do arquivo cifrado para cada um dos SADNs selecionados. Além disso, a utilização de *threads* propicia que as demais etapas previstas no protocolo possam continuar a serem executadas de forma paralela, sem a necessidade de aguardar a conclusão do envio do arquivo, cujo tempo necessário varia de acordo com o tamanho do arquivo e as características da rede utilizada.

4.1.1.4 - Cálculo do Número de Ciclos e Distribuição das Frações

Esta etapa inicia com o cálculo do número de ciclos de verificação, de acordo com o número de anos que o arquivo deve ser mantido pelos SADNs e monitorado pelo SVI. Apesar do número de ciclos empregados variar de acordo com o nível de confiança atribuído ao SADN, para fins deste cálculo foi considerado o pior caso, o nível "Desconfiança altíssima".

Para a seleção e distribuição das frações por bloco de dados dentro de cada ciclo de verificação, foi construída a classe "*Cycle*". Cada instância da classe tem a responsabilidade de sortear as frações e gerar os 256 blocos de cada ciclo. O código de endereço de cada fração é obtido por meio do algoritmo SHA1PRNG, um gerador

pseudorrandômico de números, executado por intermédio da classe "*SecureRandom*" do pacote "*java.security*" (Oracle 2016) que utiliza a semente numérica informada para adicionar entropia aos valores gerados. A classe "*Cycle*" é apresentada na Figura 4.2.

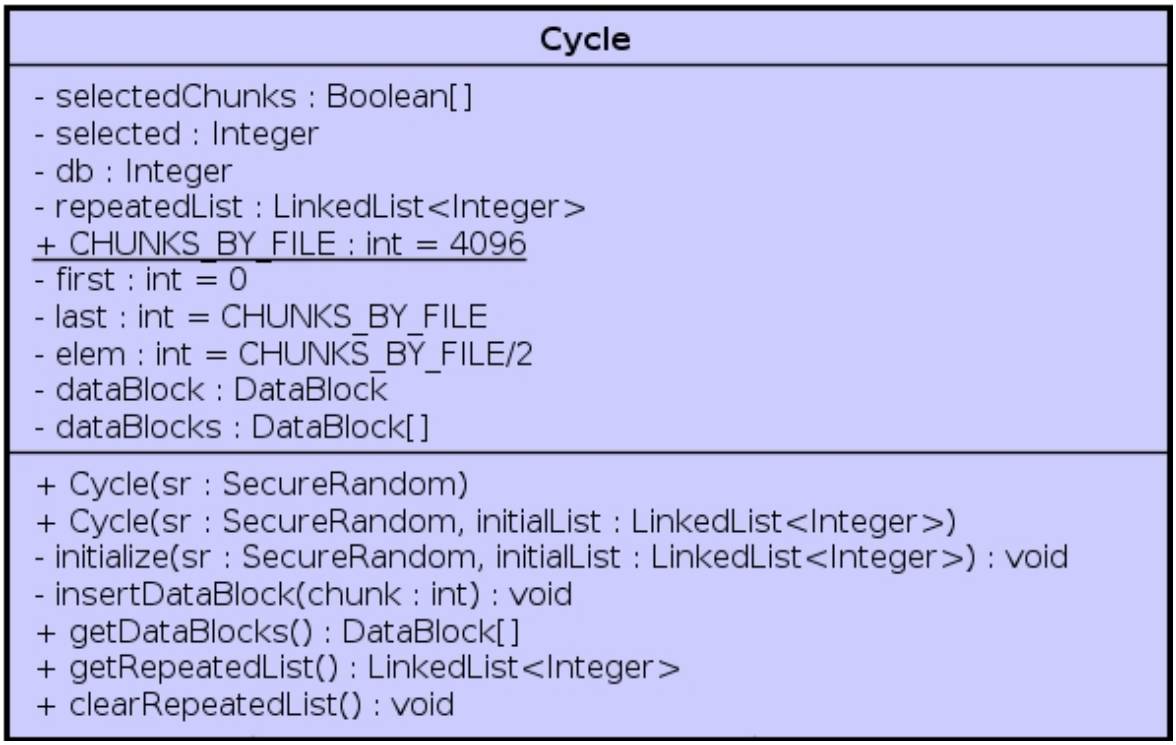


Figura 4.2 - Classe *Cycle*.

A cada iteração com a classe são gerados aleatoriamente 4096 códigos de endereço, dos quais os valores não repetidos são utilizados para a definição dos blocos de dados. É previsto que sejam realizadas tantas iterações quantas forem necessárias para definir os 256 blocos de dados. Os códigos de endereço repetidos são armazenados e repassados em ordem inversa para as instâncias seguintes da classe "*Cycle*", que utilizam os códigos de endereço recebidos na geração dos seus blocos de dados. Para o registro dos códigos de endereço selecionados e as posições que estes representam na sequência de montagem das frações dentro de cada bloco de dados foi criada a classe "*DataBlock*". Cada instância da classe "*Cycle*" mantém 256 instâncias da classe "*DataBlock*". A Figura 4.3 apresenta a classe "*DataBlock*".

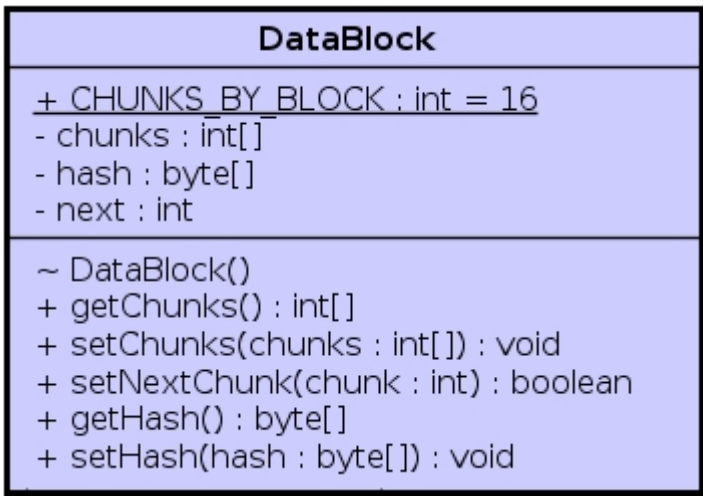


Figura 4.3 - Classe *DataBlock*.

4.1.1.5 - Geração da Tabela de Informações do Arquivo

Após a definição dos ciclos, inicia-se o processo de geração da TabInfor, para o qual é necessário o cálculo dos *hashes* que representam cada um dos blocos de dados formados pela união dos conteúdos das frações escolhidas na Seção 4.1.1.4. Para a geração dos *hashes* foi adotada a função criptográfica denominada Blake2 (Aumasson et al. 2013), escolhida devido a sua velocidade, segurança e simplicidade.

A implementação dessa etapa foi realizada por meio da classe "*InformationTable*". Na sua iniciação, a instância da classe recebe como parâmetros: os objetos da classe "*Cycle*" gerados conforme apresentado na Seção 4.1.1.4, um objeto da classe "*RandomAccessFile*" pertencente ao pacote "*java.io*" (Oracle 2016), que permite o acesso randômico ao conteúdo do arquivo cifrado e um objeto da classe "Blake2b" que implementa o algoritmo de geração *hash* utilizado. A Figura 4.4 apresenta a classe "*InformationTable*".

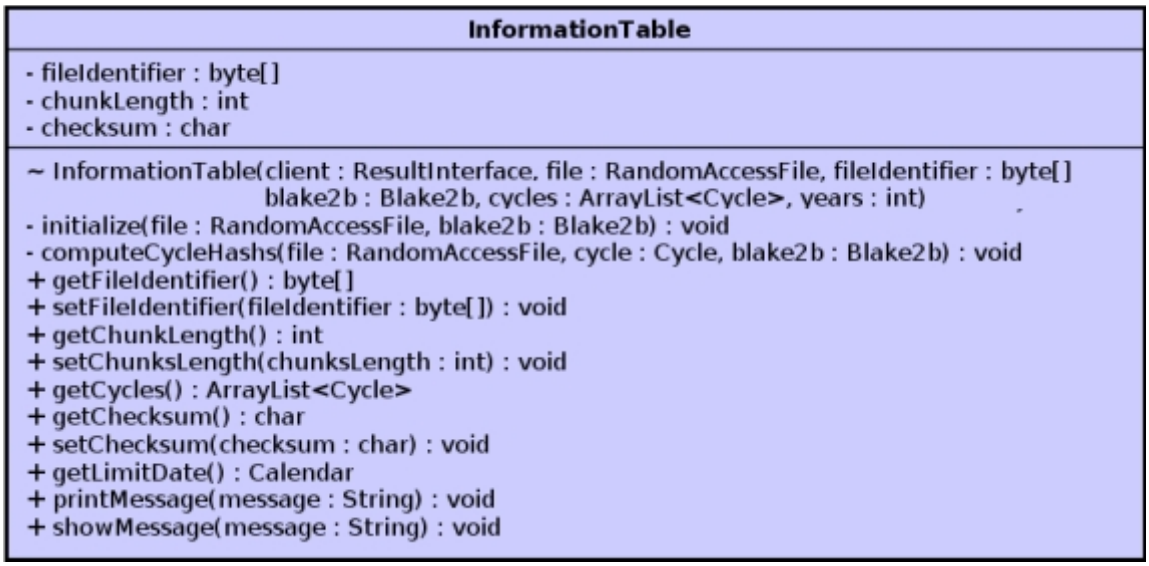


Figura 4.4 - Classe *InformationTable*.

A partir do arquivo cifrado é gerado um *hash* de 256 bits que servirá como identificador do arquivo para todos os componentes da arquitetura previstos no protocolo proposto e ainda, pela aplicação cliente para verificar a integridade do arquivo quando o mesmo for recuperado a partir do SADN. O tamanho de fração é obtido pelo resultado arredondado para cima da divisão do tamanho do arquivo por 4.096.

Para o cálculo dos *hashes* são percorridas todas as instâncias da classe "*Cycle*" e em cada uma delas, todas as 256 instâncias da classe "*DataBlock*", da qual se obtém os 16 códigos de endereço que multiplicados pelo tamanho da fração, determinam os endereços de início de cada fração a ser lida e anexada ao bloco de dados. Concluída a montagem do bloco, é realizado o cálculo do *hash* e sua inserção como atributo na respectiva instância da classe "*DataBlock*".

4.1.1.6 - Registro e Submissão das Informações Geradas

Após a conclusão da geração da TabInfor, inicia-se o processo de envio da mesma para o SVI. Para isso, o Cliente realiza uma conexão com um *web service* denominado *"VerifierService"* disponível no SVI e executa o método "*registerIT*" descrito na Seção 4.1.2.1. Após a conclusão do envio da TabInfor, o Cliente registra em sua base de dados na tabela "*stored_file*s" as informações necessárias para manter um inventário dos seus arquivos armazenados no provedor da nuvem.

A tabela "*stored_files*" é composta dos campos "*id*", "*fulloriginpath*", "*identifier*", "*iv*", "*name*", "*password*", "*salt*", "*checker_id*", os quais recebem respectivamente as seguintes informações: identificador sequencial único do registro; a pasta de origem do arquivo; o *hash* identificador do arquivo; o vetor de inicialização; o nome do arquivo; a senha utilizada para criptografar o arquivo; o *salt*; e o identificador do SVI.

Para cada registro gerado na tabela "*stored_files*" são gerados um ou mais registros na tabela "*file_contracts*" de acordo com o número de SADN escolhidos na Seção 4.1.1.1. Cada registro é composto pelos campos "*contracid*" e "*stored_files_id*". Estes campos recebem respectivamente, o identificador único do contrato com o SADN e o identificador do registro de origem na tabela "*stored_files*". O diagrama de Entidades e Relacionamentos (ER) da aplicação Cliente é apresentado na Figura 4.5.

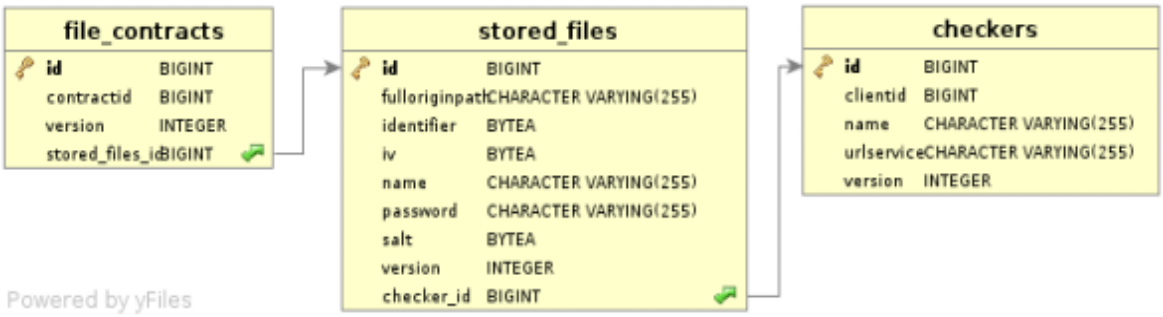


Figura 4.5 - Diagrama de entidades e relacionamentos da aplicação cliente.

4.1.1.7 - Monitoramento dos Arquivos Armazenados no Provedor da Nuvem

O módulo de monitoramento foi desenvolvido com o objetivo de disponibilizar uma ferramenta prática que permitisse gerenciar o inventário dos arquivos armazenados no provedor da nuvem, consultar os status dos arquivos atribuídos pelos SVIs de acordo com os resultados das verificações realizadas junto aos SADNs, realizar o download dos arquivos a partir dos SADNs e por fim, realizar o deciframento dos arquivos recuperados. A Figura 4.6 apresenta a tela principal do módulo de monitoramento.

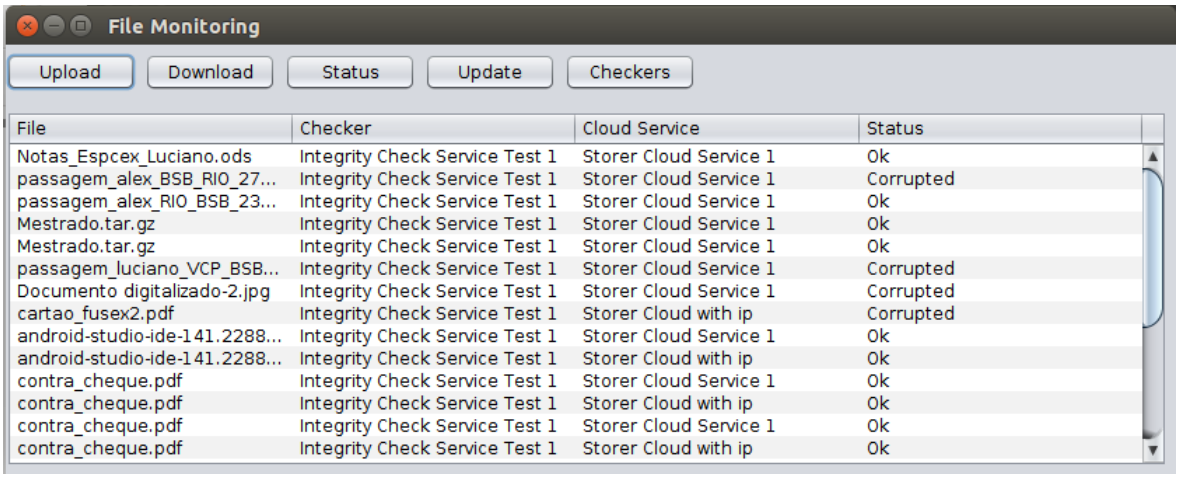


Figura 4.6: Tela principal do módulo de monitoramento.

Através do botão "*Status*" é possível obter as informações como: a pasta de origem do arquivo; o *hash* de identificação; o nome SADN onde o mesmo está armazenado; o nome do SVI responsável pelo seu monitoramento, a quantidade de ciclos de verificação que foram calculadas para o arquivo, o número do ciclo que está sendo verificado atualmente; o número de blocos de dados do ciclo atual já checados; A data final do período em que o mesmo será monitorado pelo SVI; e o seu status atual. A Figura 4.7 apresenta a tela de consulta ao status do arquivo.

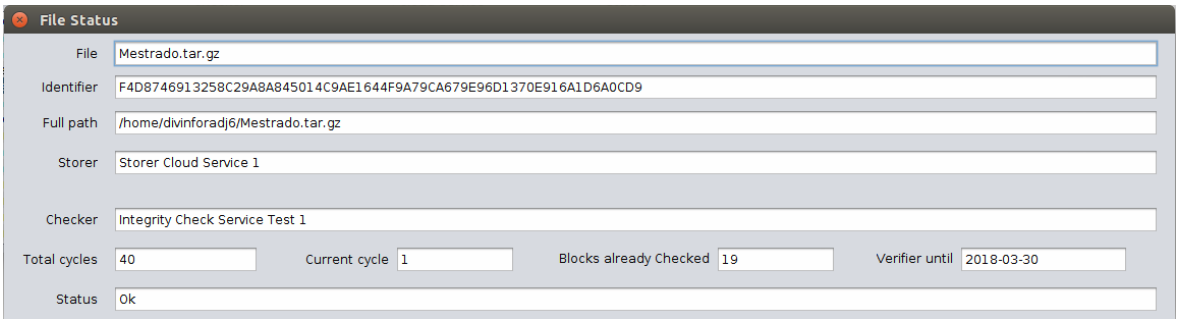


Figura 4.7 - Tela de consulta ao status do arquivo.

Outra funcionalidade disponibilizada no módulo de monitoramento é a recuperação dos arquivos armazenados na nuvem. O processo de recuperação é composto do *download* do arquivo a partir do SADN, da verificação da sua integridade, da execução do seu deciframento e do seu salvamento na estrutura de arquivos do cliente. A execução da recuperação do arquivo é realizada por meio do botão "*Download*".

Para o armazenamento do arquivo recuperado, a aplicação sugere a pasta de onde o mesmo foi obtido no momento em foi enviado para o SADN. Opcionalmente o operador pode definir qualquer outra pasta no seu sistema de arquivos. Durante o processo de recuperação, a aplicação vai apresentando as ações em andamento. A tela de execução do *download* é apresentada na Figura 4.8.

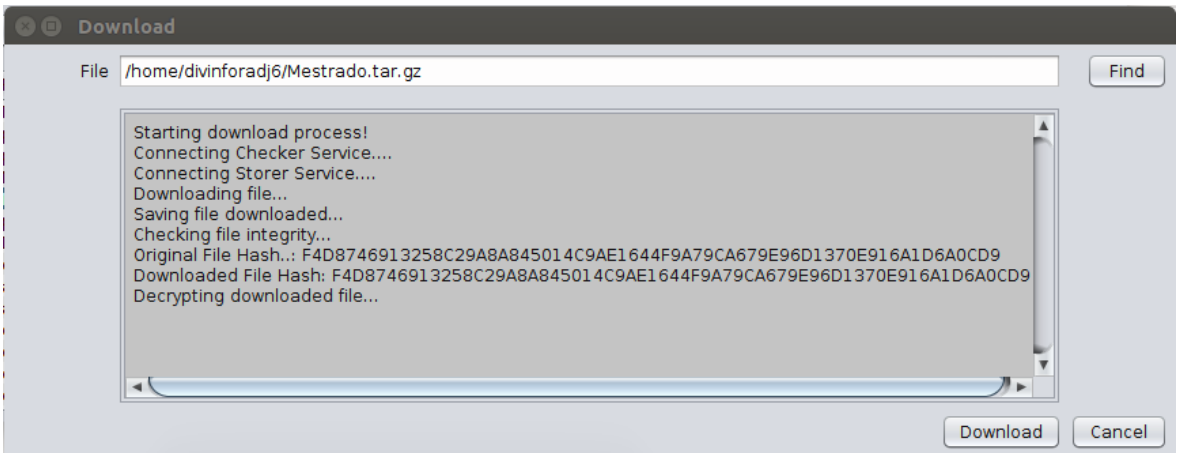


Figura 4.8 - Tela de execução do *download*.

Outra ferramenta do módulo de monitoramento é o mecanismo de atualização do status dos arquivos, acionado através do botão "*Update*". A atualização do status de cada arquivo é obtida por meio da execução do método "*getFileStatus*" disponibilizado pelo SVI no *web service* "*VerifierService*" conforme descrito na Seção 4.1.2.5.

Para facilitar o trabalho de gerência do Cliente, foi disponibilizado na tela principal do módulo de monitoramento o botão "*Upload*", conforme pode ser visualizado na Figura 4.6. Esse botão dá acesso ao processo de preparação e envio de arquivo para os SADNs descrito nas Seções 4.1.1.1, 4.1.1.2, 4.1.1.3, 4.1.1.4, 4.1.1.5 e 4.1.1.6.

4.1.1.8 - Cadastro dos Serviços de Verificação de Integridade

Por meio desta funcionalidade o Cliente gerencia o cadastro dos SVIs com os quais mantém acordo para o monitoramento de seus arquivos. O acesso ao cadastro de SVIs pode ser realizado através do botão "*Checkers*" disponível no módulo de monitoramento descrito na Seção 4.1.1.7. A Figura 4.9 apresenta a tela de cadastro de SVIs.

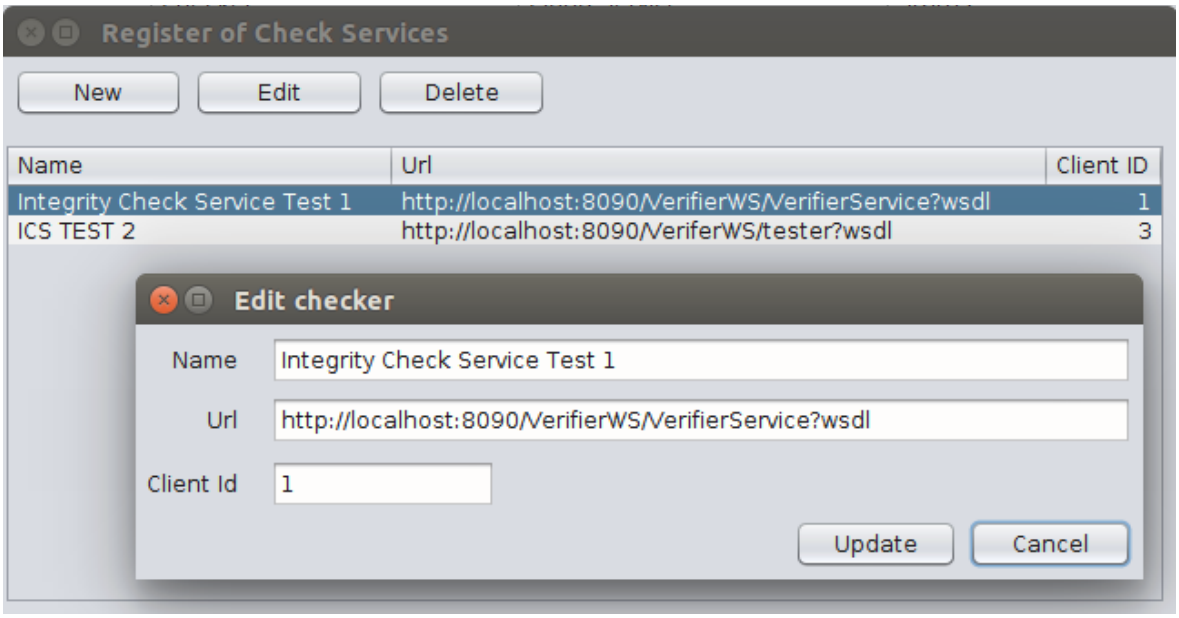


Figura 4.9 - Tela de cadastro dos serviços de verificação de integridade.

O cadastro de cada SVI é composto por um nome pelo qual o serviço é reconhecido, o endereço na internet/intranet onde o serviço está disponível (*Uniform Resource Locator* - URL) e um código fornecido pelo SVI que identifica o Cliente junto ao mesmo.

4.1.2 - Aplicação do Serviço de Verificação de Integridade

A implementação do Serviço de Verificação de Integridade (SVI) proposto no protocolo, conforme descrito na Seção 3.2.2, resultou em uma aplicação *web service* denominada *"VerifierService"*, o mesmo nome da classe que a implementa. A Figura 4.10 apresenta a classe "*VerifierService*".

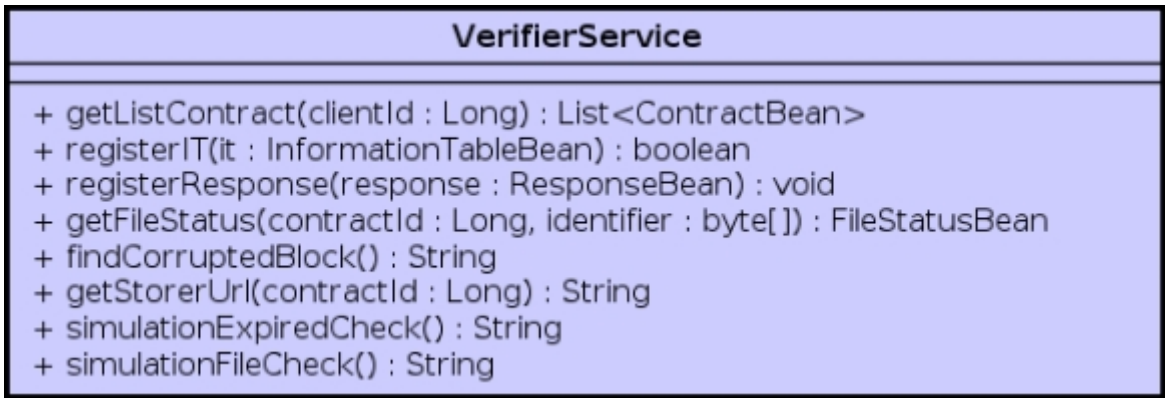


Figura 4.10 - Classe *VerifierService*.

A aplicação possui as seguintes funcionalidades: receber a TabInfor submetida pelo cliente; selecionar e enviar desafios sobre os conteúdos dos arquivos monitorados aos SADNs que os hospedam; gerenciar os desafios pendentes de resposta; receber as respostas dos desafios enviadas pelos SADNs; atualizar os níveis de confiança nos SADNs; receber e responder as requisições do Cliente sobre o status dos arquivos monitorados; receber e responder as requisições de informações sobre os SADNs com os quais o Cliente e o SVI possui acordo de monitoramento.

4.1.2.1 - Recebimento da Tabela de Informações do Arquivo

O recebimento da TabInfor de um arquivo é o ponto de partida para o monitoramento do arquivo. Para essa tarefa foi implementado um *web service* executado através do método "*registerIT*" da classe "*VerififerService*". Este método recebe a TabInfor como seu único parâmetro. As informações que compõem a TabInfor foram descritas na Seção 3.2.1.3.

Na TabInfor, os SADNs são referenciados através dos identificadores dos contratos firmados entre o Cliente e o SVI. Ao receber a TabInfor, as informações sobre o arquivo são extraídas e armazenadas na base de dados do SVI em uma tabela denominada "*archives*". O diagrama de entidades e relacionamentos da base de dados do SVI é apresentado na Figura 4.11.

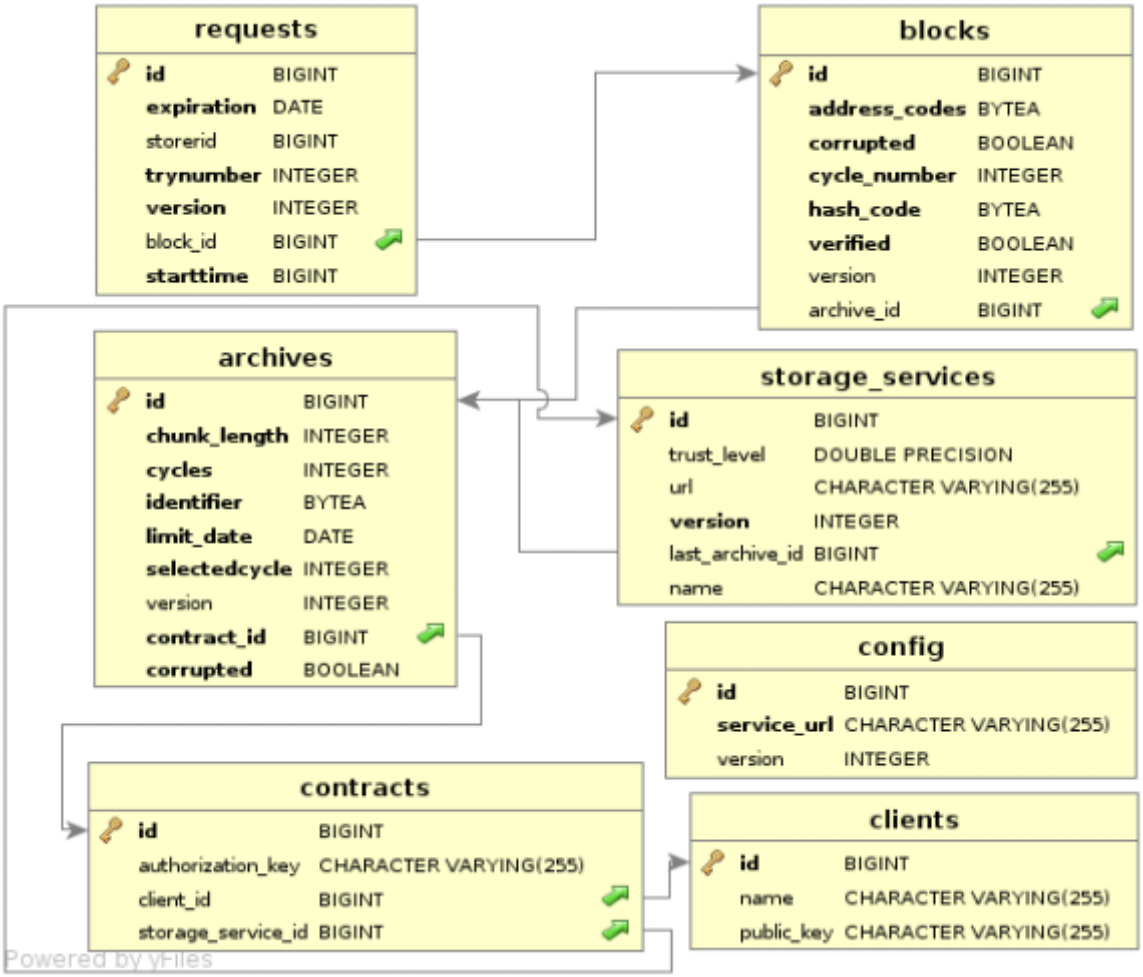


Figura 4.11 - Diagrama ER da base de dados do SVI

Para cada SADN, representado pelo identificador do contrato constante na TabInfor, é salvo um novo registro na tabela "*archives*" de forma que o monitoramento de uma cópia do arquivo não interfere no monitoramento das demais. Na tabela "*archives*" são armazenadas as seguintes informações: o *hash* identificador do arquivo; o tamanho em bytes de cada fração; o número total de ciclos gerados; e o identificador do contrato que define o SADN em que o arquivo está armazenado.

Após a criação do registro na tabela "*archives*", para cada *hash* de bloco de dados recebido na TabInfor é gerado um novo registro na tabela "*blocks*" vinculando-o ao registro da tabela "*archives*". Na tabela "*blocks*" são armazenadas as seguintes informações: o *hash* do bloco de dados; os códigos de endereço das frações que o compõem; o número do ciclo ao qual pertence; e o identificador do registro ao qual está vinculado na tabela "*archive*".

4.1.2.2 - Seleção, Geração e Envio de Desafios

O processo de seleção, geração e envio de desafios é uma atividade executada diariamente pelo SVI e compreende as ações propostas no protocolo e descritas na Seção 3.2.2.2. O processo compreende as seguintes ações: seleção dos arquivos a serem verificados em cada SADN; seleção dos blocos de dados a serem verificados em cada arquivo; geração dos desafios; e o envio dos desafios para os SADNs.

Para dar início ao processo, foi implementada a classe "*FileCheckHandler*". Nesta classe foi definido o método "*callStorageThread*", que por intermédio da anotação "*@Schedule*", pertencente ao pacote "*javax.ejb*" (Oracle 2016), é executada automaticamente pelo servidor de aplicação no primeiro minuto de cada dia. A Figura 4.12 apresenta a classe "*FileCheckHandler*".

Figura 4.12 - Classe *FileCheckHandler*.

O método realiza a leitura dos contratos ativos entre os Clientes e o SVI, que encontram-se armazenados na tabela "*contracts*", e extrai os SADNs que serão monitorados. Para cada SADN é disparado um evento "*StorageService*" que é capturado por um método denominado "*checkStorageService"* pertencente a classe "*RequestService*". A classe "*RequestService*" é apresentada na Figura 4.13.

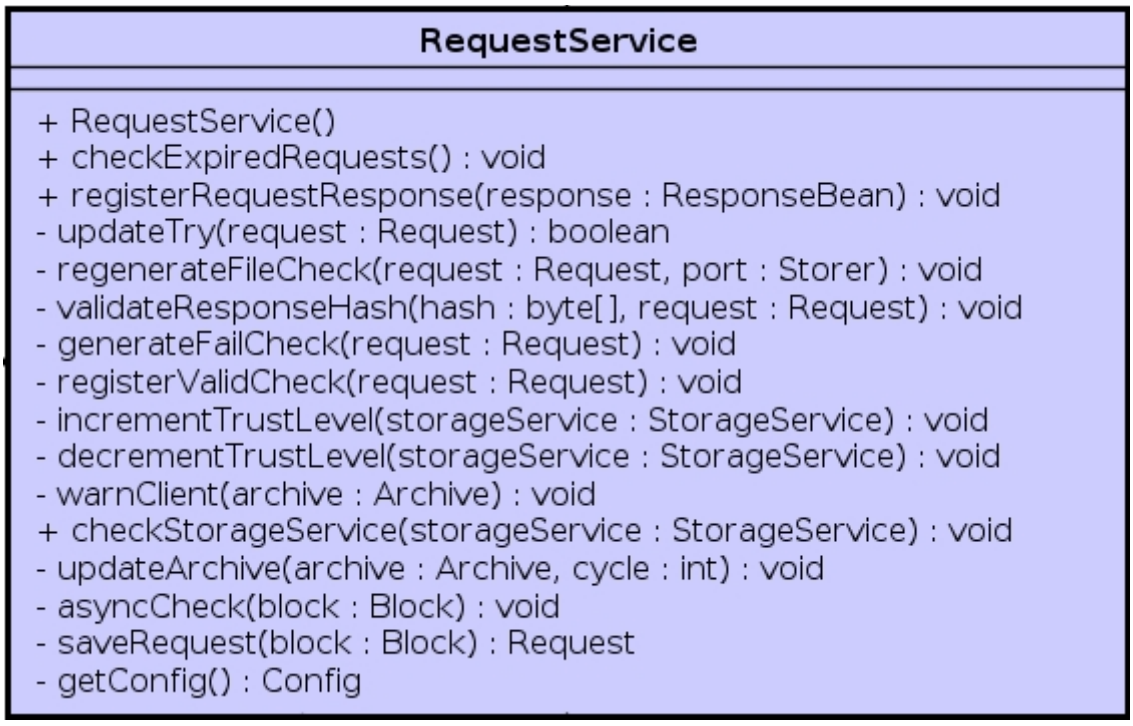


Figura 4.13 - Classe *RequestService*.

A atribuição da função de captura do evento "*StorageService*" ao método "*checkStorageService*" é realizada por meio da anotação "*@Observes*" pertencente ao pacote "*javax.enterprise.event*" (Oracle 2016). Sempre que o servidor de aplicação identifica que há um evento "*StorageService*" pendente, o método "*checkStorageService*" é assincronamente executado, recebendo como parâmetro um objeto com as informações sobre o SADN. O comportamento assíncrono na execução do método é obtido através da anotação "*@Asynchronous*" pertencente ao pacote "*javax.ejb*" (Oracle 2016).

Iniciando a execução do método é calculado o número de arquivos a serem verificados. Esse número é um percentual do total de arquivos que estão sendo monitorados no SADN. Este percentual varia de acordo com o nível de confiança atribuído pelo SVI ao SADN conforme descrito na Seção 3.2.2.4. O percentual utilizado é o definido na coluna "% dos arquivos" da Tabela 3.1. O nível de confiança utilizado é o último atribuído ao SADN pelo SVI e que foi armazenado em sua base de dados na tabela "*storage_services*".

Na sequência, são selecionados os arquivos que serão verificados. Os arquivos são selecionados seguindo a ordem cronológica da recepção de sua TabInfor pelo SVI e o seu respectivo registro na tabela "*archives*" conforme descrito na Seção 4.1.2.1. A busca pelos

arquivos parte do último já verificado, ou inicia no primeiro recebido quando for a primeira vez que se verifica os arquivos no SADN. O último arquivo é obtido no campo "*last_archive_id*" da tabela "*storage_services*".

Os arquivos selecionados são processados sequencialmente, sendo que para cada arquivo são executadas as seguintes ações: seleção do ciclo; seleção dos blocos de dados ainda não verificados pertencentes ao ciclo selecionado; geração do desafio; e a atualização do último arquivo verificado.

O processo de seleção do ciclo inicia verificando se já há um ciclo em processo de verificação. Essa informação é obtida a partir do campo "*selectedcycle*" da tabela "*archives*". Caso o campo esteja com valor nulo ou todos os blocos de dados do ciclo obtido já tenham sido verificados, um novo ciclo é escolhido de forma aleatória entre os ciclos não verificados. Esta escolha é realizada utilizando a classe "*SecureRandom*" pertencente ao pacote "*java.security*" (Oracle 2016). Após escolhido o novo ciclo, a tabela "*archives*" é atualizada.

A seleção dos blocos de dados inicia com a definição do número de blocos a serem verificados. O número é obtido na coluna "Blocos" da Tabela 3.1 de acordo com o nível de confiança atribuído ao SADN. Na sequência, os blocos de dados, ainda não verificados e pertencentes ao ciclo selecionado são obtidos a partir da tabela "*blocks*".

O identificador de cada registro lido na tabela "*blocks*" é inserido em um novo registro temporário na tabela denominada "*requests*". A finalidade desta tabela é servir como um repositório dos desafios que aguardam ser respondidos pelo SADN. Na inserção do registro na tabela "*requests*" é gerado um número sequencial que é usado como chave primária da tabela e identificador do desafio.

Na sequência é montada a estrutura de dados que compõem o desafio a ser enviado para o SADN. O desafio é composto: do seu identificador; do conjunto de códigos de endereço das frações que formam o bloco de dados a ser verificado, extraído do campo "*address_codes*" do registro de origem do desafio na tabela "*blocks*"; o tamanho em bytes das frações obtido no campo "*chunk_length*" da tabela "*archives*"; o *hash* identificador do arquivo obtido no campo "*identifier*" da tabela "*archives*"; e o endereço do *web service*

disponibilizado pelo SVI responsável por receber a resposta ao desafio gerada pelo SADN .

Concluída a montagem do desafio, é realizada uma conexão com o *web service* disponibilizado pelo SADN e executado o método denominado "*asyncFileCheck*" recebendo o desafio como parâmetro. Este método é o responsável por receber e armazenar os desafios no SADN conforme previsto pelo protocolo na Seção 3.2.3.2. A implementação do método será descrita na Seção 4.1.3.1.

Caso a execução do método "*asyncFileCheck*" retorne a exceção "*FileNotFoundException*" indicando que o arquivo não existe no SADN, são executados os seguintes procedimentos: o campo "*corrupted*" da tabela "*archives*" é atualizado para o valor "*true*"; o nível de confiança atribuído ao SADN é decrementado seguindo o processo descrito na Seção 4.1.2.4; e, por fim, uma mensagem informando o problema é enviada por e-mail para o proprietário do arquivo.

4.1.2.3 - Recebimento das Respostas aos Desafios

O processo de recebimento das respostas aos desafios foi implementada na forma de um *web service*. O método "*registerResponse*" da classe "*VerifierService*" implementa este processo conforme prevê o protocolo na Seção 3.2.2.3. O método recebe como parâmetro um objeto da classe "*ResponseBean*" contendo o identificador do desafio no SVI, o identificador do desafio no SADN e o *hash* resposta. A Figura 4.14 apresenta a classe "*ResponseBean*".

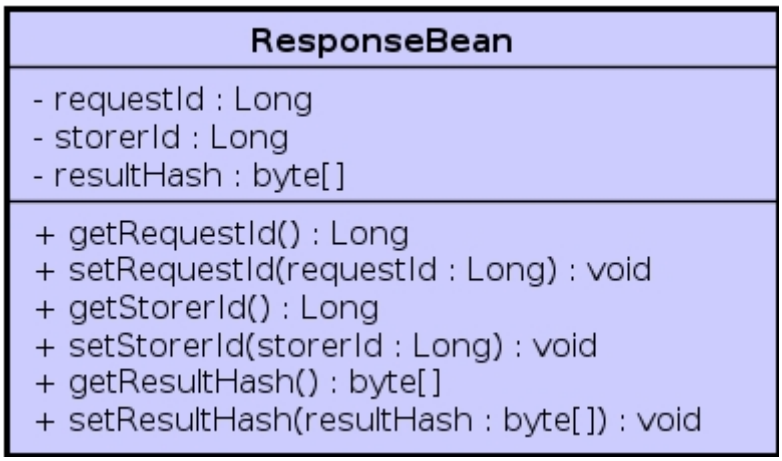


Figura 4.14 - Classe *ResponseBean*.

Quando do recebimento de uma resposta, é verificado na tabela "*requests*" se o desafio continua pendente de resposta. O desafio encontra-se pendente se ainda existir um registro na tabela "*requests*" com a chave primária igual ao identificador do desafio. Caso o registro não seja encontrado a resposta é descartada.

Ao encontrar o registro, o mesmo é bloqueado enquanto é realizada a verificação da resposta e atualização do nível de confiança no SADN. A verificação da resposta consiste da comparação do *hash* resposta com o *hash* armazenado no campo "*hash_code*" do registro que deu origem ao desafio na tabela "*blocks*".

Esta verificação foi implementada na classe "*RequestService*", iniciando pelo método "*validateResponseHash*" que testa a igualdade entre os *hashes* e executa o método "*registerValidCheck*" quando iguais ou executa o método "*generateFailCheck*" quando diferentes.

As ações executadas pelo método "*registerValidCheck*" são: atualizar o campo "*corrupted*" da tabela "*blocks*" com o valor "*false*"; excluir o registro do desafio na tabela "*requests*"; e verificar se todos os blocos de dados do ciclo do desafio já foram verificados e, neste caso, executar o método "*incrementTrustLevel*" responsável por aumentar o nível de confiança no SADN.

As ações executadas pelo método "*generateFailCheck*" são: atualizar o campo "*corrupted*" na tabela "*blocks*" com o valor "*true*"; excluir o registro do desafio na tabela "*requests*";

verificar se o valor do campo "*corrupted*" da tabela "*archive*" é "false" e, neste caso, atualizar para o valor "*true*" e executar o método "*decrementTrustLevel*" responsável por reduzir o nível de confiança no SADN; e enviar uma mensagem ao Cliente informando que o arquivo de sua propriedade está corrompido.

Para garantir que as ações executadas pelos métodos "*registerValidCheck*" e "*generateFailCheck*" sejam realizadas de forma atômica, as mesmas estão inseridas em uma transação que é iniciada e concluída dentro do respectivo método.

4.1.2.4 - Atualização do Nível de Confiança

O nível de confiança no SADN será decrementado sempre que o SVI identificar o primeiro bloco de dados corrompido em um arquivo. Os demais resultados obtidos em um arquivo sinalizado como corrompido, sejam eles positivos ou negativos, serão simplesmente ignorados. Já o incremento ocorre após todos os blocos de dados de um mesmo ciclo terem sido objetos de desafios ao SADN, e que todas as respostas a estes confirmarem a integridade dos blocos verificados.

O processo de atualização do nível de confiança inicia com a execução do método "*incrementTrustLevel*" ou do método "*decrementTrustLevel*", ambos da classe "*RequestService*". Ao serem executados, ambos utilizam a classe "*Trust*" para calcular o novo nível de confiança no SADN e atualizam esse valor no campo "*trust_level*" da tabela "*storage_services*". A classe "*Trust*" é apresentada na Figura 4.15.

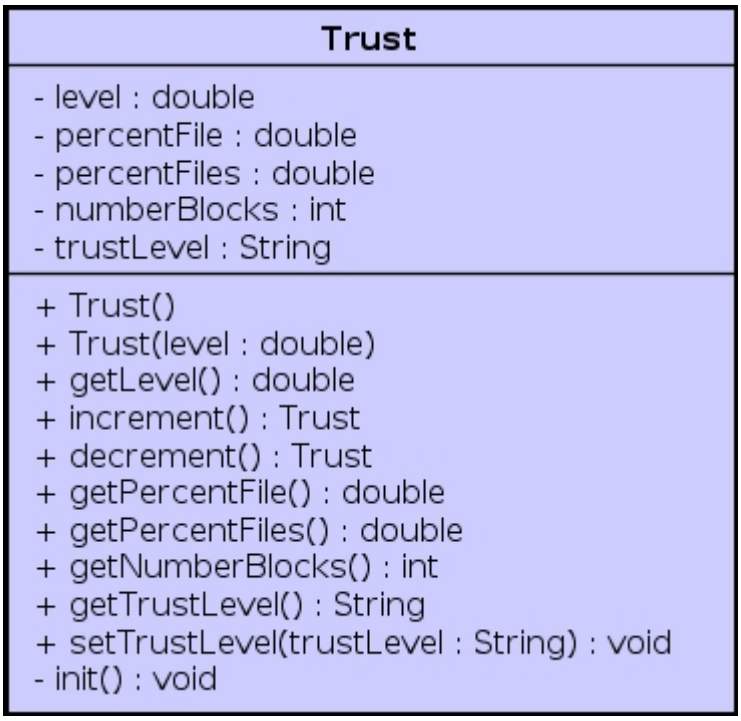


Figura 4.15 - Classe *Trust*.

A classe "*Trust*" implementa o mecanismo de cálculo do valor do nível de confiança por intermédio de dois métodos. Os métodos denominados "*increment*" e "*decrement*" implementam respectivamente os algoritmos descritos pelo protocolo na Seção 3.2.2.4 para incrementar e decrementar o nível de confiança.

4.1.2.5 - Recebimento e Resposta a Solicitação de Informações sobre um Arquivo

O recebimento de solicitações de informações sobre os arquivos monitorados pelo SVI e a geração das respectivas respostas estão implementados no *web service* "*VerifierService*" por meio do método "*getFileStatus*". O método é disponibilizado para os clientes que informam como parâmetro o *hash* identificador do arquivo a se obter as informações e o identificador do contrato que define o SADN que o hospeda.

Após receber uma requisição, o registro do arquivo na tabela "*archives*" é localizado por meio de uma chave composta, formada pelos parâmetros recebidos, o *hash* do arquivo e identificador do contrato. Do registro lido são extraídos os campos "*limit_date*", "*cycles*", "*selected_cycle*" e "*corrupted*" que representam respectivamente: a data final do período em que o arquivo será monitorado pelo SVI; o número total de ciclos pré-calculados pelo

Cliente; o ciclo que está sendo verificado no momento; e o *flag* que indica que o arquivo está ou não corrompido.

Os valores extraídos dos campos são atribuídos respectivamente aos atributos "*limitDate*", "*cycles*", "*selectedCycle*" e "*status*" de um objeto da classe "*FileStatusBean*". Além dos campos descritos, fazem parte dessa classe os atributos "*storer*" e "*checkedBlocks*". A Figura 4.16 apresenta a classe "*FileStatusBean*".

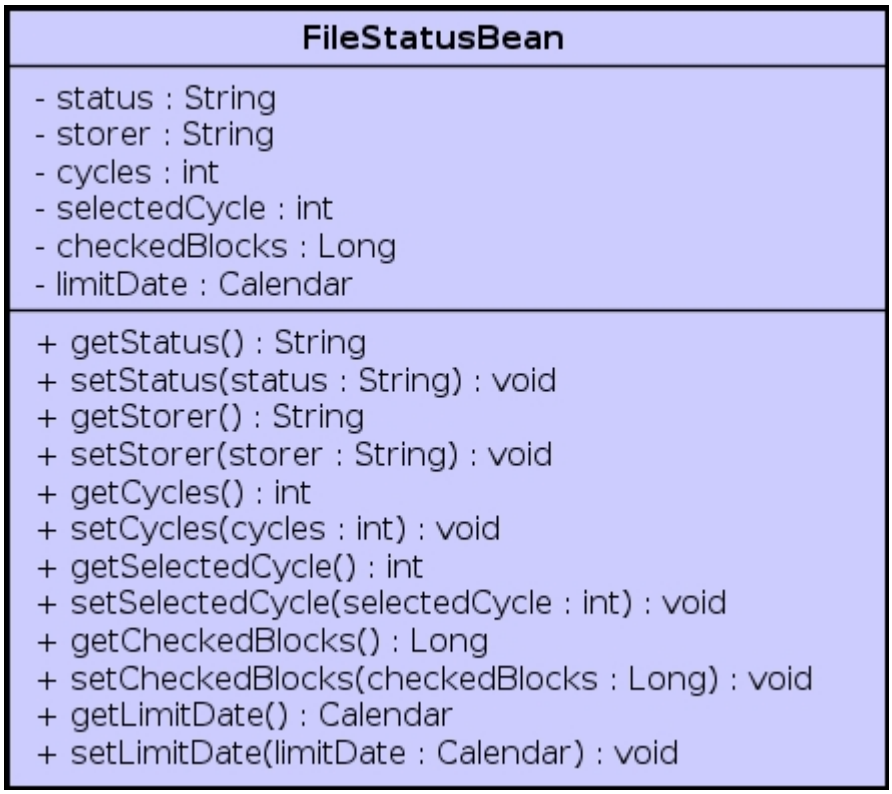


Figura 4.16 - Classe *FileStatusBean*.

O atributo *"checkedBlocks"* é obtido por meio do resultado da contagem, na tabela "*blocks*", dos registros referentes aos blocos de dados do arquivo já verificados no SADN. Já o atributo "*storer*" recebe o nome do SADN que hospeda o arquivo, informação esta obtida a partir da tabela "*StorageService*". Concluída a montagem, o objeto é enviado de forma síncrona como resposta a requisição.

4.1.2.6 - Recebimento e Resposta a Solicitação de Informações sobre Serviços de Armazenamento de Dados na Nuvem Contratados

O recebimento e geração das respostas as solicitações de informações sobre os SADNs para os quais o Cliente mantém no SVI um contrato para o monitoramento de seus arquivos e a geração das respectivas respostas foram implementados através do método "*getListContract*" também pertencente ao *web service* "*VerifierService*" disponibilizado pelo SVI. O método recebe como parâmetro o identificador do Cliente no SVI, informação fornecida pelo SVI ao Cliente após a contratação do serviço.

As informações sobre os contratos firmados entre o Cliente e o SVI são obtidos na tabela "*contracts*". Este por sua vez possui como chave estrangeira em seus registros o identificador do SADN, cujo os dados cadastrais são armazenados na tabela "*storage_services*". Após ler todos os registros dos contratos ativos com o Cliente, é montado um *array* contendo "n" objetos da classe "*ContractBean*". A Figura 4.17 apresenta a classe "*ContractBean".*

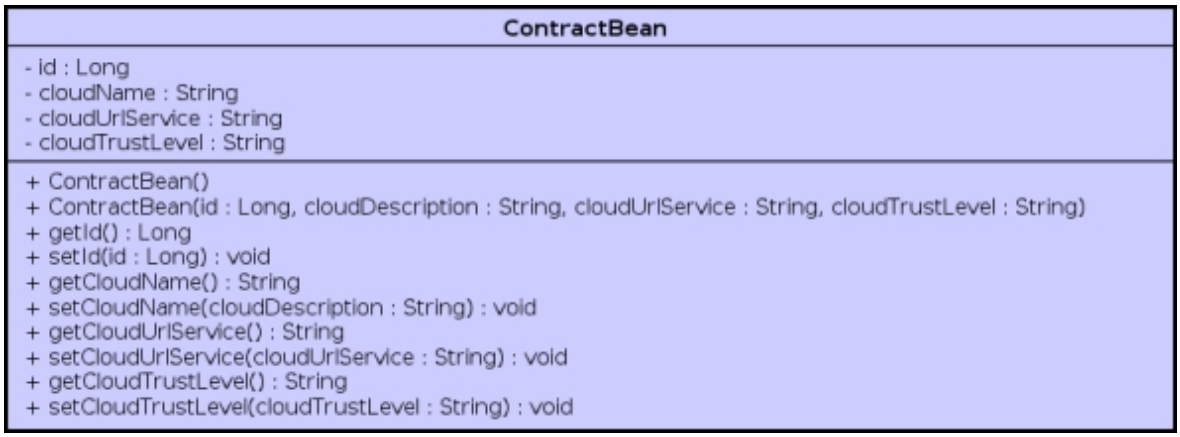


Figura 4.17 - Classe *ContractBean*.

Cada objeto "*ContractBean*" contém as informações de um SADN contratado pelo Cliente e a sua estrutura de dados é composta pelos atributos "*id*", "*cloudName*", "*cloudUrlService*" e "*cloudTrustLevel*". Estes atributos recebem respectivamente os seguintes valores: a partir da tabela "*contracts*", o identificador do contrato; e a partir da tabela "*storage_services*", o nome do SADN, o endereço do serviço no SADN responsável por receber a submissão dos arquivos e o nível de confiança atual atribuído pelo SVI ao

SADN. Concluída a montagem do *array*, este é enviado de forma síncrona como resposta ao Cliente.

4.1.3 - Aplicação do Serviço de Armazenamento de Dados na Nuvem

Para atender aos requisitos previstos no protocolo descritos na Seção 3.2.3.2, o Serviço de Armazenamento de Dados na Nuvem (SADN) necessita disponibilizar um serviço que seja capaz de receber desafios, processá-los e respondê-los com os resultados obtidos. Esse serviço é disponibilizado para o cliente na forma de um *web service* denominado "*StorerWebService*". A classe "*StorerWebService*" é apresentada na Figura 4.18.

Figura 4.18 - Classe *StorerWebService*.

A implementação completa da aplicação para o SADN contempla as seguintes funcionalidades: receber os desafios oriundos do SVI e armazená-los na base de dados do SADN; monitorar a base de dados e processar os desafios pendentes e armazenar as respostas na base de dados; e monitorar a base de dados e enviar as respostas pendentes para o SVI. Além das funcionalidades relacionadas ao protocolo, a aplicação contempla ainda as funcionalidades para o *upload* e *download* de arquivos permitindo simular as funcionalidades disponíveis nos SADN.

4.1.3.1 - Recebimento de Desafios

O recebimento de desafios é disponibilizado aos SVIs através do método "*asyncFileCheck*" implementado como um *web services* na classe "*StorerWebService*". O método recebe como parâmetro um objeto da classe denominada "*ChallengeBean*".

O objeto da classe "*ChallengeBean*" é composto pelos atributos "*identifier*", "*addressCodes*", "*chunkLength*", "*responseUrl*" e "*id*" que representam respectivamente: o *hash* identificador do arquivo; o *array* com o conjunto de códigos de endereços das frações a serem lidas no arquivo e que irão compor o bloco de dados sobre o qual será gerado o *hash* resposta; o tamanho em bytes de cada fração; a URL do *web service* do SVI responsável por receber a resposta aos desafios; e o identificador do desafio. A classe "*ChallengeBean*" é apresentada na Figura 4.19.

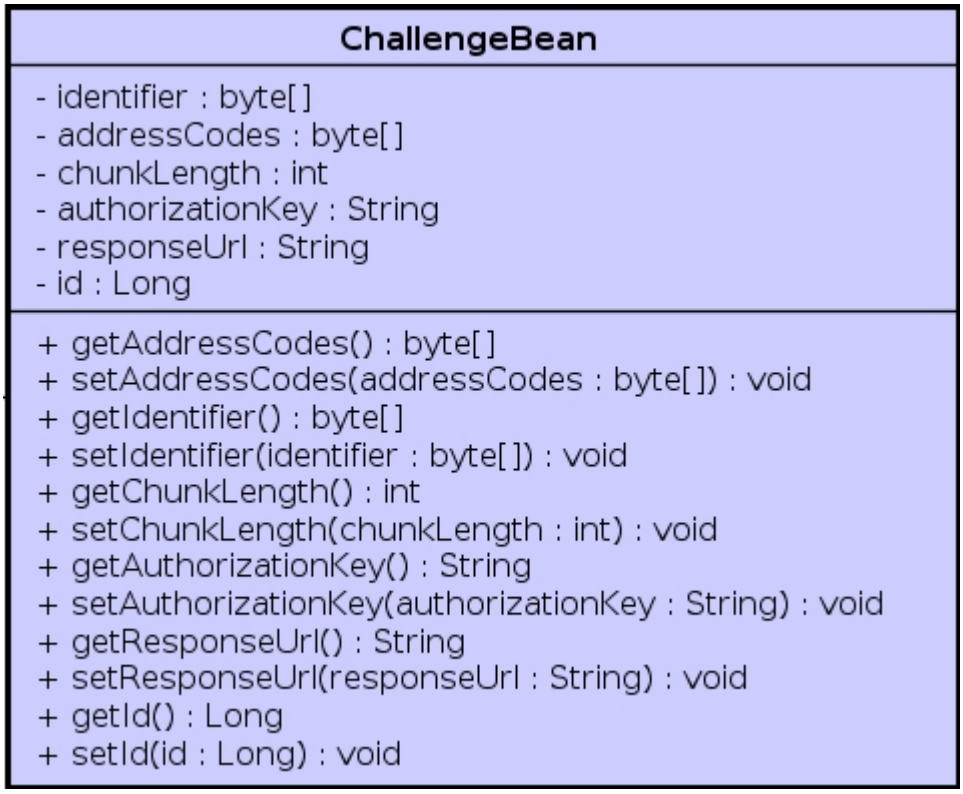


Figura 4.19 - Classe *ChallengeBean*.

Ao receber uma requisição com um desafio, é verificado se o arquivo ao qual o mesmo faz referência existe na infraestrutura de arquivos do SADN. No caso do arquivo não ser encontrado, a operação é abortada e o SVI recebe uma exceção denominada "*FileNotFoundException*". O tratamento dado a essa exceção pelo SVI é descrito na Seção 4.1.2.2.

Após o desafio recebido ser validado, as informações sobre o desafio são extraídas do objeto *ChallengeBean* recebido e armazenadas na base de dados do SADN em uma tabela denominada "*verification_requests*", atribuindo-lhe no seu campo "status" o valor

"Aguardando Processamento". Na Figura 4.20 é apresentada a tabela "*verification_requests*".

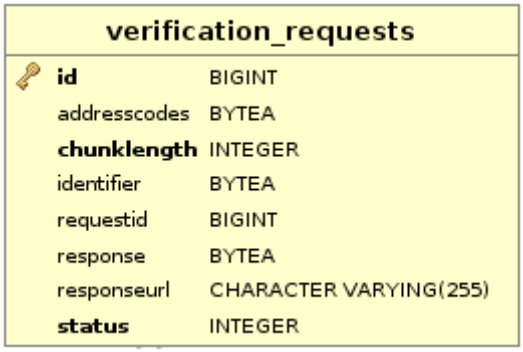


Figura 4.20 - Tabela *verification_requests*.

Ao salvar o desafio na referida tabela é gerado um identificador único para o registro que é atribuído ao campo "*id*". Na sequência, o identificador gerado é utilizado para gerar o evento "*RequestEvent*" e, concluindo a execução do método, responder ao SVI indicando que o desafio foi recebido e armazenado com sucesso.

A vantagem de armazenar os desafios recebidos para posterior processamento está na economia dos recursos computacionais do SVI. Este procedimento evita que o SVI necessite manter um processo parado, aguardando uma resposta do SADN, para cada desafio submetido. O tempo necessário para o processamento de um desafio varia de acordo o tamanho do arquivo a ser verificado, a quantidade de desafios recebidos simultaneamente e a capacidade computacional do SADN.

Outra possibilidade gerada por esse modelo é a distribuição de carga, pois os serviços responsáveis por receber, processar e responder os desafios podem ser providos por infraestruturas de *hardware* e *software* totalmente diferentes. A única exigência é que estas compartilhem acesso a uma mesma base de dados.

4.1.3.2 - Monitoramento e Processamento de Desafios Pendentes

O processamento dos desafios pendentes é realizado de duas formas, a primeira pela captura do evento "*RequestEvent*" e a segunda pelo processo de monitoramento de desafios

pendentes. O mecanismo de monitoramento foi implementado para garantir que mesmo ocorrendo uma interrupção na execução do servidor de aplicação, sem que a fila de eventos seja totalmente processada, esta possa ser recuperada no futuro.

O servidor de aplicação captura o evento "*RequestEvent*" e executa o método "*processEventRequest*" na classe "*CheckerHandler*", definido para essa função através da anotação "*@Observes*". Ao ser executado, o método busca na tabela "*verification_requests"* o registro com o identificador recebido como parâmetro no evento, e caso o mesmo ainda esteja com o valor "aguardando processamento" no seu campo "status", o método "*processRequest*" é assincronamente executado. A Figura 4.21 apresenta a classe "*CheckerHandler*".

Figura 4.21 - Classe *CheckerHandler*.

O processo de monitoramento dos desafios pendentes é realizado por intermédio do método "*detectRequests*" da classe "*CheckerHandler*". Através da anotação "*@Schedule*" este método é executado automaticamente pelo servidor de aplicação a cada 5 segundos. O método realiza a leitura dos registros existentes na tabela "*verification_requests*" com o valor do campo "*status*" igual a "aguardando processamento". Para cada registro encontrado é executado assincronamente o método "*processRequest*".

Ao ser executado, o método "*processRequest*" recebe como parâmetro o registro na tabela "*verification_requests*" através de um objeto da classe "*VerificationRequest*" denominado "*verificationRequest*". O comportamento assíncrono na execução do método é obtido através da anotação "*@Asynchronous*" na sua definição.

O método obtém os dados do desafio através da execução do método "*getChallenge*" a partir do objeto "*verificationRequest*" que retorna um objeto da classe "*ChallengeBean*" e

o repassa como parâmetro para o método "*blockCheck*". Este último método, por sua vez, se encarrega de extrair as frações de dados do arquivo, montar o bloco de dados, gerar o *hash* e retorná-lo como resposta. A classe "*VerificationRequest*" é apresentada na Figura 4.22.

Figura 4.22 - Classe *VerificationRequest*.

Na sequência, encerrando a execução do método, o registro na tabela "*verification_requests*" é atualizado alterando o valor do campo "*status*" para "processado" e atribuindo o *hash* resposta para o campo "*response*". Encerrando a execução do método é gerado um evento "*ResponseEvent*", o qual recebe como parâmetro o identificador do registro recém-atualizado.

4.1.3.3 - Geração do *Hash* Resposta

O método "*blockCheck*" recebe as informações sobre o desafio através de um objeto da classe "*ChallengeBean*" denominado "*challenge*", recebido como parâmetro. A execução do método inicia-se com o acesso ao arquivo objeto da verificação. Conforme descrito na Seção 4.1.3.6, nesta implementação, o *hash* identificador do arquivo é utilizado como nome do arquivo e o mesmo é obtido no atributo "*identifier*" do objeto "*challenge*".

Essa ação é realizada com o apoio da classe "*RandomAccessFile*" do pacote "*java.io*" (Oracle 2016). Esta classe permite o acesso direto as frações do arquivo por intermédio de um endereço de início e o tamanho em bytes da área a ser recuperada.

O bloco de dados é montado pela leitura das 16 frações do arquivo conforme definido no atributo "*addressCodes*" do objeto "*challenge*". Este atributo contém um *array* com a sequência dos 16 códigos de endereços das frações que compõem o bloco. Os códigos de endereço são números inteiros que variam entre 0 e 4095 conforme descrito no protocolo na Seção 3.2.1.3. O endereço de início de cada fração é obtido pela multiplicação do código de endereço pelo tamanho da fração obtida pela divisão do tamanho do arquivo pelo número total de frações por arquivo (4096).

Após a conclusão da leitura das frações, os dados obtidos são concatenados formando um bloco de dados. Utilizando um objeto da classe "*Blake2b*", que implementa a função criptográfica de *hash* Blake2 (Aumasson et al. 2013), é gerado a partir do referido bloco um *hash* com tamanho igual a 256 bits, que é retornado como resposta ao processo que deu origem a execução do método "*blockCheck*".

4.1.3.4 - Monitoramento e Envio de Respostas Pendentes

O envio de respostas pendentes ocorre de duas maneiras, a primeira pela captura do evento "*ResponseEvent*" e a segunda pelo processo de monitoramento de respostas pendentes. Semelhante ao modelo utilizado no monitoramento de desafios pendentes descrito na Seção 4.1.3.2, o mecanismo de monitoramento das respostas pendentes foi implementado

para garantir que mesmo ocorrendo algum problema no processamento da fila de eventos, a resposta pendente seja enviada na primeira oportunidade ao SVI que originou o desafio.

Ao ser capturado o evento "*ResponseEvent*" pelo servidor de aplicação, este executa o método "*processEventResponse*" da classe "*AnswerHandler*", definido para essa função através da anotação "*@Observes*". Ao ser executado, o método busca na tabela "*verification_requests"* o registro com o identificador recebido como parâmetro no evento, e caso o mesmo ainda esteja com o valor "processado" no seu campo "*status*", o método "*processResponse*" é executado. A classe "*AnswerHandler*" é apresentada na Figura 4.23.

Figura 4.23 - Classe *AnswerHandler*.

O processo de monitoramento das respostas pendentes é realizado por intermédio do método "*detectResponses"* da classe "*AnswerHandler*". Através da anotação "*@Schedule*" este método é executado automaticamente pelo servidor de aplicação a cada 30 minutos. O método realiza a leitura dos registros existentes na tabela "*verification_requests*" com o valor do campo "*status*" igual a "processado". Para cada registro encontrado é executado assincronamente o método "*processResponse*".

Ao ser executado, o método "*processResponse*" recebe como parâmetro o registro na tabela "*verification_requests*" através de um objeto da classe "*VerificationRequest*" denominado "*verificationRequest*". A partir desse objeto é executado o método "*getResponseUrl()*" que obtém a *URL* do serviço de recebimento de respostas disponibilizado pelo SVI conforme descrito na Seção 4.1.2.3.

Utilizando a *URL* obtida é realizada uma tentativa de conexão com o SVI. Caso a conexão seja realizada com sucesso é criada uma instância (objeto) da classe "*ResponseBean*". A classe possui os atributos "*id*", "*requestId*" e "*response*" os quais recebem os seguintes

valores respectivamente: o identificador do registro do desafio na tabela "*verification_requests*"; o identificador do desafio no SVI; e o *hash* resposta.

Na sequência, o objeto da classe "*ResponseBean*" recém-criado é enviado para o SVI por intermédio do método "*registerResponse*". Caso o procedimento de envio seja concluído com sucesso, o campo "*status*" do registro do desafio na tabela "*verification_requests*" é atualizado para o valor "resposta enviada".

4.1.3.5 - Recebimento de Solicitações de Resposta a Desafios Pendentes

Para possibilitar ao SVI requisitar a resposta de um desafio enviado anteriormente, foi disponibilizado no *web service* "*StorerWebService*" um método denominado "*getFileCheckResponse*". Este serviço foi desenvolvido com o objetivo de permitir ao SVI manter um monitoramento dos desafios enviados e no caso do tempo para o recebimento da resposta ultrapassar o esperado, o mesmo pode solicitar diretamente ao SADN a resposta a um desafio específico.

O método recebe como parâmetro o identificador do desafio gerado no SVI, por meio do qual é buscado na tabela "*verification_requests*" o seu registro no SADN. Se o registro não for encontrado é gerada uma exceção denominada "*RequestNotFoundException*", se o registro for encontrado mas o valor do campo "status" for igual a "aguardando processamento" é gerada a exceção "*AwaitingProcessException*". Nos demais casos, ou seja, quando o registro é encontrado e o seu campo "*status*" possui o valor "processado" ou "resposta enviada", o conteúdo do campo "response" é retornado como resposta.

4.1.3.6 - Recebimento e Envio de Arquivos

O recebimento e armazenamento de arquivos no sistema de arquivos do SADN e o envio de arquivos de volta para o seu proprietário são implementados no *web service* "*StorerWebService*" respectivamente pelos métodos "*fileUpload*" e "*fileDownload*". Estas funcionalidades, apesar de serem funcionalidades já disponíveis nos serviços de nuvem, foram desenvolvidas para possibilitar a simulação destas funcionalidades em outros ambientes durante o processo de implementação da arquitetura e análise da eficácia do

protocolo proposto. Não é pretensão dessa implementação substituir as funcionalidades originalmente disponíveis nos SADN.

Ambos os métodos foram implementados utilizando a anotação "*@XmlMimeType*" com o parâmetro "*application/octet-stream*" para permitir a transferências de arquivos com tamanhos superiores a 1 GB. Os arquivos são transferidos como objetos da classe "*javax.activation.DataHandler*". O Objeto é recebido como um parâmetro no método "*fileUpdate*" e enviado como resposta a execução do método "*fileDownload*".

O método "*fileUpdate*" recebe dois parâmetros, o primeiro é o *hash* identificador do arquivo e o segundo é o objeto utilizado para realizar a transferência do arquivo. Iniciando a execução do método, um novo arquivo é criado na estrutura de arquivos do SADN atribuindo-lhe como nome o *hash* identificador. Na sequência, o objeto recebido é convertido de "*DataHandler*" para um objeto da classe "*StreamingDataHandler"* pertencente ao pacote "*com.sun.xml.ws.developer*" (Oracle 2016a). Encerrando a execução do método, o objeto "*StreamingDataHandler*" através do método "*moveTo*" transfere os seus dados para o arquivo recém criado.

O método "*fileDownload*" recebe apenas um parâmetro que é o *hash* identificador do arquivo, a partir do qual, após convertido de binário para hexadecimal é utilizado como o nome do arquivo a ser localizado na estrutura de arquivos do SADN. Após localizado, o arquivo é carregado como um objeto da classe "*FileDataSource*" pertencente ao pacote "*javax.activation*" (Oracle 2016). Este objeto por sua vez é usado como parâmetro na criação de um objeto da classe "*DataHandler*", o qual é retornado como resposta ao Cliente.

4.2 - SÍNTESE DO CAPÍTULO

O objetivo deste capítulo foi apresentar a proposta de um protocolo que possibilita o monitoramento permanente da integridade de arquivos armazenados na nuvem, descrevendo passo a passo o seu funcionamento. Além disso, este capítulo apresentou a descrição completa da implementação da arquitetura abrangendo todos os papéis previstos no protocolo proposto.

5 - SIMULAÇÕES E RESULTADOS

Este capítulo apresenta um conjunto de testes e simulações executados para verificar a efetividade do protocolo proposto e a eficácia arquitetura implementada. Além disso, apresenta uma análise dos resultados obtidos.

5.1 - ESPECIFICAÇÕES DO AMBIENTE DE TESTES

O ambiente utilizado para a realização dos testes e simulações foi composto de cinco máquinas virtuais (VMs) iguais, sendo que cada uma possuía as seguintes configurações: 12 GB de memória RAM; 200 GB de espaço em disco; e o sistema operacional *Linux* usando a distribuição *Ubuntu Server* 14.04.

Cada máquina foi configurada para exercer a função de apenas um dos papéis previstos no protocolo. A distribuição dos serviços nas VMs foi realizado da seguinte maneira: uma VM no papel de Cliente, uma VM no papel de SVI e três VMs para o papel de SADN.

5.2 - TESTES E SIMULAÇÕES REALIZADAS

Os testes foram realizados utilizando 4 arquivos com tamanhos de aproximadamente 2,5, 5, 10 e 15 GB cada. Para cada arquivo foram geradas tabelas de informações para o seu armazenamento e monitoramento pelos períodos de 1, 5, 10, 20 e 30 anos. Foram utilizados arquivos com diferentes conteúdos e formatos como ISO, XVA e MKV. Para cada período de tempo, foram utilizadas chaves criptográficas de mesmo tamanho mas com valores aleatórios, fazendo com que os arquivos cifrados gerados fossem completamente diferentes.

Durante um período de aproximadamente 03 meses, foram armazenados os *logs* do monitoramento realizado pelo SVI nos arquivos armazenados nos 03 SADNs. Ao todo foram monitorados pelo SVI 60 arquivos, sendo 20 em cada SADN. Durante o período de realização dos testes foram aleatoriamente escolhidos servidores que foram desligados e religados posteriormente para simular situações de falhas.

Foram ainda realizadas simulações para avaliar o comportamento efetivo do protocolo. Dentre as simulações realizadas destacam-se a alteração proposital de conteúdo do arquivo e a alteração dos níveis de confiança atribuídos aos SADNs.

5.2.1 - Submissão de Arquivos

Nessa seção são descritos os resultados obtidos no monitoramento dos processos de preparação e submissão de arquivos para o SADN. Conforme previsto no protocolo e descrito na Seção 3.2.1.3, para cada arquivo enviado para o SADN é gerada e enviada uma TabInfor para o SVI.

No processo de preparação dos 20 arquivos submetidos para cada SADN foram armazenados arquivos textos contendo *logs* com informações sobre os tempos gastos em cada uma de suas etapas. As etapas cujos tempos foram monitorados são: cifragem do arquivo, geração do *hash* do arquivo, cálculo dos ciclos com a distribuição das frações nos blocos de dados, geração dos *hashes* dos blocos de dados, e o envio e armazenamento da TabInfor no SVI.

Nas etapas avaliadas, os tempos de execução de uma mesma etapa variam de acordo com o tamanho do arquivo processado ou com o tempo previsto para o seu armazenamento na nuvem. A Figura 5.1 apresenta a comparação entre os tempos médios de execução das etapas de cifragem e da geração de *hash* dos arquivos agrupados por tamanho.

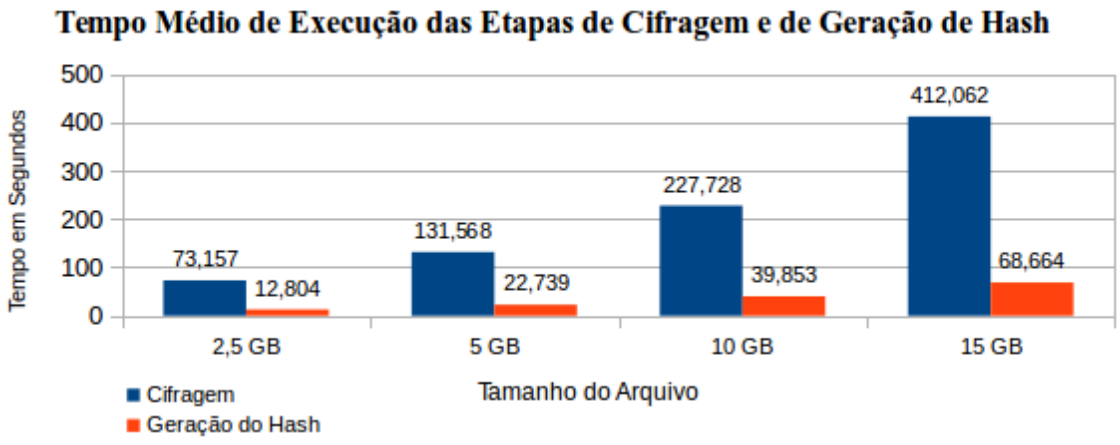


Figura 5.1 - Tempo de cifragem e de geração de *hash* por tamanho de arquivo.

A etapa de cálculo dos ciclos é a responsável pela seleção dos códigos de endereço das frações que compõem cada bloco de dados conforme implementação descrita na Seção 4.1.1.4. Esta etapa é a que gera o menor consumo de tempo dentro do conjunto de etapas analisadas. O seu tempo de execução varia exclusivamente de acordo com o período de armazenamento do arquivo. A Figura 5.2 apresenta o tempo médio de execução da etapa de cálculo dos ciclos com a distribuição das frações nos blocos de dados.

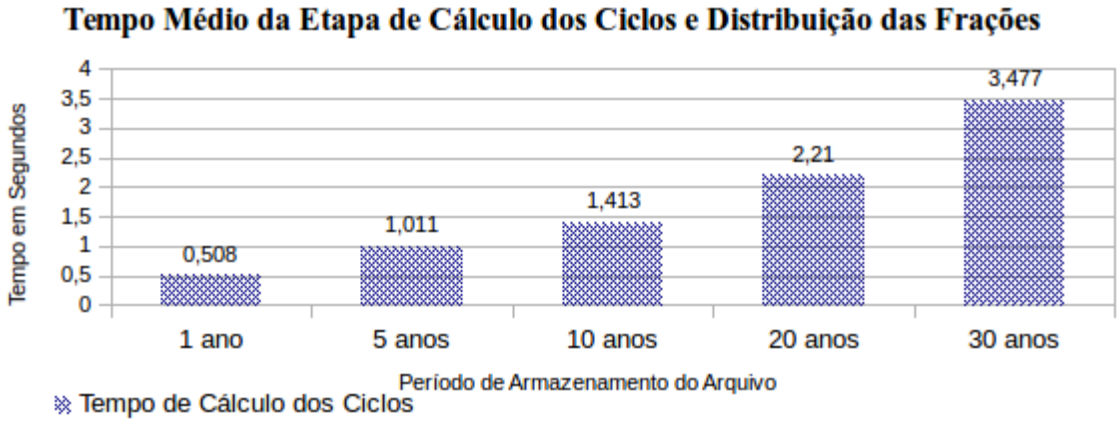


Figura 5.2 - Tempo médio da etapa de cálculo dos ciclos e distribuição das frações.

A etapa de geração dos *hashes* dos blocos de dados integra as ações de leitura randômica das frações no arquivo cifrado, montagem dos blocos de dados a partir da concatenação das frações lidas e a geração do *hash* sobre cada bloco montado. Nesse caso, os tempos de execução variam tanto em relação ao tamanho do arquivo, quanto em relação ao período de armazenamento na nuvem.

Quanto maior o arquivo, maior será o tamanho de cada bloco de dados, e quanto maior o período de armazenamento, maior a quantidade de blocos de dados a serem gerados. A Figura 5.3 apresenta o gráfico com a variação do tempo necessário para a geração dos *hashes* dos blocos de dados, conforme o tamanho do arquivo e o período de armazenamento.

Por meio deste gráfico foi possível verificar que os tempos de geração dos *hashs* para o arquivo de 15 Gb, independentemente do tempo previsto para o seu armazenamento na nuvem, cresceu de forma desproporcional aos demais tamanhos de arquivo. A partir dessa observação, é possível inferir que, para arquivos iguais ou superiores a 15 Gb, existe a

possibilidade de aprimorar as configurações do protocolo, como por exemplo, o número de frações por arquivo, de forma a obter uma melhor performance.

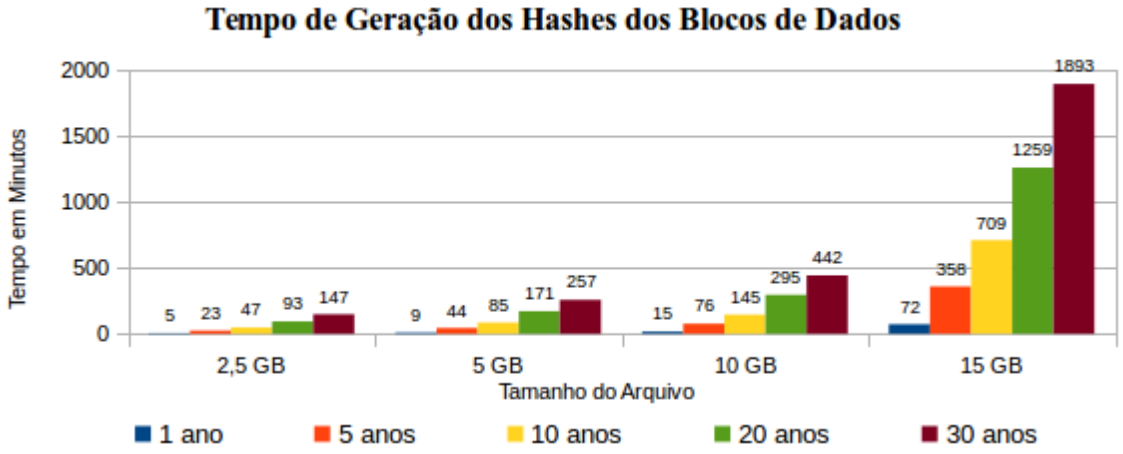


Figura 5.3 - Tempo de execução da etapa de geração dos *hashes* dos blocos de dados.

Outro importante componente no tempo total de execução do processo de submissão de um arquivo é o tempo de execução da etapa de envio da TabInfor para o SVI. Este tempo varia de acordo com a quantidade de blocos de dados gerados, que por sua vez, varia de acordo com o período de armazenamento do arquivo na nuvem.

O tempo registrado nesta etapa compreende as ações de conexão do *web service* no SVI, envio do objeto contendo a TabInfor, o armazenamento do conteúdo da TabInfor na base de dados do SVI e o recebimento da resposta conformando o recebimento e armazenamento com sucesso. A Figura 5.4 apresenta o tempo médio de execução da etapa de envio da TabInfor para o SVI conforme o período de armazenamento.

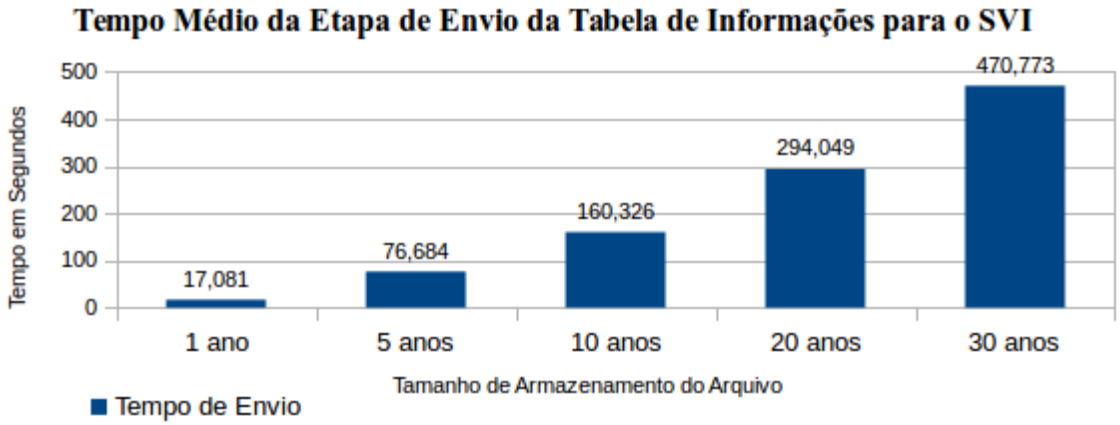


Figura 5.4 - Tempo médio de execução da etapa de envio da TabInfor para o SVI.

5.2.2 - Consumo de Banda de Rede

Nessa seção são descritas as simulações realizadas para determinar o consumo efetivo de banda de rede gerado pela execução do processo de monitoramento dos arquivos. O protocolo prevê uma variação no consumo de recursos de acordo com o nível de confiança atribuído ao SADN.

Para isso, foram atribuídos a um SADN todos os níveis de confiança previstos no protocolo. A cada nível, foi executado no SVI o processo diário de verificação dos arquivos no referido SADN. O tráfego de pacotes gerado pelo envio de desafios e recebimento das respostas foi capturado com o auxílio da ferramenta *Wireshark*.

O ambiente controlado utilizado, assim como os filtros aplicados no *Wireshark*, possibilitaram a captura única e exclusivamente dos pacotes gerados pelas aplicações SVI e SADN envolvidas. A Figura 5.5 apresenta o valor médio diário de consumo de banda de rede gerado pelo protocolo conforme o nível de confiança atribuído ao SADN.

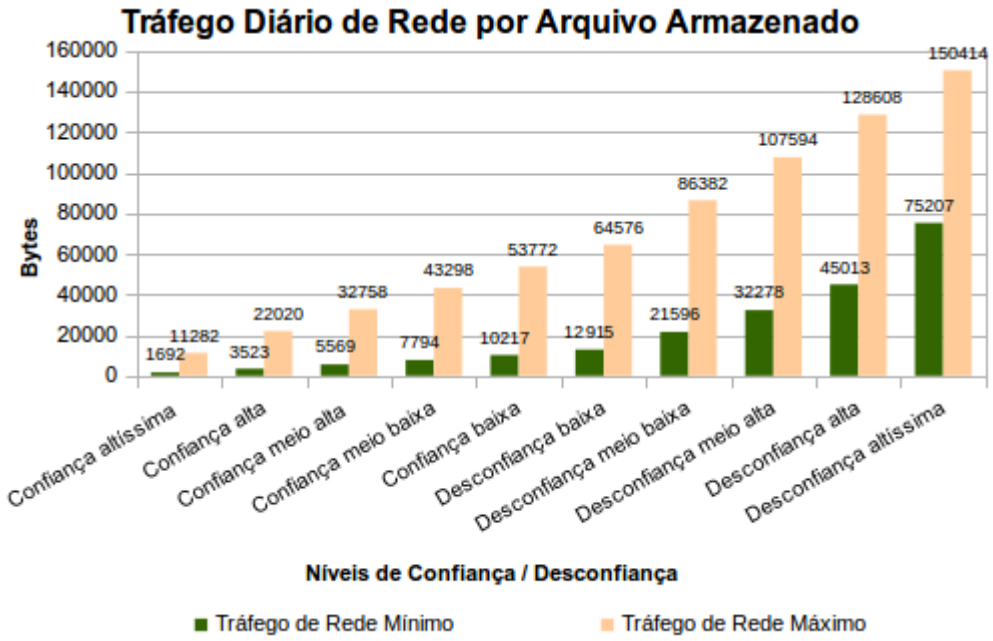


Figura 5.5 - Tráfego médio diário de rede por arquivo armazenado

O tráfego de rede máximo ocorre quando o número de arquivos armazenados no SADN é igual a 1. Nesse caso, pelo menos 1 dos blocos de dados do arquivo será verificado, independentemente do nível de confiança atribuído ao SADN. O tráfego de rede mínimo ocorre quando o valor resultante do cálculo do percentual previsto na coluna "% dos Arquivos" da Tabela 3.1, de acordo o nível de confiança sobre o número de arquivos armazenados no SADN, for um inteiro.

5.2.3 - Verificação da Integridade dos Arquivos

Nessa seção são descritos os resultados obtidos durante o monitoramento do processo de verificação da integridade dos arquivos, realizados durante aproximadamente 2 meses envolvendo um SVI e 3 SADNs. Durante esse período, as aplicações SVI e SADN registraram, em arquivos-texto, *logs* contendo o registro dos tempos gastos na execução das ações relacionadas ao processo de verificação da integridade.

Com o objetivo de não afetar a performance da aplicação e por consequência prejudicar a análise da eficiência do protocolo, o registro das ações realizadas nas aplicações para posterior análise foi realizada utilizando o sistema de registro de *log* do servidor de aplicação. Os registros de *log* gerados são formados por uma quantidade mínima de dados,

gerados apenas com as informações já disponíveis no momento da execução da ação registrada.

Para transformar os dados registrados nos *logs* em informações úteis, inicialmente foi necessário realizar uma filtragem para extrair apenas os registros gerados pelas aplicações SVI e SADN, que fossem de interesse para a análise dos resultados. A partir do resultado da filtragem, foi realizada uma importação dos registros em texto puro, oriundos do *log* para o formato de banco de dados e inserindo-os nas bases de dados dos respectivos serviços. O procedimento de importação foi realizado com apoio da ferramenta "*Spoon*" que é parte integrante do conjunto de ferramentas que compõem o *software* "*Pentaho*".

A transformação dos *logs* em registros nas respectivas bases de dados das aplicações SVI e SADN foi necessária para permitir relacionar o registro no *log* com o desafio que lhe deu origem. A verificação da integridade dos arquivos é realizada por intermédio de desafios enviados pelo SVI ao SADN, conforme descrito no protocolo na Seção 3.2.2.2, e que o tempo de resposta a cada desafio é diretamente proporcional ao tamanho do arquivo verificado. Assim, as informações sobre os resultados dos desafios obtidas a partir dos *logs,* foram agrupadas e classificadas de acordo com o tamanho dos arquivos que verificaram.

A partir do SVI é obtido o tempo total consumido ao realizar o processo de verificação de um arquivo através de um desafio. O valor registrado nos *logs* do SVI contemplam as ações realizadas entre o registro do desafio na tabela "*requests*", descrita na Seção 4.1.2.2, até o recebimento da resposta enviada pelo SADN. No SVI foram obtidos no *log* 1340 registros de verificações realizadas com sucesso. A Figura 5.6 apresenta os tempos médio, máximo e mínimo consumidos no processamento de um desafio por tamanho de arquivo verificado.

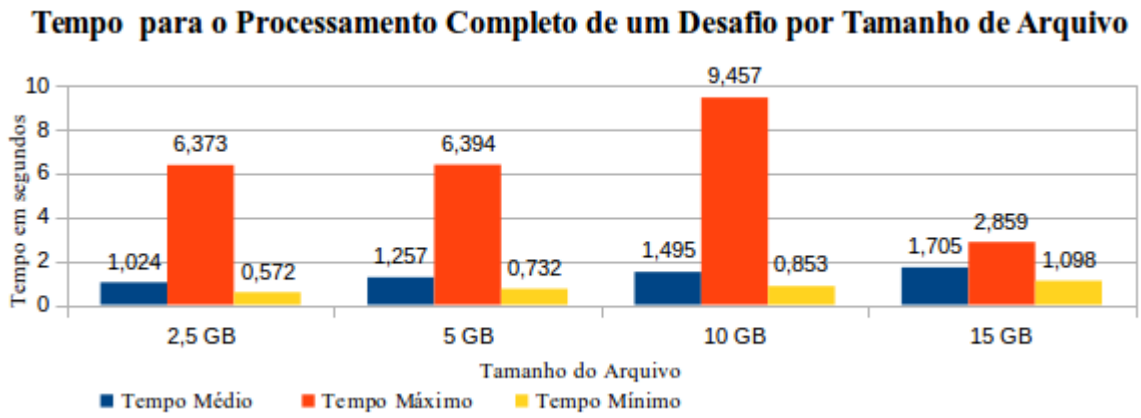


Figura 5.6 - Tempos de processamento de um desafio por tamanho de arquivo.

As ações que demandam maior quantidade de tempo no processamento de um desafio são realizadas no SADN e compreendem a remontagem do bloco de dados a partir da leitura de suas frações seguida da geração do seu *hash.* A partir do ambiente de teste composto de 3 SADNs com as mesmas características foram coletados nos *logs* dos SADNs 1, 2 e 3 respectivamente 360, 470 e 510 registros, totalizando 1340 registros, com os tempos consumidos na execução das referidas ações.

Apesar da mesma quantidade de arquivos ser enviada para todos os SADNs, o número de desafios processados por cada um variou devido ao fato que os mesmos foram desligados em momentos diversos e assim permaneceram por tempos variados. A Figura 5.7 apresenta a comparação entre os tempos médios consumidos por cada SADN na geração das respostas aos desafios recebidos do SVI.

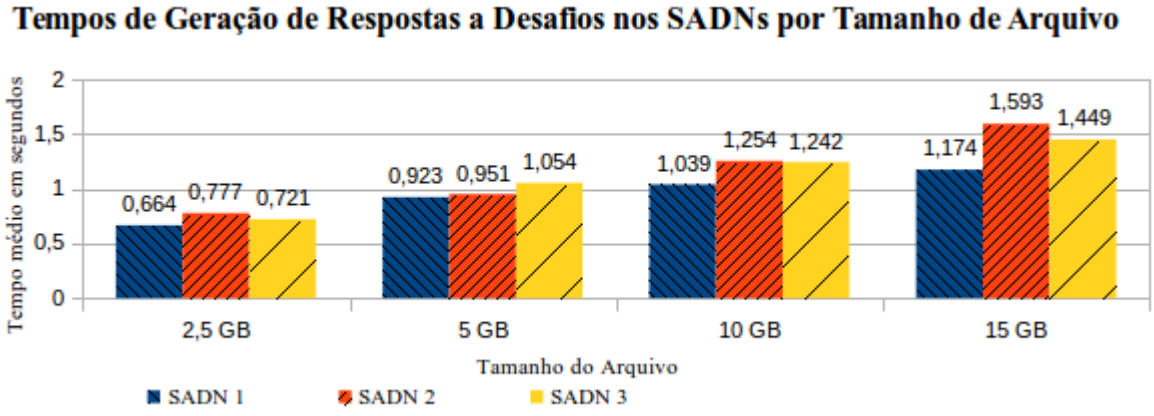


Figura 5.7 - Tempos de geração de respostas a desafios nos SADNs por tamanho.

5.2.4 - Simulação de Falhas

Nessa seção são descritos os resultados obtidos durante a simulação de falhas na integridade dos arquivos armazenados nos SADN. Para esse fim foram alterados bytes no interior dos arquivos armazenados em diferentes SADN mas com o mesmo tamanho. O objetivo desta simulação foi determinar o tempo médio necessário para que o SVI identificasse a falha.

O tamanho dos arquivos escolhidos para as simulações foi de 5 GB, sendo que, dos 15 arquivos deste tamanho existentes distribuídos nos 3 SADNs, 10 foram selecionados e distribuídos de forma aleatória em 2 grupos compostos de 5 arquivos cada. Cada grupo foi utilizado para a simulação de uma falha com diferentes características. A Tabela 5.1 apresenta as características da falha simulada e os resultados obtidos.

Tabela 5.1 - Falhas simuladas

Bytes alterados	% do Arquivo	Local da alteração	Frações afetadas	Dias para detectar a falha					
				A1	A2	A3	A4	A5	Média
4688	0.000085%	Escolha aleatória	1 ou 2	10	9	22	12	15	~14
55006658	~1%	Fim do arquivo	41	3	7	6	6	3	5

Os 5 arquivos de cada grupo foram identificados na tabela com as siglas A1, A2, A3, A4 e A5. Na simulação onde a escolha do local da alteração foi aleatória, a posição da alteração em cada um dos cinco arquivos foram diferentes, no entanto o conteúdo utilizado nas alterações de cada grupo de arquivos foram exatamente os mesmos. A coluna “Média” apresenta o tempo médio em número de dias que o SVI demorou para detectar que os arquivos monitorados estavam corrompidos.

5.2.5 - Resistência a Falhas e Fraudes

Nessa seção são apresentados os mecanismos de resistência a falhas e fraudes presentes na especificação do protocolo e na implementação da arquitetura. Na definição das regras do protocolo um dos principais requisitos considerados foi a resistência a fraudes.

Essa resistência é obtida por meio de um algoritmo de seleção e permutação que atribui a entropia necessária para tornar impraticável qualquer tentativa de prever quais frações ou sua ordem na composição de cada bloco de dados. Um ataque de força bruta, através da geração de *hashes* para todas as combinações possíveis de blocos de dados, não é viável porque, conforme demonstrado na Equação (5.1), o número de combinações possíveis das frações resulta em um total de blocos de dados diferentes de aproximadamente $6{,}09 \times 10^{57}$.

$$A_{n,p} = \frac{n!}{(n-p)!} \rightarrow A_{4096,16} = \frac{4096!}{(4096-16)!} \cong 6.09\,x\,10^{57} \qquad \text{Equação (5.1)}$$

Consequentemente, como o tamanho de *hash* usado para representar cada bloco de dados é igual a 256 bits, conforme a Equação (5.2), seriam necessários cerca de $1{,}77 \times 10^{47}$ TB de espaço em disco para armazenar os *hashes* para todos blocos de dados possíveis.

$$Espaço\,em\,Disco\,(TB) = \frac{(6.09\,x\,10^{57}) * 256}{8 * 1024^4} \cong 1.77\,x\,10^{47} \qquad \text{Equação (5.2)}$$

Com o objetivo de tornar ineficaz uma possível tentativa do SADN de fraudar as respostas aos desafios, através de um registro histórico dos desafios já previamente realizados, os desafios enviados pelo SVI ao SADN são sempre inéditos. O protocolo prevê que cada conjunto de endereços de frações que formam um bloco de dados seja utilizado uma única e exclusiva vez como um desafio para o SADN.

Conforme descrito na Seção 3.2.1.3, cada conjunto de endereços de frações que formam um bloco de dados pertencem a um ciclo de verificação, que por sua vez, compreende todas as frações de um arquivo, distribuídas em 256 blocos. Como a avaliação da confiança é incrementada sempre que um ciclo completo é verificado, o número total de conjuntos de endereços de frações devem ser suficientes para que todos os ciclos de verificação estejam completos. A Tabela 5.2 apresenta o número de ciclos de verificação e os respectivos conjuntos de endereços de frações gerados de acordo com o período de armazenamento.

Tabela 5.2 - Número de ciclos e blocos de dados por período de armazenamento

Período de Armazenamento	**Ciclos de Verificação**	**Conjuntos de Endereços de Frações / Blocos de Dados**
1 ano	20	5120
5 anos	101	25856
10 anos	202	51712
20 anos	405	103680
30 anos	608	155648

A resistência a falhas foi um dos requisitos considerados na implementação da arquitetura. Um problema comum que pode ocorrer com qualquer infraestrutura de TI é o seu desligamento, seja ele planejado por motivos técnicos ou inesperado por motivos diversos como a falta de energia ou a queima de algum equipamento. A falha pode ocorrer também quando ocorrer qualquer evento que possa interromper o funcionamento do SGBD ou do servidor de aplicação responsável por executar a aplicação SVI ou a aplicação SADN.

Nesses casos, existe a possibilidade de ocorrer a falha durante o processamento de um ou mais desafios. Como o processo de verificação ocorre de forma distribuída entre o SVI e o SADN, dependendo da forma como o mesmo fosse implementado, poderia ocorrer o travamento de um dos serviços envolvidos, pois este ficaria esperando por uma resposta que nunca chegaria.

Para garantir que os processos de verificação executados pelas aplicações SVI e SADN fossem capazes de se autorrecuperar em caso de falhas, ambos os serviços foram implementados para funcionar de forma assíncrona e fazer uso de transações para garantir a atomicidade das ações realizadas dentro de cada aplicação.

Na aplicação SVI, após o processo de seleção e montagem do desafio, conforme descrito na Seção 4.1.2.2, este é registrado em sua base de dados antes de ser enviada ao SADN. Da mesma forma, o SADN ao receber o desafio, o armazena em sua base de dados para posterior processamento. Após o seu processamento, a resposta ao desafio é armazenada junto a sua requisição na base de dados do SADN e posteriormente enviada para o SVI. Ao receber a resposta, a aplicação SVI exclui na sua base de dados o registro onde foi armazenado o desafio antes do seu envio para o SADN.

O mecanismo de autorrecuperação do SVI funciona através do método "*checkRequests*" da classe "*CheckRequestsHandler*". Este método, através da utilização da anotação "*@Schedules*" é executado automaticamente pelo servidor de aplicação a cada 30 segundos e sua função é identificar na base de dados, registros de desafios pendentes que ultrapassaram o tempo máximo de espera por uma resposta. Esse tempo inicia em 15 segundos e é dobrado a cada execução sem erros do método até acumular 10 tentativas e o desafio for considerado falho.

Após encontrar o desafio com o tempo expirado, é executado o método "*getFileCheckResponse*", disponibilizado no SADN através do *web service* "*StorerWebService*", passando como parâmetro o identificador do desafio. O SADN pode responder a requisição de 03 formas diferentes, a primeira retornando a resposta ao desafio, a segunda informando que o desafio está aguardando o processamento e a terceira informando que o desafio não se encontra na base de dados do SADN. No último caso o desafio é reenviado para o SADN. Esta funcionalidade permite que o SVI recupere-se automaticamente de falhas ocorridas durante o envio do desafio ou no recebimento de respostas.

Na aplicação SADN, foram implementados dois mecanismos de autorrecuperação, o primeiro destinado a verificar a existência de desafios recebidos e não processados e o segundo destinado a verificar a existência de respostas não enviadas ao SVI. O método "*detectRequests*" da classe "*CheckerHandler*" detecta os desafios pendentes e dispara o método responsável por gerar a resposta ao desafio conforme descrito na Seção 4.1.3.2. O método "*detectResponses*" da classe "*AnswerHandler*" detecta respostas não enviadas e dispara o método responsável por enviar a resposta ao SVI conforme descrito na Seção 4.1.3.4. Estas funcionalidades permitem que o SADN recupere-se automaticamente de falhas ocorridas durante os processos de geração e envio de respostas a desafios recebidos.

5.2.6 - Variação do Nível de Confiança Atribuído ao Serviço de Armazenamento de Dados na Nuvem

Nessa seção são apresentados os resultados obtidos a partir de simulações para determinar o comportamento do nível de confiança no decorrer do tempo. Conforme previsto na definição do protocolo, a melhoria do nível de confiança ocorre sempre que um ciclo de verificação em um arquivo é concluído sem falhas.

Por esse motivo, as primeiras simulações realizadas foram para determinar o número máximo de dias que são necessários para que o SVI conclua um ciclo de verificação em um arquivo armazenado em um SADN. A conclusão de um ciclo de verificação indica que cada uma das 4096 frações de um arquivo fizeram parte de um dos blocos de dados verificados através de 256 desafios submetidos pelo SVI ao SADN.

O tempo necessário para concluir um ciclo de verificação de um arquivo pode variar entre um valor mínimo e um valor máximo, de acordo com o número de arquivos que estão sendo monitorados simultaneamente pelo SVI em um SADN. Entretanto, o tamanho do arquivo verificado não deve influenciar de forma significativa nesse tempo, pois o número de blocos de dados verificados no arquivo a cada dia representam um percentual do seu tamanho total conforme definido na Tabela 3.1.

Por intermédio de simulações matemáticas foi possível determinar que, em um SADN classificado no pior nível de confiança, a "desconfiança altíssima", o tempo máximo para concluir um ciclo de verificação é de 38 dias. Entretanto, em um SADN classificado no maior nível de confiança, a "confiança altíssima", o tempo para concluir um ciclo pode chegar a 1.792 dias. A Figura 5.8 mostra os tempos máximos e mínimos em dias para concluir um ciclo de verificação de um arquivo para cada um dos níveis de confiança propostos no protocolo.

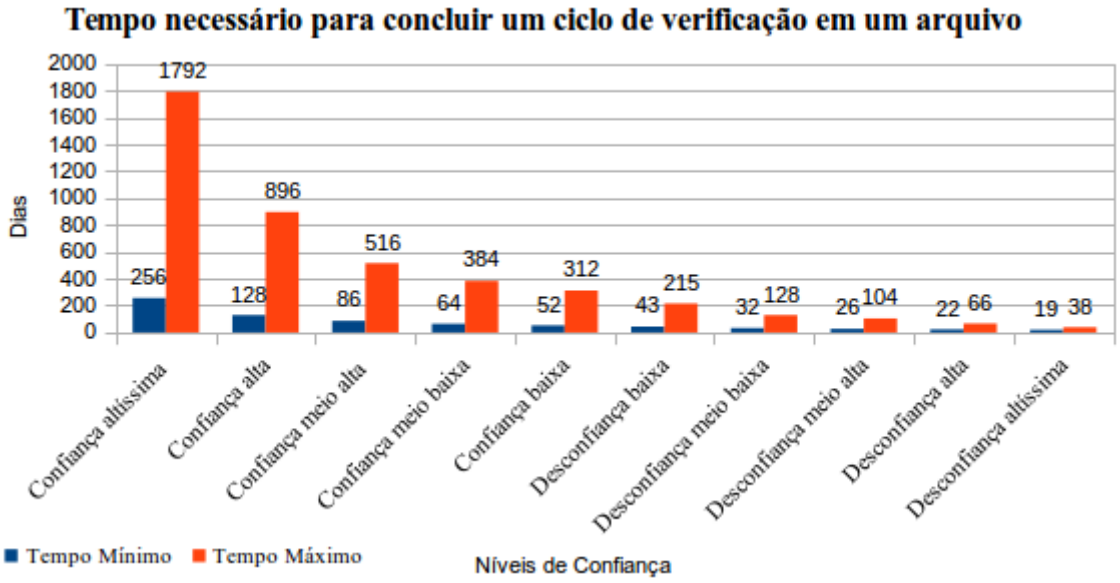


Figura 5.8 - Tempo necessário para concluir um ciclo de verificação em um arquivo.

Apesar da simulação matemática aplicada sobre as regras definidas no protocolo apontar o tempo máximo necessário para se concluir um ciclo de verificação, esse pode aumentar caso o SVI ou o SADN não possua capacidade computacional suficiente para respectivamente gerar ou responder os desafios previstos para cada dia ou ainda, quando a banda de rede disponível não comportar o tráfego de pacotes gerado. Esta situação pode

ocorrer principalmente quando o número de arquivos armazenados no SADN e monitorados pelo SVI for muito grande.

A variação no tempo para conclusão de um ciclo de verificação de acordo com o nível de confiança atribuída ao SADN é gerada pela diferença na quantidade de blocos de dados verificados a cada dia. Esta variação tem por objetivo recompensar os serviços de armazenamento que historicamente não apresentam falhas, minimizando o consumo dos recursos como processamento e banda de rede. Mais ainda, este mecanismo permite que o protocolo priorize a verificação de arquivos que estão armazenados em SADNs que já apresentaram falhas, reduzindo o tempo necessário para determinar se outros arquivos foram perdidos ou corrompidos.

Na sequência, foram realizadas simulações para determinar o número de ciclos de verificação de arquivos, concluídos sem identificar nenhuma falha, são necessários para que o nível de confiança para um SADN seja incrementado até chegar ao nível mais alto de confiança prevista na Tabela 3.1, a "Confiança altíssima". Na Figura 5.9 são apresentados os resultados da simulação, utilizando como ponto de partida a situação "Não Avaliada", ou seja, com o valor atribuído a confiança igual a zero.

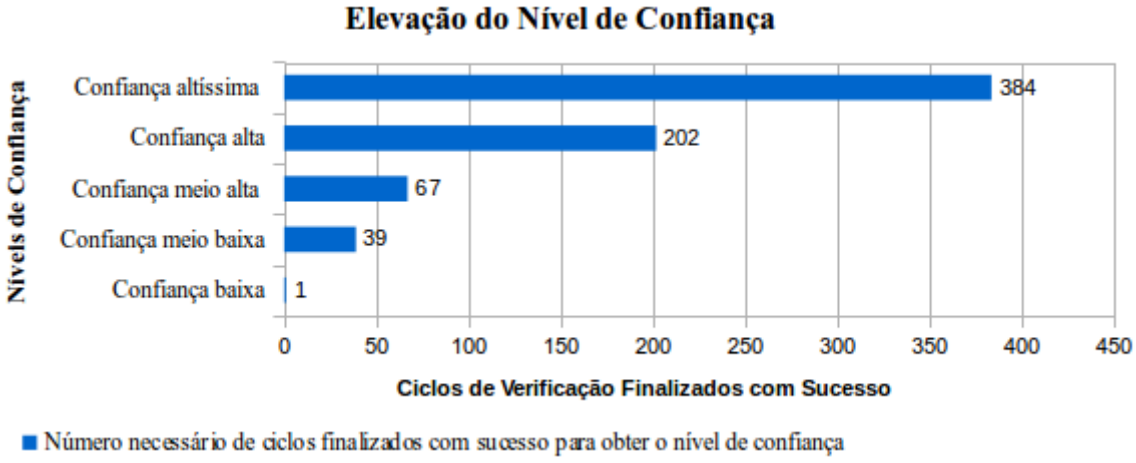


Figura 5.9 - Elevação do nível de confiança.

Ao analisar os resultados apresentados nas figuras 5.8 e 5.9, poderia se concluir que o tempo necessário para um SADN obter o nível máximo de confiança é tão grande, que seria praticamente impossível alcançá-lo. Essa conclusão é facilmente obtida através da multiplicação do número máximo de dias necessários para concluir um ciclo de verificação

no nível “Confiança alta” (896), pelo número de ciclos finalizados com sucesso para, a partir do nível anterior, atingir o nível “Confiança altíssima” (384 – 202 = 182).

Como resultado deste cálculo (182 * 896), conclui-se que seriam necessários pelo menos 163.072 dias, o que equivale a aproximadamente 453 anos. No entanto, apesar de matematicamente correta, essa avaliação está incorreta, pois na prática ela nunca ocorreria.

A explicação para este fato também é simples e está relacionada a quantidade de arquivos que estão sendo monitorados simultaneamente pelo SVI no SADN. O tempo máximo previsto para a conclusão de um ciclo de verificação de um arquivo só ocorre quando o número de arquivos monitorados em um SADN, classificado no nível “Confiança alta”, for igual a 25 ou a um múltiplo deste.

Isto decorre do fato que, de acordo com a Tabela 3.1, no nível de confiança citado, devem ser verificados, no mínimo, 16% dos arquivos a cada dia. O tempo máximo para verificação ocorre somente quando o número exato do percentual mínimo de arquivos é verificado. Entretanto, se na realização do cálculo do número de arquivos a ser verificado a cada dia, conforme descrito na Seção 3.2.1.3, o resultado obtido não for um número inteiro, o número de arquivos resultante é arredondado para o próximo inteiro, aumentando o percentual efetivamente verificado.

Considerando que o SVI esteja monitorando exatamente 25 arquivos em SADN classificado no nível “Confiança alta”, e que esses arquivos foram submetidos exatamente na mesma data, tendo em vista que as características do arquivo não influenciam no tempo para conclusão de um ciclo de verificação, todos concluirão seu ciclo de verificação na mesma data, ou seja, em 896 dias, Assim, como a cada período de 896 dias são finalizados 25 ciclos, para atingir os 182 ciclos necessários para elevar o SADN para o nível “Confiança altíssima” seriam necessários quase 20 anos.

No entanto, esta situação não descreve o pior caso, que para o nível “Confiança alta”, ocorre quando há apenas 6 arquivos sendo monitorados onde a ascensão para o próximo nível passa dos 65 anos. A tendência é que menor seja o tempo necessário para elevar o nível de confiança, quanto maior for a quantidade de arquivos monitorados, podendo haver

pequenas oscilações. A Figura 5.10 apresenta um comparativo entre os tempos necessários para ascender ao próximo nível de confiança conforme o número de arquivos monitorados.

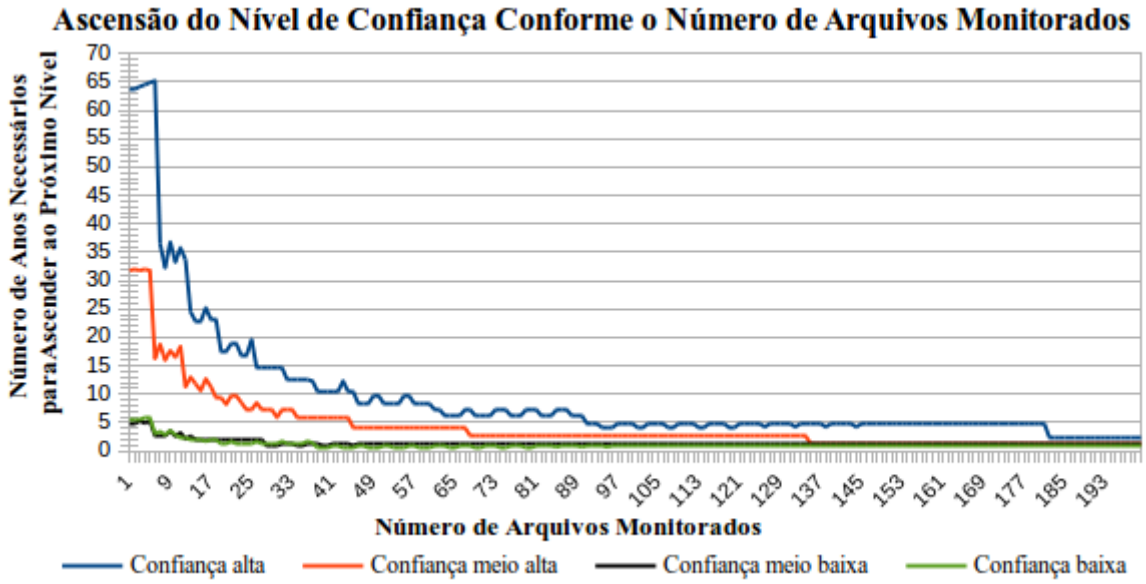


Figura 5.10 - Ascensão do nível de confiança conforme o número de arquivos monitorados.

Conforme pode ser observado na Figura 5.10, o melhor caso é obtido quando o número de arquivos monitorados é igual à quantidade de ciclos finalizados com sucesso requeridos para elevar o nível de confiança. A partir desta quantidade de arquivos, o tempo necessário para ascender ao próximo nível será sempre igual ao tempo necessário para finalizar um ciclo.

Ao contrário do processo de ascensão do nível de confiança que ocorre de forma lenta e gradual, o protocolo propõe um processo de rebaixamento do nível de confiança muito rápido. O decremento do valor da confiança atribuída ao SADN, conforme descrito na Seção 4.1.2.4, ocorre sempre que o resultado de um desafio indicar uma falha no arquivo verificado.

Para avaliar o processo de rebaixamento do nível de confiança proposto no protocolo, foram realizadas simulações para determinar o número de falhas necessárias para um SADN atingir o nível de desconfiança máxima. As simulações usaram como ponto de partida a identificação de uma falha ocorrida em um SADN avaliado em qualquer dos níveis de confiança descritas na Tabela 3.1 entre a “Confiança altíssima” e a “Confiança

baixa". Nessa situação, assim que é identificada a falha, o valor da confiança é atualizado para zero, reclassificando imediatamente o SADN no nível "Desconfiança baixa". A Figura 5.11 apresenta o número de falhas identificadas necessárias para rebaixar nível a nível a confiança até o nível máximo de desconfiança previsto, a "Desconfiança altíssima".

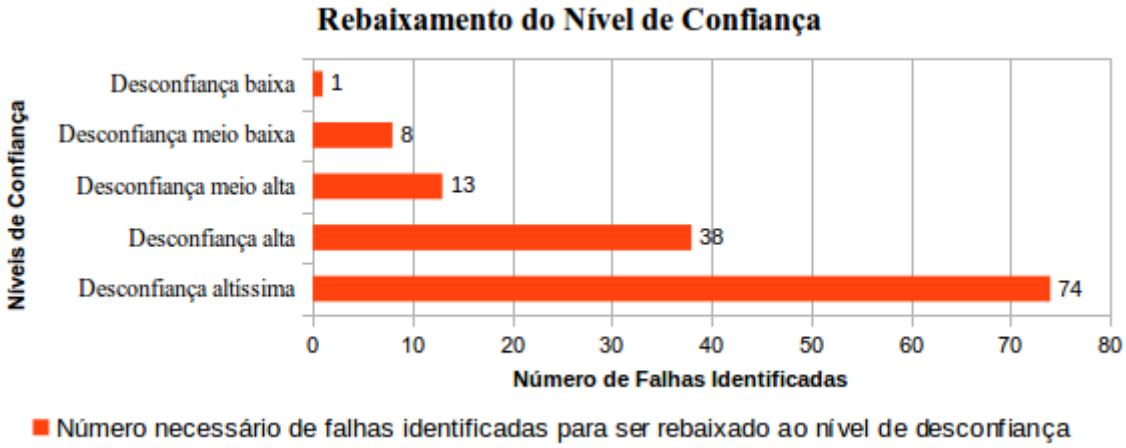


Figura 5.11 - Rebaixamento do nível de confiança.

Semelhante ao que ocorre no processo de ascensão do nível de confiança, o tempo mínimo necessário para que um SADN seja rebaixado de um nível para outro depende da quantidade de arquivos que estão sendo monitorados simultaneamente. A Figura 5.12 apresenta um comparativo entre os tempos mínimos necessários para rebaixar um SADN, considerando que todos os arquivos monitorados estão corrompidos e que a falha será identificada no primeiro conjunto de desafios enviado pelo SVI ao SADN.

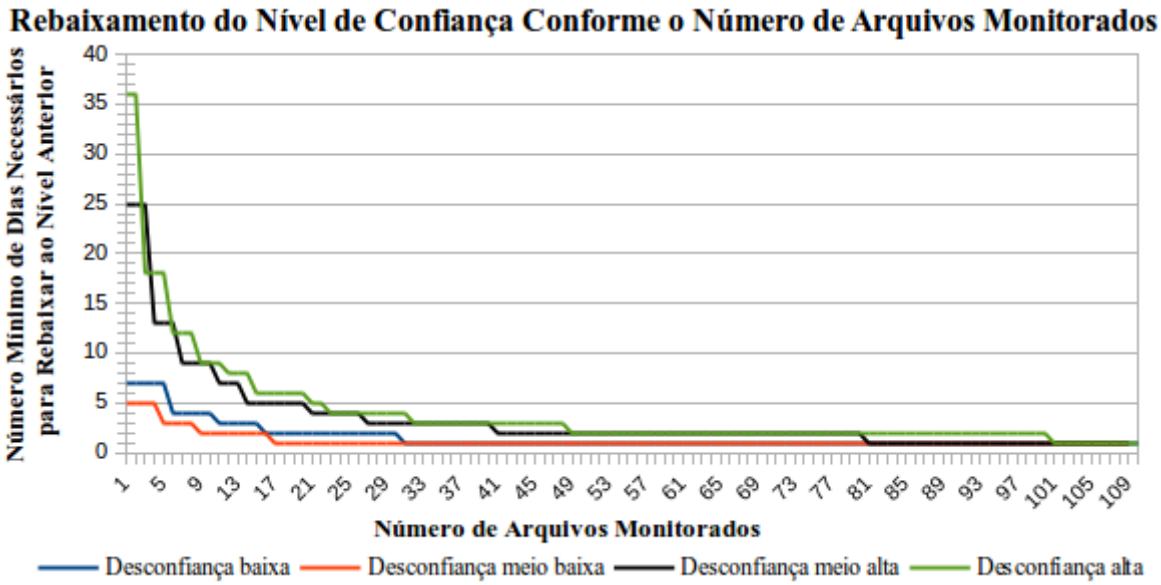


Figura 5.12 - Rebaixamento do nível de confiança conforme o número de arquivos monitorados.

Uma importante diferença entre o processo de rebaixamento em relação ao processo de ascensão do nível de confiança é que o tempo para o rebaixamento é contado em dias enquanto que o tempo para ascensão é contado em anos. Conforme pode ser visualizado na Figura 5.12, a tendência é que o tempo mínimo necessário para rebaixar um nível seja igual a um dia a partir do momento que o número de arquivos monitorados for igual ou maior que o número falhas identificadas necessárias para rebaixar um nível conforme Figura 5.11.

5.3 - SÍNTESE DO CAPÍTULO

O objetivo deste capítulo foi descrever um conjunto de testes e simulações, por meio dos quais foi possível determinar características funcionais do protocolo e das aplicações. Além disso, por intermédio da análise dos resultados foi possível confirmar a eficácia da arquitetura e confirmar características importantes do protocolo do proposto.

6 - CONCLUSÕES E RECOMENDAÇÕES

Neste trabalho, foi proposta uma arquitetura computacional, que através de um serviço terceirizado de verificação, aplica os conceitos de confiança para permitir um monitoramento qualificado e constante da integridade dos arquivos mantidos em um Serviço de Armazenamento de Dados na Nuvem (SADN), sem comprometer a confidencialidade. Baseado no comportamento de cada SADN, a frequência de verificações pode aumentar ou diminuir, reduzindo a carga de processos nos serviços que tem taxa de falha muito baixa ou verificando mais rapidamente todos os arquivos armazenados em serviços que já apresentaram algum tipo de falha.

A arquitetura implementa um protocolo projetado para atender premissas como baixo consumo de largura de banda, identificar rapidamente serviços mal comportados, prover forte resistência contra fraudes, evitar a sobrecarga do SADN, garantir a confidencialidade dos dados e ainda, fornecer previsibilidade e máxima economia de recursos ao Serviço de Verificação da Integridade (SVI). A arquitetura elimina a necessidade do cliente manter cópias dos arquivos armazenados na nuvem, garantindo a sua recuperabilidade por meio de redundância, cujo nível pode ser adequado pelo cliente, conforme a criticidade da informação a ser armazenada.

A principal característica do protocolo é fornecer um controle eficiente sobre a integridade dos arquivos salvos nos SADN, sem sobrecarregar os serviços que têm um comportamento adequado, mas atuando pró-ativamente para rapidamente identificar falhas e fornecer alertas precoces sobre arquivos corrompidos aos seus proprietários. O monitoramento da integridade é realizado de forma fracionada, sem a necessidade do SVI acessar diretamente o conteúdo do arquivo e sem a necessidade do SADN processá-lo por inteiro a cada verificação.

A implementação da arquitetura foi realizada utilizando algoritmos criptográficos, para os quais não há ataques conhecidos, e com força criptográfica suficiente para manter a confidencialidade dos dados por muitas décadas, considerando o estado da arte atual da tecnologia e o seu poder computacional. Além dos mecanismos de resistência a fraudes previstos na proposta do protocolo, na implementação da arquitetura foram inseridos

recursos para aumentar a resistência a falhas, dos quais pode-se destacar o uso de chamadas *web services* assíncronas e o monitoramento de ações pendentes tanto no SVI como SADN.

A validação fracionada do arquivo composta por blocos de dados montados a partir de frações escolhidas aleatoriamente se mostrou eficaz durante as simulações com injeção de falhas. Mesmo pequenas alterações em arquivos grandes levaram em média 14 dias para serem identificadas.

A protocolo prevê a classificação do SADNs em níveis de confiança, conforme o seu histórico de disponibilidade e a integridade dos dados por eles armazenados. O resultado dessa classificação, além de ser aplicado para desonerar a carga computacional consumida dos SADNs que não apresentaram falhas, permite ainda que os clientes possam usar essa informação, para selecionar de forma crítica o SADN mais adequado a ser contratado, conforme o nível de criticidade da informação a ser armazenada.

As simulações e testes realizados a partir da implementação da arquitetura, demonstraram que o tempo mínimo necessário para que um SADN alcance o próximo nível de confiança em determinado SVI, varia de acordo com o número de arquivos que este SVI monitora simultaneamente no SADN avaliado e o nível atual da confiança. Entretanto, a partir de um determinado número de arquivos, que no caso da ascensão são aproximadamente 180, esse tempo se estabiliza em aproximadamente 1 ano. Já no rebaixamento, apesar de apresentar o mesmo comportamento, a unidade de tempo deixa de ser “ano” e passa a ser “dia”, sendo que a estabilidade já ocorre com aproximadamente 100 arquivos e o tempo mínimo passa a ser 1 dia.

Com os resultados dos testes realizados foi possível verificar que o tempo gasto para geração da Tabela de Informações (TabInfor) do arquivo, assim como o seu tamanho, podem ser considerados aceitáveis, crescendo proporcionalmente ao tamanho do mesmo. O consumo de banda de rede para a realização do monitoramento comprovou-se bastante baixo independentemente do nível de segurança atribuído ao SADN.

A arquitetura mostrou-se durante os testes bastante robusta, respondendo de forma satisfatória as simulações de falhas. A arquitetura resistiu muito bem também a falhas não

previstas no protocolo, como o desligamento não planejado de servidores, não exigindo qualquer intervenção humana para o retorno de suas funcionalidades após o religamento dos mesmos.

Por fim, por meio dos resultados obtidos foi possível verificar que os objetivos propostos para o trabalho foram atingidos, dos quais destacam-se a verificação fracionada da integridade e o balanceamento da carga de verificações. Entretanto, também foi possível identificar que existem oportunidades de melhoria que devem ser abordadas em trabalhos futuros como a melhoria da performance do processo de geração de *hashes* em arquivos com tamanhos iguais ou superiores a 15 Gb.

6.1 - TRABALHOS FUTUROS

Como trabalho futuro, pretende-se adicionar uma funcionalidade que permita o compartilhamento de informações sobre o nível de confiança dos SADN entre os diferentes SVI. Esta funcionalidade permitiria por exemplo, que uma falha identificada em determinado SADN por um SVI, alerte os demais de forma que estes reajam pró-ativamente, dando prioridade a verificação dos arquivos armazenados no referido SADN.

Como funcionalidade adicional, objetivando aumentar a segurança e minimizar as possibilidades de ataques sobre a arquitetura, pretende-se propor um mecanismo que garanta que somente SVI autorizados pelo cliente possam submeter desafios ao SADN. Além disso, pretende-se formalizar o protocolo proposto utilizando uma linguagem formal, como redes de Petri (Jensen 1992) ou lógica BAN (Burrows, Abadi, Needham 1990).

Para complementar as análises e aumentar o nível de confiança nos resultados apresentados, pretende-se realizar testes de performance e comportamento da arquitetura em nuvens reais, providas por provedores comerciais como a *Amazon* e o *Google*. Outros testes também deverão ser realizados de forma a comparar a performance obtida pela arquitetura proposta em relação a performance obtida em mesmo ambiente por soluções propostas por outros autores.

Com objetivo de se obter melhores performances em arquivos superiores a 30 GB, pretende-se ainda, testar variações do protocolo, como o aumento da quantidade de frações por arquivo e/ou quantidade de frações por bloco de dados. Nesse mesmo sentido, pretende-se ainda aprimorar a implementação da arquitetura de forma a aumentar o nível de paralelismo dos processos, como por exemplo, o processo de geração dos *hashes* utilizados para verificar a integridade do arquivos.

REFERÊNCIAS BIBLIOGRÁFICAS

Abdul-Rahman, A., Hailes, S. (1997). "A distributed trust model." In: The 1997 New Security Paradigms Workshop, ACM, Langdale, Cumbria, United Kingdom, pp. 48-60.

Al-Jaberi, M. F., Zainal, A. (2014). "Data integrity and privacy model in cloudcomputing." In: International Symposium on Biometrics and SecurityTechnologies (ISBAST), IEEE, Kuala Lumpur, Malasia, pp. 280-284.

Albuquerque, R. O. (2008). Uma Proposta de um Modelo de Confiança Computacional para Grupos em Sistemas Distribuídos, Tese de Doutorado, Faculdade de Tecnologia, Departamento de Engenharia Elétrica, Universidade de Brasília.

Amazon Web Services, Inc. (2016). Amazon S3: Amazon Simple Storage Service. Disponível em: <http://aws.amazon.com/pt/s3>. Acesso: 10 jun. 2016.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., Zaharia, M. (2009). Above the Clouds: A Berkeley View of Cloud, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, Technical Report No. UCB/EECS-2009-28.

Aumasson, J., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C. (2013). "BLAKE2: Simpler, Smaller, Fast as MD5." In: Applied Cryptography and Network Security, Springer Berlin Heidelberg, Banff, AB, Canada, pp. 119-135.

Bellare, M., Kilian, J., Rogaway, P. (1994). "The Security of Cipher Block Chaining." In: Advances in Cryptology — CRYPTO '94: 14th Annual International Cryptology Conference, Springer Berlin Heidelberg, Santa Barbara, California, USA, pp. 341-358.

Beth, T., Borcherding, M., Klien, B. (1994). "Valuation of Trust in Open Networks." In: Computer Security — ESORICS 94: Third European Symposium on Research in Computer Security, Springer Berlin Heidelberg, Brighton, United Kingdom, pp. 1-18.

Borthakur, D. (2007). The Hadoop Distributed File System:Architecture and Design, Manual, The Apache Software Foundation, v. 11.

Bose, R. (2008). Information Theory, Coding and Cryptograph, Tata McGraw-Hill, 2nd Ed..

Burrows, M., Abadi, M., Needham, R. M. (1990). "A logic of authentication." In: ACM Transactions on Computer Systems - Journal, vol. 8, issue 1, ACM, February, pp. 18-36.

Buyya, R., Broberg, J., Goscinski, A. (2011). Cloud Computing: Principles and Paradigms, John Wiley & Sons, Hoboken, New Jersey.

Buyya, R., Ranjan, R., Calheiros, R. N. (2010). "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services." In: Algorithms and Architectures for Parallel Processing: The 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Part I, Springer Berlin Heidelberg, Busan, Korea, pp. 13-31.

Buyya, R., Ranjan, R., Calheiros, R. N. (2011). "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities." In: 7th High Performance Computing and Simulation Conference (HPCS), IEEE, Leipzig, Germany, pp. 1-11.

Canedo, E. D. (2012). Modelo de Confiança para a Troca de Arquivos em uma Nuvem Privada, Tese de Doutorado, Faculdade de Tecnologia, Departamento de Engenharia Elétrica, Universidade de Brasília.

Dabas, P., Wadhwa, D. (2014). "A recapitulation of data auditing approachesfor cloud data." In: International Journal of Computer ApplicationsTechnology and Research (IJCATR), vol. 3, issue 6, Association of Technology and Science, India, pp. 329-332.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W. (2007). "Dynamo: Amazon's Highly Available Key-value Store." In: Twenty-First Symposium on Operating Systems Principles (SIGOPS), ACM, New York, NY, USA, pp. 205-220.

Gambetta, D. (2000). Can We Trust?, Trust: Making and Breaking Cooperative Relations, v. 13, Blackwell Pub.

George, R. S., Sabitha, S. (2013). "Data anonymization and integrity checkingin cloud computing." In: Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE, Tiruchengode, India, pp. 1-5.

Ghemawat, S., Gobioff, H., Leung, S. (2003). "The Google File System." In: 19th ACM Symposium on Operating Systems Principles (SIGOPS), vol. 37, issue 5, ACM, New York, NY, USA, pp. 29-43.

Gholami, A., Arani, M. G. (2015). "A Trust Model Based on Quality of Service in Cloud Computing Environment." In: International Journal of Database Theory and Application - Journal, vol. 8, nº 5, SERSC, October, pp.161-170.

Google Inc. (2016). Google Cloud Platform - Cloud Storage: A Powerful, Simple and Cost Effective Object Storage Service. Disponível em: <https://cloud.google.com/storage>. Acesso: 10 jun. 2016.

Jendrock, E. Cervera-Navarro, R., Evans, I., Haase, K., Markito, W. (2014). Java Platform, Enterprise Edition, The Java EE Tutorial, Oracle Corporation, release 7.

Jensen, K. (1992). Coloured Petri Nets 3: Basic Concepts, Analysis Methods and Practical Use, Springer-Verlag, Berlin, Alemanha, vol. 3.

Jøsang, A., Ismail, R., Boyd (2007). "A Survey of Trust and Reputation Systems for Online Service Provision." In: Decision Support Systems - Journal, vol 43, issue 2, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, pp. 618-644.

Jøsang, A., Pope, S. (2005). "Semantic Constraints for Trust Tansitivity." In: Asia-Pacific Conference of Conceptual Modelling (APCCM), vol. 43, Australian Computer Society, Inc., Newcastle, Australia, pp. 59-68.

Josefsson, S. (2011). RFC 6070 PKCS #5: Password-Based Key Derivation Function 2 (PBKDF2) Test Vectors. Disponível em: <https://www.rfc-editor.org/rfc/rfc6070.txt>. Acesso: 23 maio 2016.

Kajeepeta, S. (2010). Multi-Tenancy in the Cloud: Why it matters. Disponível em: <http://www.computerworld.com/article/2517005/data-center/multi-tenancy-in-the-cloud--why-it-matters.html>. Acesso: 22 jun. 2016.

Kavuri, S. K. S. V. A., Kancherla, G. R., Bobba, B. R. (2014). "Data Authentication and Integrity Verification Techniques for Trusted/Untrusted Cloud Servers." In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, New Delhi, India, pp. 2590-2596.

Kay, H., Chuanhe, H., Jinhai, W., Hao, Z., Xi, W., Yilong, L., Lianzhen, Z., Bin W. (2013). "An Efficient Public Batch Auditing Protocol for Data Security in Multi-Cloud Storage." In: 8th ChinaGrid Annual Conference (ChinaGrid), IEEE, Changchun, China, pp. 51-56.

Kertész,A., Kecskeméti, G., Marosi, A., Oriol, M., Franch,X., Marco, J. (2012). "Integrated Monitoring Approach for Seamless Service Provisioning in Federated Clouds." In: 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), IEEE, Garching, Germany, pp. 567-574.

Krawczyk, H.; Bellare, M.; Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication. Disponível em: <http://tools.ietf.org/html/rfc2104?ld=APUSLPADefault>. Acesso: 23 maio 2016.

Mao, W. (2003). Modern Cryptography: Theory and Practice, Prentice Hall PTR, 1st Printing.

Marsh, S. P.(1994). Formalizing trust as a computational concept, Doctorate Thesis, Department of Computing Science and Mathematics, University of Stirling.

Marston, S., Li, S., Bandyopadhyay, S., Zhang, J., Ghalsasi, A. (2011). "Cloud computing — The business perspective." In: Decision Support Systems - Journal, vol. 51, issue 1, Elsevier B. V., April, pp. 176–189.

McGrew, D. (2011). The Use of AES-192 and AES-256 in Secure RTP: . Disponível em: <http://tools.ietf.org/html/rfc6188.html>. Acesso: 22 maio 2016.

Mell, P., Grance, T. (2011). The NIST Definition of Cloud Computing, NIST Special Publication 800-145, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

Menezes, A. J., Oorschot, P. C., Vanstone, S. A. (1996). Handbook of Applied Cryptography, CRC press, Boca Raton, USA.

Microsoft Corporation (2016). Microsoft Azure - Storage: Massively Scalable Cloud Storage for Your Applications. Disponível em: <https://azure.microsoft.com/en-us/services/storage>. Acesso: 10 jun. 2016.

Network Associates, Inc. (2000). An Introduction to Cryptography, Manual, PGP Software Documentation, Santa Clara, CA, USA.

NIST FIPS PUB 180-4 (2015). Secure Hash Standard (SHS), Federal Information Processing Standards Publication, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

NIST FIPS PUB 197 (2001). Advanced Encryption Standard (AES), Federal Information Processing Standards Publication, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

NIST FIPS PUB 202 (2015). SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, Federal Information Processing Standards Publication, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

Oracle Corporation (2016). Java™ Platform, Standard Edition 7: API Specification. Disponível em: <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>. Acesso: 21 maio 2016.

Oracle Corporation (2016a). JAX-WS RI: Architecture Document. Disponível em: <https://jax-ws.java.net/nonav/jax-ws-20-fcs/arch/overview-summary.html>. Acesso: 22 maio 2016.

Oracle Corporation (2016b). GlassFish Server Open Source Edition : Release Notes. Disponível em: <https://glassfish.java.net/docs/4.1/release-notes.pdf>. Acesso: 23 maio 2016.

Patel, J. (2007). A Trust and Reputation Model for Agent-Based Virtual Organizations, Thesis of Doctor of Philosophy, Faculty of Engineering and Applied Science, School of Electronics and Computer Science, University of Southampton.

Rivest, R. (1992). RFC1321: The MD5 message-digest algorithm. Disponível em: <http://tools.ietf.org/html/rfc1321>. Acesso: 25 jun. 2016.

Shvachko, K., Kuang, H., Radia, S., Chansler, R. (2010). "The hadoop distributed file system." In: 26th symposium on mass storage systems and technologies (MSST), IEEE, Lake TahoeIncline Villiage, NV, USA, pp. 1-10.

Stallings, W. (2005). Cryptography and Network Security Principles and Practices, Prentice Hall, Upper Saddle River, NJ, USA.

Tahta, U. E., Sen, S., Can, A. B. (2015). "GenTrust : A Genetic Trust Management Model forPeer-To-Peer Systems." In: Applied Soft Computing - Journal, vol. 34, Elsevier B. V., September, pp. 693-704.

Tandel, S. T., Shah, V. K., Hiranwal S. (2013). "An implementation of effective XML based dynamic data integrity audit service in cloud." In: International Journal of Societal Applications of Computer Science, IJSACS, vol. 2, issue 8, August, pp. 449-553.

The PostgreSQL Global Development Group (2016). PostgreSQL: The world's most advanced open source database.. Disponível em: <http://www.postgresql.org>. Acesso: 22 maio 2016.

Vaquero, M., Rodero-Merino, L., Caceres, R., Lindner, M. (2009). "A Break in the Clouds: Towards a Cloud Definition." In: ACM SIGCOMM Computer Communication Review, vol. 39, issue 1, ACM, Newsletter, January, pp. 50-55.

Wang, Q., Wang, C., Li, J., Ren, K., Lou, W. (2009). "An Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing." In: Computer Security – ESORICS 2009: 14th European Symposium on Research in Computer Security (ESORICS), Springer Berlin Heidelberg, Saint-Malo, France, pp. 355-370.

Zissis, D., Lekkas, D. (2012). "Addressing Cloud Computing Security Issues." In: Future Generation Computer Systems - Journal, vol. 28, issue 3, Elsevier B. V., March, pp. 583–592.