UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIAS DA COMPUTAÇÃO

ELISEU CASTELO BRANCO JÚNIOR

UMA ESTRATEGIA PARA ASSEGURAR A CONFIDENCIALIDADE DE DADOS ARMAZENADOS EM NUVEM

FORTALEZA

2017

ELISEU CASTELO BRANCO JÚNIOR

UMA ESTRATEGIA PARA ASSEGURAR A CONFIDENCIALIDADE DE DADOS ARMAZENADOS EM NUVEM

Tese apresentada ao Curso de Doutorado em Ciências da Computação do Programa de Pós-Graduação em Ciências da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciências da Computação. Área de Concentração: Banco de Dados

Orientador: Prof. Dr. Javam de Castro Machado

Co-Orientador: Prof. Dr. José Maria da Silva Monteiro Filho

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B813e Branco Jr., Eliseu Castelo.
Uma Estratégia para Assegurar a Confidencialidade de Dados Armazenados em Nuvem / Eliseu Castelo Branco Jr.. – 2017.
209 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2017.
Orientação: Prof. Dr. Javam de Castro Machado.
Coorientação: Prof. Dr. José Maria da Silva Monteiro Filho.

1. Privacidade. 2. Confidencialidade. 3. Computação em Nuvem. I. Título.

CDD 005

ELISEU CASTELO BRANCO JÚNIOR

UMA ESTRATEGIA PARA ASSEGURAR A CONFIDENCIALIDADE DE DADOS ARMAZENADOS EM NUVEM

Tese apresentada ao Curso de Doutorado em Ciências da Computação do Programa de Pós-Graduação em Ciências da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciências da Computação. Área de Concentração: Banco de Dados

Aprovada em: 31 de maio de 2017

BANCA EXAMINADORA

Prof. Dr. Javam de Castro Machado (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Maria da Silva Monteiro Filho (Co-Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Fernando Antônio Mota Trinta
Universidade Federal do Ceara (UFC)

Prof. Dr. Joaquim Celestino Junior
Universidade Estadual do Ceara (UECE)

Prof. Dr. João Eduardo Ferreira
Universidade de São Paulo (USP)

Ao meus pais Eliseu Castelo Branco Camurça (in memoriam) e Francisca Bezerra Castelo Branco

AGRADECIMENTOS

A Deus, por estar sempre ao meu lado, dando-me coragem para enfrentar todos os obstáculos da vida.

Aos meus pais pelos exemplos de perseverança, coragem e amor que moldaram meu caráter e personalidade para o caminho do bem, da verdade, da justiça e do amor.

A minha esposa Cláudia e meus filhos Rodrigo, Renata, Caio e Saulo pela paciência, compreensão e ajuda que me deram durante a realização do curso.

Ao meu orientador, Prof. Dr. Javam de Castro Machado, que, com sua competência teórica, auxiliou no delineamento e sistematização do trabalho. Muito obrigado pela amizade, paciência e orientação durante o curso.

Ao meu coorientador, Prof. Dr. José Maria da Silva Monteiro Filho pelo apoio e dedicada atenção a este trabalho.

Aos colegas do Banco do Nordeste do Brasil S/A pelo apoio e incentivo que tornou possível a minha participação neste curso de Doutorado.

A Universidade Corporativa do Banco do Nordeste pelo apoio logístico e financeiro para minha participação neste curso de Doutorado.

Aos professores do Mestrado e Doutorado em Ciências da Computação - MDCC, pelos valiosos ensinamentos e troca de ideias que muito contribuíram para o desenvolvimento deste trabalho.

Ao amigo professor Dr. Alberto Sampaio Lima, por ter acreditado e me feito acreditar que este projeto de doutoramento seria possível ser executado.

Aos colegas do MDCC, que pelo convívio e troca de conhecimentos se tornaram amigos para toda a vida.

A todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho. Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Aos bibliotecários da Universidade Federal do Ceará: Eliene Maria Vieira de Moura, Francisco Edvander Pires Santos, Izabel Lima dos Santos, Juliana Soares Lima, Kalline Yasmin Soares Feitosa pela revisão e discussão da formatação utilizada neste *template*.

“Ainda que genial, a invenção da imprensa é insignificante se comparada à invenção das letras.”
"Permita-me mencionar que as palavras ou nomes de todos os objetos e atos podem ser relacionados numa lista de acordo com duas maneiras diferentes, seguindo o alfabeto e seguindo a natureza[...] A primeira vai da palavra ao objeto; a segunda, do objeto à palavra.”

(Hobbes e Leibniz)

RESUMO

O armazenamento de grandes quantidades de dados confidenciais em servidores na nuvem é uma tendência para as empresas que buscam oportunidades de reduzir custos e aumentar a disponibilidade de seus serviços digitais. Contudo, nos ambientes de computação em nuvem o controle do dado deixa de ser do seu proprietário e passa a ser do provedor do serviço, o que proporciona novos desafios relacionados à privacidade, segurança e confidencialidade.

Neste contexto, diferentes soluções para assegurar a confidencialidade dos dados armazenados na nuvem foram propostas. Em geral, tais soluções utilizam criptografia, fragmentação de dados ou uma combinação dessas duas abordagens. Apesar disto, problemas relacionados à eficácia destas técnicas em relação à ataques, perda ou roubo de dados têm ocorrido nos últimos anos, causando prejuízos de milhões de dólares para empresas e clientes.

Esta tese apresenta uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem. A ciência por trás dessa abordagem utiliza conceitos da Doutrina do Ser de Hegel. A estratégia QSM-EXTRACTION baseia-se na fragmentação de um arquivo digital em fragmentos denominados objetos de informação, na decomposição desses objetos por meio da extração de suas características (Qualidade, Quantidade e Medida) e na dispersão dessas características em diferentes serviços de armazenamento em nuvem, permitindo a posterior recuperação desses dados sem perda de informação. A finalidade da estratégia proposta é inviabilizar a reconstrução do arquivo original por parte de um provedor de nuvem que possui apenas parte das características dos objetos de informações que compõem este arquivo. Desta forma, assegura-se a confidencialidade dos dados armazenados em nuvem e, por conseguinte, a privacidade dos proprietários desses dados. O trabalho aqui proposto apresenta uma nova forma de ocultar o significado dos dados armazenados na nuvem, a qual baseia-se na extração e armazenamento das características desses dados, e não nos dados em si.

Com a finalidade de demonstrar a eficiência das ideias que norteiam a estratégia proposta nesta tese, diversos experimentos foram realizados. Para executar estes experimentos, foi utilizada uma infraestrutura de nuvem privada gerida pelo OpenStack (*Openstack Cloud Operating System*). Os algoritmos que compõem a estratégia QSM-EXTRACTION foram implementados em linguagem C++. Para realizar a avaliação da eficiência da estratégia QSM-EXTRACTION, foi utilizado uma coleção de documentos criados sinteticamente, com diferentes tamanhos. Os resultados dos experimentos comprovaram a viabilidade de utilizar a abordagem proposta em cenários

típicos da computação em nuvem, nos quais a quantidade de leituras é maior que a de escritas. Adicionalmente, os experimentos mostraram que a estratégia QSM-EXTRACTION é bastante flexível, podendo ser utilizada em conjunto com as principais abordagens para confidencialidade de dados: criptografia, fragmentação e criptografia/fragmentação.

Palavras-chave: Privacidade. Confidencialidade. Computação em nuvem.

ABSTRACT

Large amounts of confidential data stored on servers in the cloud is a trend for companies looking for opportunities to reduce costs and increase the availability of their digital services. However, in cloud computing environments data control is no longer belongs to the data owner and the control belongs to the service provider, which provides new challenges related to privacy, security, and confidentiality.

In this context, privacy and security solutions for data stored in the cloud using encryption, data fragmentation or a combination of both have been proposed as best existing techniques in the scientific literature. Despite this, problems related to the effectiveness of these techniques in relation to the attacks, loss or theft of data have occurred in recent years, causing millions of dollars of damages to companies and clients.

In this thesis, we present a new approach, called QSM-EXTRACTION, to ensure the confidentiality of data in cloud storage services. The science behind this approach uses concepts from Hegel's Doctrine of Being. The QSM-EXTRACTION strategy is based on the fragmentation of a digital file into fragments called information objects, on the decomposition of these objects through the extraction of their characteristics (Quality, Quantity and Measure) and the dispersion of these characteristics in different storage services in Cloud, allowing the later retrieval of this data without loss of information.

In order to demonstrate the efficiency of the ideas that guide the strategy proposed in this thesis, several experiments were carried out. To perform these experiments, a private cloud infrastructure managed by OpenStack (Openstack Cloud Operating System) was used. The algorithms that compose the QSM-EXTRACTION strategy were implemented in C++ language. In order implement the evaluation of the efficiency of the QSM-EXTRACTION strategy, a collection of syntactically created documents of different sizes was used. The results of the experiments proved the feasibility of using the proposed approach in scenarios typical of cloud.

Keywords: Privacy. Confidentiality. Cloud Computing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	Association Computer Machinery
AES	Advanced Encryption Standard
AICPA	American Institute of Certified Public Accountants
ASCII	American Standard Code for Information Interchange
CBC	Cipher-Block Chaining
CFB	Cipher Feedback
CICA	Canadian Institute of Charted Accountants
CSA	Cloud Security Alliance
DES	Data Encryption Standard
DOS	Denial of Service
ECB	Electrônic Code Book
EDE	Encripta-Decripta-Encripta
FedRAMP	Federal Risk and Authorization Management Program
FRS	Fragmentation-Redundancy-Scattering
HIPAA	Health Insurance Portability and Accountability Act
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
PCI DSS	- Payment Card Industry Data Security Standard
PII	Personally Identifiable Information
SaaS	Software as a Service
SDC	Statistical Disclosure Control
SLA	Service Level Agreements
SM	Semi-identificadores
SSAE	Statement on Standards for Attestation Engagements

SUMÁRIO

1 INTRODUÇÃO

Esta tese apresenta uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem. A ciência por trás dessa abordagem é inspirada nos conceitos da Doutrina do Ser de Hegel. A estratégia QSM-EXTRACTION baseia-se na fragmentação de um arquivo digital em fragmentos denominados objetos de informação, na decomposição desses objetos por meio da extração de suas características e na dispersão dessas características em diferentes serviços de armazenamento em nuvem, permitindo a posterior recuperação desses dados sem perda de informação. A finalidade da estratégia proposta é inviabilizar a reconstrução do arquivo original por parte de um provedor de nuvem que possui apenas parte das características dos objetos de informações que compõem este arquivo. Desta forma, assegura-se a confidencialidade dos dados armazenados em nuvem e, por conseguinte, a privacidade dos proprietários desses dados.

1.1 Motivação

Os seres humanos valorizam a privacidade e a proteção do seu espaço pessoal de vida. Eles certamente não querem que suas informações pessoais estejam acessíveis a pessoas não autorizadas. Porém, os recentes avanços na tecnologia da informação, tais como as redes sociais e os dispositivos portáteis, têm intensificado a geração e divulgação de informações pessoais. Atualmente, uma vasta gama de aplicativos e dispositivos emitem, constantemente, informações sobre localização, relações sociais, hábitos de consumo, comportamento, etc. Segundo Keen et. al. (KEEN, 2012) “Informações pessoais são o novo petróleo, o combustível vital da nossa economia digital”. Para as organizações, dados pessoais sobre os clientes e potenciais clientes são agora um elemento essencial. A capacidade de monitoramento detalhado das ações das pessoas na *Web* e a prática difundida das empresas compartilharem e venderem dados umas para as outras afetam a privacidade de quem acessa a *Web*. O sentido e o valor da privacidade continuam a ser objeto de considerável controvérsia. O debate sobre a privacidade tem evoluído em conjunto com o desenvolvimento da tecnologia da informação.

O desenvolvimento da computação em nuvem trouxe, para o alcance de pequenas e médias empresas, o poder computacional antes acessível apenas a grandes órgãos governamentais e a grandes corporações multinacionais. Isto possibilitou aos usuários dos serviços de nuvem o acesso simples, imediato e ilimitado a recursos computacionais (RAO; YADAV, 2016). Além

disto, nos últimos anos, os serviços de armazenamento de dados em nuvem, tais como Dropbox, Google Drive, Amazon Cloud Drive, Box, iCloud, OneDrive, dentre outros, têm obtido grande popularidade. Assim, muitos usuários utilizam esses serviços para armazenar arquivos pessoais, tais como, fotos, documentos, além de arquivos de áudio e vídeo. Neste casos, o controle dos dados armazenados na nuvem deixa de ser do proprietário do dado e passa a ser do provedor do serviço de armazenamento. Apesar da maioria dos grandes provedores de computação em nuvem estarem passando por rigorosas auditorias para validar a conformidade de seus processos com normas e padrões de segurança, tais como ISO 27001, Statement on Standards for Attestation Engagements (SSAE)-16, - Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act (HIPAA), Federal Risk and Authorization Management Program (FedRAMP), entre outros, o problema de garantir a confidencialidade dos dados armazenados em relação ao provedor de nuvem persiste.

O *National Institute of Standards and Technology* (NIST) define segurança computacional como sendo “a proteção conferida a um sistema de informação automatizado, a fim de atingir os objetivos propostos de preservação da integridade, disponibilidade e confidencialidade dos recursos do sistema de informação (incluindo *hardware*, *software*, *firmware*, informações/dados e telecomunicações)". Esta definição contém 3 conceitos chaves para segurança computacional: confidencialidade, disponibilidade e integridade (GUTTMAN; ROBACK, 1995). A integridade está relacionada à garantia de que os dados manipulados mantenham todas as características originais estabelecidas pelo seu proprietário. A disponibilidade assegura que os dados estejam sempre disponíveis para o uso legítimo. Já a confidencialidade garante que os dados sejam acessados somente por entidades autorizadas pelo seu proprietário. Atualmente, a tríade CIA (*Confidentiality, Integrity and Availability*) – Confidencialidade, Integridade e Disponibilidade – orienta a análise, o planejamento e a implementação da segurança para uma determinada informação que se deseja proteger.

Em particular, a confidencialidade limita o acesso aos dados, impedindo a divulgação de dados não autorizados de forma intencional ou não intencional. Por outro lado, segundo Stallings et. al. (STALLINGS, 2010), a privacidade dos dados assegura que os indivíduos controlam ou influenciam quais informações relacionadas a eles podem ser coletadas e armazenadas por alguém e com quem elas podem ser compartilhadas. Desta forma, confidencialidade e privacidade de dados são conceitos que mantêm uma intensa relação. Ambos relacionam-se com a necessidade de assegurar que o indivíduo controle que informações relacionadas a ele podem

ser divulgadas e quem pode acessá-las. Assim, pode-se entender a confidencialidade como uma ferramenta para se obter privacidade.

Para tratar o problema de assegurar a confidencialidade de dados, várias abordagens foram propostas. Estas abordagens podem ser classificadas em três categorias (SAMARATI, 2014):

- uso de criptografia antes de enviar os dados para o servidor da nuvem;
- fragmentação vertical de dados e dispersão entre vários servidores na nuvem;
- combinação de fragmentação vertical e criptografia.

Existem questões abertas nas três abordagens previamente propostas. Em relação ao uso da criptografia, há uma troca entre a segurança dos dados e o desempenho das consultas (KANTARCIOĞLU; CLIFTON, 2005). Além disso o uso da criptografia impõe a sobrecarga de armazenar e gerenciar chaves criptográficas. Para se utilizar a fragmentação vertical é necessário, inicialmente, definir os fragmentos, separando em fragmentos diferentes os atributos com associação sensível. Contudo, este é um problema NP-difícil (SAMARATI; VIMERCATI, 2010; JOSEPH *et al.*, 2013). Adicionalmente, consultas que envolvam fragmentos diferentes devem ser executadas no cliente, o que pode gerar uma sobrecarga na comunicação entre o cliente e os provedores que armazenam os fragmentos. Já nas soluções mistas, que utilizam uma combinação da fragmentação vertical com a criptografia, somente os atributos sensíveis são encriptados e os atributos que possuem associação sensível são dispersos em fragmentos diferentes (FUGKEAW, 2012). Isto permite a execução de consultas sobre atributos não criptografados, mas ainda exige a decriptação dos atributos sensíveis. Além disto, esta abordagem continua com a necessidade de se determinar os fragmentos, o quê, como mencionado anteriormente, é um problema da categoria não polinomial difícil (NP-difícil) (pode ser reduzido ao problema de coloração do hipergrafo) (AGGARWAL, 2005).

1.2 Definição do Problema

O problema investigado neste trabalho, em sentido mais amplo, é assegurar a confidencialidade de dados armazenados em ambientes de computação em nuvem e, consequentemente, garantir a privacidade do proprietário desses dados.

As premissas aplicadas ao escopo deste problema são as seguintes:

- Os usuários da nuvem devem estar identificados e autenticados para terem acesso aos dados armazenados na nuvem;

- Os provedores de nuvem devem garantir disponibilidade, confiabilidade e integridade dos dados;
- Os provedores de nuvem são considerados “honesto-curiosos", isto é, executam corretamente os protocolos de acesso aos dados, mas têm interesse de inferir e analisar dados (incluindo índices) e o fluxo de mensagens recebidas durante o protocolo de modo a aprender informações adicionais sobre os dados.

O principal problema desta tese é o seguinte: **Como garantir a privacidade dos usuários da nuvem por meio da confidencialidade dos dados armazenados em provedores de serviços em nuvem?**

A hipótese desta tese é a seguinte: **A confidencialidade dos arquivos digitais armazenados em nuvem pode ser obtida, de forma eficiente em relação ao tempo de processamento, por meio da decomposição desses arquivos e dispersão dessas informações em diferentes provedores.**

1.3 Objetivo

O objetivo geral deste trabalho é propor uma estratégia que possa assegurar a confidencialidade dos dados armazenados em provedores de serviços em nuvem. A estratégia proposta deve inviabilizar que estes provedores (“honesto-curiosos") tenham acesso ao conteúdo original dos dados dos clientes. De fato, o quê deve ser armazenado nos provedores não são os dados originais do cliente, mas o resultado de alguma manipulação executada sobre esses dados, de forma que seja possível, posteriormente, obter o dado original a partir do que foi armazenado nos provedores. Desta forma, por meio da confidencialidade dos dados armazenados em nuvem, a estratégia irá assegurar a privacidade dos proprietários desses dados.

Para atingir este objetivo geral, foram definidos os seguintes objetivos específicos:

- investigar as técnicas de anonimização, fragmentação e criptografia de dados;
- investigar as principais métricas e indicadores relacionados a privacidade;
- projetar e implementar a estratégia proposta;
- avaliar o desempenho da estratégia proposta por meio da realização de experimentos.

1.4 Contribuições

As principais contribuições desta tese são:

1. Uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem.
2. Definição do conceito de objeto de informação.
3. Definição das características de um objeto de informação, com base nos conceitos da Doutrina do Ser de Hegel.
4. Implementação da estratégia QSM-EXTRACTION.
5. Avaliação da estratégia QSM-EXTRACTION.

A estratégia QSM-EXTRACTION apresenta ainda as seguintes propriedades:

1. Generalidade: é aplicável a vários formatos de arquivos (documentos texto, som, multimídia).
2. Flexibilidade: pode ser combinada com soluções de dispersão de arquivos (ex.: Rabin, Shamir, AONTRS, etc), fragmentação vertical e criptografia (simétrica ou assimétrica).
3. Simplicidade: a confidencialidade dos dados é obtida com o uso de operações simples de substituição e transposição, que são os blocos básicos de construção de todas as técnicas criptográficas.
4. Eficiência: aplica-se a ambientes de computação em nuvem, onde, em geral, as leituras são mais frequentes que as atualizações. Adicionalmente, a estratégia QSM-EXTRACTION não requer a definição de fragmentos, o que é um problema NP-difícil.
5. Gerenciabilidade: não requer o armazenamento e nem o gerenciamento de chaves criptográficas.
6. Utilidade: a recuperação das informações armazenadas na nuvem ocorre sem nenhuma perda de informação.

1.4.1 Artigos

Os esforços durante o processo de pesquisa para esta tese tornaram possíveis as seguintes publicações:

- Estratégias para Proteção da Privacidade de Dados Armazenados na Nuvem, Eliseu C. Branco Jr., Javam C. Machado e José Maria da Silva Monteiro Filho - Minicurso - XXIX Simpósio Brasileiro de Banco de Dados – SBBD – Curitiba – 2014 Acessível em: http://www.inf.ufpr.br/sbbd-sbsc2014/sbbd/minicursos.htm#minicurso2
- BRANCO, Eliseu Castelo; DE CASTRO MACHADO, Javam; DA SILVA MONTEIRO FILHO, José Maria. (2014) *A strategy to preserve data confidentiality in cloud storage*

services. Workshop de Teses e Dissertações em Banco de Dados (WTDBD) October 6-9, 2014 – Curitiba, PR, Brazil. Acessível em http://www.inf.ufpr.br/sbbd-sbsc2014/sbbd/proceedings/artigos/pdfs/12.pdf

- Branco, Eliseu Castelo, Monteiro, J.M., Machado, J.C., & Reis, R. (2016). *A Flexible Mechanism for Data Confidentiality in Cloud Database Scenarios.* In: *18th International Conference on Enterprise Information Systems* (ICEIS 2016). p 359-368, 25-28 Abril 2016, Roma, Itália. ISBN: 978-989-758-187-8.Acessível em http://www.scitepress.org/DigitalLibrary/PublicationsDetail.aspx?ID=69XiutC734U=&t=1
- BRANCO, Eliseu Castelo; DE CASTRO MACHADO, Javam; DA SILVA MONTEIRO FILHO, José Maria. REIS, R. C. S. (2016) *A New Approach to Preserving Data Confidentiality in the Cloud*. In: Evan Desai (Ed.). 2016. *Proceedings of the 20th International Database Engineering & Applications Symposium*. ACM, New York, NY, USA, 11-13 Julho 2016, Montreal, Canada. ACM. paginas 256-263, ISBN 978-1-4503-4118-9. Acessível em http://dl.acm.org/citation.cfm?id=2938512
- BRANCO, Eliseu Castelo; DE CASTRO MACHADO, Javam; DA SILVA MONTEIRO FILHO, José Maria. REIS, R. C. S. (2016). *A New Mechanism to Preserving Data Confidentiality in Cloud Database Scenarios.* In Livro: *SPRINGER Lecture Notes in Business Information Processing -LNBIP Series.* (aguardando publicação)

1.5 Metodologia Adotada

Para consecução dos objetivos propostos neste trabalho, foi utilizado o método **ProKnow-C** (ENSSLIN *et al.*, 2010). Este método foi criado pelo Laboratório de Metodologia Multicritério de Apoio à Decisão (LabMCDA), do Departamento de Engenharia de Produção e Sistemas da Universidade Federal de Santa Catarina e busca utilizar técnicas de análise bibliométrica, considerando o quê o pesquisador julga relevante para o tema de pesquisa. O principal resultado desta técnica é o seguinte: a validação do método utilizado para construção do conhecimento sobre o assunto pesquisado, bem como a produção de conteúdo para um portfólio bibliográfico relevante e de prestígio científico. O processo metodológico de construção do conhecimento **ProKnow-C** é mostrado na Figura 1.

A seguir são descritas as atividades realizadas em cada etapa do processo da consulta bibliográfica realizada para o levantamento das informações relacionadas ao trabalho da pesquisa.

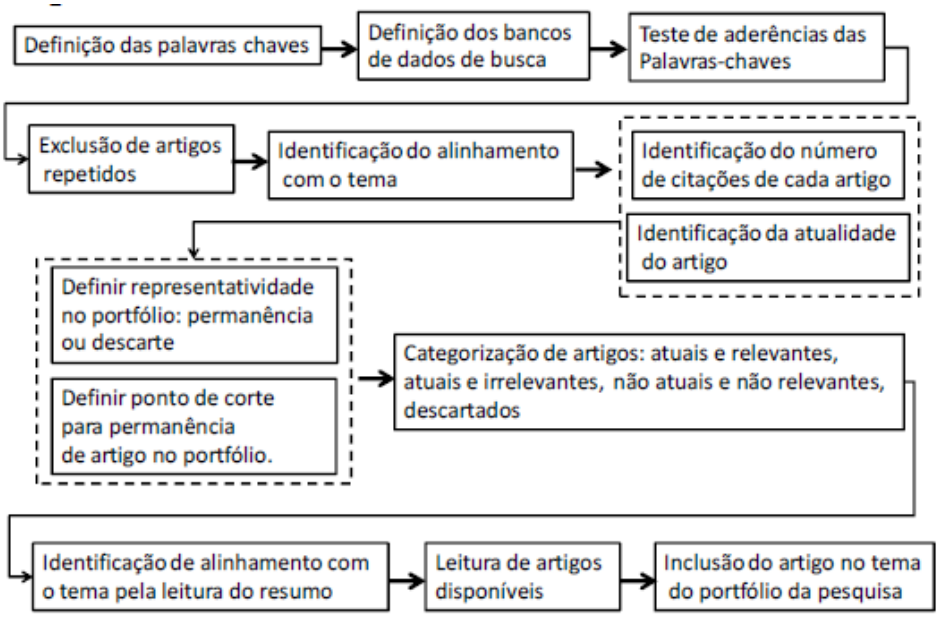

Figura 1 – Processo Metodológico ProKnow-C.

A pesquisa ocorreu no período de 2014 até 2016.

- Definição de palavras-chaves:as palavras chaves utilizadas para pesquisa foram *"cloud computing privacy"*.
- Definição de bancos de dados de busca: os bancos de dados pesquisados foram o *site ACM Digital Library* (Figura 3) e o *site* da biblioteca digital da *IEEE Xplore* (Figura 2).

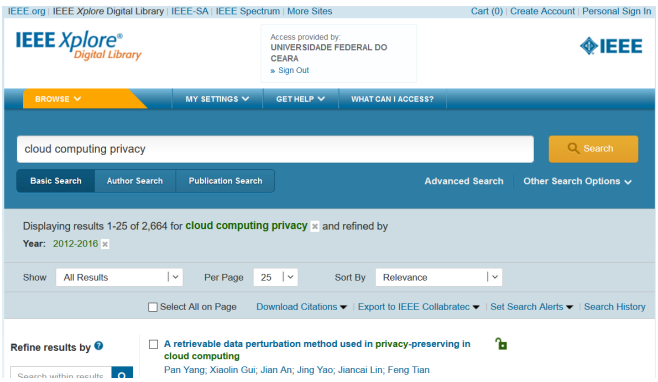

Figura 2 – Biblioteca IEEE Xplore

- Teste de aderência das palavras-chaves: Foram encontradas 2.664 artigos na biblioteca digital da *Institute of Electrical and Electronics Engineers (IEEE)* e 114.844 artigos na biblioteca da *Association Computer Machinery (ACM)*.
- Exclusão de artigos repetidos: aplicação de filtros para seleção dos artigos disponíveis que possuam maior relação com o tema de pesquisa e de relevância significativa. Para tal foi utilizado o *software* **Endnote** que permite exportar as informações do artigos selecionados nas bases (título, resumo, autores, URL, DOI). Esse software também oferece

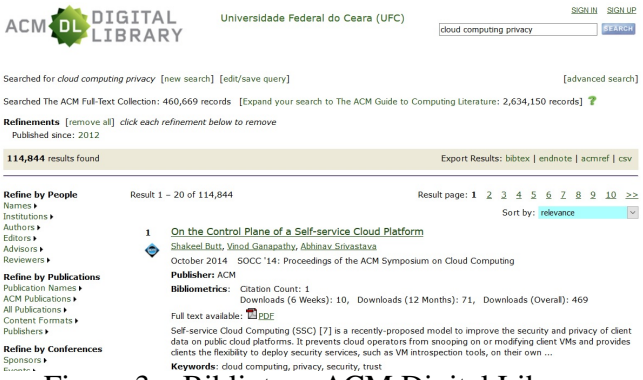

Figura 3 – Biblioteca ACM Digital Library

uma ferramenta de identificação dos artigos duplicados.

- Identificação do alinhamento com o tema: após a filtragem dos artigos, restaram 800 artigos, que foram categorizados de acordo com os temas de pesquisa sobre privacidade de dados.
- Categorização de artigos: os artigos foram classificados em *data privacy* (392), *privacy-preserving* (175), *encryption* (37), *fragmentation* (31). No total, 635 artigos foram selecionados para a etapa seguinte.
- Identificação de alinhamento com o tema pela leitura do resumo: nesta etapa foi feita a leitura do resumo dos artigos para filtragem daqueles com maior afinidade com o tema pesquisado.
- Leitura de artigos: o último passo foi a leitura dos artigos selecionados para verificação final se o seu conteúdo deverá ser incluído no portfólio da pesquisa.
- Inclusão do artigo no tema do portfólio da pesquisa: no último passo do processo, os artigos selecionados são incluídos nas referências bibliográficas da tese.

1.6 Estrutura da Tese

Este estudo está estruturado em oito capítulos, além dessa introdução. O Capítulo 2 apresenta os conceitos fundamentais sobre privacidade, segurança de dados, computação em nuvem e teoria da informação. A técnicas de anonimização de dados são discutidas no Capítulo 3. Essas técnicas procuram assegurar que um sujeito não seja unicamente caracterizado dentro de um conjunto de sujeitos. O Capítulo 4 discute os principais algoritmos relacionados à criptografia simétrica e assimétrica. As técnicas de fragmentação de dados que surgiram no âmbito da confidencialidade dos dados utilizados em sistemas distribuídos são discutidas no Capítulo 5. A principal ideia por trás destas técnicas é a de que o quê deve ser mantido privado

não é apenas a informação dos atributos, mas o relacionamento entre eles. Adicionalmente, este capítulo discute também as técnicas de dispersão de informações. Essas técnicas permitem incrementar a disponibilidade de dados por meio da replicação e distribuição destes dados. O Capítulo 6 apresenta uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem. Os experimentos realizados com a finalidade de avaliar o desempenho da estratégia QSM-EXTRACTION são apresentados e discutidos no Capítulo 7. Os principais trabalhos relacionados com privacidade e segurança de dados armazenados na nuvem são analisados no Capítulo 8. Por fim, o Capítulo 9 apresenta as conclusões desta tese e aponta direções para trabalhos futuros.

2 FUNDAMENTOS TEÓRICOS

2.1 Introdução

A computação em nuvem é uma tecnologia que tem como objetivo proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. A nuvem computacional é um modelo de computação em que dados, arquivos e aplicações residem em servidores físicos ou virtuais, os quais podem ser acessados por meio de uma rede, a partir de qualquer dispositivo compatível (fixo ou móvel), a qualquer hora, de qualquer lugar, sem a necessidade de instalação ou configuração de programas específicos.

Contudo, para que todo o potencial da computação em nuvem possa ser explorado pelas organizações, é de fundamental importância garantir a privacidade dos usuários e a segurança dos dados armazenados na nuvem. O relatório *Top Threats to Cloud Computing*, 2016, produzido pela *Cloud Security Alliance (CSA)* de (ALLIANCE, 2016), lista as 12 maiores ameaças para a computação em nuvem. Em primeiro lugar no *ranking* desta pesquisa ficou **violações de dados** e em oitavo lugar **roubo de dados**. Portanto, não será possível atingir todo o potencial da computação em nuvem sem o desenvolvimento de estratégias que assegurem a proteção da privacidade dos dados de seus usuários.

Este capítulo apresenta conceitos fundamentais sobre privacidade, segurança de dados, computação em nuvem e teoria da informação. Estes conceitos se inter-relacionam quando analisamos o desafio de se preservar a confidencialidade dos dados armazenados na nuvem. A necessidade de assegurar a privacidade das pessoas que produzem dados ou às quais os dados se referem é o que motiva os investimentos em técnicas de proteção de dados armazenados em nuvem, pois a descoberta de informações a partir do conhecimento de dados dispostos na nuvem, por parte de pessoas não autorizadas, representa o início da perda de privacidade.

2.2 Privacidade

O estudo da privacidade abrange disciplinas da filosofia à ciência política, teoria política e legal, ciência da informação e, de forma crescente, engenharia e ciência da computação. Um consenso entre os pesquisadores é que privacidade é um assunto complexo. Primeiramente, privacidade é um conceito relacionado a pessoas. Trata-se de um direito humano, como liberdade, justiça ou igualdade perante a lei. Privacidade está relacionada ao interesse que as pessoas têm

em manterem um espaço pessoal, sem interferências de outras pessoas ou organizações.

O reconhecimento do direito do indivíduo à privacidade está profundamente enraizado na história. De acordo com (LAURANT, 2003) a mais antiga referência à privacidade remonta ao Alcorão e às declarações de Maomé. A Bíblia tem numerosas referências à privacidade. Mais tarde, no século 19, o conceito de privacidade foi estendido à aparência pessoal das pessoas, provérbios, atos, crenças, pensamentos, emoções, sensações, etc.

A formulação do conceito jurídico contemporâneo de privacidade pode ser rastreada até o artigo seminal **O Direito à Privacidade** (WARREN; BRANDEIS, 1890) escrito por Samuel Warren e Louis Brandeis na Harvard Law Review em 1890. Eles enfatizaram o direito de ser deixado em paz, reagindo ferozmente ao uso antiético de câmeras portáteis e modernas máquinas de impressão para facilitar a captura e divulgação pública de informações relativas à vida privada do indivíduo.

A privacidade corporal é outro aspecto importante da vida privada a qual está relacionada com os corpos físicos das pessoas, que originalmente se refere à proteção contra a procedimentos invasivos, como revistas corporais. A implicação da privacidade corporal evoluiu com o desenvolvimento tecnológico, para abranger medidas de proteção contra os exames médicos, como testes genéticos ou testes de drogas.

O *American Institute of Certified Public Accountants (AICPA)* e o *Canadian Institute of Charted Accountants (CICA)* apresentam o conceito de privacidade da seguinte forma: “Privacidade é o conjunto de direitos e obrigações que indivíduos e organizações têm em relação à coleta, uso, retenção e divulgação de informações pessoais".

Segundo Ruth Gavison (GAVISON, 1980), privacidade é uma condição medida em termos de grau de acesso que outros tem a você, em termos de informação, atenção e proximidade. Pare ela, existem, em termos jurídicos, três princípios fundamentais e independentes que compõem a privacidade: o **sigilo** (ou segredo), o **anonimato** e o **isolamento** (ou solidão).

1. Sigilo: um dos entendimentos mais comuns da privacidade é que ela constitui o segredo de certas questões. De acordo com essa visão, a privacidade é violada pela divulgação pública de informações anteriormente ocultas. O professor Alan Westin (WESTIN, 1967) argumentou que o **controle de acesso a informações pessoais** está no coração da privacidade, mas esta é uma característica particular do **sigilo** de Gavison.
2. Anonimato: pode-se definir anonimato como uma qualidade ou condição de alguém que é anônimo, ou seja, não há como identificar o autor, por meio de seus atributos pessoais.

O principal objetivo do anonimato é esconder de terceiros a real identidade de alguém. O anonimato implica no direito de não ser registrado, entendido como direito de não ter imagens gravadas, conversas gravadas etc.

3. Isolamento: implica no direito de não ser monitorado, entendido como direito de não ser visto, ouvido etc.

Embora Gavison sustente que “a coleta, o armazenamento e a informatização da informação"estão dentro de sua concepção, essas atividades muitas vezes não revelam segredos, destroem o anonimato ou impedem a solidão. Portanto, embora Gavison evite a amplitude e a imprecisão das concepções de “acesso limitado", sua tentativa de definir o que significa “acesso"acaba sendo muito limitada (SOLOVE, 2008).

A teoria de Westin destaca as maneiras pelas quais os indivíduos se protegem limitando temporariamente o acesso a si mesmos para outros. De acordo com a teoria de Westin, a privacidade possui quatro estados que podem ser pensados como mecanismos de privacidade, que são, os significados pelos quais a privacidade é mantida. Estes estados são a solidão, a intimidade, o anonimato e a reserva. Ele também define quatro funções ou objetivos da privacidade: autonomia pessoal, liberação emocional, autoavaliação, e comunicação limitada e protegida (WESTIN, 1967). Um valor adicional fortemente relacionado com privacidade é **autonomia**, que é a habilidade de tomar decisões na vida livre da influência ou controle de outros (II, 2015).

Daniel Solove propôs uma taxonomia para privacidade que consiste em quatro grupos de atividades: (1) coleta de informações, (2) processamento de informações, (3) disseminação de informações, e (4) invasão. Cada grupo contém uma variedade de atividades que podem criar problemas de privacidade. A taxonomia é apresentada na Tabela 1, a seguir (SOLOVE, 2008).

Tabela 1 – Taxonomia de privacidade de dados

Grupo de Atividades	Atividades
1-coleta de informações	vigilância, interrogatório
2-processamento de informações	agregação, identificação, insegurança, uso secundário, exclusão
3-disseminação de informações	brecha de confidencialidade, divulgação, exposição, chantagem, apropriação, distorção
4-invasão	intrusão, interferência nas decisões

Existe necessidade de privacidade, independentemente do meio onde o sujeito esta inserido, a não ser que ele a desconheça ou não se importe com ela. A privacidade encontra

uma barreira para a sua existência no mundo virtual, na facilidade da transmissão da informação. Quando conectado a uma rede de computadores, um dispositivo (computador, *notebook*, *tablet*, celular, etc) pode estar vulnerável a todo tipo de ataque externo. As informações contidas nesse dispositivo podem ser acessadas e divulgadas para o resto da rede.

A privacidade é um termo abrangente que envolve e utiliza a segurança da informação. Nesse caso, essa segurança é necessária para os dispositivos e para os meios de transmissão de informação. No mundo virtual, a segurança relaciona-se intensamente com os dados nele armazenados. Portanto, a segurança é utilizada para garantir nesse meio certos aspectos que são apresentados no ambiente físico, como confidencialidade, autenticação, integridade, não repudio, controle de acesso e disponibilidade (STALLINGS, 2010).

A privacidade pode ser explicada com o auxílio de diferentes dimensões. Privacidade da pessoa, privacidade do comportamento da pessoa, privacidade das comunicações da pessoa e privacidade dos dados pessoais são as quatro principais dimensões da privacidade (CLARKE, 1999), (WACKS, 2010).

2.3 Segurança e Privacidade de Dados

O NIST define segurança computacional como sendo “a proteção conferida a um sistema de informação automatizado, a fim de atingir os objetivos propostos de preservação da integridade, disponibilidade e confidencialidade dos recursos do sistema de informação (incluindo *hardware*, *software*, *firmware*, informações/dados e telecomunicações)". Esta definição contém 3 conceitos chaves para segurança computacional: confidencialidade, disponibilidade e integridade (GUTTMAN; ROBACK, 1995).

1. Integridade é uma propriedade que assegura que a informação manipulada mantenha todas as características originais estabelecidas pelo seu proprietário, incluindo o controle de mudanças e a garantia do seu ciclo de vida (nascimento,manutenção e destruição). Assim. a integridade garante que a mensagem recebida é a mesma que foi transmitida e que não houve alteração de seu conteúdo de maneira intencional ou não. Integridade na computação em nuvem significa que os dados e sistemas serão acessados pelos usuários autorizados no tempo certo e de forma confiável. Algumas técnicas para assegurar integridade são serviços de filtragem de pacotes via *firewall*, gerenciamento de segurança por meio de configurações de protocolos, sistemas e tecnologias nas comunicações e serviços de detecção de intrusos.
2. Disponibilidade é uma propriedade que assegura que a informação esteja sempre dispo-

nível para o uso legítimo, ou seja, por aqueles usuários autorizados pelo proprietário da informação. Neste sentido, a disponibilidade se refere a elementos que criam estabilidade nas redes e sistemas, assegurando a conectividade quando necessária, permitindo que usuários autorizados possam acessar a rede e os sistemas. A confiabilidade de um sistema é definida como a sua probabilidade de estar operacional continuamente, durante um intervalo específico [0, t], dado que ele estava operacional em t = 0 (SIEWIOREK; SWARZ, 1992). A definição de confiabilidade de sistemas não leva em conta os tempos necessários para executar reparos ou processos de recuperação, necessários para restaurar o estado operacional de sistemas no evento de falhas. Apenas o tempo de operação contínua, sem interrupções, é considerado. A confiabilidade está diretamente relacionada ao tempo médio para falhar (*MTTF - Mean Time To Failure*). Quando sistemas ou seus elementos constituintes falham, normalmente, são reparados ou trocados por novos sistemas ou elementos, respectivamente. Assim sendo, o tempo de vida de um sistema ou elemento particular compreende períodos alternados de operação e falha. Considerando esse comportamento, pode-se medir os seguintes valores de recuperabilidade de sistemas: *MTBF - Mean Time Between Failures* - tempo médio entre falhas e *MTTR - Mean Time To Repair* - tempo médio para reparo, onde *MTBF = MTTF + MTTR* (SIEWIOREK; SWARZ, 1992). Em resumo, o tempo médio entre falhas refere-se ao tempo transcorrido entre uma falha e outra (incluindo o tempo necessário para reparo/restauração) enquanto o tempo médio para reparo engloba o intervalo que inicia no instante em que uma falha é detectada até o instante em que o sistema ou seus elementos constituintes sejam completamente reparados ou trocados. A disponibilidade de um sistema é a sua probabilidade de estar operacional no instante de tempo t (SIEWIOREK; SWARZ, 1992). Utilizando-se *MTTF* e *MTTR*, pode-se definir a disponibilidade de sistemas como uma função da frequência de ocorrência de falhas e do tempo necessário para seu reparo ou troca, isto é, a disponibilidade é uma função direta do *MTTF* e *MTTR* (especificamente, disponibilidade é *MTTF/(MTTF+MTTR)*) ou disponibilidade é tempo operando / (tempo operando + tempo parado) (SIEWIOREK; SWARZ, 1992).

3. Confidencialidade é uma propriedade que limita o acesso a informação tão somente às entidades legítimas, ou seja, àquelas autorizadas pelo proprietário da informação. Neste sentido, confidencialidade diz respeito à prevenção contra a divulgação de dados não autorizados de forma intencional ou não intencional. Segundo Krutz (KRUTZ; VINES,

2010), confidencialidade em sistemas de nuvem envolve áreas como:

- direitos de propriedade intelectual: são direitos sobre invenções, produções artísticas ou literárias que estão protegidos por leis de *copyright*;
- canais secretos: são *links* de comunicação não intencionais ou não autorizados que permitem troca de informações. Podem ser criados por meio de mensagens de temporização ou uso não apropriado de mecanismos de armazenamento;
- análise de tráfego de rede: é uma forma de violação de confidencialidade que pode ocorrer pela análise do volume, taxa, origem e destino de mensagens da rede;
- encriptação: envolve criptografar mensagens para que só possam ser lidas por usuários autorizados que possuam a chave para descriptografar as mensagens;
- inferência: normalmente está associada a segurança de banco de dados. Trata-se da capacidade de uma entidade utilizar e correlacionar informações em um nível de segurança para descobrir informações protegidas em um nível maior de segurança.

Desta forma, atualmente, a tríade CIA (*Confidentiality, Integrity and Availability*) – Confidencialidade, Integridade e Disponibilidade – orienta a análise, o planejamento e a implementação da segurança para uma determinada informação que se deseja proteger. Em particular, a confidencialidade assegura que os dados somente serão acessados por entidades autorizadas pelo seu proprietário. Desta forma, deve-se impedir divulgação de dados não autorizados de forma intencional ou não intencional. Neste sentido, a confidencialidade de dados relaciona-se às técnicas de controle de acesso, encriptação, anonimização e fragmentação. O controle de acesso é a habilidade de permitir ou negar a utilização de um objeto (uma entidade passiva, como um sistema ou arquivo) por um sujeito (uma entidade ativa, como um indivíduo ou um processo), sendo composto pelos processos de autenticação, autorização e auditoria (*accounting*) . A autenticação identifica quem acessa o sistema, a autorização determina o quê um usuário autenticado pode fazer, e a auditoria diz o quê o usuário fez. A criptografia envolve as técnicas pelas quais a informação pode ser transformada da sua forma original para outra ilegível, de forma que possa ser conhecida apenas por alguém autorizado. As estratégias de anonimização de dados buscam assegurar que um sujeito não seja unicamente caracterizado dentro de um conjunto de sujeitos. Já a fragmentação de dados busca separar, em fragmentos diferentes, atributos que utilizados em conjunto possam identificar univocamente um sujeito em conjunto de sujeitos.

Por outro lado, segundo Stallings et. al. (STALLINGS, 2010), a privacidade dos

dados assegura que os indivíduos controlam ou influenciam quais informações relacionadas a eles podem ser coletadas e armazenadas por alguém e com quem elas podem ser compartilhadas. Neste caso, a privacidade de dados na Internet é a habilidade de um usuário ou organização controlar que informações eles revelam sobre si próprios na Internet, ou seja, controlar quem pode acessar qual informação e de que forma isto pode ocorrer.

Desta forma, confidencialidade e privacidade de dados são conceitos que mantêm uma forte relação. Ambos relacionam-se com a necessidade de assegurar que o indivíduo controle que informações relacionadas a ele podem ser divulgadas e quem pode acessá-las. Neste caso, pode-se entender a confidencialidade como uma ferramenta para se obter privacidade.

A coleta de dados na Internet sem o conhecimento ou a previa autorização do usuário afronta o direito à privacidade, podendo acarretar danos irreparáveis ao usuário, o qual dispõe de poucos meios para impedir tal prática. Derivado do direito à privacidade podemos extrair o direito ao sigilo de informações. O não cumprimento deste direito decorrente da comercialização de Informações Pessoais Identificáveis *Personally Identifiable Information (PII)*, que são todas as informações relativas a uma determinada pessoa, desde características físicas até hábitos dos mais variados, de modo que do cruzamento desses dados seja possível traçar um verdadeiro perfil da respectiva pessoa - é um autêntico atentado à privacidade da pessoa humana. Assim sendo, as políticas de privacidade devem de forma bastante clara informar ao usuário sobre o tipo de informações que serão coletadas, o modo como será realizado a coleta dos dados, outrossim, como eles serão gerenciados, os motivos pelos quais os armazenam em seus bancos de dados, a possibilidade de cruzamento das informações coletadas junto a terceiros etc (VENKATARAMANAN; SHRIRAM, 2016).

Neste contexto, a proteção de dados está relacionada ao gerenciamento de informações pessoais. De modo geral, informações pessoais descrevem fatos, comunicações ou opiniões relacionadas a um indivíduo, as quais ele desejaria manter em segredo, controlando sua coleta, uso ou compartilhamento. Informações pessoais podem ser associadas a um indivíduo específico tais como nome, CPF, número do cartão de crédito, número da identidade. Algumas informações pessoais são consideradas mais sensíveis do que outras. Por exemplo, informações sobre saúde (registros médicos) são consideradas sensíveis em todas as jurisdições. Por exemplo, o conteúdo do prontuário médico só poderá ser revelado a terceiros se houver a autorização do paciente, conforme estabelece o artigo 5º da Resolução CFM n.º 1605/2000, ou se houver a anuência do Conselho Regional de Medicina da jurisdição, ex vi do artigo 8º do mesmo diploma, bem como

autorização judicial. Informações biométricas e informações sobre avaliação de desempenho no trabalho também são consideradas sensíveis. Este tipo de informação necessita de proteção adicional em relação à privacidade e segurança.

Na proporção em que grandes volumes de informações pessoais são transferidos para a nuvem, cresce a preocupação de pessoas e organizações sobre como estes dados serão armazenados e processados. O fato dos dados estarem armazenados em múltiplos locais, muitas vezes de forma transparente em relação à sua localização, provoca insegurança quanto ao grau de privacidade a que estão expostos.

A terminologia para tratar questões de privacidade de dados na nuvem inclui a noção de controlador do dado, processador do dado e sujeito proprietário do dado. Estes conceitos serão discutidos a seguir (PEARSON, 2013), (DETERMANN, 2015):

- Controlador de Dado: Uma entidade (pessoa física ou jurídica, autoridade pública, agência ou organização) que sozinha ou em conjunto com outros, determina a maneira e o propósito pela qual as informações pessoais são processadas.
- Processador de Dado: Uma entidade (pessoa física ou jurídica, autoridade pública, agência ou organização) que processa as informações pessoais de acordo com as instruções do Controlador de Dado.
- Sujeito do Dado: Um indivíduo identificado ou identificável ao qual a informação pessoal se refere, seja por identificação direta ou indireta (por exemplo por referência a um número de identificação ou por um ou mais fatores físicos, psicológicos, mentais, econômicos, culturais ou sociais).

Objetivos de proteção da privacidade de dados tem sido propostos adicionalmente aos objetivos de proteção da segurança das informações. Estes objetivos são descritos a seguir (DANEZIS *et al.*, 2015):

- Intervenibilidade: este objetivo significa que os proprietários dos dados devem estar habilitados a intervir a qualquer tempo durante o ciclo de vida do dado, podendo retirar ou alterar seu consentimento a qualquer momento.
- Desvinculação: este objetivo explica que dados de múltiplas fontes não devem ser combinados de forma a violar a privacidade dos usuários dos dados.
- Transparência: este objetivo significa que as partes interessadas devem ser informadas sobre o que acontece com os seus dados pessoais durante cada fase do ciclo de vida do dado.

Diversos estudos têm sido realizados para investigar os problemas relacionados à privacidade e segurança em ambientes de computação em nuvem. Liu estudou o assunto nas áreas de saúde e energia elétrica (LIU *et al.*, 2012). Gruschka e Jensen sugeriram modelar o ecossistema de segurança baseado em três participantes do ambiente de nuvem (GRUSCHKA; JENSEN, 2010):

- o usuário do serviço, que é a pessoa ou máquina (ex. usuário do Facebook, sensor de presença de câmera de vigilância) que está utilizando o serviço para transferência e recepção de dados para a nuvem;
- a instância do serviço, que é o sistema ou serviço (ex. Gmail, Youtube) que está sendo executado no ambiente de nuvem;
- o provedor do serviço (ex. Amazon, Google), que é quem disponibiliza a infraestrutura de rede, *hardware* e *software* para execução dos serviços ou sistemas na nuvem.

Sendo os ataques classificados em seis categorias, descritos na Tabela 2 . Em cada categoria representa-se a origem e o destino dos ataques. Por exemplo, Usuário $\rightarrow$ Provedor indica ataques de usuários a provedores de nuvem.

Tabela 2 – Tipos de Ataques na Nuvem

	Usuário do Serviço	Instância do Serviço	Provedor de Nuvem
Usuário do Serviço		Usuário $\rightarrow$ Serviço	Usuário $\rightarrow$ Provedor
Instância do Serviço	Serviço $\rightarrow$ Usuário		Serviço $\rightarrow$ Provedor
Provedor de Nuvem	Provedor $\rightarrow$ Usuário	Provedor $\rightarrow$ Serviço	

Spiekermann classificou 3 domínios técnicos para o armazenamento de dados na nuvem: esfera do usuário, esfera da organização e esfera dos provedores de serviços. O autor relacionou áreas de atividades que causam grande preocupação em relação à privacidade de dados com as 3 esferas de privacidade, conforme ilustrado na Tabela 3 (SPIEKERMANN; CRANOR, 2009):

O NIST propõe uma terminologia para a classificação de problemas relacionados à privacidade e à segurança em ambientes de computação em nuvem (JANSEN; GRANCE, 2011). A terminologia proposta contém nove áreas: governança, conformidade, confiança, arquitetura, gerenciamento de acesso e identidade, isolamento de *software*, proteção de dados, disponibilidade e resposta a incidentes. As 9 áreas de problemas de privacidade do NIST serão

Tabela 3 – Esferas de Influência associadas às preocupações com privacidade de dados

Esfera de Influência	**Preocupação com a Privacidade de Dados**
Esfera do Usuário	Coleção e armazenamento de dados não autorizados Execução não autorizada de dados Exposição de dados Entrada indesejada de dados
Esfera da Organização	Exposição de dados Mau julgamento a partir de dados parciais ou incorretos Acesso não autorizado a dados pessoais Uso não autorizado de dados por terceiros envolvidos na coleta dos dados ou por outras organizações com as quais os dados foram compartilhados
Esfera dos provedores	Uso não autorizado de dados por terceiros envolvidos na coleta dos dados Uso não autorizado por outras organizações com as quais os dados foram compartilhados Acesso não autorizado de dados pessoais Erros acidentais ou deliberados em dados pessoais Mau julgamento a partir de dados parciais ou incorretos

detalhadas a seguir:

1. Governança: trata do controle e supervisão pela organização sobre políticas, procedimentos e padrões para desenvolvimento de aplicativos e aquisição de serviços de tecnologia da informação, bem como a concepção, implementação, teste, uso e monitoramento de serviços implantados ou contratados.
2. Conformidade: refere-se à responsabilidade da organização de operar de acordo com as leis, regulamentos, padrões e especificações estabelecidos. Vários tipos de leis e regulamentações de segurança e privacidade existem dentro de diferentes países nos níveis nacional, estadual e local, tornando a conformidade uma questão potencialmente complicada para a computação em nuvem.
3. Confiança: sob o paradigma de computação em nuvem, uma organização entrega ao contratante (provedor de serviços de nuvem) o controle direto sobre muitos aspectos de segurança e privacidade e, ao fazê-lo, confere um alto nível de confiança ao provedor de nuvem. Ao mesmo tempo, as agências federais têm a responsabilidade de proteger as informações proporcionalmente ao risco e magnitude dos danos resultantes do acesso, uso, divulgação, interrupção, modificação ou destruição não autorizados, independentemente das informações serem coletadas ou mantidas por uma agência ou um contratante.
4. Arquitetura: a arquitetura do *software* e do *hardware* usados para fornecer serviços em nuvem pode variar significativamente entre provedores de nuvem pública para qualquer

modelo de serviço específico. A localização física da infra-estrutura é determinada pelo provedor de nuvem, assim como o design e a implementação da confiabilidade, *pooling* de recursos, escalabilidade e outros recursos necessários na estrutura de suporte. Os aplicativos são construídos nas interfaces de programação de serviços acessíveis pela Internet, que normalmente envolvem vários componentes da nuvem que se comunicam entre si por meio de interfaces de programação de aplicativos (APIs).

5. Gerenciamento de acesso e identidade: Dados sensíveis e a privacidade da informação tornaram-se cada vez mais uma área de preocupação para as organizações. Os aspectos de identidade e autenticação do gerenciamento de identidade implicam o uso, manutenção e proteção de dados pessoais coletados dos usuários. Impedir o acesso não autorizado a recursos de informação na nuvem também é uma consideração importante. Uma questão recorrente é que a estrutura de identificação e autenticação organizacional pode não se estender naturalmente para uma nuvem pública e pode não ser possível ampliar ou alterar a estrutura existente para oferecer suporte a serviços de identificação e autenticação em nuvem.
6. Isolamento de *software*: São necessários altos graus de multi-inquilinato em um grande número de plataformas para que a computação em nuvem alcance a flexibilidade imaginada de provisionamento sob demanda de serviços confiáveis e os benefícios e eficiência de custo devido à economia de escala. Para atingir as altas escalas de consumo desejadas, os provedores de nuvem devem garantir a entrega dinâmica e flexível do serviço e o isolamento dos recursos dos inquilinos. Aplicativos implantados em máquinas virtuais convidadas permanecem suscetíveis a ataques e comprometimento de recursos, bem como suas contrapartes não virtualizadas.
7. Proteção de dados: os dados armazenados em uma nuvem pública normalmente residem em um ambiente compartilhado com dados de outros clientes. As organizações que colocam dados sensíveis e regulados em uma nuvem pública devem, portanto, ter em conta os meios pelos quais o acesso aos dados é controlado e os dados são mantidos seguros. Preocupações semelhantes existem para dados migrados dentro ou entre nuvens.
8. Disponibilidade: trata-se da extensão em que o conjunto completo de recursos computacionais de uma organização é acessível e utilizável. A disponibilidade pode ser afetada temporária ou permanentemente, e uma perda pode ser parcial ou completa. Os ataques de negação de serviço, interrupções de equipamentos e desastres naturais são ameaças à

disponibilidade. A preocupação é que o tempo de indisponibilidade, que é imprevisível, possa afetar as operações da organização.

9. Resposta a incidentes: compreende um método organizado para lidar com as consequências de um ataque contra a segurança de um sistema de computador. O papel do provedor de nuvem é vital na realização de atividades de resposta a incidentes, incluindo verificação de incidentes, análise de ataques, contenção, coleta e preservação de dados, remediação de problemas e restauração de serviços.

Em relação a proteção de dados, o NIST recomenda que sejam avaliadas a adequação de soluções de gerenciamento do provedor de nuvem para dados organizacionais envolvidos e a capacidade de controlar o acesso e proteger dados em repouso, em movimento e em uso, incluindo protocolos para descarte dos dados.

A seguir serão apresentados as principais abordagens para desenvolvimento de sistemas que considera a preservação da privacidade como uma característica importante no planejamento do sistema.

2.4 Privacidade por *Design*

Embora a tecnologia da informação seja normalmente vista como a causa dos problemas de privacidade, também existem várias maneiras como ela pode ajudar a resolver estes problemas. Existem regras, diretrizes ou melhores práticas que podem ser usados para projetar sistemas de preservação da privacidade. Tais possibilidades variam de metodologias de projeto eticamente informadas ao uso de criptografia para proteger informações pessoais de uso não autorizado. A estratégia de **Privacidade por *Design*** (PbD) proposta por (CAVOUKIAN, 2011) define um conjunto de sete princípios que orientam o desenho do sistema descritos a seguir (BORRETT *et al.*, 2016):

1. Proativo não reativo; Evitável não remediável. PbD não espera pela ocorrência de riscos ou infrações de privacidade para iniciar procedimentos de prevenção ou mitigação.
2. Privacidade como configuração padrão. Os dados pessoais são mandatoriamente e automaticamente protegidos em qualquer sistema de gerenciamento de dados ou prática de negócios.
3. Privacidade embutida no projeto. A privacidade é um componente essencial da funcionalidade básica de um sistema por ser incorporada na arquitetura.
4. Funcionalidade total: soma positiva, soma não zero. Por estar embutida na arquitetura

do sistema, a privacidade não deve restringir o acesso aos dados; ambos privacidade e segurança podem ser acomodados.

5. Segurança de ponta a ponta: proteção completa do ciclo de vida. Ao incorporar a privacidade antes do primeiro elemento de informação ser adicionado ao sistema, todos os dados são protegidos até serem destruídos no final do processo.
6. Visibilidade e transparência. As partes interessadas estão seguras de que seus dados estão protegidos de acordo com promessas e objetivos declarados; os componentes e funcionamento do sistema são visíveis e transparentes.
7. Respeito pela privacidade do usuário. Ao exigir que a concepção dos sistemas coloque os interesses dos usuários em primeiro lugar, habilita opções de fácil utilização e permanece centrada no usuário.

Os valores de privacidade são levados em consideração como uma forma de construção de interfaces mais amigáveis nos sistemas de informação. Estes princípios tem como ponto central o lema de que a proteção dos dados precisa ser tratada de forma proativa, ao invés de reativa, fazendo com que a privacidade por *design* seja preventiva e não simplesmente uma medida paliativa. Avança a visão de que a privacidade não pode ser assegurada unicamente pela conformidade com as estruturas regulatórias, mas deve se tornar o modo de operação padrão de uma organização. A proteção do dado deve ser uma preocupação central em todas as fases do ciclo de vida de desenvolvimento do *software*. (CLARKE, 2009) propõe uma abordagem semelhante, sugerindo "um processo sistemático para avaliação dos potenciais efeitos da privacidade em uma iniciativa, projeto ou sistema proposto".

Algumas soluções específicas para privacidade por *design* se propõem a aumentar o nível de consciência e consentimento do usuário. Estas soluções podem ser vistas como uma tentativa de aplicar a noção de "consentimento informado" para questões de privacidade com uso de tecnologia (PIETERS, 2011). A reponsabilização do usuário na tomada de decisão de definir o nível de privacidade dos seus dados esbarra na capacidade limitada de lidar com tais escolhas, e se a interface proporcionar muitas opções de configuração de privacidade, pode facilmente provocar um problema da sobrecarga de informação (HOVEN *et al.*, 2012). Abordagens de *design* sócio-técnico têm sido propostas nos últimos anos para modelagem de sistemas utilizando princípios de privacidade por *design*. (DEGELING *et al.*, 2016; NOTARIO *et al.*, 2015; HERRMANN *et al.*, 2016)

2.5 Computação em Nuvem

Computação em nuvem *Cloud Computing* é uma tendência recente de tecnologia cujo objetivo é proporcionar serviços de Tecnologia da Informação (TI) sob demanda com pagamento baseado no uso. Tendências anteriores a computação em nuvem foram limitadas a uma determinada classe de usuários ou focadas em tornar disponível uma demanda específica de recursos de TI, principalmente de informática (BUYYA *et al.*, 2010). Apesar de ser uma tecnologia recente, o conceito de Computação em Nuvem é antigo. O termo *Cloud Computing* surgiu pela primeira vez em 1997, em uma palestra acadêmica ministrada por Ramnath Chellappa. Porém, o conceito é associado ao nome de John Mccarthy, pioneiro na tecnologia de Inteligência Artificial e criador da linguagem de programação LISP. Em 1960, ele disse que “a computação pode algum dia ser considerada como uma utilidade pública". A origem do termo vem dos diagramas das antigas redes de dados ISDN (*Services Digital Network*, ou rede de serviços digitais) e *Frame Relay*, projetadas pelas operadoras de telefonia. A interligação entre ambas era mostrada por desenhos de nuvens, para sinalizar algo que estava fora do alcance das empresas (HU, 2015).

Para que seja possível discutir em detalhes os principais aspectos relacionados à confidencialidade dos dados armazenados na nuvem, é preciso definir o que é computação em nuvem. Existem várias definições para computação em nuvem. Contudo, neste trabalho, a definição utilizada será a seguinte (HON *et al.*, 2011):

- a computação em nuvem fornece acesso flexível, independente de localização, para recursos de computação que são rapidamente alocados ou liberados em resposta à demanda;
- serviços (especialmente infraestrutura) são abstraídos e virtualizados, geralmente sendo alocados como um *pool* de recursos compartilhados com diversos clientes;
- tarifas, quando cobradas, geralmente são calculadas com base no acesso, de forma proporcional, aos recursos utilizados.

As aplicações da nuvem incluem processamento e integração de dados, serviços de armazenamento e comunicação, que são normalmente disponibilizados sob demanda e taxados com base no uso. A movimentação de dados e programas para a nuvem aumenta o poder de processamento e armazenamento dos usuários e oferece as seguintes vantagens:

1. Flexibilidade ilimitada para acessar milhões de bancos de dados e programas na nuvem, com possibilidade de combinar e customizar serviços.
2. Novas possibilidade de trabalho colaborativo entre os usuários da nuvem, por meio de

compartilhamento *online* de dados e aplicações na nuvem.

3. Melhor confiabilidade e segurança no armazenamento dos dados na nuvem, em relação ao armazenamento dos dados nos dispositivos dos usuários, que podem ser perdidos ou roubados ou apresentarem falhas de hardware que resultem em perda de dados.
4. Ubiquidade: usuários podem acessar seus dados de qualquer lugar e a qualquer momento, desde que estejam conectados à internet.
5. Simplificação do *hardware* dos dispositivos dos usuários, que necessitam de menor poder de processamento e armazenamento de dados.

A computação em nuvem é composta de sete características fundamentais: três modelos de serviço e quatro abordagens de implantação (MELL; GRANCE, 2011). Os modelos de serviços de nuvem são disponibilizados normalmente nas seguintes plataformas:

1. Plataforma como Serviço – PaaS: provimento de recursos de *hardware*, tipicamente máquinas virtuais, que podem ser carregadas com o sistema operacional e *softwares* dos usuários.
2. Infraestrutura como Serviço – Iaas: provimento de infraestrutura de *hardware*, incluindo plataformas virtuais, interconectividade, etc, onde as aplicações dos usuários podem ser instaladas.
3. *Software* como Serviço – SaaS: disponibilização de toda infraestrutura de *hardware*, software para execução de aplicações de *software* na Internet pelos usuários de nuvem.

A implantação dos modelos pode seguir uma abordagem (i) pública – os recursos são dinamicamente disponibilizados na Internet para o público em geral, (ii) privada – os recursos são acessíveis apenas dentro de uma determinada corporação ou uma instituição cientifica, (iii) comunitária – recursos compartilhados por organizações com interesses comuns ou (iv) híbrida – qualquer tipo de combinação entre as categorias anteriores.

O mecanismo de virtualização é amplamente utilizado na camada de infraestrutura da nuvem, porque permite a flexibilização do uso da camada de *hardware*. Máquinas virtuais proveem capacidade de processamento independente em máquinas isoladas, que podem ser instanciadas e destruídas sob demanda. Dessa forma, o ambiente de máquinas virtuais constitui uma base bastante adequada para a construção de infraestruturas de computação em nuvem (GROBAUER *et al.*, 2011). O modelo de serviço operacional, em conjunto com as tecnologias utilizadas para prover serviços do ambiente de computação em nuvem, apresenta diferentes níveis de riscos em comparação aos ambientes tradicionais de tecnologia de informação (BARDIN

et al., 2009). O provimento de recursos sob demanda para o processamento e armazenamento intensivo de dados está sujeito a falhas de segurança, abusos com relação à privacidade, violação de direitos autorais, etc.

As aplicações de maior sucesso e visibilidade na nuvem são serviços desenvolvidos para consumidores pessoais, tais como *e-mails*, redes sociais, e mundos virtuais (RAGHAVAN *et al.*, 2007). Para manter estes serviços, *terabytes* de dados são coletados, sendo que a maior parte são informações pessoais, que precisam de privacidade. As formas como os provedores de serviços de nuvem irão tratar os problemas de privacidade serão um fator decisivo para o desenvolvimento da tecnologia de computação em nuvem.

As garantias fornecidas pelo provedor para seus consumidores podem ser definidas por meio de contratos em nível de serviço (*Service Level Agreements (SLA)*). A maioria dos provedores de nuvem oferecem garantias de tempo de atividade (*uptime*) em seus SLAs. O SLA é um documento contratual cujas clausulas, muitas vezes, não conseguem expressar as regras em nível computacional (TESHOME *et al.*, 2016). Por exemplo: como fornecer garantias de tempo de atividade para uma transação se esta envolve um fluxo de dados por meio da Internet? Um dos problemas fundamentais causados pela adoção de computação em nuvem é o asseguramento de que os serviços e informações armazenados na nuvem serão mantidos de acordo com os acordos de serviços firmados com os usuários. A política de segurança dos provedores de serviços de computação em nuvem deve garantir que terceiros não irão ter acesso à plataforma de processamento, memória ou arquivos de disco de um usuário da nuvem.

Os requisitos de segurança de dados podem variar de acordo com o modelo de serviço fornecido pelo provedor de nuvem (Software as a Service (SaaS), Platform as a Service (PaaS) ou Infrastructure as a Service (IaaS)), o modelo de nuvem (pública, privada, comunitária ou híbrida) e o nível de tolerância a riscos da organização (WINKLER, 2011). Os dados na nuvem podem estar em repouso, armazenados nos servidores da nuvem, ou em movimento, sendo processados ou transferidos do disco para a memória ou de uma máquina para outra através da rede. Preocupações com a segurança dos dados devido à diminuição do controle pelo proprietário dos dados, quando estes não estão mais sendo gerenciados dentro das fronteiras da organização têm motivado a pesquisa de soluções de segurança e privacidade para disponibilização de conteúdo sensível em nuvens públicas e mistas.

Existem técnicas de segurança de dados utilizadas em ambientes tradicionais de TI que podem ser utilizadas em ambiente de Computação em Nuvem, tais como autenticação,

gestão de identidade, controle de acesso, criptografia, descarte seguro, verificação da integridade e mascaramento de dados para geração de base de testes.

Além dos riscos e ameaças inerentes aos ambientes tradicionais de TI nas organizações, o ambiente de Computação em Nuvem possui seu próprio conjunto de problemas de segurança, classificados por Krutz em sete categorias: segurança de rede, interfaces, segurança de dados, virtualização, governança, conformidade e questões legais (KRUTZ; VINES, 2010). Os princípios fundamentais da segurança da informação: confidencialidade, integridade e disponibilidade definem a postura de segurança de uma organização e influenciam os controles e processos de segurança que podem ser adotados para minimizar os riscos. Estes princípios se aplicam também aos processos executados na nuvem.

O processo de desenvolvimento e implantação de aplicações para a plataforma de Computação em Nuvem, que seguem o modelo *software* como um serviço (*SaaS*), deve considerar os seguintes aspectos de segurança em relação aos dados armazenados na nuvem (SUBASHINI; KAVITHA, 2011):

- Segurança dos dados: no modelo *SaaS*, os dados são armazenados fora dos limites da infraestrutura de tecnologia da organização, por isso o provedor de nuvem deve prover mecanismos que garantam a confidencialidade dos dados. Por exemplo, isso pode ser feito utilizando técnicas de criptografia forte e mecanismos de ajuste preciso para autorização e controle de acesso.
- Segurança da rede: os dados do cliente são processados pelas aplicações *SaaS* e armazenados nos servidores da nuvem. A transferência dos dados da organização para a nuvem deve ser protegida para evitar perda de informação sensível. Por exemplo, pelo uso de técnicas de encriptação do tráfego de rede, tais como *Secure Socket Layer (SSL)* e *Transport Layer Security (TLS)*.
- Localização dos dados: no modelo *SaaS*, o cliente utiliza as aplicações *SaaS* para processar seus dados, mas não sabe onde os dados serão armazenados. Isto pode ser um problema, devido à legislação sobre privacidade em alguns países proibir que os dados sejam armazenados fora de seus limites geográficos. O que ocorre, por exemplo, em relação ao armazenamento de dados médicos na nuvem, em alguns países da União Europeia.
- Integridade dos dados: o modelo *SaaS* é composto por aplicações multi-inquilino hospedadas na nuvem. Estas aplicações utilizam interfaces baseadas em *API-Application Program Interfaces* XML para expor suas funcionalidades sob a forma de serviços Web (*web ser-*

vices). Embora existam padrões para gerenciar a integridade das transações com *web services*, tais como *WS-Transaction e WS-Reliability*, estes padrões não são amplamente utilizados pelos desenvolvedores de aplicações *SaaS*.

- Segregação dos dados: os dados de vários clientes podem estar armazenados no mesmo servidor ou banco de dados no modelo *SaaS*. A aplicação *SaaS* deve garantir a segregação, no nível físico e na camada de aplicação, dos dados dos clientes.
- Acesso aos dados: o ambiente muilti-inquilino da nuvem pode gerar problemas relacionados à falta de flexibilidade de aplicações *SaaS* para incorporar políticas específicas de acesso a dados pelos usuários de organizações clientes do serviço *SaaS*.

2.6 Teoria da Informação

Os princípios da Teoria da Informação tratam do problema da codificação, o qual trata de buscar formas eficientes de representação da informação, que proporcionem maior segurança e privacidade (ex. criptografia), proteção contra erros e economia de símbolos utilizados no armazenamento dos dados (ex. compressão). A Teoria da Informação apresenta modelos matemáticos para representação da linguagem, que é compreendida como um conjunto de características da informação que se deseja codificar. O matemático e engenheiro Claude Shannon deu-nos uma maneira pura de definir a informação em 1948 (SHANNON, 1949). Ele mostrou que a quantidade de informação em algo como um fluxo de *bits* ou letras está relacionada com a sua entropia, uma medida de desordem. Quanto maior a entropia, maior a informação. Por exemplo, um fluxo de números de três bits que são sempre 000 contém menos informação do que um fluxo em que os números também podem ser 001, 101 ou 111. A informação (autoinformação) é medida em função da probabilidade de ocorrência de mensagens ou símbolos individuais. Na Teoria da Informação, o conceito de informação está associado à ideia de incerteza, surpresa, dificuldade e entropia (GLEICK, 2011):

- Incerteza: A informação é intimamente associada à incerteza, que por sua vez pode ser medida ao contar o número de mensagens possíveis. Se uma única mensagem for possível, não há incerteza e, portanto, não há informação.
- Surpresa: Algumas mensagens podem ser mais prováveis do que outras, e informação implica surpresa. A imprevisibilidade é uma maneira de se referir às probabilidades de ocorrência dos símbolos.
- Dificuldade: O significativo é a dificuldade de transmitir a mensagem de um ponto a outro.

- Entropia: Informação é entropia. A entropia é a medida da desordem na termodinâmica.

Seja A um símbolo ou mensagem. A autoinformação (I_A) de A é definida como sendo a quantidade de informação contida em A, representada pela fórmula a seguir:

$$\boxed{I_A = f(P_A) = log_2 \frac{1}{P_A} = -log_2 P_A}$$

onde P_A é a probabilidade de ocorrência do símbolo ou mensagem A.

Seja X um arquivo de dados com M símbolos diferentes e estatisticamente independentes, a informação média associada aos M símbolos de X é a média ponderada das autoinformações de cada símbolo. A informação é medida como uma função que soma as probabilidades com um peso logarítmico (a base 2). Trata-se do logaritmo médio da probabilidade da mensagem. Na prática, é uma medida da incerteza. A entropia é a informação média por símbolo, também conhecida como incerteza de X e designa-se por H(X), que é denominada de entropia da mensagem, entropia de Shannon ou apenas informação (SHANNON, 2001):

$$\boxed{H(X) = \sum_{j=1}^{M} P_j I_j = -\sum_{j=1}^{M} P_j log_2 P_j}$$

onde I_j é a medida da quantidade de informação do símbolo j e P_j é a probabilidade de ocorrência de j no arquivo X.

As técnicas de criptografia produzem um sistema de sigilo que proporciona privacidade para os dados. Do ponto de vista do analista criptográfico, destacou Shannon, “um sistema de sigilo é quase idêntico a um sistema de comunicação ruidoso"(SHANNON, 1949). O fluxo de dados deve parecer aleatório ou estocástico. Na análise criptográfica, a redundância é o ponto fraco do sistema de sigilo. Todos os sistemas de sigilo tem em comum o uso de uma palavra em código, ou frase, ou um livro inteiro, ou algo ainda mais complexo, mas mesmo assim uma fonte de caracteres conhecida tanto pelo emissor como pelo receptor, que é um conhecimento partilhado distinto da mensagem em si.

A Teoria da Informação, por meio de métodos algébricos, teoremas e comprovações, definiu uma forma rigorosa de avaliar o grau de segurança de qualquer sistema de sigilo, definindo assim os princípios da criptografia. Shannon provou que cifras perfeitas eram possíveis, no sentido de que uma mensagem de comprimento infinito não ajudaria um decifrador de códigos. Numa cifra perfeita, todas as chaves devem apresentar igual probabilidade de ocorrência, consistindo na prática em um fluxo aleatório de caracteres, cada chave só pode ser utilizada uma vez e deve ser do mesmo comprimento da mensagem inteira.

2.7 Conclusão

Este capítulo apresentou conceitos básicos sobre privacidade, destacando, principalmente, a relação entre privacidade e confidencialidade de dados. As preocupações relacionados a violações e roubos de dados no ambiente de computação em nuvem ressaltam a importância de medidas preventivas para garantir a confidencialidade dos dados armazenados neste ambiente.

O histórico da evolução do conceito de privacidade mostra sua complexidade e multidimensionalidade, ao mesmo tempo que destaca a sua importância como um valor humano, tal como liberdade de expressão ou igualdade. O uso de novas tecnologias da informação que proporcionam a produção de enormes volumes de dados pessoais aumenta as preocupações quanto ao grau de confidencialidade destes dados.

A Teoria da Informação oferece uma forma de medição da quantidade de informação contida nos dados, associando esta medida com uma função de probabilidade de ocorrência de símbolos. Preservar a confidencialidade de dados pessoais, muitas vezes, implica em restringir ou minimizar a quantidade de conhecimento da informação que estes dados podem revelar sobre seus proprietários. A ideia de proteção dos dados por meio da ocultação ou diminuição das informações que estes contêm é explorada por técnicas como anonimização, criptografia, e fragmentação que serão apresentadas nos próximos capítulos.

3 ANONIMIZAÇÃO DE DADOS

3.1 Introdução

Organizações públicas e privadas têm, cada vez mais, sido cobradas para publicar seus dados “brutos"em formato eletrônico, em vez de disponibilizarem apenas dados estatísticos ou agregados. Esses dados “brutos” são denominados microdados (*microdata*). Neste caso, antes de sua publicação, os dados devem ser “sanitizados”, com a remoção de identificadores explícitos, tais como nomes, endereços e números de telefone. Para isso, pode-se utilizar técnicas de anonimização.

O termo anonimato, que vem do adjetivo “anônimo”, representa o fato do sujeito não ser unicamente caracterizado dentro de um conjunto de sujeitos. Neste caso, afirma-se que o conjunto está anonimizado. O conceito de sujeito refere-se a uma entidade ativa, como uma pessoa ou um computador. Conjunto de sujeitos pode ser um grupo de pessoas ou uma rede de computadores (PFITZMANN; KöHNTOPP, 2005). Um registro ou transação é considerado anônimo quando seus dados, individualmente ou combinado com outros dados, não podem ser associados a um sujeito particular (CLARKE, 1999).

Os dados sensíveis armazenados em sistemas de banco de dados relacionais sofrem riscos de divulgação não autorizada. Por este motivo, tais dados precisam ser protegidos. Os dados são normalmente armazenados em uma única relação **r**, definida por um esquema relacional $\mathbf{R(a_1,a_2,a_3,...,a_n)}$, onde $\mathbf{a}_i$ é um atributo no domínio $\mathbf{D}_i$, com **i = 1 , ... , n**. Na perspectiva da divulgação de dados de indivíduos, os atributos em **R** podem ser classificados da seguinte forma (CAMENISCH *et al.*, 2011):

- Identificadores: atributos que identificam unicamente os indivíduos (ex.: CPF, Nome, Número da Identidade);
- Semi-identificadores (SM): atributos que podem ser combinados com informações externas para expor alguns ou todos os indivíduos, ou ainda reduzir a incerteza sobre suas identidades (ex.: data do nascimento, CEP, cargo, função, tipo sanguíneo);
- Atributos sensíveis: atributos que contêm informações sensíveis sobre os indivíduos (ex.: salário, exames médicos, lançamentos do cartão de crédito).

Os atributos que podem revelar informações sobre o proprietário dos dados devem ser anonimizados para proteger sua privacidade. O acesso a estas informações deve ser permitido apenas às pessoas e aos processos autorizados. Normalmente, os atributos identificadores são

excluídos, por causa da forte relação que possuem com o proprietário dos dados; os atributos semi-identificadores normalmente são generalizados e os atributos sensíveis geralmente sofrem algum tipo de perturbação randômica ou são generalizados.

3.2 Técnicas de Anonimização

As operações de anonimização têm o objetivo de modificar os microdados antes que estes sejam publicados para que requisitos específicos de privacidade sejam atendidos. Existem 5 tipos de operações de anonimização (FUNG *et al.*, 2010): generalização, supressão, anatomização, permutação e perturbação. Estas operações serão descritas a seguir:

3.2.1 Generalização

Esta técnica propõe a substituição de valores dos atributos semi-identificadores por valores semanticamente semelhantes, porém com uma classificação taxonômica mais geral. Por exemplo: a categoria "profissional" é mais geral do que "engenheiro" ou "médico". A operação reversa de generalização é a especialização. Na literatura sobre este assunto existem cinco esquemas de generalização que serão apresentados utilizando como exemplo a árvore de classificação taxonômica da Figura 4.

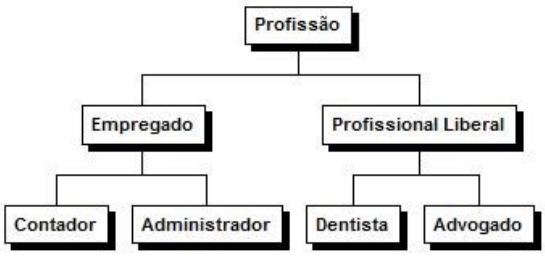

Figura 4 – Árvore de Taxonomia de Profissões

- esquema de generalização de domínio completo: neste esquema, todos os valores de um atributo são generalizados para o mesmo nível da árvore de taxonomia (LEFEVRE *et al.*, 2006). Para os valores do atributo "Profissão", anonimizados de acordo com a árvore de taxonomia da Figura 4, se "Contador" e "Administrador" forem anonimizados para "Empregado", então "Dentista" e "Advogado" serão anonimizados para "Profissional Liberal". O espaço de busca representa as opções de classificação dos atributos que podem ser consultadas após a execução da operação de generalização. No caso da generalização de domínio completo, O espaço de busca é menor do que o dos outros esquemas e o nível

de distorção da informação é maior por causa da exigência de todos os valores estarem anonimizados no mesmo nível da árvore de taxonomia;

- esquema de generalização de subárvore: neste esquema, a exigência de que todos os valores de um atributo estejam no mesmo nível da árvore de taxonomia aplica-se apenas aos valores das subárvores (IYENGAR, 2002). Por exemplo, caso o valor do atributo "Administrador" seja generalizado para "Empregado", o valor de "Contador" também deverá ser generalizado, mas os valores de "Dentista" e "Advogado" não precisam ser anonimizados, por estarem em outra subárvore da árvore de taxonomia. Comparativamente ao esquema de generalização de domínio completo, este esquema gera menor nível distorção da informação e maior espaço de busca dos atributos generalizados;
- esquema de generalização de irmãos (*siblings*): neste esquema, não há a obrigatoriedade de que todos os valores de um atributo que estejam em uma subárvore sejam generalizados (LEFEVRE *et al.*, 2005). Por exemplo, o atributo com valor "Contador" poderia ser generalizado para "Empregado", entretanto o atributo com valor "Administrador" permaneceria como está. Este esquema produz menos distorção do que o esquema de generalização de subárvore, porque precisa atuar apenas sobre os valores de atributos que violam o limite de anonimização especificado. O limite de anonimização representa a quantidade mínima de registros com valores semi-identificadores ou sensíveis idênticos na tabela. Por exemplo, caso fosse estabelecido que deveriam existir no mínimo **k** registros com mesmo valor de atributo na tabela e existissem **(k - 1)** registros com valores de atributo "Contador", "Coordenador" e "Consultor", todos estes registros seriam generalizados para "Empregado", mas se a quantidade de registros com valor de "Administrador" fosse maior do que k (limite de anonimização), então estes registros não sofreriam anonimização;
- esquema de generalização de células: nos esquemas anteriores, se o valor de um atributo é generalizado, então todos os demais registros da tabela com o mesmo valor também são generalizados (WANG; FUNG, 2006). Neste esquema, a generalização não é global. Algumas instâncias do atributo podem permanecer não generalizadas na tabela. Por exemplo, o atributo "Contador" pode existir nos registros 3 e 5 de uma tabela e ser anonimizado para "Empregado"apenas no registro 3.

3.2.2 Supressão

Esta técnica propõe a substituição de valores do atributo por um valor especial, indicando que o valor substituído não será disponibilizado na tabela anonimizada. A operação reversa da supressão é a divulgação. Existem também diferentes esquemas de supressão.

- Supressão de registro (SAMARATI, 2001): refere-se a supressão do registro inteiro da tabela;
- Supressão de valor: remove todas as instâncias de um determinado valor do atributo na tabela, por exemplo, remover todos os registros da tabela SALÁRIO cujo atributo valor-salario seja $\geqslant$ 5.000 (IYENGAR, 2002) (WANG *et al.*, 2007)
- Supressão de célula ou supressão local: remove apenas algumas instâncias de um determinado valor do atributo na tabela, por exemplo: remover 50% dos registros da tabela SALÁRIO cujo atributo valor-salario seja $\geqslant$ 5.000
  (MEYERSON; WILLIAMS, 2004; BEDI; MAHAJAN, 2016);

3.2.3 Anatomização

Este método desassocia as informações entre os semi-identificadores e os atributos sensíveis. Há a separação dos valores desses tipos de atributos em 2 tabelas e a utilização de um atributo comum (chave) para relacionamento entre os registros das 2 tabelas que têm seus atributos particionados ou generalizados. Cada grupo de registros anonimizados ou particionados é composto por atributos sensíveis com **n** valores distintos dentro do grupo. As duas tabelas possuem um atributo comum (Identificador de Grupo-IG). Todos os registros do mesmo grupo terão o mesmo valor IG em ambas as tabelas. Quanto maior for o valor de **n**, mais difícil a descoberta de atributos sensíveis dos usuários dos dados.

3.2.4 Permutação

Este método desfaz a ligação existente entre um semi-identificador (**SM**) e um atributo sensível numérico pelo particionamento de grupos contendo registros com atributos **SM** com mesmo valor e embaralhamento dos valores dos atributos sensíveis dentro de cada grupo. Por exemplo: em uma tabela com os atributos [nome da rua] e [número], realizar a operação de permutação do atributo [número] dos registos que tem o mesmo valor do atributo [nome da rua].

3.2.5 Perturbação

Este método substitui os valores originais dos dados por valores sintéticos que mantenham as informações estatísticas inalteradas, de forma que uma computação sobre os dados perturbados não seja muito diferente da mesma computação feita sobre os dados originais (XIAO *et al.*, 2009). A Perturbação é utilizada para preservação de privacidade em mineração de dados (*data mining*) ou para substituição de valores reais por fictícios. A ideia geral é alterar randomicamente os dados para disfarçar informações sensíveis enquanto se preservam as características que são críticas para o modelo de dados. Neste caso, o conjunto anonimizado representa apenas uma síntese dos dados originais, não correspondendo aos valores reais das entidades representadas.

Os métodos mais comuns de perturbação randômica são os seguintes:

- *Random Data Perturbation* (RDP): esta técnica adiciona números aleatórios (ruído), de forma randômica, aos dados numéricos sensíveis. Desta forma, mesmo que um atacante consiga identificar um valor individual de um atributo confidencial, o valor verdadeiro não será revelado. A maioria dos métodos utilizados para adicionar ruído randômico são casos especiais de mascaramento de matriz. Por exemplo, seja o conjunto de dados D, o conjunto Z dos dados randomizados é computado como $Z = A \times D \times B + C$, onde A é uma máscara de transformação de registro, B é um máscara de transformação de atributo e C é um máscara de deslocamento (ruído). A é uma matriz de operadores de linha que transformam os valores dos dados de D. B é uma matriz de operadores de coluna que transformam os valores dos dados de D. A matriz C adiciona ruído estocástico ou sistemático aos valores de $A \times D \times B$. Os valores de (A, B e C) dependem dos valores particulares do conjunto D (DOMINGO-FERRER, 2008);
- Troca de Dados: realiza a troca de valores de atributos sensíveis entre os registros da tabela, mantendo as características das contagens de frequências dos atributos;
- Geração de Dados Sintéticos: constrói um modelo estatístico para os dados e publica dados sintéticos do modelo ao invés dos dados originais;
- Condensação de Dados (AGGARWAL; PHILIP, 2004): condensa os dados em múltiplos grupos de tamanhos predefinidos. Informações estatísticas sobre média e correlações entre diferentes dimensões de cada grupo são preservadas. Dentro de um grupo não é possível distinguir diferenças entre os registros. Cada grupo tem um tamanho mínimo k, que é o nível de privacidade obtido com esta técnica.

O mascaramento de dados é um método de perturbação utilizado para disponibilizar bases de dados para teste ou treinamento de usuários, com informações que pareçam reais, mas nada revelem sobre ninguém. Isto protege a confidencialidade dos dados pessoais presentes no banco de dados, bem como outras informações sensíveis que não possam ser colocadas à disposição para a equipe de testes ou usuários em treinamento. Algumas técnicas de mascaramento de dados são descritas a seguir (LANE, 2012):

- Substituição: substituição randômica de conteúdo por informações similares, mas sem nenhuma relação com o dado real. Como exemplo, podemos citar a substituição de sobrenome de família por outro proveniente de uma grande lista randômica de sobrenomes;
- Embaralhamento (*Shuffling*): substituição randômica semelhante ao item anterior, com a diferença de que o dado é derivado da própria coluna da tabela. Assim, o valor do atributo A em uma determinada tupla t_1 é substituído pelo valor do atributo A em uma outra tupla t_n, selecionada randomicamente, onde $n \neq 1$;
- *Blurring*: esta técnica é aplicada a dados numéricos e datas. A técnica altera o valor do dado por alguma percentagem randômica do seu valor real. Logo, pode-se alterar uma determinada data somando-se ou diminuindo-se um determinado número de dias, determinado randomicamente, 120 dias, por exemplo; valores de salários podem ser substituídos por um valor calculado a partir do valor original, aplicando-se, para mais ou para menos, uma percentagem do valor original, selecionada randomicamente, por exemplo, 10% do valor inicial;
- Anulação/Truncagem (*Redaction/Nulling*): esta técnica substitui os dados sensíveis por valores nulos (NULL). A técnica é utilizada quando os dados existentes na tabela não são requeridos para teste ou treinamento.

As técnicas de anonimização são utilizadas individualmente ou de forma combinada para criação de modelos de anonimização. Alguns dos principais modelos de anonimização são descritos na próxima seção.

3.3 Modelos de Anonimização

A anonimização visa ocultar a identidade ou dados confidenciais das pessoas, de forma que, ainda que a informação seja divulgada e útil para análise por parte de receptores, a privacidade individual seja preservada (QUEIROZ *et al.*, 2016). Existem vários modelos de anonimização com a finalidade de proteger a privacidade do proprietário do dado, ao se

disponibilizar estes dados publicamente (ZHANG *et al.*, 2015) , (GAI *et al.*, 2015), (DUBOVITSKAYA *et al.*, 2015). Os principais modelos de anonimização encontrados na literatura são *k*-anonimato (*k-anonymity*), *l*-diversidade (*l-diversity*), *LKC*-Privacidade (*LKC-Privacy*), *t*-proximidade (*t-closeness*) e *b*-semelhante (*b-likeness*).

3.3.1 k-anonimato

O modelo *k*-anonimato requer que qualquer combinação de atributos semi-identificadores (SI) seja compartilhada por pelo menos *k* registros em um banco de dados anonimizado, onde *k* é um valor inteiro positivo definido pelo proprietário dos dados, possivelmente como resultado de negociações com outras partes interessadas (SAMARATI, 2001). Um valor alto de *k* indica que o banco anonimizado tem baixo risco de divulgação, porque a probabilidade de reidentificar um registro é de $1/k$, embora isto não proteja o banco contra divulgação de atributos. Mesmo que o atacante não tenha capacidade de reidentificar o registro, ele pode descobrir atributos sensíveis no banco anonimizado (BASSO *et al.*, 2016).

Samarati (SAMARATI; SWEENEY, 1998) apresenta dois esquemas de transformação dos dados por generalização e supressão. O primeiro esquema substitui os valores de atributos semi-identificadores por valores menos específicos, mas semanticamente consistentes, que os representam. Como exemplo, pode-se trocar datas no formato dd.mm.aaaa pelo formato mm.aaaa suprimindo o valor referente ao dia. A supressão é um caso extremo de generalização, o qual anula alguns valores de atributos semi-identificadores ou até mesmo exclui registros da tabela. A supressão deve ser utilizada como uma forma de moderação para a técnica de generalização, quando sua utilização provocar um grande aumento de generalização dos atributos semi-identificadores em um conjunto pequeno de registros com menos de k ocorrências. Fung propõe discretizar os atributos SI que apresentem valores contínuos, substituindo-os por um intervalo que contenha estes valores, por exemplo substituir o preço de produtos de supermercado por uma faixa de valores [1 a 3], [3 a 7] (FUNG *et al.*, 2007).

Para limitar o risco de divulgação em uma tabela anonimizada, *k*-anonimato define a propriedade de cada registro ser indistinguível em relação a $k-1$ outros registros em relação aos atributos semi-identificadores, o quê protege a tabela anonimizada contra divulgação de identidade, mas é insuficiente para proteção contra divulgação de atributo. Após a generalização dos atributos semi-identificadores, a estratégia de supressão de dados é utilizada caso existam atributos SI com quantidade de registros inferior ao limite *k*. Por exemplo, supondo $k = 50$, se

existirem na tabela privada apenas 30 registros com uma determinada data de infração, estes serão excluídos da tabela anonimizada.

O modelo *k*-anonimato assume como pressuposto que cada registro representa apenas um indivíduo. O problema aqui posto é que se vários registros na tabela representarem um único indivíduo, um grupo de k registros pode representar menos do que k indivíduos, colocando em risco a proteção da privacidade de algum indivíduo. Para resolver esta questão, Wang e Fung propuseram o modelo (x,y)-anonimato ((x,y)-anonymity) em que x e y representam conjuntos de atributos disjuntos, onde cada valor de x descreve um conjunto de registros (ex.: x = data nascimento) e está ligado a pelo menos k valores distintos de y (ex.: y = data infração). A associação entre x e y proposta pelo modelo dificulta a descoberta de atributos sensíveis (WANG; FUNG, 2006).

3.3.2 l-diversidade

O modelo *l*-diversidade captura o risco da descoberta de atributos sensíveis em um banco de dados anonimizado. O modelo *l*-diversidade requer que, para cada combinação de atributos SI, deva existir pelo menos ***l*** valores "bem representados" para cada atributo sensível. Isso significa que não devem existir atributos com valores que ocorram em grande ou pequena quantidade, pois isso poderia indicar o quê aquele valor estaria representando. Por exemplo, um atributo sensível na tabela DOENÇAS que tivesse apenas 2 valores, onde um valor está presente em 99% dos registros e o outro em 1%, poderia revelar quais doenças estes valores estariam representando. Formalmente, esse modelo requer que o número de valores distintos de atributos sensíveis em cada classe de equivalência seja no mínimo ***l***. A definição de *l*-diversidade é a seguinte: "Um grupo SI é l-diverso se contiver pelo menos l valores bem representados para os atributos sensíveis. Uma tabela é l-diversa se cada grupo SI for l-diverso". O modelo garante privacidade, mesmo quando não são conhecidas quais informações o atacante possui, pois garante a existência de pelo menos ***l*** valores de atributos sensíveis em cada grupo SI (LI *et al.*, 2007), (ZHANG *et al.*, 2016).

O modelo *l*-diversidade utiliza um conceito de privacidade baseado na distribuição de probabilidade do conhecimento do adversário sobre os atributos sensíveis e utiliza técnicas de inferência Bayesianas para estimar a privacidade, com a definição de duas medidas: CP_1 e CP_2 descritas a seguir.

- crença a priori (CP_1): a probabilidade da crença "a priori"do adversário da descoberta de

um atributo sensível, a partir de um semi-identificador externo;

- crença a posteriori (CP_2): a probabilidade da crença do adversário, após acessar a tabela anonimizada, da descoberta de um atributo sensível, a partir do conhecimento dos atributos anonimizados (semi-identificadores).

Considerando o conhecimento anterior do adversário sobre os dados e as informações disponíveis na tabela anonimizada, existem dois tipos de divulgação, as quais são descritas a seguir:

- divulgação positiva: ocorre quando o adversário consegue identificar o valor de um atributo sensível com alta probabilidade de acerto, seja pela falta de diversidade nos valores dos atributos sensíveis dos grupos SI ou pelo grande conhecimento anterior do adversário sobre os atributos semi-identificadores de um indivíduo;
- divulgação negativa: ocorre quando o adversário consegue, com alta probabilidade de acerto, eliminar corretamente valores prováveis de atributos sensíveis.

A interpretação do princípio de valores "bem representados" para os atributos de cada grupo SI pelo modelo *l*-diversidade originou 3 variações desta métrica (LI *et al.*, 2007), (SEI; OHSUGA, 2014):

- *l*-diversidade com valores distintos: neste caso existem l valores distintos para cada grupo SI. Um grupo SI pode ter um valor que apareça mais frequentemente que outros valores, possibilitando ao atacante re-identificar este valor ao sujeito que está presente no grupo SI. Este ataque é denominado ataque de probabilidade de inferência;
- *l*-diversidade com entropia: neste caso, a entropia da tabela inteira deve ser pelo menos log(l) e a entropia de cada grupo SI deve ser maior ou igual a log(l). Esta definição é mais forte do que a definição anterior e pode ser muito restritiva se existirem poucos valores com alta frequência de ocorrência na tabela. A entropia de cada grupo SI é definida pelo índice de diversidade de Shannon: Entropia(SI) = H(X) (ver tópico 2.3 da tese) onde P_j é a fração de registros do grupo SI que contem atributo sensível com valor igual a s;
- *(c,l)*-diversidade recursivo: Considere um grupo SI em que existem diversos valores para o atributo **s**, dado pelo conjunto **(s_1,...,s_m)**. Considere o conjunto de contagem destes valores (Ex.: "r_1 = count(*) where s=s_1"), ordenados em ordem decrescente **(r_1,...r_m)**. Neste caso, dada uma constante **c**, cada grupo SI satisfaz recursivamente *(c,l)-diversidade* se $\mathbf{r_1} < \mathbf{c}(\mathbf{r}_l + \mathbf{r}_{l+1} + ... + \mathbf{r}_m)$. Este procedimento assegura que valores muito frequentes não apareçam tão frequentemente e que valores mais raros não apareçam tão raramente nos grupos SI.

Um grupo SI satisfaz *(c,l)*-diversidade recursivo se for possível eliminar um valor sensível e mesmo assim o grupo SI continuar *(c,l-1)*-diverso.

O modelo *l*-diversidade apresenta alguns problemas e vulnerabilidades, os quais são descritos a seguir:

- o modelo é limitado na pressuposição do conhecimento do adversário sobre os atributos sensíveis. Por exemplo, não considera a possibilidade do adversário obter informações sobre um atributo sensível a partir da informação da frequência da distribuição global deste atributo na tabela;
- o modelo assume que todos os atributos sensíveis são categorizados, desconsiderando atributos numéricos, nos quais, apenas pode ser suficiente a descoberta de valores aproximados;
- o modelo é vulnerável ao ataque de assimetria, o que ocorre quando existe grande assimetria na distribuição dos valores dos atributos sensíveis. Por exemplo, um atributo com 2 valores em que existe 99% de ocorrência de um valor e 1% de ocorrência do outro valor;
- o modelo é vulnerável ao ataque de similaridade: ocorre quando os valores em um grupo SI são distintos mas semanticamente equivalentes. Por exemplo, o atributo salário poderia ser discretizado por faixa de valores, mas as faixas de valores mais altas indicariam que os indivíduos ocupavam funções de chefia, enquanto faixas de valores mais baixas poderiam indicar pessoas recém contratadas que ocupavam funções operacionais.

O modelo *l*-diversidade não leva em consideração a proximidade semântica dos valores dos atributos sensíveis, por isso, apesar de garantir diversidade para os valores dos grupos SI, não evita a descoberta de informação probabilística sobre os atributos em um grupo, quando a distribuição dos valores dentro do grupo é muito diferente da distribuição na tabela inteira.

3.3.3 LKC-Privacidade

Uma das maneiras de evitar o ataque de ligação de atributo é utilizar a técnica de generalização de dados em grupos SI, de forma que cada grupo contenha k registros com os mesmos valores de semi-identificadores e diversificação dos atributos sensíveis para desorientar inferências do atacante sobre atributos da vítima conhecidos por ele. O problema de aplicar esta técnica quando a quantidade de atributos semi-identificadores é muito grande é que a maior parte dos atributos tem que ser suprimida para se obter k-anonimização, o que diminui a qualidade dos dados anonimizados (FUNG *et al.*, 2010). Este problema foi identificado por Aggarwal

(AGGARWAL, 2005) e é conhecido como problema da alta dimensionalidade dos dados em k-anonimização. O modelo *LKC*-Privacidade foi criado como uma tentativa de resolver este problema (MOHAMMED *et al.*, 2009). Este modelo parte do pressuposto de que o atacante não possui todas as informações dos atributos semi-identificadores do seu alvo. Neste caso, é razoável supor que o atacante possui conhecimento de pelo menos **L** valores de atributos semi-identificadores.

O modelo *LKC*-Privacidade assegura que, em um conjunto S de valores de atributos semi-identificadores, cada combinação de valores de tamanho máximo L em uma tabela T seja compartilhada por pelo menos K registros, e a confiança da inferência de qualquer valor sensível em S não seja maior do que um valor probabilístico C que limita o espaço amostral de possibilidade da descoberta do proprietário do atributo, onde L, K e C são valores limites definidos. O modelo limita a probabilidade de sucesso na identificação do registro da vítima a ser menor ou igual a 1/K e a probabilidade de sucesso no ataque de ligação de atributo a ser menor ou igual a C, considerando que o conhecimento prévio do adversário não excede o valor de L. O modelo *LKC*-Privacidade é adequado para anonimização de dados com alta dimensionalidade por possuir as seguintes propriedades:

- requer que apenas um subconjunto de atributos semi-identificadores seja compartilhado por **k** registros. Este relaxamento da restrição tradicional de *k*-anonimato baseia-se na premissa de que o adversário tem limitado conhecimento dos atributos sensíveis da vítima;
- generaliza vários modelos tradicionais de *k*-anonimato. Por exemplo: *k*-anonimato é um caso especial de *LKC*-Privacidade onde L = Conjunto de Todos os Atributos semi-identificadores, K = k e C = 100 *l*-diversidade é um caso especial de *LKC*-Privacidade em que L= conjunto de todos os atributos semi-identificadores, K = 1 e C = 1/l;
- é flexível para ajustar o dilema entre privacidade de dados e utilidade de dados. Aumentando **L** e **K** ou diminuindo **C** pode-se aumentar a privacidade, embora isso reduza a utilidade dos dados;
- é um modelo de privacidade geral que evita ataques de ligação de registro e ligação de atributos. É aplicável para anonimização de dados com ou sem atributos sensíveis.

3.3.4 t-proximidade

Este modelo propõe-se a corrigir algumas limitações de *l*-diversidade no que diz respeito à proteção contra divulgação de atributo. O objetivo é limitar o risco de descoberta

a um nível aceitável. O modelo *t*-proximidade utiliza o conceito de “conhecimento global de retaguarda", que pressupõe que o adversário pode inferir informações sobre atributos sensíveis, a partir do conhecimento da frequência de ocorrência destes atributos na tabela. Como os dados anonimizados disponibilizados devem conter a maior parte ou todos os registros da tabela original, é possível para o atacante calcular a medida da distribuição do atributo sensível em relação ao total de registros da tabela. Esse modelo objetiva assegurar que a distribuição de um atributo sensível em cada grupo de atributos seja próxima à sua distribuição na tabela inteira.

O modelo *t*-proximidade estima o risco de divulgação computando a distância entre a distribuição de atributos confidenciais dentro do grupo SI e a tabela inteira. Esta métrica requer que a distribuição de um atributo sensível em qualquer grupo SI seja um valor próximo do valor da distribuição do atributo em relação à tabela inteira (LI *et al.*, 2007). Sendo Q a medida da distribuição do atributo sensível em toda a tabela e P a medida da distribuição do atributo sensível em um grupo SI, quanto mais próximas estas medidas estiverem, menor será o conhecimento que o atacante poderá ter sobre indivíduos específicos e maior será o grau de privacidade dos grupos SI. A distância entre as duas distribuições não pode ser maior que um limite t.

Desta forma, o modelo *t*-proximidade limita as possibilidades de um adversário obter informações sobre atributos sensíveis pela análise da distribuição de valores globais destes atributos. A medida da distância variacional é utilizada para calcular a distância entre dois atributos semi-identificadores $P = (p_1,p_2,p_3,...,p_m)$ e $Q =(q_1,q_2,q_3,...,q_m)$, definida pela métrica *Earth-Mover Distance (EMD)*, que mede a quantidade mínima de esforço necessário para transformar uma distribuição de probabilidade em outra, por meio do movimento da distribuição de massa entre pontos de um espaço probabilístico. Nesta caso, uma unidade de trabalho corresponde a mover uma quantidade unitária de massa de probabilidade por uma unidade de distância (LIANG; YUAN, 2013). O valor de EMD entre as distribuições de dois atributos semi-identificadores **P** e **Q** em um espaço normalizado é um número entre 0 e 1. O espaço equidistante entre atributos semi-identificadores significa que a distância entre cada par de valores distintos de atributos semi-identificadores é igual a 1. Neste caso, a fórmula para calcular o valor de EMD entre duas distribuições de atributos semi-identificadores neste espaço é a seguinte:

$$EMD[P,Q] = \sum \frac{1}{2} |p_i - q_i|$$

3.3.5 b-semelhante

O modelo *b*-semelhante assegura que a estimativa de um atacante sobre o valor de um atributo sensível não aumentará em termos relativos, mais que um limite **b** pré-estabelecido, após o atacante tomar conhecimento dos dados anonimizados publicados (CAO; KARRAS, 2012).

A definição básica de *b*-semelhante é de que dada uma tabela T que contem atributos sensíveis (grupo SI), seja V = (v_1, v_2, v_3, ..., v_m) o domínio de SI e P = (p_1, p_2, p_3, ..., p_m) a distribuição global de SI em T. Uma classe de equivalência G com distribuição de atributos sensíveis Q = (q_1, q_2, q_3, ..., q_m) satisfaz um limite básico *b*-semelhante, se e somente se $max(D(p,q)|p_i \in P, p_i < q_i) \leqslant b$, onde b > 0 é um limite e D é uma função de distância entre p_i e q_i. A distância D deve ser grande o suficiente para proteger os dados de ataques de assimetria e de similaridade. Esta técnica difere das anteriores em relação ao uso da função de distância D para estabelecer o limite de distância máximo, ao invés da distância cumulativa entre os atributos sensíveis. É utilizada uma medida relativa, ao invés das medidas absolutas utilizadas pelas funções cumulativas de diferenças de frequências dos outros modelos anteriores. D é calculado pela fórmula descrita a seguir:

$$\boxed{D(p_i, q_i) = \frac{q_i - p_i}{2}}$$

O modelo *b*-semelhante apresenta-se como uma solução ao problema da exposição de privacidade de valores de atributos sensíveis que ocorrem com menor frequência. Em geral, modelos de privacidade, como o *t*-proximidade, que utilizam funções cumulativas de diferenças de frequências entre as distribuições não conseguem fornecer uma relação compreensível entre o limite t e a privacidade proporcionada pelo modelo. Tais modelos não dão atenção aos valores de atributos sensíveis que são menos frequentes e que são mais vulneráveis a exposição de privacidade.

A restrição imposta à função $D(p_i,q_i)$ de ser menor ou igual ao limite b, tem como consequência, a criação de um limite superior para a frequência de $v_i \in V$ em qualquer classe de equivalência G, conforme descrito na expressão

$$\boxed{\frac{(q_i - p_i)}{p_i} \leqslant b \Rightarrow q_i \leqslant p_i \times (1+b)}$$

Esta função representa um limite de proteção de privacidade compreensível apenas se $p_i \times (1+b) < 1$, neste caso, valores de $p_i < 1/(1+b)$ devem ser monitorados, pois pode

ocorrer de tais valores assumirem valor igual a 1 na classe de equivalência, tornando possível ao atacante, que saiba que o registro da vítima está presente na classe de equivalência, a inferência do valor atributo sensível com 100% de confiança.

3.4 Métricas de Privacidade

Os modelos apresentados na seção anterior definem métricas para limitar o risco da descoberta de informações privativas. Estas métricas impõem requisitos mínimos na associação de um indivíduo e seus dados pessoais. A medida que esses métodos aumentam a proteção da privacidade, eles diminuem a qualidade e utilidade dos dados. A proteção provida pelas técnicas de anonimização normalmente implica em algum grau de modificação de dados, variando entre nenhuma modificação (máxima utilidade, mas nenhuma proteção à divulgação de dados privados) e criptografia de dados (proteção máxima, mas sem utilidade para o usuário sem a chave criptográfica). O estudo das técnicas de anonimização pode ser classificado em 3 categorias (DOMINGO-FERRER, 2008):

- proteção de dados relacionais estáticos: os dados privados estão agregados em tabelas que são publicadas em bancos de dados estatísticos. Em seguida, técnicas de anonimização são utilizadas para garantir a privacidade dos dados pessoais (GIESSING, 2004), (WILLENBORG; WAAL, 2012);
- proteção de bancos de dados dinâmicos: Neste caso, os dados privados, ao invés de serem anonimizados para publicação, são disponibilizados sob a forma de dados estatísticos, que podem ser consultados sob diversas formas (somas, médias, etc). O risco da anonimização de dados dinâmicos é que as informações agregadas obtidas por meio de sucessivas consultas podem revelar informações pessoais sobre usuários dos dados (DUNCAN *et al.*, 2001). Para minimizar esse risco, as soluções propostas são as seguintes: i) realizar perturbação randômica das saídas das consultas; ii) limitar o acesso a algumas consultas, caso seja necessário obter a resposta exata das consultas (GOPAL *et al.*, 1998); iii) retornar respostas corretas dentro de um intervalo determinado, ao invés de retornar apenas o valor correto da consulta (GOPAL *et al.*, 2002);
- proteção de microdados: os dados são publicados na sua forma original. Estes dados são chamados de microdados. Esta área de pesquisa é muito recente e ainda está em evolução.

Como o objetivo é impedir que dados privados sejam ligados aos proprietários dos dados, os microdados devem ser pré-processados para remoção de todos os identificadores. As

técnicas de anonimização utilizam um conjunto M de microdados como entrada e produzem outro conjunto $M^{'}$ de saída, com o objetivo de minimizar o risco de divulgação de dados privados e possibilitar consultas estatísticas sobre o conjunto $M^{'}$ com obtenção de resultados iguais ou semelhantes às mesmas consultas feitas ao conjunto M.

Os métodos de proteção dos microdados podem ser classificados em duas categorias: mascaramento e geração de dados estatísticos sintéticos. A categoria de mascaramento é dividida em duas subcategorias: métodos perturbativos e métodos não perturbativos, descritos a seguir:

- métodos perturbativos distorcem o microdado antes da publicação. Isto deve ser feito de forma que as estatísticas calculadas sobre os dados perturbados não difiram significativamente das estatísticas que seriam obtidas no conjunto dos dados originais;
- métodos não perturbativos não alteram os dados, ao invés disto, eles produzem supressões parciais ou reduções de detalhes dos dados originais.

A maior parte dos trabalhos sobre métricas de privacidade têm focado em *k*-anonimato e suas variações: *l*-diversidade e *t*-proximidade, os quais garantem um limite de proteção (*k*, *l* e *t*) respectivamente para os dados publicados. Estas abordagens definem um requisito mínimo (cenário de pior caso) que cada combinação de registros da tabela deve satisfazer. Outra abordagem recente proposta por (AGRAWAL; AGGARWAL, 2001) utiliza o conceito de informação mútua entre o dado original e o dado anonimizado. A informação mútua tem a vantagem de expressar as diferentes medidas e limites de risco em um único indicador bem definido, permitindo o uso de ampla variedade de ferramentas da teoria da informação para otimização de risco, por exemplo o problema da distorção entre privacidade e utilidade do dado (REBOLLO-MONEDERO *et al.*, 2010).

A avaliação da qualidade das técnicas de anonimização necessita de métricas para tornar possível avaliar o risco da descoberta de dados pessoais anonimizados e sua utilidade. Bezzi (BEZZI, 2010) propõe uma nova forma de medir o risco de descoberta do dado utilizando conceitos da teoria da informação. A nova métrica proposta utiliza o conceito de informação de *one-symbol*, que calcula a contribuição de cada entrada para o risco de descoberta do dado e permite avaliar o grau de privacidade oferecido pelas técnicas de anonimização.

3.4.1 Medidas Estatísticas de Anonimização

Existem 3 técnicas que usam medidas estatísticas para calcular o grau de privacidade obtido utilizando-se técnicas de anonimização.

Consultas com Restrição: Esta técnica utiliza um parâmetro k de limitação da quantidade de registros retornados por uma consulta. Para um banco de dados de tamanho N, todas as consultas, que retornarem menos que k registros ou mais do que (N - k) registros, são rejeitadas. Esta técnica antecipa *k*-anonimato, retornando um grande conjunto de registros para cada consulta. Em comparação à técnica de supressão, suprime consultas ao invés de suprimir registros (AGGARWAL; PHILIP, 2008).

Anonimização por Variância: A privacidade é medida pela variância do dado perturbado. Neste caso, é adicionado um ruído ao dado usando-se a técnica de perturbação. Quanto maior for a perturbação, maior será o nível de proteção do valor do dado e maior será a garantia de anonimização, porém a utilidade do dado é inversamente proporcional ao grau de anonimização obtido.

Anonimização por Multiplicidade: as técnicas de privacidade baseadas em generalização “borram"os dados por meio da generalização dos valores. Neste caso, o conjunto de dados generalizado ainda continua a fornecer informações estatísticas úteis, sem comprometer a privacidade dos dados pessoais dos indivíduos. A medida de privacidade é o tamanho do intervalo mínimo dos registros generalizados. Em um banco de dados k-anônimo, nenhuma consulta pode retornar menos do que k registros. Caso seja possível estimar com c% de confiança que um valor x está dentro de um intervalo [x1, x2], então o tamanho do intervalo (x2-x1) define a quantidade de privacidade obtida com nível c% de confiança.

3.4.2 Medidas Probabilísticas de Anonimato

O vazamento de informações privativas pode ocorrer a partir do conhecimento do adversário sobre informações agregadas em bancos de dados perturbados ou sobre o método de perturbação utilizado para modificar os dados. Por exemplo, a anonimização de um atributo V por perturbação com um valor randômico escolhido em um intervalo entre $[-1,1]$ e com um nível de confiança de 100%, usando a métrica de anonimização por multiplicidade, fornece uma medida de privacidade igual a 2, que é o tamanho do intervalo $[-1,1]$. Caso o atacante ao analisar os dados anonimizados constate que a distribuição de valores do atributo V tenha 50% de probabilidade de estar dentro do intervalo $[0,1]$ e 50% de probabilidade de estar dentro do intervalo $[4,5]$, ele poderá inferir a partir dos dados perturbados V^* , em qual dos dois intervalos de tamanho 1 o valor de V está presente. Neste caso, consegue reduzir a medida efetiva da privacidade para 1.

Ao incorporar informações de *background*, muda-se o foco das medidas de anonimização. Ao invés de medir a probabilidade de alguns dados serem liberados, deve-se medir a quantidade de novas informações aprendidas por um adversário em relação ao seu conhecimento a priori dos dados. Para tal, é necessário mais do que apenas informações sobre a variância de um valor perturbado. É preciso noções mais precisas sobre vazamento de informações. Esta análise é independente da forma de anonimização utilizada. Seja perturbação aleatória ou generalização, em ambos os casos, as medidas probabilísticas são resultados da computação de funções de distribuições definidas nos dados (AGGARWAL; PHILIP, 2008).

3.4.3 Medidas Baseadas em Perturbação Randômica

Agrawal (AGRAWAL; AGGARWAL, 2001) propõe uma métrica de privacidade que considera o conhecimento, por parte do atacante, do registro individual perturbado e da distribuição agregada dos dados. Esta métrica é baseada no conceito de informação mútua entre dados numéricos originais e perturbados. A medida de entropia H(A) é utilizada para representar a quantidade de incerteza (grau de privacidade) de uma variável randômica A. H(A|B) representa a entropia condicional de A a partir do conhecimento de B, que pode ser interpretada como a quantidade de privacidade “deixada"em A depois que B é revelado. Como a entropia é representada em bits de informação, usa-se a expressão $2^{H(A)}$ para representar a medida de privacidade de A. Usando-se esta notação, o percentual de privacidade revelada para um adversário que conhece B pode ser expressa da seguinte forma: $P(A|B) = 1 - 2^{H(A|B)}/2^{H(A)} = 1 - 2^{-I(A;B)}$ onde I(A;B) = H(A) - $H(A|B)$ é a informação mútua entre as variáveis randômicas A e B. Quando os dados representam categorias formadas por coleções de transações, cada transação sendo definida por um conjuntos de itens, os dados categorizados normalmente são representados por um vetor binário. Por exemplo, em uma aplicação de carrinho de compras, uma transação é formada pelo conjunto de itens comprados por um cliente. O que precisa de privacidade é a associação entre o item i e a transação t. Existem duas maneiras de medir a privacidade neste contexto:

- utilizar uma métrica de violação de privacidade baseada na probabilidade de uma propriedade do dado de entrada ser descoberta a partir da análise das propriedades do dado perturbado na saída. Um conjunto de itens A provoca uma violação de nível p na privacidade se para algum item a e algum item i, temos

  $$a \in A, i \in 1...\mathbb{N} \Rightarrow P\lfloor a \in t_i | A \subseteq t_i \rfloor \geqslant p$$

Neste caso, o fato de A estar contido na transação perturbada $t_i^{'}$ está vazando informação sobre o item a pertencer à transação de entrada t_i (EVFIMIEVSKI *et al.*, 2004);

- medir a privacidade em termos de probabilidade de reconstrução correta do bit original, a partir do bit perturbado, que pode ser calculada usando o teorema de Bayes e é parametrizada pela probabilidade de inversão de um bit (definida como uma constante p). Privacidade é obtida pela fixação de um valor para p que minimize a probabilidade de reconstrução o dado perturbado(RIZVI; HARITSA, 2002).

3.5 Conclusão

Este capítulo apresentou os principais modelos e técnicas de anonimização de dados, além de algumas métricas de medição da privacidade de dados que utilizam abordagens estatísticas, probabilísticas e baseadas em perturbação randômica. A anonimização tem sido a estratégia utilizada para divulgação de dados sensíveis sem o comprometimento da privacidade dos proprietários dos dados. Neste caso, o risco da descoberta de informações privativas passa a ser calculado em função da probabilidade de reidentificação de 1 entre n registros idênticos. O modelo *k*-anonimato protege a descoberta de atributos sensíveis de dados disponibilizados. Outras abordagens (*l*-diversidade, *t*-proximidade) aumentam a proteção de *k*-anonimato, quando os dados disponibilizados podem ser utilizados para inferir informações sobre os proprietários dos dados.

As maiores dificuldades enfrentadas pelos modelos de anonimização são determinar quais são os atributos sensíveis e bloquear inferências decorrentes de conhecimento externo, ou seja, que vai além do relacionamento entre os identificadores e atributos sensíveis, os quais podem permitir ligações de identificadores ou outros atributos com os proprietários dos dados.

Para situações que exigem um maior grau de proteção dos dados são utilizadas técnicas de criptografia. Estas técnicas serão discutidas em detalhes no capítulo seguinte.

4 CRIPTOGRAFIA

4.1 Introdução

A criptografia, do grego **kryptós** (esconder) e **grápho** (escrita), estuda formas de se esconder o significado de uma mensagem a ser transmitida entre um emissor e um receptor, de maneira que terceiros não consigam entender o que é comunicado. Desde que foi inventada, na antiguidade, até o advento da computação, a criptografia foi usada principalmente para proteger o sigilo das comunicações. Em outras palavras, sua função é evitar que terceiros se inteirassem do conteúdo da mensagem. Assim, as inúmeras técnicas de criptografia desenvolvidas inicialmente tinham, basicamente, aplicações militares e diplomáticas.

Existem registros de textos criptografados esculpidos em pedra, no Egito, desde 1900 a.c. e a primeira descrição de um sistema de criptografia militar conhecida data do ano 475 a.c. durante a guerra entre Esparta e Pérsia (KAHN, 1996). A criptografia avançou significativamente em períodos de guerras, por necessidade de transmissão de mensagens de forma secreta entre as forças aliadas. No entanto, com a evolução da tecnologia da informação e dos sistemas de computação, “criptografar"e “descriptografar"se tornaram atividades corriqueiras na vida de usuários de tecnologias móveis, aplicações de nuvem e transações comerciais na internet.

Durante a II Guerra Mundial, as comunicações via rádio impulsionaram bastante o uso da criptoanálise, que é um conjunto de técnicas e métodos para a decifração de caracteres de uma escrita de sistema desconhecido sem conhecer a chave secreta usada para encriptar a mensagem. Se o telégrafo havia tornado as comunicações militares mais eficientes apesar de aumentar a possibilidade de interceptação, o rádio potencializou diversas vezes tanto a eficiência da comunicação quanto a da interceptação. Neste caso, não era mais necessário ter acesso físico à linha de comunicação para interceptar as mensagens: bastava sintonizar na mesma frequência do inimigo. Dessa forma, o rádio introduziu dois novos fatores para o desenvolvimento da criptoanálise: quantidade e continuidade. Atualmente, as redes de dados sem fio que utilizam equipamentos de radiofrequência ou infravermelho são mais suscetíveis a interceptações. A criptologia é a ciência que estuda a criptografia e a criptoanálise. A criptografia abrange as técnicas matemáticas relacionadas a aspectos de segurança da informação tais como confidencialidade, integridade de dados, autenticação de entidade e autenticação de origem de dados (SCHNEIER, 2015). Friedman (FRIEDMAN, 1987) criou o termo “criptoanálise"em 1920. Ele foi o primeiro a associar o estudo da criptologia à estatística em seu livro *The Index of*

Coincidence. Antes dele, todas as contagens de frequência, características linguísticas, enfim, técnicas desenvolvidas eram peculiares e particulares da criptologia.

Shannon (SHANNON, 2001), com sua Teoria Matemática da Informação, teve grande influência na criptologia contemporânea, por meio das demonstrações matemáticas de sistemas criptográficos inquebráveis e do conceito de redundância, que influenciaram os sistemas criptográficos da IBM e o projeto de desenvolvimento do algoritmo *Data Encryption Standard (DES)*.

Além de garantir o sigilo das comunicações, a criptografia também é utilizada para autenticação do remetente da mensagem e para garantir que a mensagem recebida não foi alterada. Outra aplicação que surgiu com as transações eletrônicas foi a da assinatura digital, necessária a partir do momento em que documentos digitais passaram a substituir documentos em papel em transações comerciais e bancárias como, por exemplo, transferência eletrônica de fundos. A criptografia pode ser usada não apenas na proteção dos dados, mas também na proteção dos programas (software), por exemplo, evitando a execução de copias não autorizadas de um programa.

A seguir, apresenta-se algumas aplicações mais representativas do uso da criptografia, explicando-se por que esse uso se faz necessário e como ele é feito.

- **proteção de senhas**: ocorre na grande maioria dos sistemas de computadores multiusuários, cada usuário autorizado a acessá-lo deve ter em seu poder uma senha, a qual é fornecida pelo usuário quando este deseja utilizar o sistema. A fim de proteger o conjunto de senhas dos usuários, estas são armazenadas criptografadas.
- **proteção de software**: a criptografia pode ser utilizada para impedir que cópias ilegais sejam executadas.
- **transações bancárias e de cartão de crédito**: transações são, em geral, criptografadas para garantir a confidencialidade dos dados financeiros dos clientes.
- **sigilo/autenticação de informações transmitidas**: trata-se de garantir o sigilo das informações transmitidas através de canais inseguros, sujeitos à “escuta", como linhas telefônicas, redes sem fio, redes cabeadas, radiodifusão, teleconferências, etc.
- **sigilo/autenticação de informações armazenadas**: informações sigilosas (como por exemplo, dados dos pacientes de um hospital) que precisam ser armazenadas em dispositivos de armazenamento tais como discos rígidos, discos óticos, memória *flash*, etc, devem ser criptografadas antes de seu armazenamento.

4.2 Técnicas de Criptografia Clássicas

Para “camuflar"uma mensagem, em geral, a ser transmitida por um meio inseguro de maneira que apenas o destinatário consiga entendê-la, é preciso que apenas o destinatário e o remetente tenham conhecimento de um dado necessário para a correta conversão da mensagem. Esse dado é a chave. Assume-se que qualquer outra pessoa conhece o algoritmo utilizado para encriptar/decriptar e que apenas a chave é secreta.

A **Criptografia Simétrica**, também conhecida como criptografia convencional ou criptografia de chave única, pode ser definida como um esquema consistindo de um conjunto de transformações de encriptação $\mathbf{E}_e$ e decriptação D_d, tais que: ($\mathbf{E}_e$ **: e** $\in$ **K**) e ($\mathbf{D}_d$ **: d** $\in$ **K**), onde **K** é o espaço de chaves. O esquema é classificado como simétrico porque a chave utilizada na operação de encriptação (e) deve ser a mesma utilizada na decriptação (d), ou seja, $d = e$.

A **Criptografia de Chave Pública** pode ser definida da seguinte forma: Seja ($\mathbf{E}_e$ **: e** $\in$ **K**) um conjunto de transformações de encriptação, e seja ($\mathbf{D}_d$ **: d** $\in$ **K**) o conjunto correspondente de transformações de decriptação, onde **K** é o espaço de chaves. Neste esquema, a chave utilizada na operação de encriptação (e) deve ser a diferente da chave utilizada na decriptação (d), ou seja, $d \neq e$. Assim, dado um texto cifrado **c** $\in$ **C**, e o espaço de chaves K, é computacionalmente inviável encontrar a chave d ou a mensagem **m** $\in$ **M** tal que $\mathbf{E}_e$**(m) = c**. Adicionalmente, dado **e** é inviável determinar a chave **d** correspondente de decriptação. Sendo que **e** e **d** são apenas formas simples de descrever as funções de encriptação e decriptação, respectivamente (STALLINGS, 2006).

A **Criptografia Simétrica** foi o único tipo de criptografia em uso antes do desenvolvimento da **Criptografia de Chave Pública** na década de 1970. Estas duas técnicas mantêm-se como os dois tipos de criptografia mais amplamente utilizados (STALLINGS, 2006).

Um criptossistema é um sistema para cifragem (encriptação) e decifragem (decriptação) (PFLEEGER; PFLEEGER, 2006). Geralmente, um criptossistema funciona como o descrito na Figura 5: um emissor utiliza uma chave secreta para criptografar a mensagem e envia esta chave por um canal seguro para o receptor, enquanto a mensagem cifrada é enviada por um canal inseguro. O receptor após receber a mensagem cifrada, utiliza a chave secreta para descriptografar a mensagem e ter acesso ao texto original.

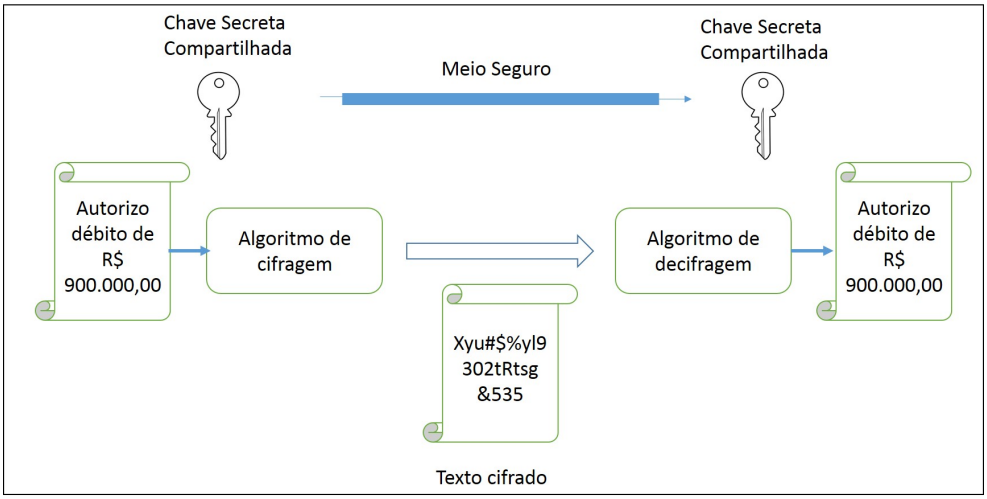

Figura 5 – Criptossistema Genérico

4.2.1 Cifras Simétricas

O esquema de criptografia simétrica possui 5 componentes:

1. Texto simples: mensagem a ser transmitida, que pode ser algum texto, numeral, um programa executável ou qualquer outro tipo de informação utilizada como entrada no algoritmo.
2. Algoritmo de encriptação: algoritmo que realiza várias substituições e transformações no texto simples com a finalidade de “camuflar"a mensagem.
3. Chave secreta: a chave secreta é usada como entrada do algoritmo de encriptação. É um valor independente do algoritmo e do texto simples. O algoritmo irá produzir diferentes saídas dependendo da chave utilizada como entrada. As substituições e transformações do algoritmo dependem do valor da chave.
4. Texto cifrado: é a mensagem em formato não compreensível, correspondente a um conjunto aleatório de símbolos, que é produzida como saída pelo algoritmo de encriptação, que depende do texto simples de entrada e da chave secreta. Para uma mesma mensagem, dois textos diferentes de entrada irão produzir dois textos cifrados diferentes. O texto cifrado assemelha-se a uma distribuição randômica de dados, sendo assim ininteligível.
5. Algoritmo de descriptografia: É essencialmente o algoritmo de criptografia executado de forma reversa. Utiliza o texto cifrado e a chave secreta como entrada e produz como saída o texto simples.

Sistemas criptográficos (criptossistemas) podem ser analisados por três dimensões independentes: tipo de operação, quantidade de chaves e modo de processamento. A seguir são

descritas estas três características: (STALLINGS, 2006):

1. Os tipos de operações utilizadas para transformar o texto simples no texto cifrado. Em geral, os algoritmos são baseados em dois princípios gerais: **substituição**, em que um elemento (bit, letra, grupo de bits ou letras) é mapeado em outro elemento e **transposição**, em que vários elementos são rearranjados, mudando a ordem em que se encontram originalmente dentro da mensagem. Nenhuma informação deve ser perdida nestas transformações. Este é um requisito fundamental da criptografia. A maioria dos criptossistemas utilizam múltiplos estágios de substituição e transposição.
2. O número de chaves utilizada. Se o emissor e receptor usam a mesma chave, o criptossistema é denominado simétrico ou de chave simples ou ainda de chave secreta ou encriptação convencional. Se o emissor e receptor usam chaves diferentes, o criptossistema é denominado de assimétrico, de duas chaves ou encriptação de chave pública.
3. O modo como um bloco de texto simples (não cifrado) é processado. Um algoritmo que processa um conjunto de dados por unidade de tempo, produzindo como saída um bloco de texto cifrado é chamado de **Cifra de Bloco**. Um algoritmo que processa os elementos de entrada continuamente, produzindo uma cifra de saída por unidade tempo, à medida que avança, é chamado de **Cifra de Fluxo**.

4.2.2 Criptanálise e Ataque de Força Bruta

A força de criptografia é medida pela quantidade de tempo e recursos que se exige para se descobrir a mensagem original a partir do texto cifrado. O resultado de **criptografia forte** é o texto cifrado que é muito difícil decifrar sem o conhecimento da chave e algoritmo de decodificação. Por exemplo, dado todo do poder de computação do planeta atualmente alocado em tempo integral, não seria possível decifrar o resultado da criptografia forte antes do fim do universo. O **criptógrafo** tem como objetivo desenvolver criptossistemas para proteger as informações contra acesso não autorizado ao seu conteúdo. Enquanto o atacante de um criptossistema (**criptoanalista**) tem como meta descobrir a chave em uso para pode recuperar o texto simples original, a partir do texto cifrado. Existem duas abordagens para ataque a criptossistemas convencionais (STALLINGS, 2006):

1. Criptanálise: este tipo de ataque explora características dos algoritmos de encriptação/decriptação do criptossistema mais algum conhecimento das características gerais do texto simples ou comparação de pares de texto simples e texto cifrado. O objetivo é deduzir um

texto simples específico ou a chave que está sendo utilizada.

2. Ataque de força bruta: este tipo de ataque testa possíveis chaves em uma parte do texto cifrado, utilizando o algoritmo de decriptação, até que um texto inteligível seja obtido. Na média, a chave é descoberta após metade das chaves possíveis serem testadas aleatoriamente.

Se qualquer um destes tipos de ataques obtiver sucesso, as consequências são graves: todas as mensagens criptografadas com essa chave estarão comprometidas. A Tabela 4 resume os vários tipos de ataques criptoanalíticos com base na quantidade de informação conhecida pelo criptoanalista. O problema mais difícil é apresentado quando tudo o que está disponível é apenas o texto cifrado. Em alguns casos, nem mesmo o algoritmo de criptografia é conhecido, mas em geral, podemos supor que o oponente sabe o algoritmo usado para criptografia. Um ataque possível sob estas circunstâncias é o de força bruta, testando todas as chaves possíveis. Se o espaço-amostral da chave é muito grande, este tipo de ataque se torna impraticável. Assim, o oponente deve tentar analisar o texto cifrado, geralmente aplicando vários testes estatísticos. Para usar essa abordagem, o oponente deve ter algum conhecimento sobre as características do texto simples que está oculto, por exemplo, se é texto, imagem, arquivo executável, etc.

Tabela 4 – Tipos de Ataques em Mensagens Encriptadas

Tipo de Ataque	Conhecimento do Criptoanalista
Somente texto cifrado	Algoritmo de encriptação e texto cifrado
Conhecimento de texto simples	Algoritmo de encriptação, texto cifrado e um ou mais pares de texto simples-texto cifrado
Escolha de texto simples	Algoritmo de encriptação, texto cifrado e mensagens em texto simples escolhidas pelo criptoanalista, com o correspondente texto cifrado
Escolha de texto cifrado	Algoritmo de encriptação, texto cifrado e mensagens em texto cifrado escolhidas pelo criptoanalista, com o correspondente texto simples decriptado
Escolha de texto	Algoritmo de encriptação, texto cifrado, mensagens em texto simples escolhidas pelo criptoanalista, com o correspondente texto cifrado e mensagens em texto cifrado escolhidas pelo criptoanalista, com o correspondente texto simples decriptado

Um criptossistemas é considerado seguro se o texto cifrado gerado pelo esquema não contiver informações suficientes para determinar exclusivamente o texto simples correspondente, não importa o quanto o texto cifrado esteja disponível. Portanto,espera-se que um algoritmo de criptografia atenda a um ou ambos critérios descritos a seguir:

- O custo de quebrar a cifra excede o valor das informações criptografadas.
- O tempo necessário para quebrar a cifra excede a vida útil da informação.

Um criptossistema é considerado computacionalmente seguro se qualquer um dos dois critérios anteriores forem atendidos. Infelizmente, é muito difícil estimar a quantidade de esforço necessária para descriptografar o texto cifrado com sucesso. As formas de criptoanálise para criptossistemas simétricos são projetadas para explorar o fato de que vestígios de estrutura ou padrão no texto simples podem sobreviver à criptografia e ser discerníveis no texto cifrado (STALLINGS, 2006). Com exceção de esquema conhecido como *one-time pad* (descrito mais adiante neste capítulo), não há algoritmo de criptografia que seja incondicionalmente seguro (DELFS; KNEBL, 2015).

A discussão sobre a segurança dos métodos criptográficos tem pelo menos o consenso entre as partes de que não existe segurança perfeita nas técnicas utilizadas atualmente. Por exemplo, como o caso Apple/FBI de 2016 mostrou, a criptografia é sempre quebrável (BAY, 2017). Vários estudiosos e profissionais afirmaram que o conflito entre criptógrafos e criptoanalistas é perpétuo. Não existe tal coisa como segurança perfeita e total quando se trata de proteger ativos digitais e informações pessoais (ASSANTE, 2014),(ELLISON, 2000; BUHRMAN *et al.*, 2006).

No entanto, com o surgimento da computação quântica, a criptografia está se tornando tão avançada que a criptografia inquebrável e impenetrável pode muito bem se tornar uma realidade. Outras tecnologias também estão se tornando disponíveis, que são descritas como inquebráveis, criadas por empresas privadas como a Apple, e agências governamentais como a DARPA. Em outras palavras, a criptografia pode não ser inquebrável para sempre, mas é muito provável que a criptografia seja inquebrável por períodos significativos de tempo, dependendo da dinâmica de evolução das novas tecnologias, como a computação quântica (BAY, 2017).

4.2.3 Técnicas de Substituição

Uma técnica de substituição é aquela em que as letras de texto simples são substituídas por outras letras ou por números ou símbolos. Se o texto simples for visto como uma sequência de bits, então a substituição envolve a substituição de padrões de bits de texto simples por padrões de bits de texto criptografado.

Um caso particular de substituição simples é a Cifra de Cesar (acredita-se que Júlio Cesar foi o primeiro a utilizá-la), que significa deslocar 3 posições no alfabeto. Por exemplo a

letra A é substituída por D, B por E, C por F etc. Generalizando, pode-se deslocar o alfabeto de x posições, neste caso x é a chave, e pode variar de 1 a 25. Neste caso, o método de força bruta não é inviável, uma vez que só existem somente 25 chaves possíveis.

Cifra de Substituição Simples

Definição: Seja A um alfabeto de q símbolos e M o conjunto de todas as sequências de t símbolos sobre A. Seja K o conjunto de todas as permutações no conjunto A. Defina para cada e $\in$ K uma transformação de encriptação E_e como:

$E_e(m) = (e(m_1), e(m_2), ..., e(m_t)) = (c_1, c_2, ..., c_t) = c$,

onde m =$(m_1, m_2, ..., m_t) \in$ M. Para cada símbolo em uma t-tupla substitua-o por outro símbolo de A de acordo com uma permutação fixa e. Para decriptar c =$(c_1, c_2, ..., c_t)$, compute a permutação inversa d = e^{-1}

$D_d(c) = (d(c_1), d(c_2), ..., d(c_t)) = (m_1, m_2, ..., m_t) = m$.

E_e é chamado de cifra de substituição simples ou cifra de substituição monoalfabética.

Em se tratando da substituição monoalfabética de maneira geral, a criptanálise é trivial, pois apesar do número de chaves ser exponencial, dependendo da quantidade de símbolos utilizada na mensagem (no caso de um texto escrito em português, temos 26 letras, que permutadas criam 26! = 4,03 x 10^{26} combinações) a distribuição de frequência da mensagem original é mantida. Em uma mensagem escrita em português, por exemplo, as letras que mais aparecem são E, A, O, S, I, R, N e T (essas letras formam 69,7% do texto) e a que menos aparece é o Z. Quando o criptoanalista analisa o texto encriptado por substituição monoalfabética, irá utilizar esta informação para inferir a coincidência entre um símbolo cifrado e uma letra do alfabeto: se J é a letra que mais aparece no criptograma, provavelmente J substitui E, e assim por diante.

Cifra de Substituição Homofônica

Definição: Considere uma mensagem **A** com **n** símbolos, Sendo $\mathbf{n} > \mathbf{0}$. Para cada símbolo $\mathbf{a} \in \mathbf{A}$, associado com um conjunto **H(a)** composto por uma lista de **t** símbolos, com a restrição de que os conjuntos **H(a)**, **a** $\in$ A sejam mutuamente disjuntos. Isto é $\forall$**a,b** $\in$ **H : H(a)** $\neq$ **H(b)** $\Rightarrow$ **H(a)** $\cap$ **H(b)** $= \varnothing$. Uma cifra de substituição homofônica substitui cada símbolo **a** de um bloco da mensagem em texto simples por um símbolo de **H(a)** escolhido de forma randômica. Para decriptar a cifra **c** de **t** símbolos, deve-se determinar um **a** $\in$ A de tal forma que **c** $\in$ **A**. A chave para a cifra consiste na lista de símbolos de **H(a)**.

Como exemplo de aplicação de cifra de substituição homofônica, considere o conjunto A = a,b, H(a) = 00,10 e H(b) = 01,11. A mensagem **ab** pode ser encriptada da seguinte forma: 0001, 0011, 1001 ou 1011. Como os símbolos não ocorrem com igual frequência nas mensagens de texto simples, uma cifra homofônica pode ser usada para tornar a frequência de ocorrência de símbolos de texto cifrado mais uniforme, à custa da expansão dos dados. Isto torna a decriptação mais difícil, em comparação com as cifras de substituição simples.

Cifra de Substituição Polialfabética

Definição: Uma cifra de substituição polialfabética é uma cifra de bloco de tamanho t sobre um alfabeto A que tem as seguintes propriedades:

- o espaço amostral da chave K consiste de todos os conjuntos de pares ordenados de t permutações $(p_1, p_2, ..., p_t)$ onde cada permutação p_i está definida no conjunto A;
- encriptação da mensagem m = $(m_1, m_2, ..., m_t)$ com a chave e = $(p_1, p_2, ..., p_t)$ é dado por E_e(m) = $(p_1(m_1), p_2(m_2), ..., p_t(m_t))$; e
- a chave de decriptação associada com e = $(p_1, p_2, ..., p_t)$ é d = $(p_1^{-1}, p_2^{-1}, ..., p_t^{-1})$.

A substituição polialfabética surgiu durante o período da Renascença e foi considerada inquebrável durante muito tempo (KAHN, 1996). Nesse caso, cada letra do texto original é substituída por um alfabeto diferente. Geralmente a chave consiste de **N** alfabetos o que faz com que a enésima primeira letra do texto original seja substituída pelo mesmo alfabeto utilizado na primeira. A substituição polialfabética mais popular é a de Vigenère, que consiste numa tabela com o alfabeto deslocado de **0** até **N-1** posições (sendo **N** o número de letras do alfabeto utilizado) e de uma palavra-chave que indica quais as linhas da tabela serão aplicadas ao texto simples.

Considere o exemplo mostrado na Tabela 5. A palavra-chave é “secreto", o texto da mensagem é “você foi selecionado"e a tabela de Vigenère exibida na Figura 6.

As cifras polialfabética tem como vantagem sobre as cifras de substituição simples o fato de não preservar a frequência de ocorrência dos símbolos. Entretanto, estas cifras não são significativamente mais difíceis de serem descobertas utilizando-se as técnicas de criptoanálise. A abordagem utilizada é semelhante às das cifras de substituição simples. De fato, uma vez que o tamanho t da chave é descoberto, as cifras podem ser divididas em t grupos (onde um grupo i, $1 \leq i \leq t$, consiste das cifras derivadas usando permutação p_i), e a análise de frequência pode ser feita para cada grupo.

Tabela 5 – Exemplo substituição polialfabética

chave	s	e	c	r	e	t	o	s	e	c	r	e	t	o	s	e	c	r
mensagem	v	o	c	e	f	o	i	s	e	l	e	c	i	o	n	a	d	o
criptograma	p	s	e	t	j	j	x	m	i	n	v	g	d	e	h	e	f	h

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A
> C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C
> E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
> O	O	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
> R	R	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
> S	S	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
> T	T	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
X	X	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	X

Figura 6 – Tabela de Vigenère

4.2.4 Técnicas de Transposição

As técnicas de transposição utilizam permutação dos símbolos da mensagem. Uma permutação em um conjunto de símbolos finitos S é uma bijeção de S para ele mesmo (p:S $\rightarrow$ S). Por exemplo, seja S = 1,2,3,4,5. Uma permutação p:S $\rightarrow$ S é definida da seguinte forma:

p(1) = 3, p(2) = 5, p(3)= 4, p(4)=2, p(5)=1.

Uma permutação pode ser descrita como uma matriz:

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 2 & 1 \end{pmatrix}$$

Onde a linha superior na matriz é o domínio e a linha inferior é a imagem sob o mapeamento p. Como as permutações são bijeções, elas têm inversas. Se uma permutação é escrita como uma matriz, seu inverso é facilmente encontrado trocando as linhas na matriz e reordenando os elementos na nova linha superior se desejado (a linha inferior teria que ser reordenada correspondentemente). A inversa de p é p^{-1}. Por exemplo:

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 1 & 3 & 2 \end{pmatrix}$$

Uma cifra de transposição é facilmente reconhecida porque tem as mesmas frequên-

cias das letras do texto original. Para o tipo de transposição colunar que acabamos de mostrar. Uma forma de tornar está técnica mais segura é realizar mais de uma fase de transposição. O resultado é uma permutação mais complexa que não é facilmente reconstruída.

4.2.5 One Time Pad

A técnica "*one time pad*"é uma proposta de um criptossistema que produz segurança máxima para proteção da chave. A ideia é usar uma chave aleatória que seja tão longa quanto a mensagem, para que a chave não precise ser repetida. Além disso, a chave deve ser usada para criptografar e descriptografar uma única mensagem e, em seguida, é descartada. Cada nova mensagem requer uma nova chave do mesmo comprimento que a nova mensagem. Shannon provou que esta técnica produz um "segredo perfeito", no sentido de que um ataque probabilístico para tentar distinguir entre dois possíveis candidatos a chave não fornece nenhuma informação adicional ao atacante (DELFS; KNEBL, 2015).

Um algoritmo E que, para uma entrada em texto simples m $\in$ M, produz uma cifra criptografada de saída c $\in$ C, é considerado de encriptação randômica se E for um algoritmo probabilístico não determinístico. O comportamento aleatório do algoritmo E é resultado da escolha aleatória de uma chave única (para cada mensagem a ser criptografada, uma nova chave aleatória deve ser escolhida, independentemente das opções anteriores). A seguir é apresentada a definição da Cifra de Vernam que é o exemplo clássico de uma cifra aleatória (e provavelmente segura):

Cifra de Vernam: Seja n $\in \mathbb{N}$ e M := C := $\{0,1\}^n$. A encriptação randômica **E** encripta a mensagem m $\in$ M pela operação lógica de ou exclusivo (*xor*) bit a bit com uma sequência de bits escolhida aleatoriamente e uniformemente $k \xleftarrow{u} \{0,1\}^n$ do mesmo comprimento da mensagem m, E(m):= m $\oplus$ k é chamada de cifra Vernam *one-time pad*.

A noção clássica de segurança de um algoritmo de encriptação está baseada nos trabalhos de Claude Shannon sobre a teoria da informação. Considerando o algoritmo de encriptação E que mapeia texto simples m $\in$ M em cifras c $\in$ C. Assumimos que as mensagens são encriptadas de acordo com alguma distribuição de probabilidade. Por exemplo M é considerado o espaço de probabilidade. A distribuição em M e o algoritmo E induzem a distribuição de probabilidade em M x C e C. Considere prob(m|c) a probabilidade de m ser o texto simples se c for o texto cifrado. Sem perda de generalidade, assume-se que prob(m) $>$ 0 para todo m $\in$ M e prob(c) $>$ 0 para todo c $\in$ C. Segundo Shannon, a encriptação E é um segredo perfeito se C e M

são independentes, ou seja, a distribuição de M x C é o produto da distribuição de M e C.

prob(m,c) = prob(m) x prob (c) para todo m $\in$ M e c $\in$ C.

4.3 Princípios de Criptossistemas de Chave Secreta

Os métodos descritos a seguir são, como todos os descritos anteriormente, simétricos. Isso significa que a mesma chave que é usada para cifrar é usada para decifrar.

4.3.1 Princípios de Cifras de Fluxo

Uma cifra de fluxo é aquela que criptografa uma sequência de símbolos (bits ou bytes) um a um por vez. Cifras de fluxo são, de certa forma, cifras de bloco muito simples com comprimento de bloco igual a um. A utilidade desta técnica está no fato de que a transformação criptográfica pode mudar para cada símbolo de texto não criptografado. Em situações onde os erros de transmissão são altamente prováveis, cifras de fluxo são vantajosas porque não têm propagação de erro. Exemplos de cifras de fluxo clássicas são a cifra Vigenère e a cifra de Vernam. Em uma situação ideal, uma versão de "*one time pad*"da cifra Vernam seria usada, em que a chave de fluxo é tão longa quanto o fluxo de bits de texto simples. Se a chave criptográfica de fluxo for aleatória, então essa cifra é inquebrável por qualquer meio que não seja a aquisição da chave (MENEZES *et al.*, 1996). No entanto, a chave de fluxo deve ser fornecida a ambos os usuários com antecedência por meio de algum canal independente e seguro. Isto cria problemas logísticos se o tráfego de dados for muito grande.

Consequentemente, por razões práticas, o gerador de fluxo de bits tem de ser implementado como um procedimento algorítmico, de modo que o fluxo de bits criptográfico possa ser produzido por ambos os utilizadores. Nesta abordagem, ilustrada na Figura 7, o gerador de fluxo de bits é um algoritmo controlado por chave e deve produzir um fluxo de bits que se assemelhe a uma sequência aleatória de bits, sendo assim considerado criptograficamente forte. Esta técnica necessita que a chave de geração seja compartilhada entre as partes que estão trocando as mensagens (STALLINGS, 2006).

Cifras de fluxo podem gerar longas sequências de aparência aleatória a partir de uma chave pequena. Isso pode ser conseguido com o uso de registros de deslocamento, tradução para *Linear Feedback Shift Register (LFSR)* (DELFS; KNEBL, 2015).

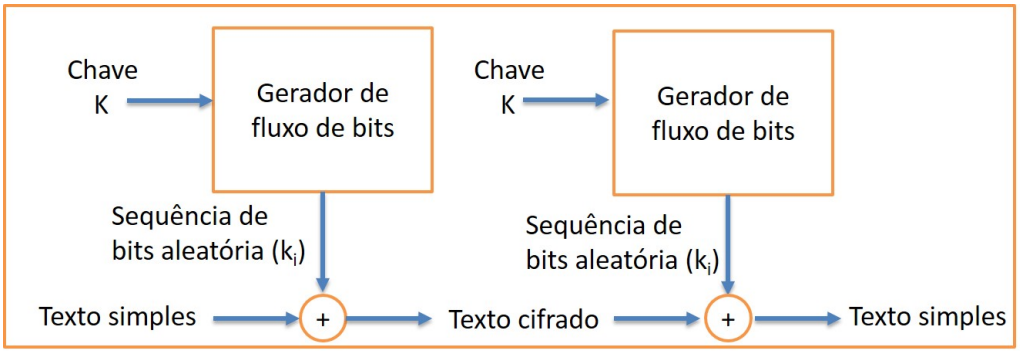

Figura 7 – Cifra de Fluxo

4.3.2 Princípios de Cifras de Blocos

Uma cifra de bloco é um criptossistema que divide as mensagens de texto simples a serem transmitidas em cadeias (chamados blocos) de um comprimento fixo **t** sobre um alfabeto **A** e criptografa um bloco de cada vez, produzindo um bloco de texto cifrado de igual comprimento. Tipicamente, é utilizado um bloco de tamanho 64 ou 128 bits. Duas importantes classes de cifras de bloco são cifras de substituição, cifras de transposição.

As cifras de produto são obtidas pela combinação das técnicas de substituição e transposição. As cifras simples de substituição e de transposição individualmente não proporcionam um nível de segurança muito elevado. No entanto, ao combinar estas duas transformações é possível obter cifras fortes, que são muito difíceis de serem descobertas pelo criptoanalista. A maioria dos sistemas de chave secreta utilizam cifras de produto.

Seja $\mathbf{E}_k$ uma operação de encriptação utilizando a chave**k**. Um exemplo de cifra de produto é uma composição de $\mathbf{t} \geq \mathbf{2}$ transformações $\mathbf{E}_{k1}, \mathbf{E}_{k2}, \cdots, \mathbf{E}_{kt}$ onde para cada $\mathbf{E}_{ki}$, $\mathbf{1} \geq \mathbf{i} \geq \mathbf{t}$, é uma substituição ou uma cifra de transposição. Normalmente a composição de uma substituição e uma transposição é chamada de “rodada"(DELFS; KNEBL, 2015).

Exemplo de cifra de produto: Seja **M = C = K** o conjunto de strings binárias de tamanho 6 bits. O número de elementos em **M** é $2^6 = 64$. Seja $\mathbf{m} = (\mathbf{m}_1\mathbf{m}_2...\mathbf{m}_6)$ e defina:

$E_k^{(1)}(m) = m \oplus k$, onde $k \in K$,

$E^{(2)}(m) = (m_4\ m_5\ m_6\ m_1\ m_2\ m_3)$.

Aqui, $\oplus$ é uma operação lógica de ou-exclusivo (XOR). $E_k^{(1)}$ é uma cifra de substituição polialfabética e $E^{(2)}$ é uma cifra de transposição (não envolvendo a chave). a aplicação de $E_k^{(1)}$ seguida da aplicação de $E^{(2)}$ é uma rodada.

Uma cifra de bloco trabalha com um bloco de texto simples de **n** bits para produzir um bloco cifrado de **n** bits. Existem $\mathbf{2}^n$ possíveis blocos de texto simples e, para a encriptação

ser reversível, cada bloco de texto simples deve produzir apenas um único bloco de texto cifrado. Tal transformação é chamada reversível, ou não singular. Este tipo de transformação reversível limita a quantidade de mapeamentos entre o texto simples e a cifra em $\mathbf{2^n!}$ combinações. Para o primeiro bloco de texto simples, pode-se escolher $\mathbf{2^n}$ blocos de texto cifrado, para o segundo bloco, temos $\mathbf{2^n - 1}$ e assim por diante. Para o último bloco de texto simples temos apenas 1 bloco de texto cifrado para mapear. Feistel refere-se a isto como a cifra de bloco ideal, porque permite o número máximo de possíveis mapeamentos de criptografia a partir do bloco de texto simples (STALLINGS, 2006).

Existe um problema prático com a cifra de bloco ideal. Para tamanhos de bloco pequenos, tal como **n = 4**, o sistema torna-se equivalente a uma cifra de substituição clássica. Esses sistemas, são vulneráveis a uma análise estatística de texto simples. Essa fraqueza não é inerente ao uso de uma cifra de substituição, mas resulta do uso de um bloco de tamanho pequeno. Se **n** for suficientemente grande e houver uma substituição arbitrária reversível entre o texto simples e o texto cifrado, então as características estatísticas do texto simples da fonte serão mascaradas de tal forma que este tipo de criptoanálise se torna inviável. Uma cifra de substituição reversível arbitrária (a cifra de bloco ideal) para um tamanho de bloco grande não é prática sob o ponto de vista de implementação e desempenho.

A Cifra de Feistel

Feistel propôs uma técnica de aproximação da cifra de bloco ideal, utilizando o conceito de uma cifra de produto, que é a execução de duas ou mais cifras simples em sequência de tal forma que o resultado final ou produto seja criptograficamente mais forte do que qualquer um dos componentes de cifra (STALLINGS, 2006).

A cifra de Feistel consiste em uma cifra de bloco com uma chave de tamanho **k** bits e um bloco de **n** bits de tamanho, permitindo um total de $\mathbf{2^k}$ possíveis transformações, ao invés de $\mathbf{2^n!}$ transformações disponíveis na cifra ideal. A proposta de Feistel alterna o uso das técnicas de substituição (confusão) e permutação (difusão) sobre os blocos de texto simples para gerar os blocos de texto cifrado. Trata-se de uma implementação prática das ideias de Shannon que utiliza as funções de confusão e difusão, as quais são descritas a seguir:

- confusão: esta técnica procura tornar a relação entre as estatísticas do texto cifrado e o valor da chave de criptografia tão complexa quanto possível, para impedir as tentativas de descoberta da chave. Mesmo que o atacante obtenha alguma informação sobre as estatísticas do texto cifrado, a maneira pela qual a chave foi usada para produzir esse

texto cifrado é tão complexa, que torna difícil deduzir o valor da chave. Isso é conseguido através da utilização de um algoritmo complexo de substituição.

- difusão: a estrutura estatística do texto simples é dissipada em uma longa faixa de valores estatísticos das cifras. Isto é conseguido quando cada símbolo do texto simples afeta vários símbolos do texto cifrado. Um elemento de difusão simples é a permutação de bits, que é frequentemente usada no DES. A difusão pode ser conseguida executando repetidamente alguma permutação nos dados seguida pela aplicação de uma função a essa permutação. Com efeito, consegue-se que bits de diferentes posições no bloco de texto simples contribuam para um único bit no bloco de texto cifrado.

Segundo Stallings, as técnicas de difusão e a confusão foram tão bem sucedidas ao captar a essência da aleatoriedade desejada dos atributos de uma cifra em bloco que se tornaram a pedra angular do design de cifra de bloco moderno (STALLINGS, 2006).

A estrutura de iterações (rodadas) proposta por Feistel é chamada de rede de Feistel. A seguir será explicado como esta rede funciona para transformar um bloco de texto simples em um cifra, conforme Figura 8.

1. O bloco é dividido em 2 sub-blocos de mesmo comprimento L_0 (esquerda) e R_0 (direita)
2. As duas metades passam por 16 rodadas de processamento/combinação
3. A entrada (L_{n-1}, R_{n-1}) da n-ésima rodada é transferida para a saída da n rodada, tanto como a subchave
4. Uma substituição é feita na metade da esquerda através da aplicação de uma função $f_i()$ na parte da direita. $f_i()$ é uma uma função complexa, tipicamente não linear, que produz um valor dado um sub-bloco e uma subchave K_i derivada da chave global K usada pelo algoritmo.

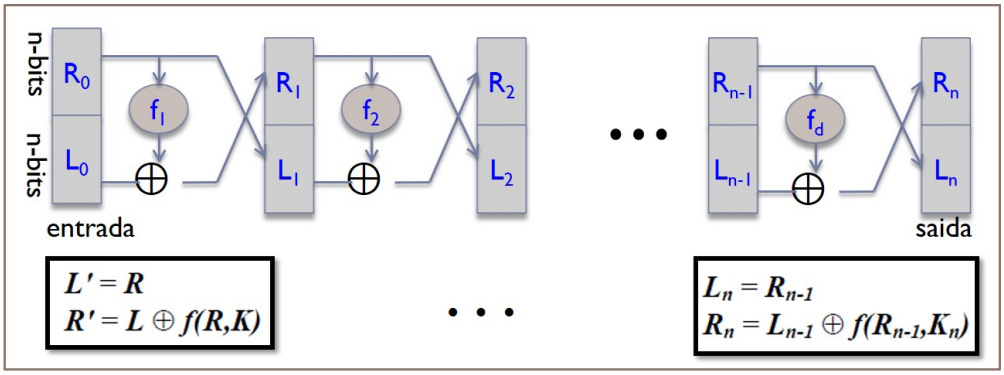

Figura 8 – Rede de Feistel

O processo de descriptografia dos dados é feito de forma inversa, utilizando-se a

mesma chave.

1. Para cada rodada i = 16, 15, 14, ..., 3, 2, 1 compute
2. $L_{n-1} = R_n$
3. $R_{n-1} = L_n \oplus f(R_n, K_n)$
4. Então (L_0,R_0) é o texto simples novamente.

A inversibilidade de uma rede de Feistel é independente das características da função f(), o que facilita a utilização de funções não lineares que mais facilmente podem tornar o criptossistema mais complexo. Existem vários criptossistemas que usam a rede de Feistel, que se diferenciam pela função f() que usam. Como por exemplo o DES(Data Encryption Standard), Lucifer, FEAL, Khufu, LOKI, GOST, Blowfish entre outros (SCHNEIER, 2015).

4.3.3 DES (Data Encrpytion Standard)

O criptossistema adotado em 1977 pelo NBS (*Nationa1 Bureau of Standards*) americano, conhecido como DES (*Data Encryption Standard*), baseia-se no LUCIFER, desenvolvido na IBM por Horst Feistel. O DES é uma cifra simétrica por blocos que cifra blocos de 64 bits de texto simples usando uma chave de 56 bits. O processo de cifrar e decifrar é o mesmo e consiste numa série de transposições e substituições. A cifra pode ser aplicada a blocos de bits de diversas maneiras, tendo sido apresentados inicialmente quatro modos de cifra, descritos a seguir (FIPS, 1999):

- *Electrônic Code Book (ECB)*: cada bloco é cifrado individualmente, conforme ilustrado na Figura 9;

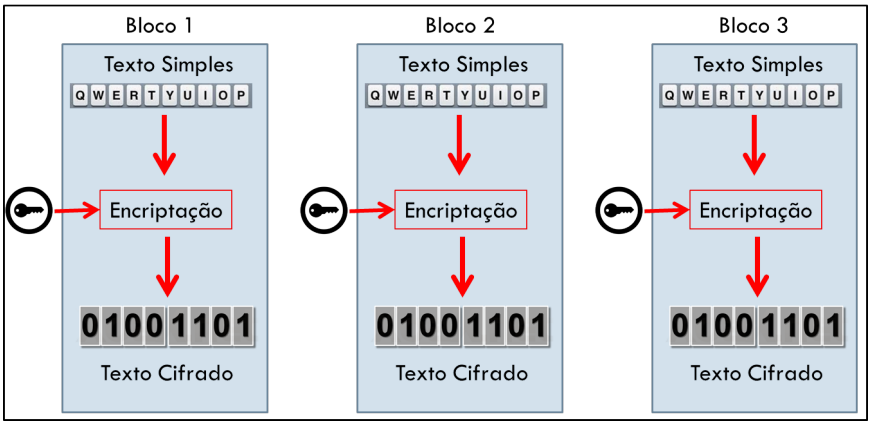

Figura 9 – Electronic Code Book (ECB)

- *Cipher-Block Chaining (CBC)*: é realizada uma operação de XOR com cada bloco de texto anterior antes de ser encriptado. É utilizado um vetor de inicialização para o primeiro

bloco, conforme ilustrado na Figura 10;

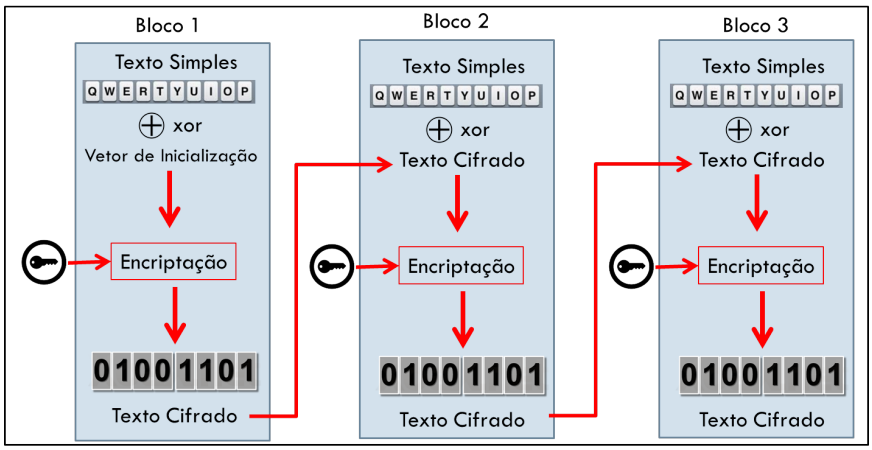

Figura 10 – Cipher-Block Chaining (CBC)

- *Cipher Feedback (CFB)*: Este modo é mais adequado para cifrar quantidades muito pequenas de dados (*bytes* ou blocos pequenos), como por exemplo, *bytes* individuais que formam um *stream* de *bytes*. Neste modo, é necessário um vetor de inicialização R para dar início ao processo. Esse vetor de inicialização funcionará como um registrador de deslocamento R (*shift register*). O byte da extremidade mais à esquerda do registrador de deslocamento R é selecionado. Uma operação XOR é feita com o byte da vez, do bloco. O registrador R é deslocado 8 bits à esquerda, fazendo com que o seu byte mais à esquerda seja excluído e o byte cifrado atual seja inserido na posição que ficou vaga na extremidade do registrador mais à direita, criando assim um movimento circular no registrador, conforme ilustrado na Figura 11.

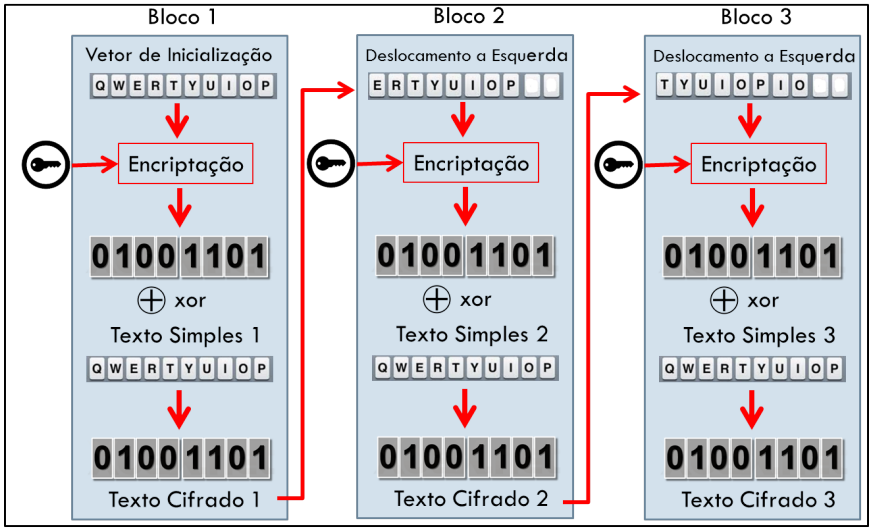

Figura 11 – Cipher Feedback (CFB)

A desvantagem do método CFB é que se um bit do texto cifrado for invertido acidentalmente durante a transmissão, os bytes no registrador de deslocamento R no receptor, serão danificados, enquanto o byte defeituoso estiver no registrador de deslocamento.

Internamente, a operação de cifra segue os princípios de confusão e difusão de Shannon, usando permutação, substituição, expansão e compressão de blocos em 16 interações.

Cada rede de Feistel utiliza uma subchave própria de 48 bits derivada da chave de 56 bits. A não linearidade da função interna de cada rede Feistel é assegurada por unidades de substituição chamadas caixas S (S-boxes). A Figura 12 ilustra de forma resumida, o modo de operação do DES.

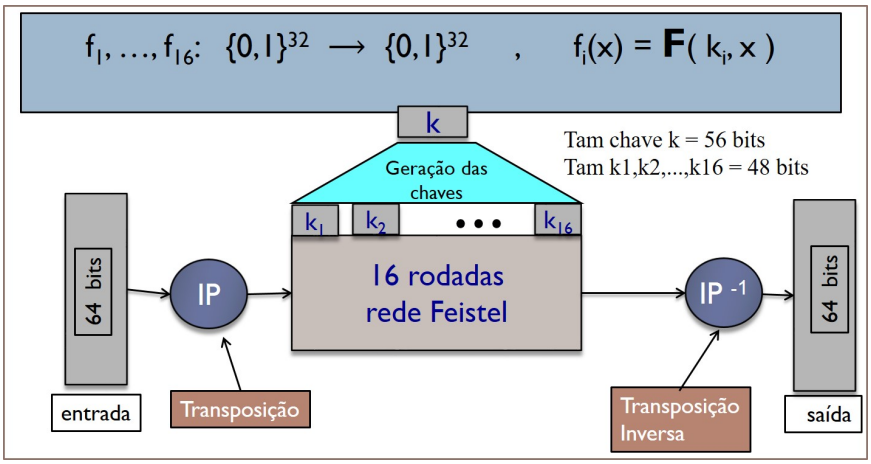

Figura 12 – Operação do Algoritmo DES

No início do processo, o bloco de texto simples passa por uma transposição inicial (*Initial Permutation - IP*) que permuta os 64 bits de entrada de acordo com a matriz de posições apresentada na Figura 13. O bit 1 é trocado pelo bit 58, o bit 2 é trocado pelo bit 50, o processo segue até o final, quando o bit 64 é trocado pelo bit 7. Isto é seguido por uma fase que consiste em dezesseis rodadas da mesma função, que envolve a permutação e as funções da substituição.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Figura 13 – Tabela de Transposição Inicial do DES

A transposição inversa (IP^{-1}), realizada ao final das 16 rodadas da rede de Feistel, permuta os 64 bits de saída de acordo com a matriz de posições apresentada na Figura 14. O bit 1 é trocado pelo bit 40, o bit 2 é trocado pelo bit 8, o processo segue até o final, quando o bit 64 é trocado pelo bit 25.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Figura 14 – Tabela de Transposição Final do DES

O detalhamento da rodada da rede de Feistel que é implementada pelo DES é apresentado na Figura 15. As metades esquerda e direita de cada valor intermediário de 64 bits são tratadas como quantidades separadas de 32 bits, rotuladas L (esquerda) e R (direita). A seguir, serão apresentados os passos da função f() do DES (FIPS, 1999):

- Passo 1: O bloco direito de 32 bits é expandido para 48 bits (replicação e movimentação de bits).
- Passo 2: A chave K de 56 bits é dividida em 2 blocos de 28 bits que são girados a esquerda em uma quantidade de bits que depende do número da iteração. K_i é derivada destes blocos girados pela aplicação de uma transposição de 56 bits sobre eles, que gera uma chave comprimida de 48 bits.
- Passo 3: É feita uma operação de OU-Exclusivo (XOR) entre os 48 bits de R e a chave K_i
- Passo 4: Os 48 bits de saída da operação de XOR são processados por um função de substituição (caixa S-Box), que produz 32 bits.
- Passo 5: Os bits de saída da caixa S-Box são passados para uma função de permutação (P-Box) que produz 32 bits de saída.

A segurança do DES está em função do tamanho da chave e do número de rodadas de permutação/substituição. A segurança do algoritmo depende fortemente das caixas *S-boxes*. Tudo o mais no DES é linear. O DES só pode ser atacado utilizando pesquisa exaustiva, o que hoje em dia é viável tecnicamente e economicamente, por sua chave ter apenas 56 bits. A solução mais usada para superar esta deficiência é usar cifra múltipla (triple DES).

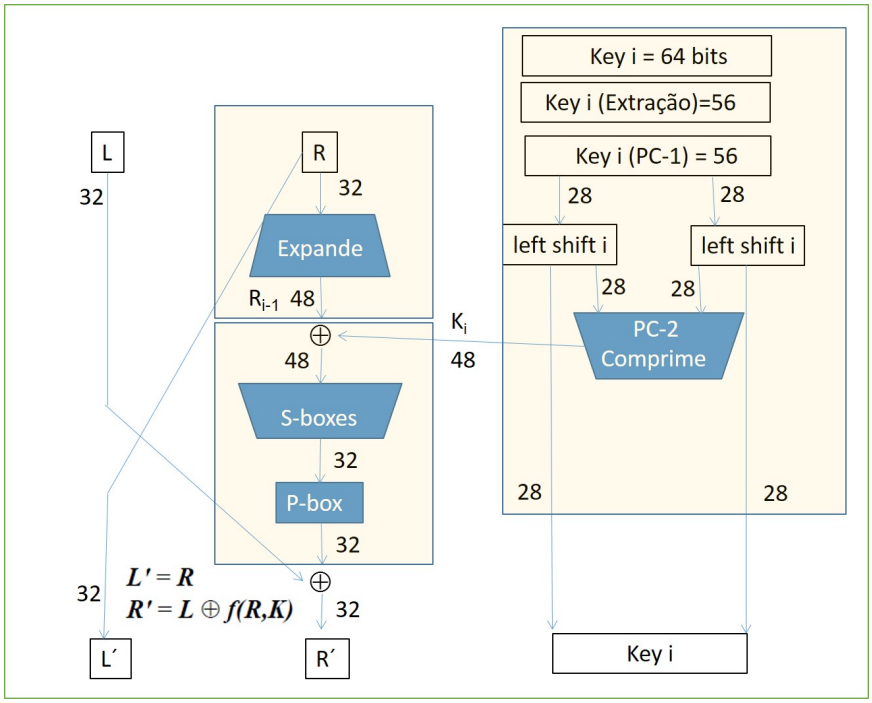

Figura 15 – Rodada da Rede de Feistel no Algoritmo DES

A cifra múltipla consiste em cifrar um texto mais do que uma vez, usando em cada cifra uma chave diferente. A segurança será tanto maior quanto maior for o número de cifras independentes aplicadas, mas o desempenho do algoritmo será menor. É preciso encontrar um ponto de equilíbrio entre desempenho e segurança, efetuando-se apenas as cifras necessárias e suficientes para garantir os níveis de segurança desejados.

A cifra dupla foi inicialmente proposta, mas posteriormente abandonada porque não reforça significativamente a segurança (MITCHELL, 2016). De fato, ao se usar chaves com n bits, consegue-se descobrir as chaves em 2^{n+1} tentativas e não em 2^{2n}, como seria de se esperar.

A cifra tripla foi proposta em substituição à cifra dupla, usando três operações de encriptação e decriptação e uma a três chaves distintas. O modo mais usual de cifra tripla é o Encripta-Decripta-Encripta (EDE), onde se efetuam, sequencialmente, uma encriptação, uma decriptação e uma encriptação. A Tabela 6 mostra operações de cifra tripla com variações na quantidade de chaves.ANSI X9.52

Tabela 6 – Modo de tripla cifra EDE com 1,e ou 3 chaves

Número de chaves	Cifra tripla EDE	Decifra tripla EDE
1	$E_k(D_k(E_k(m))) \rightarrow c$	$D_k(E_k(D_k(c))) \rightarrow m$
2	$E_{k1}(D_{k2}(E_{k1}(m))) \rightarrow c$	$D_{k1}(E_{k2}(D_{k1}(c))) \rightarrow m$
3	$E_{k3}(D_{k2}(E_{k1}(m)))$	$D_{k1}(E_{k2}(D_{k3}(c))) \rightarrow m$

O modo EDE permite compatibilizar cifras simples com cifras triplas usando uma

única chave, o que permite compatibilidade com aplicações ou equipamentos antigos. Ao usar a cifra tripla EDE apenas com uma chave, o resultado é o equivalente a uma cifra simples (porque a cifra e decifra iniciais se anulam). A cifra tripla EDE é usada fundamentalmente com o DES (3DES-EDE) para minimizar problemas decorrentes do comprimento reduzido das chaves de 56 bits.

4.3.4 AES

O desenvolvimento de um padrão de encriptação avançado (AES) foi proposto em 1997 pelo *National Institute of Standards and Technology (NIST)*. O NIST encorajou potenciais participantes em todo o mundo a apresentar propostas para a nova norma. Os requisitos para as propostas de algoritmo do concurso foram as seguintes:

- O algoritmo deveria ser definido publicamente;
- deveria suportar tamanho de bloco de pelo menos 16 bytes (128 bits);
- deveria permitir que o tamanho da chave pudesse aumentar 3 tamanhos de chave de 16, 24 ou 32 bytes, correspondentes a 128, 192 ou 256 bits;
- deveria ser implementado em hardware e software;
- deveria ser disponibilizado livremente.

O processo de seleção foi dividido em duas rodadas. Na primeira rodada, 15 das 21 propostas apresentadas foram aceitas como candidatas AES. Os candidatos foram avaliados por uma discussão pública. A comunidade criptográfica internacional foi solicitada a fazer comentários sobre as cifras de bloco propostas. Cinco candidatos foram escolhidos para a segunda rodada: MARS (IBM), RC6 (RSA), Rijndael (Daemen e Rijmen), Serpent (Anderson, Biham e Knudsen) e Twofish (Counterpane). Foram realizadas três “Conferências de Candidatura AES"internacionais e, em outubro de 2000, o NIST selecionou a cifra de Rijndael para ser a AES.

A Figura 16 (STALLINGS, 2006) mostra a estrutura geral do processo de criptografia AES. A entrada para os algoritmos de criptografia e descriptografia é um único bloco de 128 bits. Este bloco é copiado para a matriz “Estado", que é modificada em cada estágio de criptografia ou descriptografia. Após o estágio final, a matriz “Estado"é copiada para uma matriz de saída (GUERON *et al.*, 2016).

Da mesma forma, a chave é representada como uma matriz quadrada de 16 bytes. Esta chave é então expandida para um matriz de bytes de 44 palavras de 4 bytes cada uma (w_0,

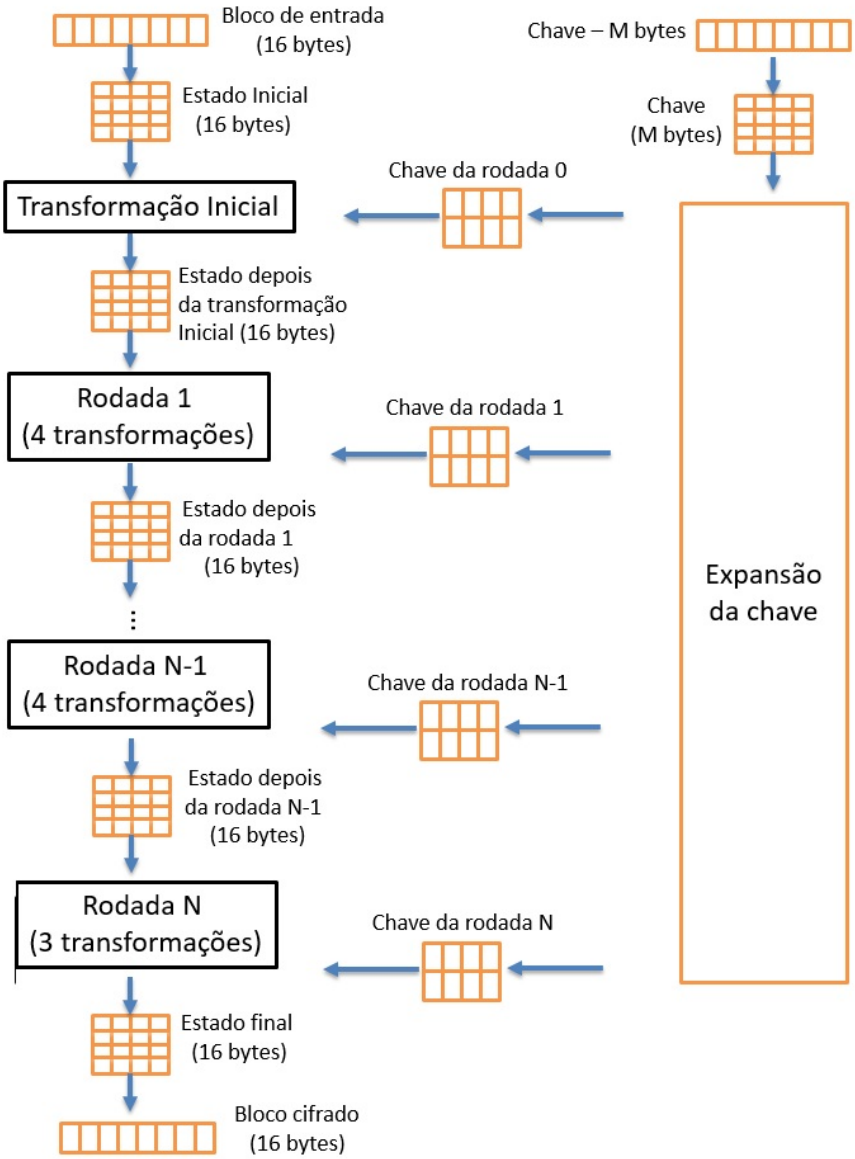

Figura 16 – Processo de Encriptação do AES

w_1,..., w_{43}), totalizando 176 bytes (1408 bits), conforme ilustrado na Figura 17.

Figura 17 – Expansão da Chave do AES

A cifra consiste em N rodadas, onde a quantidade N depende do comprimento da chave: 10 rodadas para uma chave de 16 bytes, 12 rodadas para uma chave de 24 bytes e 14 rodadas para uma chave de 32 bytes. As primeiras N-1 rodadas executam 4 funções distintas de transformação: SubBytes, ShiftRows, MixColumns e AddRoundKey, que serão descritas posteriormente. Há uma transformação única inicial (Rodada 0) antes da primeira rodada e uma

rodada final com apenas três transformações. Cada transformação toma uma ou mais matrizes 4x4 como entrada e produz uma matriz 4x4 como saída. A Figura 16 mostra que a saída de cada rodada é uma matriz 4x4, sendo a saída da rodada final o texto cifrado. Além disso, a função de expansão de chave gera N+1 chaves de rodadas, cada uma das quais é uma matriz 4x4 distinta. Cada chave de rodada serve como uma das entradas para a transformação da função AddRoundKey em cada rodada (STALLINGS, 2010).

4.4 Princípios de Criptossistemas de Chave Pública

O desenvolvimento de criptossistemas de chave pública ou assimétricos foi sem dúvida uma solução interessante para o problema de distribuição de chaves num sistema criptográfico. Estes criptossistemas utilizam uma função assimétrica em que uma chave pública é usada para encriptar e outra chave privada distinta é usada para decriptar. Além disso há uma dificuldade em descobrir uma chave a partir da outra, baseada na dificuldade de fatorar números primos muito grandes, de forma que uma chave não pode ser deduzida a partir da outra, ou pelo menos que a chave pública não forneça nenhuma informação sobre a chave privada. Alguns desses criptossistemas ainda permitem a assinatura digital, ou seja, que um usuário tenha certeza (e possa prová-lo perante um juiz) de que a mensagem recebida foi enviada pelo emissor da mensagem. Em outras palavras, a mensagem pode ser “assinada"pelo remetente.

Os sistemas de chave pública foram inicialmente propostos por Diffie e Hellman em 1976 (DIFFIE; HELLMAN, 2006). Esses criptossistemas baseiam-se em problemas conhecidos como NP-completos, para os quais não se conhece nenhuma solução que execute em tempo polinomial, mas uma vez dada uma provável solução é fácil verificar se ela é verdadeira ou não.

Os criptossistemas de chave publica baseiam-se na ideia de que cada usuário possui um par de chaves (**d**, **e**) sendo **d** a chave privada e portanto secreta do usuário (conhecida também como chave de descriptografia) e **e** a chave publica que pode ser fornecida a qualquer pessoa (conhecida também como chave de criptografia).

Como exemplo para entendimento do criptossistema de chave pública, suponha que existe uma caixa inquebrável com uma fechadura que permite a entrada de duas chaves (**d**,**e**). Estas chaves possuem a seguinte propriedade: a chave **e** só permite o movimento de giro para a esquerda enquanto que a chave **d** só permite movimentos giratórios para a direita. Suponha que a caixa esteja aberta quando a tranca da fechadura está na posição central, ou seja, qualquer outra posição fecha terminantemente a caixa. Dessa maneira, um emissor pode colocar uma mensagem

na caixa, trancá-la com sua chave, enviá-la através de um meio qualquer e ter certeza de que ela só será aberta pelo destinatário. Qualquer uma das chaves pode trancar a caixa (codificar a mensagem), usando a outra para abri-la (decodificar a mensagem). Por causa da complexidade computacional da encriptação assimétrica, ela é tipicamente usada apenas para transferir uma chave de encriptação simétrica pela qual a mensagem (e normalmente a mensagem inteira) é encriptada.

O modelo de criptografia de chave pública pode ser representado de forma mais genérica. Dado um par de chaves (**d**,**e**) e uma mensagem M, devemos ter as seguintes relações (STALLINGS, 2010):

- Denotando por D() a aplicação da chave particular **d**, que transforma M em D(M) = C, onde C é a mensagem criptografada. Então, E(C) = M onde E() denota a aplicação da chave **e**. Ou seja, E(D(M)) = M (**e** é a chave inversa da chave **d**).
- O calculo do par de chaves (d, e) é computacionalmente fácil, ou seja, pode ser dado por um algoritmo de tempo polinomial.
- É computacionalmente difícil calcular **d** a partir do conhecimento de **e** e do algoritmo de criptografia.
- Os cálculos de D() e E() são computacionalmente fáceis para quem conhece as chaves.
- É computacionalmente difícil calcular D() sem conhecer a chave **d**.
- Qualquer uma das duas chaves relacionadas pode ser utilizada para a encriptação, com a outra sendo utilizada para a decriptação.

A Tabela 7 sumariza alguns aspectos importantes das técnicas de criptografia simétrica e pública. A chave utilizada na criptografia simétrica será denominada de **chave secreta** e as chaves utilizadas na criptografia assimétrica serão chamadas de **chave pública** e **chave privada**.

Os criptossistemas de chaves publicas, além de resolverem o problema da distribuição de chaves que ocorrem nos criptossistemas de chave secreta ou simétricos, permitem solucionar problemas conhecidos como:

- **Autenticacão de Destino**: esconder informações sigilosas das pessoas que controlam as linhas de comunicação e os computadores intermediários (provedores), garantindo que só o verdadeiro destinatário consiga ler a informação enviada.
- **Autenticacão da Origem**: evitar que uma pessoa mal-intencionada personalize ou falsifique a identidade do emissor enviando informação para o destinatário, ou seja, o destinatário

Tabela 7 – Criptografia Simétrica e Assimétrica

Criptografia Simétrica	Criptografia Assimétrica
Necessita para funcionar: 1. o mesmo algoritmo com a mesma chave é utilizado para criptografar e descriptografar 2. o emissor e receptor devem compartilhar o algoritmo e a chave	Necessita para funcionar: 1. um algoritmo é utilizado para encriptação e decriptação com um par de chaves, uma para encriptar e outra para decriptar 2. o emissor e receptor devem uma das chaves do par de chaves (não a mesma chave)
Necessita para segurança: 1. as chaves devem ser mantidas secretas 2. deve ser impossível ou impraticável decifrar a mensagem se nenhuma outra informação estiver disponível 3. conhecimento do algoritmo mais amostras de texto criptografado devem ser insuficientes para determinar as chaves	Necessita para segurança: 1. uma das duas chaves deve ser mantida secreta 2. deve ser impossível ou impraticável decifrar a mensagem se nenhuma outra informação estiver disponível 3. conhecimento do algoritmo, mais uma das chaves, mais amostras de texto criptografado devem ser insuficientes para determinar a outra chave

quer ter certeza de que foi o verdadeiro emissor que enviou a informação.

- **Detecção de Integridade de Informação**: Evitar que uma pessoa mal-intencionada altere parte da informação que transita na linha de comunicação, antes de chegar ao destinatário ou após o recebimento e armazenamento no computador deste. Assim, o emissor gostaria de detectar se alguma alteração foi feita na linha ou no local de armazenamento dos dados.

Para entendermos como a criptografia de chave publica auxilia na resolução dos problemas acima mencionados, introduziremos a seguir o criptossistema RSA. Para tanto, denotaremos daqui para frente Bob como sendo emissor ou remetente de uma mensagem $M \in \mathbb{Z}_N$, Alice como sendo a receptora ou destinatária desta mensagem e Carlos um mal-intencionado que deseja ler informação sigilosa alheia. Denotaremos C como um texto ilegível, resultado da codificação da mensagem M (que supomos legível).

4.4.1 Criptossistema RSA

O algoritmo RSA, criado em 1976, leva o sobrenome dos criadores: Ronald **R**ivest, Adi **S**hamir e Leonard **A**dleman. O esquema do algoritmo RSA é uma cifra de bloco em que um texto e uma cifra são representados como inteiros entre 0 e (n-1) para algum valor de **n** entre

1024 e 21.024 bits. RSA utiliza equações exponenciais. O tamanho do bloco deve ser menor ou igual a $\log_2(n) + 1$. Na prática, o bloco tem tamanho de **i** bits, onde $2^i < n < 2^{i+1}$. Um valor típico para **n** é 1024 bits, ou 309 dígitos decimais. A segurança deste algoritmo se baseia no fato de não existir até agora um algoritmo computacionalmente eficiente que fatore um número que seja produto de dois números primos muito grandes.

Em linhas gerais, podemos descrever três algoritmos que constituem o criptossistema RSA. O primeiro é responsável pela geração do par de chaves pública e privada, o segundo e o terceiro são responsáveis pela criptografia e descriptografia de uma dado bloco de texto $M \in Z_{\mathbb{N}}$. Estes algoritmos são descritos a seguir:

O algoritmo de geração de chaves utiliza um parâmetro de segurança **n** como entrada, que indica o tamanho em bits do módulo RSA. Para obter um par de chaves, um usuário ou uma entidade U calcula, através de um algoritmo probabilístico (STALLINGS, 2006), dois primos **p** e **q** de $\lfloor n/2 \rfloor$ bits e um inteiro $\mathbf{N} \leftarrow pq$ conhecido como módulo RSA (assim **N** possui n bits). A seguir, U escolhe um $e \in Z$ relativamente primo à função totiente de Euler $\phi(N) = (p-1)(q-1)$. O inteiro **e** é escolhido de tal forma que $1 < e < \phi(N)$ e MDC(e , $\phi(N)$) = 1 (**e** e $\phi(N)$ são primos entre si). O inteiro **e** é chamado expoente de criptografia, e é normalmente fixado com o valor 65537 (BONEH; SHACHAM, 2002). O valor de d é calculado pelo inverso multiplicativo modular, com a fórmula $d = e^{-1} \pmod{\phi(N)}$. A chave pública de U é dada por (N;e) e a chave privada de U é dada por (N;d), satisfazendo $ed = 1 \bmod \phi(N)$.

O segredo da chave é preservado pelos números **p** e **q**. Para que um espião descubra o segredo, ele deverá primeiro descobrir quem são **p** e **q**. Uma vez que esses números são muito grandes, será extremamente trabalhoso fatorar **N** (que é um dado público).

Os algoritmos de encriptação e decriptação são executados da seguinte forma para um bloco de texto M e um bloco de cifra C:

$C = M^e \bmod n.$

$M = C^d \bmod n = (M^e)^d \bmod n = (M^{ed}) \bmod n.$

O emissor e receptor da mensagem devem conhecer o valor de **n**. O emissor conhece o valor de **e**, e somente o receptor conhece o valor de **d**. Para que este sistema funcione satisfatoriamente, os seguintes requisitos devem ser atendidos:

- deve ser possível encontrar os valores de **e**, **d** e **n**, de tal forma que $(M^{ed}) \bmod n = M$ para todo $M < \mathbf{n}$.
- é relativamente fácil calcular $(M^e) \bmod n$ e $(C^d) \bmod n$ para todos os valores de $M < n$.

- é inviável calcular **d**, apenas com o conhecimento de **e** e **n**.

Um exemplo do algoritmo RSA é mostrado na Figura 18. Para este exemplo as chaves são geradas da seguinte maneira:

- Escolher p = 47 e q = 59
- Calcular n = p x q = 47 x 59 = 2773
- Calcular $\phi(2773) = \phi(47x59)$ = (47 - 1)x(59 - 1) = 46 x 58 = 2668
- Escolher e de forma que $1 < e < \phi(2773)$. Considerar e = 17
- Calcular d de forma que de = $1 \equiv$ mod(2668): d $\equiv 17^{-1}$ (mod 2668) $\rightarrow$ d = 157 (Prova: 157 x 17 = 2669 $\rightarrow$ 2669 $\equiv$ 1 (mod 2668))
- Encriptar 920: C $\equiv 920^{17}$ (mod 2773) $\rightarrow$ C = 948
- Decriptar: 948: m $\equiv 948^{157}$ (mod 2773) $\rightarrow$ m= 920

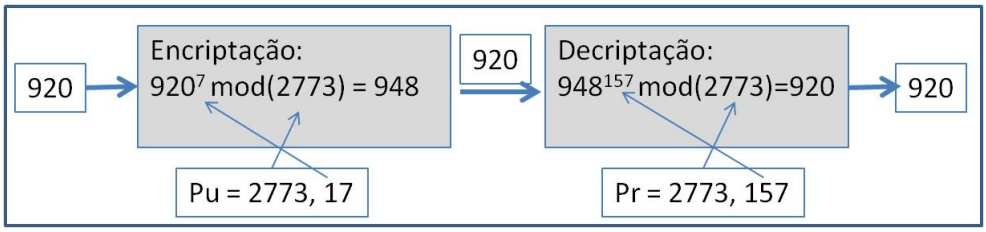

Figura 18 – Exemplo do Algoritmo RSA

A segurança do algoritmo RSA pode ser atacada de quatro formas diferentes: força bruta, ataques matemáticos, ataques temporais, escolha de texto criptografado. O ataque de força bruta significa testar todas as possíveis chaves privadas. Os ataques matemáticos consistem na tentativa de tentar fatorar os dois números primos **p** e **q**. Os ataques temporais dependem do tempo de execução do algoritmo de decriptação. Os ataques de texto criptografado escolhido (*adaptative-choosen-plaintext*) exploram propriedades do algoritmo RSA (STALLINGS, 2006).

O ataque de força bruta falha na dificuldade de fatoração de números inteiros extensos, mas com o aumento do poder de processamento dos computadores, tarefa de fatoração de números primos extensos, vem se tornando exequível. Alguns exemplos disto são os seguintes: em 1999 uma chave RSA com 512 bits foi fatorada em 6 meses e em 2009, uma chave de 768 bits foi fatorada em 4 anos. Desde setembro de 2014, a recomendação do NIST para o tamanho da chave RSA é de 3072 bits (BARKER *et al.*, 2014).

O ataque temporal leva em consideração a diferença de tempo para realização da operação de exponenciação modular pelo computador. Uma exponenciação modular é realizada bit a bit, com uma multiplicação modular realizada em cada iteração, existindo uma multiplicação

modular adicional executada para cada 1 bit. Portanto, há uma diferença de tempo na execução do algoritmo de descriptografia, que é mais lento ao processar o bit 1 e mais rápido ao processar o bit 0. Na prática, as implementações de exponenciação modular não têm variações de temporização extremas, nas quais o tempo de execução de uma única iteração pode exceder o tempo de execução médio de todo o algoritmo. No entanto, há variações suficientes para tornar este ataque prático (KOCHER, 1996).

4.5 Modelos de Preservação da Privacidade

4.5.1 Busca Criptográfica

A Busca Criptográfica (*Searchable Encryption*) é uma técnica que provê funcionalidades de pesquisa em dados encriptados sem requerer a chave de encriptação (XIA *et al.*, 2016). Esta técnica utiliza duas partes: um cliente e um servidor que armazena um banco de dados D encriptado, onde o cliente possui uma chave de acesso Q e a utiliza para obter o resultado da consulta Q(D) sem revelar o texto e o resultado da consulta para o servidor. Uma chave de acesso é um conjunto de palavras codificadas que estão relacionadas a palavras-chaves associadas aos registros da tabela pesquisada no banco de dados. A consulta retornará os registros em que houver coincidência entre as palavras da chave de acesso Q e as palavras dos registros da tabela. Como exemplo de um cenário de uso de busca encriptada, suponha que um determinado cliente deseja armazenar seus dados médicos criptografados em um banco de dados na nuvem, de forma que possa recuperar os registros seletivamente. Para usar a busca criptográfica, o cliente associa e criptografa um conjunto de palavras-chaves para cada registro da tabela, por exemplo, tipo da doença. Os registros dos dados médicos também são criptografados usando algum esquema de criptografia padrão. As palavras-chaves e os dados médicos são armazenados em uma tabela no banco de dados. Para consultar registros que estejam associados com a palavra “diabetes", o cliente cria uma chave de consulta Q usando a palavra “diabetes"criptografada e envia a consulta para o servidor, que verifica cada palavra-chave da tabela e seleciona os registros onde existe correspondência entre a chave de consulta e a palavra-chave “diabetes"criptografada, retornando estes registros para o cliente, caso existam. Neste caso, o servidor obtém a informação de quais registros foram retornados, mas não aprende nada sobre o conteúdo destes registros.

Esquemas de busca criptográfica podem utilizar criptossistemas baseados em chave simétrica ou chave pública, que são adequados para atributos multiusuário, em que qualquer

Tabela 8 – Comparação entre Esquemas de Busca Criptográfica

	Busca criptográfica com chave simétrica	**Busca criptográfica com chave pública**
Construção do texto cifrado pesquisável	Criado por uma chave secreta	Criado por parâmetros públicos
Gerenciamento da chave	Atributos de usuário único	Atributos de multiusuário
Funcionalidade	Busca por um palavra chave	Busca por uma palavra chave e decriptação parcial dos dados
Desempenho	Mais eficiente	Menos eficiente

cliente pode encriptar os dados utilizando parâmetros públicos, mas somente um usuário pode realizar consultas aos dados. Nos criptossistemas de chave simétrica, apenas o proprietário da chave secreta pode criar as palavras-chaves. A Tabela 8, adaptada de (Sedghi, 2012), mostra uma comparação entre os criptossistemas de criptografia de chave pública e chave simétrica

4.5.2 Private Information Retrieval

Para proteger a privacidade do padrão de acesso a dados, a intenção de cada operação de acesso a dados deve ficar escondida de forma que quem estiver observando a transação não obtenha nenhuma informação significativa (SUN; JAFAR, 2017). *Private Information Retrieval*-PIR é uma técnica de consulta em bancos de dados públicos não criptografados com proteção à violação de privacidade de acesso dos usuários. Uma violação de privacidade de acesso ocorre quando, além de aprender as propriedades dos dados estatísticos agregados, o provedor de nuvem pode, com alta probabilidade de acerto, saber determinada informação privada do usuário a partir de dados criptografados armazenados. Protocolos PIR permitem que clientes recuperem informações de bancos de dados públicos ou privados sem revelarem para os servidores de banco de dados quais registros são recuperados. Ao proteger o conteúdo das consultas, PIR pode ser aplicado em importantes domínios de aplicações, tais como banco de dados de patentes, banco de dados farmacêuticos, censo online, serviços baseados em localização e análise de comportamento online para propaganda pela rede (OLUMOFIN; GOLDBERG, 2012). Um esquema PIR modela o banco de dados como uma *string* binária x = $x_1, x_2, x_3, \cdots, x_n$ de tamanho n. Cópias idênticas desta *string* são armazenadas em k servidores, sendo k $\geq$ 2. Os usuários possuem um índice i (um inteiro entre 1 e n) e estão interessados em obter o valor do bit x_i fazem consultas aleatórias aos servidores e obtém respostas com as quais podem computar o bit x_i. As consultas realizadas

aos servidores são distribuídas independentemente do valor de i para que os servidores não obtenham nenhuma informação sobre i. As consultas não recuperam necessariamente um bit em particular ou conjuntos de bits. Elas podem definir funções computadas pelos servidores, como por exemplo, uma consulta pode especificar um conjunto de índices entre 1 e n e a resposta do servidor pode ser o resultado de uma operação xor dos bits que possuem estes índices. O parâmetro de maior relevância nos esquemas PIR é a complexidade da comunicação entre o usuário e os servidores. Os protocolos mais eficientes para comunicação com 2 servidores têm complexidade de comunicação de $O(n^3)$ (CHOR *et al.*, 1998). Devido ao fato dos esquemas PIR utilizarem dados não criptografados, eles não são adequados para uso em ambientes não confiáveis de nuvem (YANG *et al.*, 2011).

4.5.3 Secure Multiparty Computation

Secure Multiparty Computation - SMC é uma técnica de processamento distribuído de dados, com garantia de privacidade. No SMC, um conjunto de partes interessadas deseja avaliar alguma função de interesse comum ao grupo e para tal processa dados individuais privados sem revelar estes dados uns aos outros. Apenas a saída da função é disponibilizada para todas as partes. O processamento de dados de forma colaborativa é muitas vezes necessário em ambiente de nuvem. No processamento distribuído, as partes podem ser adversários passivos que tentam obter informação “extra"sobre os dados de outras partes. Neste método, cada cliente C_i possui uma entrada privada x_i, e todos os clientes computam uma função pública $f(x_1,x_2,x_3,...,x_n)$ sem revelar x_i para os outros, exceto o que pode ser derivado da entrada ou saída da função (CHOUDHURY; PATRA, 2017).

4.6 Conclusão

Este capítulo apresentou as principais técnicas criptográficas utilizadas para garantir a confidencialidade de dados. No ambiente de nuvem, os dados podem estar em repouso, armazenados em bancos de dados no servidores da nuvem, ou em movimento, sendo transmitidos entre os usuários da nuvem e os provedores ou sendo processados nos computadores. A criptografia é a técnica mais antiga utilizada para comunicação sigilosa de informações, que está em constante evolução para garantir a comunicação nas redes de computadores.

A transmissão segura de dados sigilosos é um velho e importante problema. Seja

por uma questão de segurança militar ou institucional, seja pela transmissão de informações comerciais e bancárias, a comunicação segura continua sendo um problema estratégico para qualquer sociedade moderna. Hoje, toda a segurança de compras e transações bancárias feitas pela internet baseia-se na dificuldade de os computadores fatorarem números muito grandes. Até o mais veloz deles poderia levar anos e anos para decodificar essas informações confidenciais.

A criptografia está evoluindo para proteger grandes volumes de dados (*big data*) contra atacantes com recursos computacionais cada vez maiores. Assim que um computador quântico de grande capacidade de processamento entrar em funcionamento, a transmissão segura de dados sigilosos estará com seus dias contados, pois essas máquinas poderão fazer esses cálculos matemáticos em minutos ou até segundos. Para resolver este problema, evitando que a chave seja interceptada ou adulterada, a criptografia quântica já está sendo testada com sucesso. A técnica de distribuição de chaves quânticas (*Quantum Key Distribution-QKD*) resolve este problema ao criar uma chave de uso único feita de fótons emaranhados (SIBSON *et al.*, 2017; WANG *et al.*, 2017).

Pode-se perceber que a criptografia é bastante adequada para proteção de dados em movimento, pois não há a necessidade de que a chave seja guardada após a transmissão ser concluída, mas para segurança de dados armazenados, há o inconveniente de a chave ter que ser guardada pelo mesmo período de tempo em que os dados forem mantidos. Isto pode aumentar a probabilidade de perda ou esquecimento da chave pelo usuário ou perda ou roubo da chave armazenada nos computadores da nuvem com o passar do tempo, diminuindo a segurança que a criptografia pode proporcionar.

No próximo capítulo serão apresentadas as técnicas de fragmentação de dados, as quais constituem uma abordagem alternativa e complementar à criptografia. A fragmentação de dados é solução interessante para manter a confidencialidade de dados em repouso.

5 FRAGMENTAÇÃO DE DADOS

5.1 Introdução

Este capítulo apresenta as técnicas de fragmentação de dados que surgiram no âmbito da confidencialidade dos dados utilizados em sistemas distribuídos. A principal ideia por trás destas técnicas é a de que o quê deve ser mantido privado não é apenas a informação dos atributos, mas o relacionamento entre eles. Desta forma, a fragmentação de dados busca separar, em fragmentos diferentes, atributos que utilizados em conjunto, ou seja, relacionados, possam identificar univocamente um sujeito em conjunto de sujeitos. Além disto, este capítulo discute também as técnicas de **Dispersão de Informações** Essas técnicas permitem incrementar a disponibilidade de dados por meio da replicação e distribuição destes dados.

5.2 Dispersão de Informações

Dispersão de informação é uma estratégia comumente utilizada para melhorar a disponibilidade, confidencialidade e integridade de dados armazenados em nuvem. Esta estratégia decompõe um arquivo em **n** partes e armazenar estas partes em diferentes sites. A recuperação de **k** das **n** partes, sendo $\mathbf{k} \leq \mathbf{n}$ são suficientes para reconstruir o arquivo original. Uma premissa desta estratégia é que um conjunto menor do que **k-1** sites não irão cooperar para reconstruir o arquivo ou não serão comprometidos simultaneamente por um invasor, o quê garante a confidencialidade do arquivo. As principais técnicas de dispersão de informação apresentadas por (SLAMANIG; HANSER, 2012) são descritas a seguir:

- a) *Shamir's Secret Sharing*: o objetivo de um algoritmo de **partilha de segredos** é permitir que **k** processos combinem suas partes e revelem determinado segredo **s**, garantindo simultaneamente que um esforço combinado de **(k-1)** processos maliciosos não consiga obter qualquer informação relevante de **s**. Está técnica propõe dividir os dados em **n** partes, de modo que possam ser recuperados a partir de qualquer **k** partes, sendo $k \leq n$, mas que não revele nenhuma informação a partir de **(k - 1)** partes dos dados. A técnica proposta por Shamir (SHAMIR, 1979) é utilizada para construção de esquemas de gerenciamento de chaves para sistemas criptográficos.

  O algoritmo de Shamir faz parte de um conjunto de algoritmos denominados **criptografia de limiar**, que surgiu para prover tolerância a falhas e intrusões. Assume-se que o atacante

tem limitações computacionais, ou seja, não consegue quebrar a segurança das chaves criptográficas utilizadas.

O esquema da criptografia de limiar é baseado em duas propriedade dos polinômios (BLAKLEY, 1979):

1. dados quaisquer **d+1** pontos distintos da curva definida por um polinômio, é possível determinar qualquer outro ponto desse polinômio;
2. se os índices do polinômio forem todos desconhecidos, o conhecimento de até **d** pontos da curva não revela nenhuma informação sobre outros pontos

Assume-se que os dados podem ser representados por um número (**d**). O objetivo é dividir o número (**d**) em **n** partes $(d_1, d_2, d_3, \ldots, d_n)$, de tal forma que o conhecimento de qualquer **k** ou mais $\mathbf{d}_i$ partes, torne (**d**) computável.

Seja dado um polinômio **p(x)** de grau (**d**):

$$\boxed{p(x) = a_0 + a_1 x + \ldots + a_d x^k}$$

O algoritmo de Shamir considera a existência de um processo que pretende guardar o segredo (**d**). Esse processo define um polinômio de grau **q = k - 1** de índices $\mathbf{a}_i$ aleatórios, exceto $\mathbf{a}_0$ **= p(0) = d**. Para tal, utiliza-se um polinômio **q(x)** de grau $(k-1)$:

$$\boxed{q(x) = a_0 + a_1 x + \ldots + a_{k-1} x^{k-1}}$$

Depois, calcula-se **p(x)** para **n** valores diferentes e aleatórios de **x** e distribui-se cada uma dessas partes para um dos **n** processos. Atendendo às duas propriedades dos polinômios acima, o segredo pode ser reconstituído por **k** processos mas não por **(k - 1)**. A reconstituição de (**d**) é feita usando a interpolação de Lagrange da seguinte forma:

$$\boxed{p(0) = \sum_{i=1}^{k} (p(x_i) \prod_{j \neq i} \frac{0 - x_j}{x_i - x_j})}$$

A desvantagem desta técnica é o aumento do espaço de armazenamento por um fator de **n** e a falta de capacidade de detecção e correção de erros. Além disto, o algoritmo de Shamir tem duas limitações (CORREIA MIGUEL PUPO E SOUSA, 2010):

1. o processo não tem como saber se o seu fragmento está ou não corrompido. Vale destacar que se o fragmento estiver corrompido, sua combinação com outros $(k-1)$ fragmentos não irá reconstruir o segredo (arquivo). Esta limitação foi superada posteriormente com algoritmos de **partilha de segredos verificável**;

2. a segunda limitação é que se uma das demais **(**$k-1$**)** partes partes utilizadas para reconstruir o segredo estiver corrompida, o segredo não é reconstituído. Além disso, dado um determinado fragmento **f**, sendo **f** um dos **(**$k-1$**)** fragmentos recebidos, não é possível determinar se **f** é integro ou não.

- b) Rabin's Approach: utiliza o conceito de **algoritmos de dispersão de informação** (*information dispersal algorithms - IDAs*). Propõe o uso de códigos de apagamento (*erasure code*) não sistemáticos ao invés de compartilhamento de segredo polinomial (RABIN, 1989a). A ideia é dividir um arquivo **A** em **d** fragmentos de forma que seja suficiente apenas **k** fragmentos para reconstruí-lo, mas **(k-1)** fragmentos não sejam suficientes para o fazer. Pretende-se distribuir fragmentos de **A** para **d** processos ativos, de forma que a recuperação de **A** possa ocorrer na presença de **k** processos ativos, onde **d** e **k** são parâmetros que satisfazem $\mathbf{1} \leqslant \mathbf{k} \leqslant \mathbf{d}$. O esquema assume que os processos ativos funcionam corretamente, ou seja retornam fragmentos não modificados. Essa técnica adiciona algum grau de redundância às informações contidas no arquivo **A**, antes de fragmentar o arquivo em **d** partes. Cada parte possui tamanho $\mathbf{d} \div \mathbf{k}$, que representa o tamanho ótimo do fragmento. Esquemas de dispersão de informação podem ser implementados de diversas maneiras, que correspondem ao conceito de "códigos de apagamento"na teoria de códigos de correção de erros (RABIN, 1989b). Apesar de eficiente no uso de espaço e tempo de processamento, esta técnica não garante confidencialidade dos dados, por isso recomenda-se criptografar os dados antes de usar a técnica. Esta técnica reduz o aumento do espaço de armazenamento para um fator de $\mathbf{d} \div \mathbf{k}$, por este motivo foi aperfeiçoada para funcionar em sistemas distribuídos em trabalhos posteriores (KRAWCZYK, 1993; ALON *et al.*, 2000; GARAY *et al.*, 2000).
- c) *Secret Sharing Made Short*: esta técnica faz uso de um esquema de encriptação simétrico seguro:IND-CPA, que proporciona indistinguibilidade, pois o adversário não consegue aprender nenhuma informação do texto claro **x**, a partir de uma cifra **y**, além de ser resistente a *chosen plaintext attack - CPA*, situação em que o adversário pode selecionar cifras de textos claros a sua escolha. Nesta técnica, uma chave secreta simétrica randômica **sk** é utilizada para encriptar o arquivo **A**. Ao resultado da criptografia, aplica-se um algoritmo de dispersão de informação (por exemplo: *Rabin's Approach*) para criar um **vetor(V)** contendo **n** palavras-chaves como elementos, depois aplica-se uma técnica de compartilhamento de segredo polinomial para computar **n** compartilhamentos $s_1, \cdots, s_n$

da chave sk. Finalmente o conjunto $(vetor(v_i), s_i)$ é distribuído para **i** sites, onde **i** $\leqslant$ **n**.

- d) AONT-RS: trata-se de uma extensão do algoritmo de Rabin, adicionando confidencialidade. Esta técnica evita utilizar compartilhamento de segredo polinomial e, ao invés disto, utiliza uma variante do algoritmo AONT (*All-or-nothing transform*) de Rivest, como um passo de pré-processamento dos dados antes de usar um algoritmo de dispersão de informação. A técnica funciona da seguinte maneira: o arquivo **A** é visto como um vetor **(V)** com $(k+1)$ elementos, onde o elemento $\mathbf{v}_{(k+1)}$ é um valor de verificação de integridade (*checksum*). Uma chave randômica **sk** é escolhida de um esquema de criptografia simétrica para criptografar os elementos do vetor(V), criando o vetor(E) da seguinte forma: $e_i = v_i \; xor \; E(i+1, sk)$. Após criptografar todos os $k+1$ elementos do vetor(D), um novo elemento é inserido no vetor $v_{k+2} = sk \; xor \; H(e_1 \parallel \cdots \parallel e_{k+1})$, onde **H** é uma função criptográfica *hash*. A chave **sk** estará distribuída nos $(k+1)$ elementos do vetor(V). O passo seguinte é aplicação de um código de apagamento Reed-Solomon (RS) para produzir uma matriz de dispersão **M** onde as primeiras $(k+2)$ colunas compõem uma matriz identidade. Por fim, os fragmentos gerados a partir de **M** e **E** são distribuídos para os diferentes sites.

A Tabela 9 ilustra as principais técnicas de dispersão de informações apresentadas e os sistemas de armazenamento que as utilizam (RHEA *et al.*, 2001; GOODSON *et al.*, 2004), (STORER *et al.*, 2008), (STORER *et al.*, 2009), (SUBBIAH; BLOUGH, 2005).

Tabela 9 – Sistemas de Armazenamento que utilizam técnicas de dispersão

Sistema de Armazenamento de Arquivos	Técnica de Dispersão
Pergamum	*Rabin's Approach*
OceanStore	*Rabin's Approach*
POSTHARDS	Shamir's Secret Sharing
PASIS	Rabin's Approach
GridSharing	*Shamir's Secret Sharing*

5.3 Fragmentação de Banco de Dados

A técnica de fragmentação de banco de dados tem sido usada para aumentar o nível de paralelismo no acesso a dados, permitindo acesso simultâneo a diferentes fragmentos. Dessa forma, pode-se dividir uma consulta em partes que serão executadas simultaneamente sobre esses diferentes fragmentos. Outro fator que motiva a fragmentação de dados é o grande volume

de dados predominantemente armazenado em bancos de dados XML, que demandam cada vez mais maior desempenho no processamento de consultas em tais ambientes, motivando diversos estudos nesta área (FIGUEIREDO *et al.*, 2010; RODRIGUES *et al.*, 2011; SILVA *et al.*, 2012). Neste sentido, técnicas de distribuição de dados pelos diferentes nós de uma rede segundo técnicas de fragmentação e de alocação de dados e processamento paralelo de consultas sobre bases de dados têm sido adotadas com grande sucesso (KLING *et al.*, 2010; KLING *et al.*, 2011; BIDOIT *et al.*, 2013).

Para tal, existem duas técnicas básicas de fragmentação: Fragmentação Horizontal e a Fragmentação Vertical. Estas duas técnicas podem ser combinadas e utilizadas de forma conjunta na mesma tabela (Fragmentação Híbrida), ou em tabelas distintas (Fragmentação Mista). A escolha da técnica mais apropriada para fragmentar cada tabela que compõe a base de dados deve levar em conta a analise de diversas informações relativas à aplicação, tais como: as consultas mais frequentes e suas respectivas frequências; características das tabelas que formam a base de dados, seus relacionamentos com outras tabelas e suas cardinalidades (quantidade de registros existentes em uma tabela).

Independentemente da estratégia utilizada, é necessário garantir que a fragmentação mantenha a correção da base de dados, por meio de três regras, a saber: i) a completude; ii) a reconstrução e iii) a disjunção dos fragmentos (SANTOS, 2013).

1. A completude garante que se uma relação R é decomposta entre os fragmentos F_R= R_1, R_2, R_3, ..., R_n, então cada item encontrado em R, deve necessariamente ser encontrado em algum R_i.
2. A reconstrução uma relação R garante que se R é decomposta entre os fragmentos F_R= R_1, R_2, R_3, ..., R_n, deve existir um operador relacional que consiga reconstruir R a partir de suas partes.
3. A disjunção garante que se uma relação R é decomposta horizontalmente entre os fragmentos F_R= R_1, R_2, R_3, ..., R_n, então qualquer item d_i encontrado em R_j, não pode ser encontrado em nenhum outro fragmento R_{ki}, para k $\neq$ j.

5.3.1 Fragmentação Horizontal

Na técnica de **Fragmentação Horizontal - FH**, os fragmentos de uma tabela podem ser determinados através de funções de fragmentação definidas de acordo com as características da aplicação. Um fragmento horizontal de uma relação é um subconjunto das tuplas daquela

relação. As tuplas que pertencem a um determinado fragmento são especificadas por uma condição sobre um ou mais atributos da relação. Frequentemente, apenas um único atributo é envolvido. Por exemplo, uma tabela de EMPREGADOS poderia ser fragmentada em duas partições: aqueles que possuem salário acima de 5.000 e aqueles com salário menor ou igual a 5.000. Temos então a definição das duas partições de EMPREGADO, especificadas por duas funções de fragmentação distintas:

EMPREGADO1 = F[salário $<$ 5000] (EMPREGADO)

EMPREGADO2 = F[salário $\geqslant$ 5000] (EMPREGADO)

A Fragmentação Horizontal divide uma relação agrupando linhas para criar subconjuntos de tuplas, onde cada subconjunto possui um significado lógico. Então, esses fragmentos podem ser armazenados em diferentes localizações. A Fragmentação Horizontal pode ser realizada de forma primária (Fragmentação Horizontal Primária - FHP) ou derivada (FHD - Fragmentação Horizontal Derivada). Na FHD, a fragmentação de uma relação não é baseada nas propriedades dos seus próprios atributos mas em função da fragmentação de outra relação. Ela é usada para facilitar as operações de junção e navegação entre fragmentos. Já a fragmentação primária (FHP), utiliza três técnicas para a geração de fragmentos: faixa de valores, *hashing* e circular, as quais são descritas a seguir:

- Fragmentação Horizontal por Faixa de Valores: os fragmentos são obtidos em função dos valores dos atributos da relação tomando-se em consideração os predicados existentes nas aplicações dos usuários. Em geral, escolhe-se um ou mais atributos baseado nas cláusulas dos predicados, sendo a faixa de valores obtida tomando como base o domínio dos atributos escolhidos.
- Fragmentação Horizontal por Função de *Hashing*: A relação é fragmentada de acordo com o valor de uma função de *hashing* aplicada sobre um atributo, de forma a relacionar cada tupla da relação a um fragmento.
- Fragmentação Horizontal Circular: significa a associação sequencial de cada tupla da relação a um fragmento sem levar em consideração o valor de nenhum atributo. Desta forma, se a relação contiver **n** tuplas/objetos e o número de fragmentos a serem gerados for **m**, cada fragmento receberá **n/m** tuplas.

5.3.2 Fragmentação Vertical

Por questões de privacidade, pode não ser adequado alocar todos os atributos de uma relação em um único site, o que indicaria a necessidade de um tipo diferente de fragmentação. A Fragmentação Vertical é o processo que divide uma relação, agrupando em cada fragmento um subconjunto dos atributos existentes na relação original e replicando-se em cada fragmento a chave primária/identificação do objeto da relação/classe de forma a permitir a sua reconstrução a partir da junção dos fragmentos. O seu objetivo é identificar fragmentos de tal forma que uma determinada aplicação possa utilizar apenas um dos fragmentos, uma vez que todos os atributos necessários para esta aplicação estariam no mesmo fragmento. Entretanto, caso uma aplicação necessite de acesso à dados armazenados em mais de um fragmento, a fragmentação vertical não se mostra uma boa solução, uma vez que será necessário a realização de junções entre os fragmentos para a obtenção do resultado desejado.

Dado um conjunto de restrições de confidencialidade, a técnica de Fragmentação Vertical divide uma determinada relação do banco de dados em fragmentos de modo que os atributos de uma determinada restrição não apareçam juntos em nenhum fragmento.

Um fragmento vertical de uma relação mantém apenas um subconjunto dos atributos da relação. Por exemplo, pode-se fragmentar a relação empregado em dois fragmentos verticais. O primeiro fragmento inclui as informações pessoais (nome, data de nascimento, endereço e sexo) e o segundo inclui as informações relacionadas ao trabalho (código do funcionário, valor do salário, cargo, função). Essa fragmentação vertical não é exatamente adequada porque, se os dois fragmentos forem armazenados separadamente, não poderemos juntar de volta as tuplas originais de empregado, uma vez que não há atributo comum entre os dois fragmentos. É necessário incluir a chave primária ou algum atributo que seja chave candidata em cada fragmento vertical de forma que a relação completa possa ser reconstruída a partir dos fragmentos. Consequentemente, uma solução possível seria adicionar o atributo “código do funcionário” ao fragmento das informações pessoais.

5.3.3 Fragmentação Híbrida

Também chamada de fragmentação mista ou aninhada, a Fragmentação Híbrida representa a aplicação das duas técnicas descritas anteriormente, uma após a outra. Ela é utilizada porque, na maioria dos casos, uma Fragmentação Vertical ou Horizontal não será

suficiente para atender aos requisitos de segurança e privacidade do usuário.

A seguir será apresentado um exemplo do uso de Fragmentação Horizontal e Vertical para armazenamento de dados sobre funcionários e terceirizados de uma empresa em dois provedores de nuvem, garantindo a privacidade dos proprietários dos dados, desde que os provedores não trabalhem em conjunto para obter informações sobre os dados armazenados. Neste caso existe uma relação de EMPREGADOS que armazena informações pessoais e funcionais de funcionários e terceirizados da empresa. Deseja-se armazenar estas informações com privacidade em uma infraestrutura de nuvem pública, de modo que informações sobre o salários dos empregados não seja divulgada para os provedores de nuvem. Os atributos considerados de associação sensível são aqueles que podem revelar informações sobre o proprietário dos dados. Neste caso, são NOME-EMPREGADO e SALARIO-EMPREGADO. Estes atributos deverão ser armazenados em fragmentos diferentes, os quais devem ser localizados em infraestruturas de nuvem que sejam fisicamente e administrativamente distintas. As relações EMPREGADOS-TB1, EMPREGADOS-TB2 e EMPREGADOS-TB3 representam fragmentos da relação EMPREGADOS particionados horizontalmente e verticalmente, que utilizam a chave primária ID-EMPREGADO como elemento comum entre os fragmentos, a fim de permitir a posterior recomposição da relação EMPREGADOS.

A seguir será apresentado um exemplo de fragmentação horizontal e vertical. Será utilizado no exemplo a relação EMPREGADOS. A fragmentação vertical irá separar os atributos que identificam o empregado (nome e cpf) dos atributos relacionados ao trabalho (categoria e salário). A fragmentação horizontal irá particionar os atributos de categoria e salário em duas relações (EMP-TB2 e EMP-TB3) utilizando uma regra de faixa de salário e tipo de categoria.

Passo 1: Criação de link de acesso remoto.

Cada fragmento será uma relação em um determinado banco de dados. Para que dois bancos se comuniquem, é necessário a definição de um *link*. Com o *link* definido é possível que um banco de dados acesse objetos em outro banco de dados. A sintaxe para a criação de um *link* na linguagem *Structured Query Language* - SQL do Banco de dados Oracle é apresentada a seguir.

```
CREATE DATABASE LINK REMOTE_DB_MAQUINA2
CONNECT TO SYSTEM
IDENTIFIED BY ‘‘keymaquina2’’
USING REMOTE
```

```
CREATE DATABASE LINK REMOTE_DB_MAQUINA3
CONNECT TO SYSTEM
IDENTIFIED BY ‘‘keymaquina3’’
USING REMOTE
```

Na criação de um *link*, é obrigatória a especificação de um usuário e senha, tais parâmetros deverão receber o usuário e senha do banco de dados da máquina que se deseja acessar.

Passo 2: Fragmentação da Tabela EMPREGADOS.

A tabela EMPREGADOS será fragmentada em 3 tabelas: EMP-TB1, EMP-TB2 e EMP-TB3. A seguir é apresentado a estrutura destas tabelas.

```
CREATE TABLE EMPREGADOS(
ID-EMPREGADO INT PRIMARY KEY,
NOME-EMPREGADO VARCHAR(80) NOT NULL,
CPF-EMPREGADO VARCHAR(14) NOT NULL,
CATEGORIA-EMPREGADO VARCHAR(20) NOT NULL,
SALARIO-EMPREGADO NUMERIC(8,2) NOT NULL
);

CREATE TABLE EMP-TB1(
ID-EMPREGADO INT PRIMARY KEY,
NOME-EMPREGADO VARCHAR(80) NOT NULL,
CPF-EMPREGADO VARCHAR(14) NOT NULL,
);

CREATE TABLE EMP-TB2(
ID-EMPREGADO INT PRIMARY KEY,
CATEGORIA-EMPREGADO VARCHAR(20) NOT NULL,
SALARIO-EMPREGADO NUMERIC(8,2) NOT NULL
);
```

```
CREATE TABLE EMP-TB3(
ID-EMPREGADO INT PRIMARY KEY,
CATEGORIA-EMPREGADO VARCHAR(20) NOT NULL,
SALARIO-EMPREGADO NUMERIC(8,2) NOT NULL
);
```

Passo 3: Criação de Predicado de Fragmentação Horizontal.

Um predicado de fragmentação é criado para definir uma condição para o particionamento da relação original. O atributo categoria-empregado da tabela EMPREGADO será usado para definição do predicado **P**, definindo a condição para que os dados profissionais (Categoria-Empregado e Salário-Empregado) dos empregados sejam enviados para o banco de dados da máquina 2 ou 3:

```
P: (categoria-empregado = ‘‘terceirizado") AND
   (salario-empregado < 3000)
```

Conforme o predicado, no banco de dados da relação EMP-TB2 apenas serão inseridos funcionários terceirizados e que recebem salário inferior a R$ 3.000,00. Os demais registros de funcionário ou terceirizados que recebem mais que R$ 3.000,00 serão armazenados na relação EMP-TB3.

A técnica de fragmentação mostra-se eficiente por permitir que apenas parte de uma relação seja replicada, reduzindo assim o tempo e espaço com a administração de relações completas (PRESSMAN; MAXIM, 2016). Permite também atender mais rapidamente às solicitações de usuários, isso pode ser feito com o particionamento definido sobre a localidade do usuário, onde os dados estarão disponíveis em locais próximos a ele.

5.4 Conclusão

Este capítulo apresentou as principais técnicas de fragmentação de dados que são utilizadas para prover confidencialidade a dados distribuídos. Em geral geral, as técnicas de fragmentação de dados são utilizadas em conjunto com estratégias de dispersão da informação. As estratégias de dispersão possuem a finalidade principal de aumentar a disponibilidade dos dados armazenados em sistemas distribuídos. Recentemente, as técnicas de fragmentação de

dados têm sido utilizadas em conjunto com as técnicas de criptografia. Esse uso combinado permite explorar diferentes combinações entre os graus de confidencialidade e utilidade dos dados.

No próximo capítulo, será apresentada uma estratégia **QSM-EXTRACTION**, que pode ser utilizada para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem. A estratégia **QSM-EXTRACTION** se baseia na extração de características dos dados a serem armazenados na nuvem. Essa estratégia utiliza ainda técnicas de fragmentação, decomposição e dispersão de dados, com a finalidade de aumentar o grau de confidencialidade e disponibilidade dos dados.

6 CONFIDENCIALIDADE DE DADOS BASEADA NA EXTRAÇÃO DE CARACTERÍSTICAS

6.1 Introdução

Este capítulo apresenta uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem. A ciência por trás dessa abordagem utiliza conceitos da Doutrina do Ser de Hegel. A estratégia QSM-EXTRACTION baseia-se na fragmentação de um arquivo digital em fragmentos denominados objetos de informação, na decomposição desses objetos por meio da extração de suas características (Qualidade, Quantidade e Medida) e na dispersão dessas características em diferentes serviços de armazenamento em nuvem, permitindo a posterior recuperação desses dados sem perda de informação. A finalidade da estratégia proposta é inviabilizar a reconstrução do arquivo original por parte de um provedor de nuvem que possui apenas parte das características dos objetos de informações que compõem este arquivo. Desta forma, assegura-se a confidencialidade dos dados armazenados em nuvem e, por conseguinte, a privacidade dos proprietários desses dados.

A estratégia QSM-EXTRACTION aplica-se a arquivos digitais que possuem as seguintes características:

- os arquivos armazenam as informações por meio de símbolos que são representados na forma de *bytes*, ou seja, os símbolos contidos no arquivo são representados por arranjos de 8 *bits* que podem formar 256 números na base 2, de 0_{10} $(00000000)_2$ a 255_{10} $(11111111)_2$;
- o tamanho de cada arquivo deve ser maior ou igual a 256 *bytes*, sendo que não há restrição de limite de tamanho máximo;
- os arquivos estão compactados. A compactação, ou compressão sem perdas, é amplamente utilizada, com o objetivo de reduzir os custos de armazenamento. Alguns exemplos de arquivos compactados são documentos (.doc, .pdf, .ps), arquivos de imagem (.jpg, .bmp, .eps), arquivos de som (.wav, .mp3, .wma), arquivos de vídeo (.wmv, .avi., .flv), dentre outros formatos (.zip, .rar. .lhz).

6.2 Teoria Hegeliana do Ser

A ideia principal da estratégia QSM-EXTRACTION foi inspirada nas ideias do filósofo alemão Georg Wilhelm Friedrich Hegel, publicadas no livro **Enciclopédia das Ciências**

Filosóficas em 1817. A Doutrina do Ser trata da Lógica do Ser aborda três conceitos principais: determinidade (qualidade), a grandeza (quantidade) e a medida. O estudo da lógica de Hegel sobre a formação dos objetos foi fonte de inspiração para o desenvolvimento de uma técnica de extração das características de qualidade, quantidade e medida de um arquivo digital, visto como um objeto de informação.

Segundo Hegel, o SER possui três propriedades fundamentais que o determinam: a **Qualidade**, a **Quantidade** e a **Medida** (HEGEL, 1995). Em seu livro, Hegel divide a Lógica em 3 partes: A Doutrina do Ser, A Doutrina da Essência e a Doutrina do Conceito e da Ideia. Na parte da Doutrina do Ser, ele apresenta as propriedades do "Ser determinado", que representam os objetos que compõem a realidade. Essas propriedades são apresentadas da seguinte forma:

1. **Qualidade** é "o Ser como uma determinidade, enquanto imediata ou existente", ou seja o Ser determinado por suas características, posto como algo que existe, algo que está-aí. A qualidade transforma o Ser abstrato em Ser real, determinado. Segundo Hegel, "Algo mediante a sua qualidade é, em primeiro lugar, finito, e, em segundo lugar, mutável, de maneira que a finidade e a mutabilidade pertencem ao seu ser (HEGEL, 1995)".
2. **Quantidade** representa a extensão do ser, não levando em conta as determinações do ser (qualidade). O conceito da quantidade considera exclusivamente os números que quantificam as características (qualidades) do ser, sem determinar quais são estas características. Segundo Hegel, "A quantidade posta essencialmente com a determinidade exclusiva, que nela está contida, é o *quantum*; quantidade limitada (HEGEL, 1995)".
3. Segundo Hegel, "a **Medida** é o *quantum* qualitativo,..., um *quantum* a que está vinculado um ser determinado ou uma qualidade (HEGEL, 1995)". A qualidade e a quantidade estão representadas na medida por números que indicam proporcionalidade, posicionamento e extensão do ser determinado.

A Doutrina do Ser de Hegel define alguns conceitos:

- **Ser determinado**: é representado por substantivos concretos ou abstratos. Por exemplo: um carro, uma casa, um unicórnio, um pensamento, entre outros. O Ser determinado é um **"objeto"** real ou imaginário. Tradicionalmente, substantivos abstratos são considerados como opostos de substantivos concretos. Contudo, atualmente, a ideia de oposição está sendo substituída por uma visão que defende a existência de diferentes graus de concretude e abstração (MIRANDA; FARIAS, 2011).
- **Qualidade**: é representada por adjetivos, propriedades ou características do objeto. Por

exemplo ao se considerar um carro, suas qualidades podem ser cor, marca, modelo, motor, rodas, buzina, consumo de combustível.

- **Quantidade**: é representada por numerais que indicam a extensão da presença dos atributos da qualidade no objeto. Por exemplo, um carro possui 4 rodas, 2 retrovisores, 1 motor, 1 buzina, 8 km/l de consumo.
- **Medida**: é representada por numerais que mostram a relação entre os atributos da qualidade e da quantidade. Por exemplo, as medidas de um carro podem ser a posição das rodas em relação à carroceria, a distância do bancos traseiros em relação aos bancos dianteiros, a proporção do tamanho do aro em relação ao tamanho dos pneus.

6.3 Conceituação de Objeto de Informação

As informações contidas nos arquivos digitais são representadas sob a forma de *byte*s. Existem várias codificações propostas para os arquivos digitais, uma das mais conhecidas e utilizadas é o Código Padrão Americano para o Intercâmbio de Informação (*American Standard Code for Information Interchange - ASCII*), apresentado no Apêndice E, que codifica um conjunto de 128 sinais: 95 sinais gráficos (letras do alfabeto latino, sinais de pontuação e sinais matemáticos) e 33 sinais de controle, utilizando portanto apenas 7 bits para representar todos os seus símbolos. Outro padrão internacional para codificação de informações em arquivos digitais é o padrão **Unicode**, que foi criado em 1989 para permitir aos computadores representar e manipular, de forma consistente, texto de qualquer sistema de escrita existente. O **Unicode** define dois métodos de mapeamento de códigos **Unicode** em códigos de implementação (CONSORTIUM, 2000):

- UTF-8 (*Unicode Transformation Format*), que utiliza entre um e quatro bytes por código.
- UTF-16 que representa o conjunto UCS (*Universal Character Set*), que utiliza uma codificação 16-bit de largura variável. Ela pode incluir uma ou duas palavras 16-bit para suportar outros caracteres.

Neste trabalho, propõe-se analisar as propriedades de qualidade, quantidade e medida dos arquivos digitais, considerando a representação de 1 símbolo por *byte*. Neste caso, o arquivo digital será tratado como um conjunto de **n** partes, onde cada parte possui 256 *bytes* de tamanho. Caso a parte final do arquivo seja menor do que 256 *bytes*, esta será complementada com *bytes* quaisquer, para completar a sequência de 256 bytes. Cada parte de 256 *bytes* sequenciais de um determinado arquivo será denominada de objeto de informação, que chamaremos de **iobjeto**. Um objeto de informação pode armazenar entre 1 e 256 símbolos diferentes. De um modo geral,

as três características de um objeto de informação são as seguintes:

- **Qualidade** representa os diferentes símbolos (*bytes*) que compõem o arquivo digital;
- **Quantidade** representa a quantidade de vezes em que cada *byte* está presente no objeto de informação;
- **Medida** representa a informação sobre a ordem em que os *bytes* estão dispostos sequencialmente no objeto de informação.

Definição 1: (objeto de informação). Um objeto de informação é uma fragmento de um arquivo digital contendo 256 *bytes* sequenciais. Mais formalmente, um objeto de informação é definido da seguinte forma: Seja um arquivo F = $< b_1, b_2, \cdots, b_n >$, em que b_i é um *byte*, 1 $\leqslant i \leqslant$ n e n $\geqslant$ 256. Um objeto de informação iOBJ = $< b_j, b_{j+1}, \cdots, b_{j+255} >$ onde j $\geqslant$ 1, j+255 $\leqslant$ n e iOBJ $\subset$ F.

Para exemplificar a definição de objeto de informação, considere um arquivo **F** contendo 768 *bytes*. **F** = $< b_1, b_2, \cdots, b_{768} >$, em que b_i é um *byte*. Neste caso, o arquivo **F** será fragmentado em 3 objetos de informação ($iOBJ_1$, $iOBJ_2$ e $iOBJ_3$), onde $iOBJ_1$ possui os primeiros 256 *bytes* sequenciais (de b_1 até b_{256}), $iOBJ_2$ possui os próximos 256 *bytes* sequenciais (de b_{257} até b_{512}) e $iOBJ_3$ possui os últimos 256 *bytes* sequenciais (de b_{513} até b_{612}).

Definição 2: (Qualidade). Qualidade é o conjunto de *bytes* diversos que compõem um determinado objeto de informação. Seja iOBJ um objeto de informação, **Q(iOBJ)** denota a propriedade da qualidade do iOBJ. **Q(iOBJ)** é um vetor ordenado que contem **m** *bytes* diversos presentes em iOBJ. Mais formalmente, **Q(iOBJ)** = $\{b_1, b_2, b_3, \cdots, b_m\}$ tal que $1 \leqslant b_i \leqslant 256$ e $i \neq j \rightarrow b_i \neq b_j$, onde b_i é um *byte* existente em iOBJ.

Definição 3: (Quantidade). Quantidade contém informações sobre o número de vezes que cada *byte* diverso aparece em um determinado objeto de informação. Seja iOBJ um objeto de informação, **S(iOBJ)** denota a propriedade da quantidade (extensão) do iOBJ. **S(iOBJ)** é um vetor de 256 posições que contém, para cada *byte* diverso b_j existente em **Q(iOBJ)**, o número de vezes que b_i aparece em iOBJ. Mais formalmente, **S(iOBJ)** = $\{s_1, s_2, s_3, \cdots, s_m\}$ tal que $1 \leqslant s_i \leqslant 256$, onde s_i representa o número de vezes que b_i aparece em iOBJ.

Definição 4: (Medida). Seja iOBJ um objeto de informação, M(iOBJ) denota a propriedade da medida de iOBJ. **M(iOBJ[256][256])** contém, para cada *byte* diverso b_j presente em **Q(iOBJ)**, um vetor m_{b_j} que armazena as posições nas quais o *byte* b_j aparece em iOBJ. Mais formalmente, M(iOBJ) = $\{m_{b_1}, m_{b_2}, \cdots, m_{b_m}\}$, tal que, m = 256 e $1 \leqslant size(m_{b_i}) \leqslant 256$.

A Medida armazena as posições ocupadas pelos *bytes* no iOBJ. Essas informações

estão em uma matriz esparsa de 256 linhas por 256 colunas. As linhas representam a qualidade dos *bytes* existentes no objeto de informação e as colunas representam a quantidade de posições ocupadas pelos *bytes* no objeto de informação. Caso exista apenas um byte que se repete 256 dentro do objeto de informação, os 256 elementos da linha da matriz da Medida correspondente a este byte ficarão totalmente preenchidos, e as demais linhas da matriz ficarão vazias. Na situação oposta, caso todos os 256 bytes do objeto de informação sejam diferentes uns dos outros, a primeira coluna da matriz da Medida ficará completamente preenchida, ficando vazias, as demais colunas da matriz.

6.4 QSM-EXTRACTION: Estratégia de Armazenamento Confiável de Dados na Nuvem

As técnicas de confidencialidade para armazenamento de arquivos na nuvem apresentadas nos capítulos anteriores mostram que, de modo geral, a anonimização de dados realiza operações de generalização, supressão, encriptação e perturbação para esconder o significado do conteúdo dos arquivos. As técnicas criptográficas utilizam operações de substituição e transposição com o mesmo objetivo e a fragmentação usa técnicas de particionamento dos arquivos para dispersar as informações em partes distintas da nuvem, obedecendo a um conjunto de restrições de confidencialidade. Essas abordagens atuam sobre os dados contidos nos arquivos, modificando-os ou particionando-os para ocultar seu significado de pessoas ou sistemas computacionais não autorizados a acessá-los.

O trabalho aqui proposto apresenta uma nova forma de ocultar o significado dos dados armazenados na nuvem, por meio da extração de suas características (Qualidade, Quantidade e Medida) e do armazenamento destas informações em servidores distintos na nuvem. Assume-se que tais características são armazenadas em três provedores de armazenamento de dados em nuvem diferentes, onde cada provedor armazena uma determinada característica. Os usuários são identificados e autorizados para acessar os dados armazenados na nuvem. Assume-se também que os usuários utilizam canais autenticados com os provedores. Já os provedores devem assegurar a disponibilidade e a integridade dos dados e são considerados honestos, porém curiosos, isto é, executam os protocolos de modo apropriado, mas têm interesse em obter informações extra sobre os dados e podem analisar o fluxo de mensagens trocadas com o cliente, além de inferir informações sobre os dados. Considera-se que existe uma infraestrutura segura de comunicação que permite que os atores realizem mutua autenticação e utilizem canais de comunicação seguros.

A diferenciação entre a proposta desta tese e as abordagens previamente existentes é

que esta atua sobre as propriedades do arquivo digital, considerando-o como uma coleção de objetos de informação que possuem as características de Qualidade, Quantidade e Medida.

As técnicas atuais de proteção da confidencialidade dos dados e privacidade dos proprietários dos dados (generalização, supressão, encriptação e perturbação), propostas pela comunidade acadêmica, podem ser usadas e combinadas para aumentar a confidencialidade dos dados armazenados na nuvem. Todas estas técnicas operam usando uma abordagem baseada no dado, representado por um conjunto de 8 ou 16 bits, que são usados para representar símbolos (letras, números ou caracteres especiais). A estratégia QSM-EXTRACTION difere das técnicas existentes por não atuar sobre o dado em si, e sim sobre suas três características fundamentais: Qualidade, Quantidade e Medida.

Um visão geral da estratégia QSM-EXTRACTION é apresentada na Figura 19. Neste cenário temos 5 atores: um cliente, que deseja armazenar seus arquivos pessoais na nuvem pública, três provedores de serviços de armazenamento de dados em nuvens públicas e uma Terceira Parte Confiável (TPC) que é responsável pelo processamento dos algoritmos que compõem a estratégia QSM-EXTRACTION, os quais irão assegurar a confidencialidade dos dados, além do controle das comunicações entre o cliente e os provedores de armazenamento em nuvem.

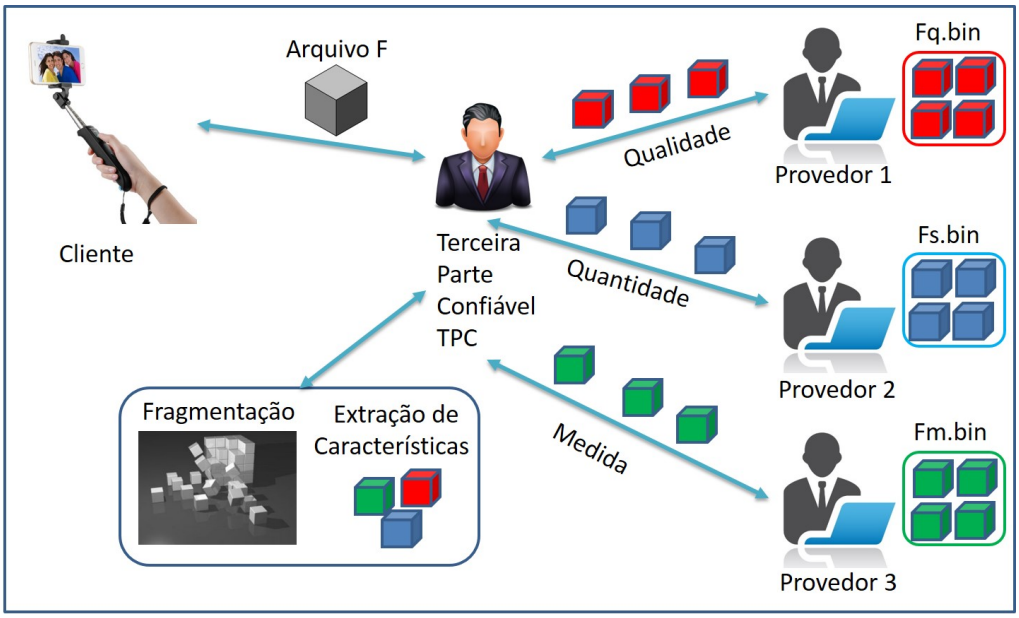

Figura 19 – Visão Geral do Armazenamento de Dados nas Nuvens

As fases e etapas do processo de criação e armazenamento dos objetos de informação na infraestrutura de armazenamento da nuvem são as seguintes:

1. Fase de Fragmentação

- Fragmentação do arquivo em objetos de informação;

2. Fase de Decomposição
   - Extração dos atributos de Qualidade, Quantidade e Medida dos objetos de informação;
   - Anonimização da medida;
   - Anonimização da qualidade;
   - Anonimização da quantidade
   - Compactação da medida;
3. Fase de Dispersão
   - Envio dos três arquivos contendo as características dos objetos de informação que compõem o arquivo **F** para três servidores distintos na nuvem.
   - Armazenamento das informações sobre a qualidade dos objetos de informação no arquivo **Fq.bin**, por exemplo na nuvem 1;
   - Armazenamento das informações sobre a quantidade dos objetos de informação no arquivo **Fs.bin**, por exemplo na nuvem 2;
   - Armazenamento das informações sobre a medida dos objetos de informação no arquivo **Fm.bin**, por exemplo na nuvem 3.

O detalhamento dessas fases é descrito a seguir, juntamente com os algoritmos utilizados e um exemplo de execução.

6.4.1 Fase 1: Fragmentação

A fase de fragmentação representa a quebra do arquivo de entrada **F** em uma sequência de **n** objetos de informação. Então, é possível representar um arquivo **F** como um conjunto ordenado $\{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$ de objetos de informação. A Figura 20 apresenta uma visão geral da fase de fragmentação.

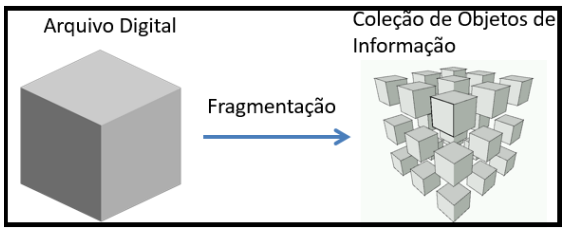

Figura 20 – Fase de Fragmentação

O Algoritmo 1, apresentado a seguir em pseudo-código, é uma representação em

alto nível da etapa de fragmentação. Para fins didáticos, assume-se que o tamanho do arquivo F é múltiplo de 256 *bytes*. Vale destacar que pode-se utilizar qualquer múltiplo de 256 como tamanho de bloco de leitura e gravação. Nos dispositivos mais comuns de armazenamento de arquivos digitais, tais como, *hard disk*, *drivers* de DVD-ROM, *flash drivers* e dispositivos de armazenamento em geral, o modo de leitura e gravação de dados é feito por meio de blocos de dados. O tamanho dos blocos normalmente varia de 512 *bytes* a 32 *Kbytes*. Dispositivo de blocos é aquele que armazena informação em bloco de tamanho fixo, cada um com seu próprio endereço. A propriedade essencial de um dispositivo de blocos é que cada bloco pode ser lido/escrito independente dos demais (TANENBAUM; WOODHULL, 2009), i.e, o bloco é uma unidade de entrada/saída. O Apêndice A apresenta o algoritmo de fragmentação de forma mais detalhada.

Implementação de (Coleção de iOBJs) ← Frag(F) **(Algoritmo 1)**. O Algoritmo 1 recebe como entrada um arquivo F. Percorre esse arquivo sequencialmente extraindo blocos de 256 bytes, que constituem os objetos de informação. A saída do algoritmo é uma coleção de objetos de informação.

Algoritmo 1: (iOBJ) ← Frag(F)

```
   Entrada  : Arquivo F
   Saída    : Coleção de objetos de informação referentes ao arquivo F
 1 início
 2     abrir_arquivo(F);
 3     i ← 1;
 4     Enquanto não for fim do arquivo F faça
 5         iOBJ_i ← extrair_256_bytes(F);
 6         i ← i + 1;
 7     fim-enquanto
 8     fechar_arquivo(F);
 9     Retorna iOBJ
10 fim
```

6.4.2 Fase 2: Decomposição

A Figura 21 exibe uma visão geral do processamento do arquivo original **F** com a finalidade de gerar os três arquivos contendo as características de Qualidade, Quantidade e Medida (**Fq.bin**, **Fs.bin** e **Fm.bin**, respectivamente). Nesta figura, pode-se observar que a fase de decomposição recebe como entrada um conjunto de objetos de informação, os quais compõem um determinado arquivo **F**. Em seguida, extrai-se as características de Qualidade, Quantidade e Medida de cada um desses objetos de informação. Por fim, as características dos objetos de informação são concatenadas para gerar os arquivos **Fq.bin**, **Fs.bin** e **Fm.bin**, os

quais representam, respectivamente, as características de Qualidade, Quantidade e Medida do arquivo **F**.

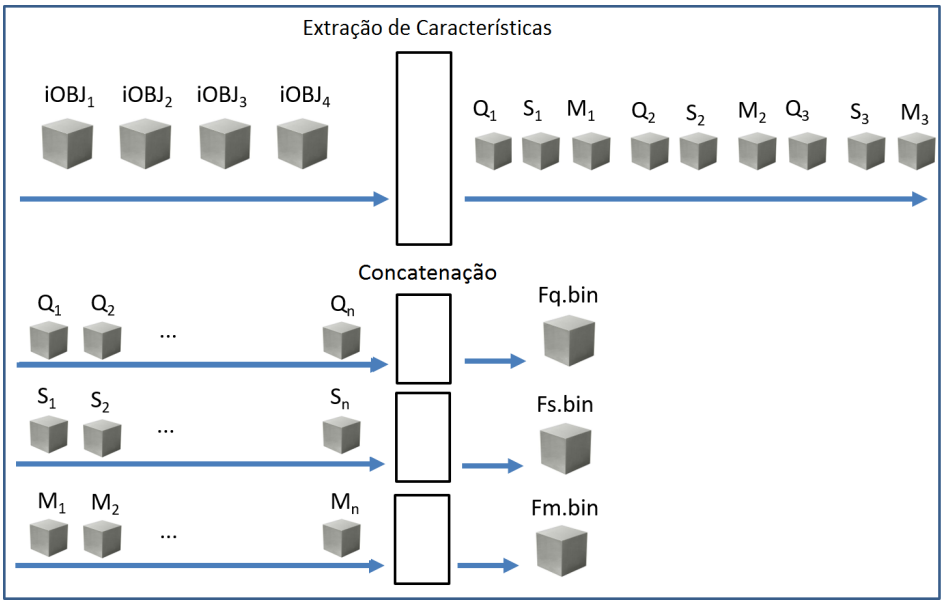

Figura 21 – Visão Geral da Fase de Decomposição

O processo de decomposição de um objeto de informação é ilustrado na Figura 22. Inicialmente, quando um objeto de informação é recebido, extrai-se suas características de Qualidade, Quantidade e Medida. Em seguida, realiza-se a anonimização da Medida. Para isso, é utilizada a função $\mathbf{Fp}_1$, que realiza o embaralhamento das posições dos elementos da Medida utilizando informações da Qualidade e da Quantidade, como parâmetros de permutação para gerar $\mathbf{M}_E$. Posteriormente, os elementos da Medida, após o embaralhamento, são utilizados como parâmetros de entrada das funções que irão anonimizar a Quantidade e a Qualidade (funções $\mathbf{Fp}_2$ **e** $\mathbf{Fp}_3$, respectivamente). Por fim, os elementos de $\mathbf{M}_E$ são anonimizados, utilizando-se a técnica de generalização (função **Fa**), proporcionando redução de espaço de armazenamento em disco e possibilitando maior confidencialidade aos dados de $\mathbf{M}_E$.

A fase de decomposição tem como entrada um conjunto ordenado $\{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$ de objetos de informação e gera como saída três arquivos: **Fq.bin**, **Fs.bin** e **Fm.bin**, que representam, respectivamente, a Qualidade, a Quantidade e a medida dos objetos de informação do arquivo **F**. É importante destacar que as informações de cada um destes três arquivos não devem revelar nenhuma informação sobre o conteúdo dos outros dois arquivos. Mais especificamente, as informações sobre a característica **Qualidade**, as quais serão armazenadas no arquivo **Fq.bin**, revelam quais *bytes* estão presentes em **F**, mas não revelam a ordem em que estes *bytes* estão dispostos, nem a frequência em que ocorrem. As informações sobre a

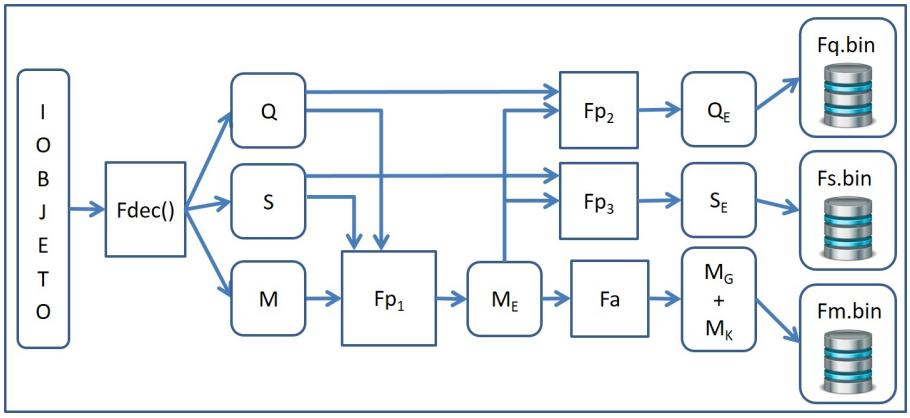

Figura 22 – Decomposição de um Objeto de Informação

característica **Quantidade**, as quais serão armazenadas no arquivo **Fs.bin**, revelam se um *byte* que está presente em **F** ocorre uma ou várias vezes, mas não revelam qual é este *byte* nem quais posições ocupa. Finalmente, as informações sobre a característica **Medida**, as quais serão armazenadas no arquivo **Fm.bin**, revelam as posições ocupadas pelos *bytes* em **F**, mas não revelam quais são os *byte*s, nem sua frequência de ocorrência. Desta forma, a confidencialidade das informações pode ser assegurada pela extração das características dos objetos de informação que compõem o arquivo F. Os algoritmos relativos ao processo de decomposição dos objetos de informação serão apresentados a seguir.

Implementação de (Q, S, M) ←Fdec(iobjeto) **(Algoritmo 2)**. O algoritmo de decomposição, descrito a seguir, é a rotina principal que coordena todo o processo de extração das características de um determinado objeto de informação. O Algoritmo 2 recebe como entrada um objeto de informação. Em seguida, chama as rotinas que realizam a extração das características de Qualidade, Quantidade e Medida. Por fim, concatena os resultados recebidos nos arquivos **Fq.bin**, **Fs.bin** e **Fm.bin**, respectivamente.

Algoritmo 2: (Q, S, M) ← Fdec(iOBJ)

Entrada : Objeto de informação iOBJ
Saída : Qualidade(Q), Quantidade(S) e Medida(M)

```
1 início
2     Q ← Extrair_Q(iobjeto);
3     S ← Extrair_S(iobjeto);
4     M ← Extrair_M(iobjeto);
5     Insere_no_Final(Fq.bin, Q);
6     Insere_no_Final(Fs.bin, S);
7     Insere_no_Final(Fm.bin, M);
8 fim
```

6.4.2.1 Etapa 1: Extração da Qualidade, Quantidade e Medida

Dado um arquivo F, onde $F = \{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$. Inicialmente, a fase de decomposição realiza a extração, para cada objeto de informação iOBJ, onde $1 \leqslant k \leqslant n$ e $iOBJ_k \in F$, das propriedades $Q(iOBJ_k)$, $S(iOBJ_k)$ e $M(iOBJ_k)$.

Implementação de (Qualidade) ← Extrair_Q(iOBJ) **(Algoritmo 3)**. O Algoritmo 3 recebe como entrada um objeto de informação iOBJ. Em seguida, processa sequencialmente os bytes de iOBJ, extraindo a informação do valor decimal de cada byte e armazenando essa informação no vetor de bytes **Q** de 256 posições. Para otimização do espaço de armazenamento, as informações da Qualidade são representadas por um vetor de bits Q_{BA} de 256 posições, onde o bit 1 na posição $\mathbf{Q_{BA}[pos]}$ do vetor representa a existência do byte de valor decimal ***pos*** no objeto de informação iOBJ. Observe que os vetores Q e Q_{BA} representam, de maneiras distintas, as mesmas informações.

Algoritmo 3: (Q) ← Extrair_Q(*iOBJ*)

```
   Entrada : Um objeto de informação iOBJ
   Saída   : Q(iOBJ), Q_BA(iOBJ)
 1 início
 2     cria_vetor_de_bytes(Q);
 3     cria_vetor_de_bits(Q_BA);
 4     preenche_vetor(Q_BA, 0);
 5     preenche_vetor(Q, 0);
 6     n ← 0;
 7     para byte = 0 to 255 faça
 8         pos ← decimal(iOBJ[byte]);
 9         Q_BA[pos] ← 1;
10     fim para
11     para bit = 0 to 255 faça
12         se Q_BA[bit] = 1 então
13             Q[n] ← pos;
14             n ← n + 1;
15         fim se
16     fim para
17     Retorna Q(iOBJ), Q_BA(iOBJ)
18 fim
```

Implementação de (Quantidade) ← Extrair_S(iOBJ) **(Algoritmo 4)**. Os bytes que compõem um determinado objeto de informação podem ou não se repetir. Devido a este fato, é necessário determinar se os bytes existem isoladamente ou em grupo dentro de um determinado objeto de informação. O vetor **S**, de 256 posições, armazena o número de vezes em que os bytes ocorrem em um determinado objeto de informação. Para otimização do espaço de

armazenamento, as informações da quantidade são representadas por um vetor de bits S_{BA} de 256 posições, onde o bit 0, na posição $\mathbf{Q_{BA}[pos]}$ do vetor, indica que o byte de valor decimal ***pos*** ocorre apenas uma vez no objeto de informação. Já o bit 1, nessa posição, indica que o byte de valor decimal ***pos*** ocorre mais de uma vez no objeto de informação. Observe que os vetores S e S_{BA} representam, de maneiras distintas, as mesmas informações.

Algoritmo 4: (S) ← Extrair_S(*iOBJ*)

```
   Entrada  : Um objeto de informação iOBJ
   Saída    : S(iOBJ), S_BA(iOBJ)
 1 início
 2     cria_vetor_de_bytes(S);
 3     cria_vetor_de_bits(S_BA);
 4     cria_vetor_de_bytes(Temp)//** de 256 posições **// ;
 5     preenche_vetor(S_BA,0);
 6     preenche_vetor(S,0);
 7     preenche_vetor(Temp,0);
 8     n ← 0;
 9     para byte = 0 to 255 faça
10         pos ← decimal(iOBJ[byte]);
11         cont ← Temp[pos];
12         Temp[pos] ← cont + 1;
13     fim para
14     para i = 0 to 255 faça
15         se Temp[i] > 1 então
16             S_BA[i] ← 1;
17             S[n] ← Temp[i];
18             n ← n + 1;
19         fim se
20     fim para
21     Retorna S(iOBJ), S_BA(iOBJ)
22 fim
```

Implementação de (Medida) ← Extrair_M(iOBJ) **(Algoritmo 5)**.

A Medida representa as posições ocupadas pelos *bytes* da Qualidade em um determinado objeto de informação. A Medida é representada por uma matriz M de 256 linhas por 256 colunas, e por um vetor M_{ALL}, de 256 posições. A matriz M é esparsa.

Algoritmo 5: (M) ← Extrair_M(iOBJ)

```
   Entrada : Um objeto de informação iOBJ
   Saída   : M_ALL(iOBJ)
 1 início
 2     cria_matriz_de_bytes(M);
 3     cria_vetor_de_bytes(M_ALL);
 4     cria_vetor_de_bytes(Temp)//** de 256 posições **// ;
 5     preenche_vetor(M_ALL, 0);
 6     preenche_vetor(M, 0);
 7     preenche_vetor(Temp, −1);
 8     x ← 0;
 9     para i = 0 to 255 faça
10         pos ← decimal(iOBJ[i]);
11         cont ← Temp[pos];
12         Temp[pos] ← cont + 1;
13         cont ← Temp[pos];
14         M[pos][cont] ← i;
15     fim para
16     para i = 0 to 255 faça
17         se Temp[i] > 0 então
18             ultimo ← M[i][Temp[i]];
19             M[i][Temp[i]] ← M[i][Temp[i − 1]];
20             M[i][Temp[i − 1]] ← ultimo;
21             para n = 1 to Temp[i] faça
22                 M_ALL[x] ← M[i][n];
23                 x ← x + 1;
24             fim para
25         fim se
26     fim para
27     Retorna M_ALL(iOBJ)
28 fim
```

A primeira dimensão da matriz M representa a Qualidade e a segunda dimensão representa a Quantidade. Por exemplo, se M[32][1] = 24, isto significa que o byte decimal 32 ocupa, na primeira vez que aparece no objeto de informação, a vigésima quarta posição. M[32][7] = 30 significa que o byte 32 ocupa, na sétima vez que aparece no objeto de informação, a trigésima posição. Os valores das posições dos bytes estão dispostos em ordem crescente. Os valores dos 2 últimos elementos são invertidos para indicar o final do conjunto (linhas 18 a 20). Isto é feito para determinar o final de um conjunto de posições referentes a um determinado *byte* e o início das posições referentes a outro *byte*.

A Tabela 10 apresenta uma ilustração da extração da medida de um determinado objeto de informação que contém apenas 4 bytes diferentes, sendo que cada byte ocorre 64 vezes

no iOBJ$_k$. Ex: IOBJ$_k$ = A,A,A,...,A,B,B,B,...,B,C,C,C,...,C,DDD,...,D

Tabela 10 – Ilustração da Extração da Medida

Qualidade	Quantidade	Medida
A	64	1, 2, 3, ... , **64, 63**
B	64	65,66,67, ... , **128, 127**
C	64	129, 130, 131, ... , **192, 191**
D	64	193,194, 195, ... , **256, 255**

A seguir será apresentado um exemplo da execução que ilustra a extração das características de um objeto de informação. A Quantidade e a Qualidade são representadas por vetores de *bits*, enquanto a medida é representada por um vetor de bytes composto por números que indicam a posição dos bytes no objeto de informação.

Exemplo 1: Considere um objeto de informação iOBJ que contem o seguinte texto: “Ninguém será objeto de ingerências arbitrarias em sua vida privada, sua família, seu domicilio ou sua correspondência, nem de ataques a sua honra ou a sua reputação. Toda pessoa tem direito a proteção da lei contra tais ingerências ou ataques."(artigo 12 da Declaração Universal dos Direitos Humanos).

Observe que o objeto de informação contem 31 *bytes* diversos, onde cada *byte* representa um símbolo. Então a característica da Qualidade do iOBJ é um vetor com 31 elementos, o código *American Standard Code for Information Interchange (ASCII)* e o símbolo correspondente são apresentados a seguir:

Q(iOBJ) = {**32(espaço)**, 34("), 40(( ), 41( )), 44(,), 46(.), 49(1), 50(2), 78(N), 84(T),97(a), 98(b), 99(c), 100(d), 101(e), 102(f), 103(g), 104(h), 105(i), 106(j), 108(l), 109(m), 110(n), 111(o), 112(p), 113(q), 114(r), 115(s), 116(t), 117(u), **118(v)**}

Assim, a característica Quantidade do iOBJ é também um vetor **S** de 31 elementos, em que cada elemento de **S** indica a quantidade de vezes em que os *bytes* de **Q(iOBJ)** aparecem no **iOBJ**:

S(iOBJ) = { **40**, 2, 1, 1, 3, 2, 1, 1, 1, 1, 34, 2, 8, 9, 22, 1 4, 1, 20, 1, 3, 6, 10, 18, 5, 2, 15, 16, 11, 13, **2**}

Deve-se observar cuidadosamente a relação entre as características de Qualidade e Quantidade. Pode-se observar que, por exemplo, o caractere “espaço"(ASCII 32), que é o primeiro elemento em **Q(iOBJ)**, representado como Q(iOBJ)[1], aparece 40 vezes no iOBJ, e o caractere “v"(ASCII 118), o último elemento em **Q(iOBJ)**, representado como $Q(iOBJ)[31]$, ocorre 2 vezes no *iOBJ*.

Neste cenário, as informações da posição dos *bytes* no iOBJ são armazenadas temporariamente em uma matriz $M(iOBJ)$ que contém, neste exemplo, 31 vetores correspondentes aos 31 elementos da Qualidade. Essas informações representam a proporção em que os diferentes *bytes* estão presentes no $iOBJ$. A matriz $M(iOBJ)$ é apresentado a seguir:

$M(iOBJ)$ = {{ 8, 13, 20, 23, 35, 47, 50, 54, 59, 68, 72, 81, 85, 95, 98, 102, 119, 123, 126, 134, 136, 140, 146, 149, 151, 155, 166, 171, 178, 182, 190, 192, 201, 204, 208, 215, 220, 232, 252, 235}, {244, 0}, {245}, {255}, {67, 118, 80}, {243, 165}, {253}, {254}, {1}, {167}, {12, 33, 36, 42, 45, 53, 58, 64, 66, 71, 74, 79, 101, 117, 127, 129, 135, 139, 145, 150, 154, 161, 163, 170, 177, 191, 199, 203, 214, 217, 230, 236, 246, 238}, {38, 15}, {31, 90, 103, 115, 162, 198, 228, 209}, {21, 57, 65, 86, 112, 124, 169, 202, 183},{6, 10, 17, 22, 27, 29, 48, 83, 107, 113, 121, 125, 132, 157, 173, 180, 186, 197, 206, 224, 241, 226}, {73}, {4, 26, 250, 223}, {141}, {2, 24, 32, 39, 44, 56, 62, 76, 78, 89, 91, 93, 116, 184, 187, 207, 218, 221, 249, 229}, {16}, {77, 205, 92}, {7, 49, 75, 88, 181, 122 }, {3, 25, 30, 111, 114, 120, 143, 211, 227, 222}, {14, 19, 87, 94, 96, 104, 110, 142, 147, 164, 168, 176, 189, 195, 200, 210, 251, 233}, {60, 109, 158, 193, 172}, {239, 130}, {11, 28, 37, 41, 43, 61, 105, 106, 144, 156, 185, 194, 213, 247, 225}, {9, 34, 46, 51, 69, 82, 99, 108, 133, 137, 152, 174, 175, 219, 242, 231}, {18, 40, 128, 160, 179, 188, 196, 212, 216, 248, 237}, {5, 52, 70, 84, 97, 100, 131, 138, 148, 153, 159, 240, 234}, **{63, 55}**}

Observe, por exemplo, que o caractere “v"(ASCII 118), que é o 31º elemento em **Q(iOBJ)**, ocorre 2 vezes ($Q(iOBJ)[31] = 2$) no $iOBJ$, nas posições 63 e 55, que estão representadas pelo último vetor em $M(iOBJ)$.

A matriz $M(iOBJ)$ será transformado em um vetor unidimensional para esconder a “quantificação"dos bytes da Qualidade. Esse novo vetor será denominado $M_{ALL}(iOBJ)$ e conterá todos os elementos presentes em $M(iOBJ)$. É importante ressaltar que $M_{ALL}(iOBJ)$ deverá sempre conter 256 números diferentes (0 a 255). Neste exemplo, o vetor $M_{ALL}(iOBJ)$ é descrito a seguir:

$M_{ALL}(iOBJ)$ = { 8 , 13 , 20 , 23 , 35 , 47 , 50 , 54, 59 , 68 , 72 , 81 , 85 , 95 , 98 ,102, 119 ,123 ,126 ,134 ,136 ,140 ,146 ,149, 151 ,155 ,166 ,171 ,178 ,182 ,190 ,192, 201 ,204 ,208 ,215 ,220 ,232 ,252 ,235, 244 , 0 ,245 ,255 , 67 ,118 , 80 ,243, 165 ,253 ,254 , 1 ,167 , 12 , 33 , 36, 42 , 45 , 53 , 58 , 64 , 66 , 71 , 74, 79 ,101 ,117 ,127 ,129 ,135 ,139 ,145, 150 ,154 ,161 ,163 ,170 ,177 ,191 ,199, 203 ,214 ,217 ,230 ,236 ,246 ,238 , 38, 15 , 31 , 90 ,103 ,115 ,162 ,198 ,228, 209 , 21 , 57 , 65 , 86 ,112 ,124 ,169, 202 ,183 , 6 , 10 , 17 , 22 , 27 , 29, 48 , 83 ,107 ,113 ,121 ,125 ,132 ,157, 173 ,180 ,186 ,197 ,206 ,224 ,241 ,226, 73 , 4 , 26 ,250 ,223 ,141 , 2 , 24, 32 , 39

, 44 , 56 , 62 , 76 , 78 , 89, 91 , 93 ,116 ,184 ,187 ,207 ,218 ,221, 249 ,229 , 16 , 77 ,205 , 92 , 7 , 49, 75 , 88 ,181 ,122 , 3 , 25 , 30 ,111, 114 ,120 ,143 ,211 ,227 ,222 , 14 , 19, 87 , 94 , 96 ,104 ,110 ,142 ,147 ,164, 168 ,176 ,189 ,195 ,200 ,210 ,251 ,233, 60 ,109 ,158 ,193 ,172 ,239 ,130 , 11, 28 , 37 , 41 , 43 , 61 ,105 ,106 ,144, 156 ,185 ,194 ,213 ,247 ,225 , 9 , 34, 46 , 51 , 69 , 82 , 99 ,108 ,133 ,137, 152 ,174 ,175 ,219 ,242 ,231 , 18 , 40, 128 ,160 ,179 ,188 ,196 ,212 ,216 ,248, 237 , 5 , 52 , 70 , 84 , 97 ,100 ,131, 138 ,148 ,153 ,159 ,240 ,234 , 63 , 55 }

A seguir serão descritos os exemplos dos vetores de bits $Q_{BA}(iOBJ)$ e $S_{BA}(iOBJ)$, que otimizam o espaço de armazenamento das informações da Qualidade e da Quantidade nos arquivos **Fq.bin** e **Fs.bin**, respectivamente.

Geração do Vetor de bits $Q_{BA}(iOBJ)$ para armazenar as informações da Qualidade.

As informações referentes à característica Qualidade, que estavam armazenadas no vetor **Q(iOBJ)**, serão codificadas em um vetor de bits da seguinte forma: Um determinado *iOBJ* pode conter no máximo 256 *bytes* distintos, então, Q_{BA} $(iOBJ)$ deve ser um vetor de tamanho 256 bits. Cada posição em $Q_{BA}(iOBJ)$ representa um *byte*, de forma ordenada. Se um determinado *byte* ocorre no *iOBJ*, a posição correspondente ao seu código decimal ASCII em $Q_{BA}(iOBJ)$ armazenará um bit 1, caso contrário armazenará um bit 0.

O vetor $Q_{BA}(iOBJ)$, criado a partir de **Q(iOBJ)** ilustrado no Exemplo 1, é apresentado a seguir:

$Q_{BA}(iOBJ)$ = { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 ,
1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 ,
1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }

Pode-se notar que $Q_{BA}(iOBJ)_1 = 0$, uma vez que o primeiro caractere na tabela ASCII é o símbolo “nulo"e ele não está presente no *iOBJ*. Além disso, observa-se que $Q_{BA}(iOBJ)_{32} = 1$, uma vez que o 32º símbolo na tabela ASCII é o caractere “espaço"que está presente no *iOBJ*.

Geração do Vetor de bits $S_{BA}(iOBJ_k)$ para armazenar as informações da Quantidade.

As informações da Quantidade, que estavam armazenadas no vetor **S(iOBJ)**, serão codificadas em um vetor de bits da seguinte forma: A partir do vetor **S(iOBJ)**, será gerado um vetor de bits, denominado $S_{BA}(iOBJ)$, para representar **S(iOBJ)**. $S_{BA}(iOBJ)$ = $\{bit_1$, bit_2, bit_3, $\cdots$, $bit_q\}$ de tal forma que, q é a quantidade de *byte*s diversos presentes em *iOBJ*, ou seja, q é igual ao tamanho de **Q(iOBJ)**. Cada bit em $S_{BA}(iOBJ)$ representa um elemento de **Q(iOBJ)**. Então, o primeiro elemento em $S_{BA}(iOBJ)$ caracteriza o primeiro elemento em **Q(iOBJ)**, e assim sucessivamente. Se o *byte* b_i ocorre apenas uma vez no iOBJ, a posição i em $S_{BA}(iOBJ)_i$ irá armazenar o bit 0, caso contrário, irá armazenar o bit 1. O vetor S_{BA} tem tamanho de 256 bits para suportar objetos de informação com 256 *bytes* diversos. Se q for menor do que 256, então S(iOBJ)[j] será preenchido com 0 para todo j maior do que q, ou seja, completa-se o vetor S_{BA} com zeros. O vetor $S_{BA}(iOBJ)$, criado a partir de **S(iOBJ)** é apresentado a seguir:

S_{BA} = { **1** , 1 , 0 , 0 , 1 , 1 , **0** , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 }.

Note que, $S_{BA}(iOBJ)_1 = 1$ porque o primeiro elemento de **Q(iOBJ)** é o caractere “espaço", que ocorre 40 vezes no *iOBJ*. Além disso, observe que $S_{BA}(iOBJ)_{10} = 0$ uma vez que o 10º elemento de **Q(iOBJ)** é o caractere “N"que aparece apenas uma vez no *iOBJ*.

Ao final desta etapa as características do objeto de informação já foram extraídas e estão armazenadas em dois vetores de bits (Q_{BA} e S$_{BA}$) e um vetor de *bytes* (M_{ALL}). Vale destacar que os vetores Q e Q_{BA} representam, de maneiras distintas, as mesmas informações. O mesmo ocorre com S e S_{BA}.

Desta forma, dado um determinado objeto de informação *iOBJ*, suas características de Qualidade, Quantidade e Medida foram extraídas e armazenadas, respectivamente, nos vetores Q_{BA}, S_{BA} e M_{ALL}. É importante enfatizar que o vetor Q_{BA} revela que bytes ocorrem no *iOBJ*,

porém não informa a quantidade de vezes ou as posições em que cada byte ocorre no *iOBJ*. Assim, é computacionalmente inviável reconstruir o objeto de informação *iOBJ* utilizando-se somente Q_{BA}. Já o vetor S_{BA} revela apenas se os bytes que compõem o *iOBJ* ocorrem uma única vez ou mais de uma vez, contudo não informa quais são os bytes e nem a ordem em que estes ocorrem no *iOBJ*. Portanto, é computacionalmente inviável reconstruir o objeto de informação *iOBJ* utilizando-se somente S_{BA}. Por fim, M_{ALL} pode revelar a quantidade de vezes que um determinado byte ocorre no *iOBJ* (Quantidade) e em que posições esses bytes ocorrem em *iOBJ*. Contudo, não informa quais são esses bytes. Observe que no exemplo discutido anteriormente, os dois primeiros valores de M_{ALL} são: 8 e 13. Neste caso, 8 e 13 podem ser as duas primeiras posições de um mesmo byte ou a primeira posição de dois bytes distintos que ocorrem uma única vez no *iOBJ*. Portanto, recompor o objeto de informação *iOBJ* utilizando-se somente um dos três vetores, Q_{BA}, S_{BA} ou M_{ALL}, não é uma tarefa trivial. Entretanto, técnicas probalísticas baseadas na frequência da ocorrência dos símbolos poderiam ser utilizadas para inferir a Qualidade do *iOBJ* a partir de M_{ALL} e assim recompor o objeto de informação *iOBJ*. Por este motivo, o vetor M_{ALL} será anonimizado na próxima etapa da fase de decomposição. Uma discussão mais detalhada sobre a segurança das características de Qualidade, Quantidade e Medida pode ser encontrada na Seção 6.5.

6.4.2.2 Etapa 2: Anonimização da Medida

Os elementos da Medida representam as posições ocupadas pelos conjuntos de *bytes* da Qualidade. Os valores referentes ao um mesmo *byte* estão dispostos em ordem crescente, invertendo-se as posições dos 2 últimos para indicar o final das posições de um determinado *byte*. A análise da ordenação dos elementos de M_{ALL} poderia revelar valores da quantidade ($\mathbf{S}_{BA}$) a um possível atacante.

No Exemplo 1, o primeiro byte do vetor **Q** era o símbolo de “espaço"e estava presente em 40 posições no vetor M ({8, 13, 20, 23, 35, 47, 50, 54, 59, 68, 72, 81, 85, 95, 98, 102, 119, 123, 126, 134, 136, 140, 146, 149, 151, 155, 166, 171, 178, 182, 190, 192, 201, 204, 208, 215, 220, 232, 252, 235}). Para Um atacante que conhecesse o algoritmo e estivesse de posse do arquivo **Fm.bin** da Medida, poderia inferir que o primeiro byte do iOBJ ocorreria 40 vezes, pois o valor anterior ao elemento 235 é 252, o que indica final da sequência de posições. Assim sendo, deduziria o valor da quantidade deste byte. Se soubesse que tratava-se de um arquivo texto, poderia inferir que, um byte que se repete 40 vezes em um texto de 256 palavras provavelmente

seria o caractere “espaço". Para evitar este tipo de dedução, as posições do vetor da medida devem ser embaralhadas.

Esta etapa tem por finalidade embaralhar as posições dos elementos da Medida (vetor M_{ALL}), com o objetivo de anonimizar o seu conteúdo. A ideia principal é reordenar a posição dos elementos da Medida para que a informação das posições de um byte que esteja presente mais de uma vez em um objeto de informação seja embaralhada. O objetivo desta etapa é dificultar a recomposição de um objeto de informação *iOBJ* a partir de sua Medida $M_{ALL}(iOBJ)$, aumentando assim o nível de confidencialidade da estratégia proposta. Neste sentido, os elementos de M_{ALL} são permutados usando Q, $\mathbf{Q}_{BA}$ e $\mathbf{S}_{BA}$ como entrada da função Fp_1 para embaralhamento dos elementos de $M_{ALL}[256]$, gerando o vetor M_E de 256 posições. O Algoritmo 6 descreve a rotina Fp_1.

Implementação de $(\mathrm{M}_E) \leftarrow \mathrm{Fp}_1(\mathrm{M}_{ALL}, \mathrm{Q}, \mathrm{Q}_{BA}, \mathrm{S}_{BA})$ (**Algoritmo 6**). Este algoritmo implementa uma permutação dos elementos da Medida armazenados no vetor M_{ALL}, utilizando as informações da Qualidade (vetores Q e Q_{BA}) e da Quantidade (vetor S_{BA}) como parâmetros da função de permutação.

Inicialmente, cria-se no vetor M_E uma cópia do vetor M_{ALL}. O vetor M_E será embaralhado e retornado como saída pelo algoritmo. Logo, M_E será o vetor M_{ALL} embaralhado. A primeira parte do algoritmo, linhas 8 a 14, cria e preenche um vetor Q_T, contendo 259 posições. A ideia principal é preencher Q_T com os elementos de Q em ordem inversa. Contudo, normalmente, o vetor Q tem um tamanho menor que 256, uma vez que um objeto de informação dificilmente contém 256 símbolos diferentes. Por este motivo, repete-se a cópia dos elementos de Q, em ordem inversa, para Q_T até que este esteja completamente preenchido. Para isso, a informação do tamanho do vetor Q é utilizada (linhas 4 e 12) na variável ***n***. Considerando o vetor Q ilustrado no Exemplo 1, o vetor Q_T correspondente é mostrado a seguir:

Q_T = { 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109,

108, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 84, 78, 50, 49, 46, 44, 41, 40, 34, 32, 118, 117, 116, 115 }

Em seguida, a estrutura de repetição ilustrada na linha 16 executa 256 permutações em M_E. Em cada permutação é utilizado um índice diferente. O valor desse índice é armazenado na variável *indice_permutacao* e é calculado da seguinte forma:

- Se $Q_{BA}(iOBJ_i) = 0$, ou seja, o símbolo ASCII cujo valor decimal é i não está presente no $iOBJ$, o valor do índice será o resultado de uma operação XOR entre $Q_T[i]$ e $Q_T[i+1]$;
- Se $Q_{BA}(iOBJ_i) = 1$, ou seja, o símbolo ASCII cujo valor decimal é i está presente no $iOBJ$, duas alternativas são analisadas:
  - Se $S_{BA}[i] = 0$, ou seja, o símbolo ASCII cujo valor decimal é i ocorre uma única vez no $iOBJ$, o valor do índice será o resultado de uma operação XOR entre $Q_T[i]$, $Q_T[i+1]$ e $Q_T[i+2]$;
  - Se $S_{BA}[i] = 1$, ou seja, o símbolo ASCII cujo valor decimal é i ocorre mais de uma vez no $iOBJ$, o valor do índice será o resultado de uma operação XOR entre $Q_T[i]$, $Q_T[i+1]$, $Q_T[i+2]$ e $Q_T[i+2]$;

Para ilustrar o funcionamento destas permutações, considere $i = 0$. Neste caso, $Q_{BA}[0] = 0$. Logo o valor da variável *indice* será computado por meio de uma operação XOR entre $Q_T[0]$ (118) e $Q_T[1]$ (117), resultando no valor 3 ($indice = 3$). Desta forma, o valor de $M_E[i]$ (onde, $M_E[indice] = M_E[0] = 8$) será permutado com o valor de $M_E[indice]$ (onde, $M_E[indice]$ = $M_E[3] = 23$). Agora, considere $i = 32$. Neste caso, $Q_{BA}[32] = 1$. Portanto, deve-se verificar o valor de $S_{BA}[32]$. Como $S_{BA}[32] = 0$, o valor da variável *indice* será computado por meio de uma operação XOR entre $Q_T[32]$ (117), $Q_T[33]$ (116) e $Q_T[34]$ (115), resultando no valor 114 (ou seja, $indice = 114$). Desta forma, o valor de $M_E[i]$ (ou seja, $M_E[32]$) será permutado com o valor de $M_E[indice]$ (ou seja, $M_E[114]$). Vale destacar que uma mesma posição pode ser permutada diversas vezes, dependendo dos valores em Q_T. A saída do Algoritmo 6 será o vetor M_E, o qual representa o vetor M_{ALL} anonimizado, por meio de 256 permutações.

Algoritmo 6: $(M_E) \leftarrow Fp_1(M_{ALL}, Q, Q_{BA}, S_{BA})$

Entrada : Q, Q_{BA}, S_{BA}, M_{ALL}
Saída : M_E

```
1  início
2      cria_vetor_de_bytes(Q_T[259]);
3      cria_vetor_de_bytes(M_E[256]);
4      n ← tamanho(Q);
5      para i = 0 to 255 faça
6          *** cria uma cópia de M_ALL ***;
7          M_E[i] ← M_ALL[i];
8      fim para
9      para i = 0 to 259 faça
10         Q_T[i] ← Q[n];
11         n ← n − 1;
12         se n < 0 então
13             n ← tamanho(Q);
14         fim se
15     fim para
16     indice ← 0;
17     para i = 0 to 255 faça
18         se Q_BA[i] = 0 então
19             *** byte i não existe no iOBJ ***;
20             n ← Q_T[i] ⊕ Q_T[i+1];
21         fim se
22         senão
23             *** Q_BA[i] = 1 byte i existe no iOBJ ***
24         fim se
25         ;
26         se S_BA[i] = 0 então
27             *** há ocorrência do byte apenas uma vez no iOBJ ***
28         fim se
29         ;
30         n ← Q_T[i] ⊕ Q_T[i+1] ⊕ Q_T[i+2];
31         senão
32             *** Q_BA[i] = 1 há ocorrência do byte mais de uma vez no iOBJ ***
33         fim se
34         ;
35         n ← Q_T[i] ⊕ Q_T[i+1] ⊕ Q_T[i+2] ⊕ Q_T[i+3];
36         Temp ← M_E[i];
37         M_E[i] ← M_E[indice];
38         M_E[indice] ← Temp;
39     fim para
40     Retorna M_E
41 fim
```

Ainda seguindo o Exemplo 1, observe que os 40 primeiros elementos de M_{ALL},

marcados em negrito a seguir, representam as posições do primeiro byte de Q (símbolo de “espaço"). Esses elementos são apresentados sublinhados no vetor M_E. Pode-se constatar o efeito da função Fp_1 neste exemplo, distribuindo esses elementos entre os demais elementos do vetor M_E. A Tabela 11 apresenta as propriedades do caractere “espaço"antes e depois de serem embaralhadas pela função Fp_1.

1	$M_{ALL}(iOBJ)$ = { **8 , 13 , 20 , 23 , 35 , 47 , 50 , 54, 59 , 68 , 72 , 81 , 85 , 95 , 98 ,102, 119 ,123 ,126 ,134 ,136 ,140 ,146 ,149, 151 ,155 ,166 ,171 ,178 ,182 ,190 ,192, 201 ,204 ,208 ,215 ,220 ,232 ,252 ,235**, 244 , 0 ,245 ,255 , 67 ,118 , 80 ,243, 165 ,253 ,254 , 1 ,167 , 12 , 33 , 36, 42 , 45 , 53 , 58 , 64 , 66 , 71 , 74, 79 ,101 ,117 ,127 ,129 ,135 ,139 ,145, 150 ,154 ,161 ,163 ,170 ,177 ,191 ,199, 203 ,214 ,217 ,230 ,236 ,246 ,238 , 38, 15 , 31 , 90 ,103 ,115 ,162 ,198 ,228, 209 , 21 , 57 , 65 , 86 ,112 ,124 ,169, 202 ,183 , 6 , 10 , 17 , 22 , 27 , 29, 48 , 83 ,107 ,113 ,121 ,125 ,132 ,157, 173 ,180 ,186 ,197 ,206 ,224 ,241 ,226, 73 , 4 , 26 ,250 ,223 ,141 , 2 , 24, 32 , 39 , 44 , 56 , 62 , 76 , 78 , 89, 91 , 93 ,116 ,184 ,187 ,207 ,218 ,221, 249 ,229 , 16 , 77 ,205 , 92 , 7 , 49, 75 , 88 ,181 ,122 , 3 , 25 , 30 ,111, 114 ,120 ,143 ,211 ,227 ,222 , 14 , 19, 87 , 94 , 96 ,104 ,110 ,142 ,147 ,164, 168 ,176 ,189 ,195 ,200 ,210 ,251 ,233, 60 ,109 ,158 ,193 ,172 ,239 ,130 , 11, 28 , 37 , 41 , 43 , 61 ,105 ,106 ,144, 156 ,185 ,194 ,213 ,247 ,225 , 9 , 34, 46 , 51 , 69 , 82 , 99 ,108 ,133 ,137, 152 ,174 ,175 ,219 ,242 ,231 , 18 , 40, 128 ,160 ,179 ,188 ,196 ,212 ,216 ,248, 237 , 5 , 52 , 70 , 84 , 97 ,100 ,131, 138 ,148 ,153 ,159 ,240 ,234 , 63 , 55 }
2	$M_E(iOBJ)$ = { 13 , 55 ,216 , 63 , 20 ,188 ,148 , 5 , 50 , 35 ,212 , 59 , 102 , 81 , 72 ,240 , 23 , 85 , 98 , 12 , 166 ,206 , 126 , 47 , 54 , 192 , 18 , 8 , 151 ,246 , 136 , 97 , 48 , 146 , 27 ,121 , 190 , 204 , 220 , 68 , 208 ,124 ,125 ,107 ,169 ,113 ,112 ,118 , 232 ,197 , 10 , 83 ,165 ,231 , 178 , 155 ,243 , 171 , 33 , 182 , 42 , 149 , 64 ,167 , 71 , 134 , 79 , 74 ,117 , 235 ,127 ,129 ,245 ,145 ,139 ,154 , 66 , 65 , 38 , 215 , 123 , 1 ,161 ,101 , 252 ,248 ,177 , 17 , 53 , 36 ,238 ,170 , 90 ,217 ,115 ,230 , 198 , 29 , 80 , 21 , 67 , 6 ,183 ,202 , 57 ,244 , 0 ,209 , 45 ,253 ,254 ,163 , 255 , 140 ,236 , 201 , 119 , 95 ,157 , 31 , 86 ,103 ,173 ,162 , 40 ,228 ,214 , 22 , 241 ,135 ,226 , 73 ,150 ,250 , 26 ,141 ,180 , 24 , 2 ,199 ,203 ,186 , 44 ,224 , 15 , 58 , 39 ,191 , 91 ,132 ,116 , 32 ,218 , 78 ,249 , 89 , 16 ,229 ,205 , 4 , 92 , 7 ,223 , 88 , 75 ,122 ,221 , 25 , 3 , 56 , 62 , 76 ,114 , 77 ,187 , 93 , 111 ,184 , 14 ,207 , 87 , 30 ,110 ,227 ,147 ,222 ,168 ,164 ,189 , 49 ,195 ,200 , 181 ,233 ,251 ,109 ,142 ,193 ,158 ,120 ,143 ,211 ,130 ,176 , 96 , 19 ,239 , 94 , 61 ,104 ,106 ,172 ,194 , 41 ,247 , 43 , 9 ,225 , 46 ,210 , 51 , 69 , 60 ,108 , 99 ,137 ,213 ,174 ,152 , 11 , 28 , 37 ,242 , 34 ,156 ,105 ,219 ,144 ,179 ,185 , 196 ,175 ,237 ,128 , 52 ,160 , 84 , 70 ,100 , 82 ,131 ,138 ,133 ,159 ,153 ,234 }

Por fim, vale a pena destacar que o algoritmo utilizado para anonimizar o vetor M_{ALL}, Algoritmo 6, o qual produz como saída o vetor M_E, é determinístico. Além disso, dado o vetor M_E, para se recompor o objeto de informação iOBJ será necessário gerar novamente o vetor M_{ALL}, o que só será viável se utilizarmos os vetores $\mathbf{Q}$, $\mathbf{Q}_{BA}$ e $\mathbf{S}_{BA}$. Desta forma, é computacionalmente inviável recompor o objeto de informação iOBJ utilizando-se somente a Medida anonimizada (M_E).

6.4.2.3 Etapa 3: Anonimização da Qualidade.

O vetor Q_{BA}, que representa a Qualidade, indica quais bytes (símbolos ASCII) estão presentes no objeto de informação. Para não revelar esta informação, os elementos do vetor Q_{BA} serão embaralhados, utilizando como índice de permutação a Medida anonimizada, ou seja, o vetor M_E. O Algoritmo 7 descreve a rotina Fp_2. Este algoritmo tem como entrada os vetores Q_{BA} e M_E. A saída deste algoritmo será o vetor Q_{BA} anonimizado, o qual será denominado Q_E. A rotina Fp_2 utiliza os elementos de M_E como imagem da função bijetora de permutação das posições dos elementos de Q_{BA}.

Tabela 11 – Propriedades do Caractere “espaço"no Exemplo 1

Qualidade	Quantidade	Medida M_{ALL}
(espaço)	40	{8, 13, 20, 23, 35, 47, 50, 54, 59, 68, 72, 81, 85, 95, 98, 102, 119, 123, 126, 134, 136, 140, 146, 149, 151, 155, 166, 171, 178, 182, 190, 192, 201, 204, 208, 215, 220, 232, 252, 235}
Qualidade	Quantidade	Medida M_E
(espaço)	40	{13, ..., 20, ...,50, 35, ..., 59, 102, 81, 72, ..., 23, 85, 98, ..., 166, ..., 126, 47, 54, 192, ..., 8, 151, ..., 136, ..., 146, ..., 190, 204, 220, 68, 208, ..., 232, ..., 178, 155, ..., 171, ..., 182, ..., 149, ..., 134, .., 235, ..., 215, 123, ..., 252, ..., 140, ..., 201, 119, 95 }

Algoritmo 7: $(Q_E) \leftarrow Fp_2(Q_{BA}, M_E)$

```
   Entrada  : Q_BA, M_E
   Saída    : Q_E
1  início
2      para i = 0 to 255 faça
3          Q_E[M_E[i]] ← Q_BA[i];
4      fim para
5      Retorna Q_E;
6  fim
```

Vale recordar que o vetor Q_{BA} é um vetor de bits de tamanho 256, onde seus elementos possuem valor 0 ou 1 (indicando se um determinado byte ocorre ou não no objeto de informação iOBJ). Além disso, o vetor M_E é um vetor de 256 posições, que armazena exatamente os valores de 0 a 255 (uma vez que este representa as posições de um objeto de informação). O algoritmo percorre os elementos do vetor Q_{BA} e copia cada elemento $Q_{BA}[i]$ para a posição $M_E[i]$ no vetor Q_E. Como os elementos de M_E são exatamente os valores de 0 a 255, sem repetição, o vetor Q_E terá exatamente os mesmos valores de Q_{BA}, só que em posições diferentes.

Observe que, dado o vetor Q_E, para se recompor o objeto de informação *iOBJ* será necessário gerar novamente o vetor Q_{BA}, o que só será viável se utilizarmos o vetor M_E. Desta forma, é computacionalmente inviável recompor o objeto de informação *iOBJ* utilizando-se somente a Qualidade anonimizada (Q_E).

A seguir, é apresentado um exemplo da permutação realizada pelo Algoritmo 7 sobre o vetor de bits Q_{BA} descrito no Exemplo 1, tendo como parâmetro de entrada o vetor M_E, gerando como saída o vetor Q_E, que representa a Quantidade embaralhada.

1 Q_{BA} = { 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 1 , 0 , 1 ,
0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 1
, 1 , 1 , , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }

2 M_E = { 13 , 55 , 216 , 63 , 20 ,188 ,148 , 5 , 50 , 35 , 212 , 59 , 102 , 81 , 72 , 240 , 23 , 85 , 98 , 12 , 166 ,206 , 126 , 47 ,
54 , 192 , 18 , 8 , 151 ,246 , 136 , 97 , 48 , 146 , 27 ,121 , 190 , 204 , 220 , 68 , 208 ,124 ,125 ,107 ,169 ,113 ,112 ,118
, 232 ,197 , 10 , 83 ,165 ,231 , 178 , 155 ,243 ,171 , 33 ,182 , 42 , 149 , 64 ,167 , 71 , 134 , 79 , 74 ,117 , 235 ,127 ,129
,245 ,145 ,139 ,154 , 66 , 65 , 38 , 215 , 123 , 1 ,161 ,101 , 252 ,248 ,177 , 17 , 53 , 36 ,238 ,170 , 90 ,217 ,115 ,230 ,
198 , 29 , 80 , 21 , 67 , 6 ,183 ,202 , 57 ,244 , 0 ,209 , 45 ,253 ,254 ,163 , 255 , 140 ,236 , 201 , 119 , 95 ,157 , 31 , 86
,103 ,173 ,162 , 40 ,228 ,214 , 22 , 241 ,135 ,226 , 73 ,150 ,250 , 26 ,141 ,180 , 24 , 2 ,199 ,203 ,186 , 44 ,224 , 15 , 58
, 39 ,191 , 91 ,132 ,116 , 32 ,218 , 78 ,249 , 89 , 16 ,229 ,205 , 4 , 92 , 7 ,223 , 88 , 75 ,122 ,221 , 25 , 3 , 56 , 62 , 76
,114 , 77 ,187 , 93 , 111 ,184 , 14 ,207 , 87 , 30 ,110 ,227 ,147 ,222 ,168 ,164 ,189 , 49 ,195 ,200 , 181 ,233 ,251 ,109
,142 ,193 ,158 ,120 ,143 ,211 ,130 ,176 , 96 , 19 ,239 , 94 , 61 ,104 ,106 ,172 ,194 , 41 ,247 , 43 , 9 ,225 , 46 ,210 , 51
, 69 , 60 ,108 , 99 ,137 ,213 ,174 ,152 , 11 , 28 , 37 ,242 , 34 ,156 ,105 ,219 ,144 ,179 ,185 , 196 ,175 ,237 ,128 , 52
,160 , 84 , 70 ,100 , 82 ,131 ,138 ,133 ,159 ,153 ,234 }

3 Q_E = { 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 ,
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1}

6.4.2.4 Etapa 4: Anonimização da Quantidade

No vetor S_{BA}, um valor 1 em uma determinada posição indica que o byte (símbolos ASCII) associado a esta posição ocorre mais de uma vez no objeto de informação. Já um valor zero em uma determinada posição pode indicar tanto que o byte associado a esta posição ocorre apenas uma vez quanto que o byte não ocorre no objeto de informação.

Para não revelar esta informação, os elementos do vetor S_{BA} serão embaralhados, utilizando como índice de permutação a Medida anonimizada, ou seja, o vetor M_E. O Algoritmo 8 descreve a rotina Fp_3. Este algoritmo tem como entrada os vetores S_{BA} e M_E. A saída deste algoritmo será o vetor S_{BA} anonimizado, o qual será denominada S_E. A rotina Fp_3 utiliza os elementos de M_E como imagem da função de bijetora de permutação das posições dos elementos de S_{BA}.

Algoritmo 8: (S_E) ← Fp_3(S_{BA}, M_E)

```
Entrada  :S_BA[256], M_E[256]
Saída    :S_E[256]
início
    para i = 0 to 255 faça
        S_E[M_E[i]] ← S_BA[i];
    fim para
    Retorna S_E[256];
fim
```

Vale recordar que o vetor S_{BA} é um vetor de bits de tamanho 256, onde seus elementos possuem valor 0 ou 1 (indicando se um determinado byte ocorre mais de uma vez, uma única vez ou se este não ocorre no objeto de informação iOBJ). Além disso, o vetor M_E é um vetor de 256 posições, que armazena exatamente os valores de 0 a 255 (uma vez que este representa as posições de um objeto de informação). O algoritmo percorre os elementos do vetor S_{BA} e copia cada elemento $S_{BA}[i]$ para a posição $S_E[i]$ no vetor S_E. Como os elementos de M_E são exatamente os valores de 0 a 255, sem repetição, o vetor S_E terá exatamente os mesmos valores de S_{BA}, só que em posições diferentes.

Observe que, dado o vetor S_E, para se recompor o objeto de informação *iOBJ* será necessário gerar novamente o vetor S_{BA}, o que só será viável se utilizarmos o vetor M_E. Desta forma, é computacionalmente inviável recompor o objeto de informação *iOBJ* utilizando-se somente a Quantidade anonimizada (S_E).

A seguir, é apresentado um exemplo da permutação realizada pelo Algoritmo 8 sobre o vetor de bits S_{BA} descrito no Exemplo 1, tendo como parâmetro de entrada o vetor M_E, gerando como saída o vetor S_E.

---

1 S_{BA} = { 1 , 1 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1, 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }

2 M_E = { 13 , 55 , 216 , 63 , 20 ,188 ,148 , 5 , 50 , 35 , 212 , 59 , 102 , 81 , 72 , 240 , 23 , 85 , 98 , 12 , 166 ,206 , 126 , 47 ,
54 , 192 , 18 , 8 , 151 ,246 , 136 , 97 , 48 , 146 , 27 ,121 , 190 , 204 , 220 , 68 , 208 ,124 ,125 ,107 ,169 ,113 ,112 ,118
, 232 ,197 , 10 , 83 ,165 ,231 , 178 , 155 ,243 ,171 , 33 ,182 , 42 , 149 , 64 ,167 , 71 , 134 , 79 , 74 ,117 , 235 ,127 ,129
,245 ,145 ,139 ,154 , 66 , 65 , 38 , 215 , 123 , 1 ,161 ,101 , 252 ,248 ,177 , 17 , 53 , 36 ,238 ,170 , 90 ,217 ,115 ,230 ,
198 , 29 , 80 , 21 , 67 , 6 ,183 ,202 , 57 ,244 , 0 ,209 , 45 ,253 ,254 ,163 , 255 , 140 ,236 , 201 , 119 , 95 ,157 , 31 , 86
,103 ,173 ,162 , 40 ,228 ,214 , 22 , 241 ,135 ,226 , 73 ,150 ,250 , 26 ,141 ,180 , 24 , 2 ,199 ,203 ,186 , 44 ,224 , 15 , 58
, 39 ,191 , 91 ,132 ,116 , 32 ,218 , 78 ,249 , 89 , 16 ,229 ,205 , 4 , 92 , 7 ,223 , 88 , 75 ,122 ,221 , 25 , 3 , 56 , 62 , 76
,114 , 77 ,187 , 93 , 111 ,184 , 14 ,207 , 87 , 30 ,110 ,227 ,147 ,222 ,168 ,164 ,189 , 49 ,195 ,200 , 181 ,233 ,251 ,109
,142 ,193 ,158 ,120 ,143 ,211 ,130 ,176 , 96 , 19 ,239 , 94 , 61 ,104 ,106 ,172 ,194 , 41 ,247 , 43 , 9 ,225 , 46 ,210 , 51
, 69 , 60 ,108 , 99 ,137 ,213 ,174 ,152 , 11 , 28 , 37 ,242 , 34 ,156 ,105 ,219 ,144 ,179 ,185 , 196 ,175 ,237 ,128 , 52
,160 , 84 , 70 ,100 , 82 ,131 ,138 ,133 ,159 ,153 ,234 }

3 S_E = {0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0
, 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0
, 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 ,
0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 , 0 }

6.4.2.5 Etapa 5: Compactação da Medida

Esta etapa utiliza aritmética modular para, a partir do vetor M_E, gerar dois vetores de *bits*, M_G e M_K, de tal forma que o espaço utilizado para o armazenamento desses dois vetores seja menor que o espaço necessário para armazenar M_E. Assim, o objetivo desta etapa é reduzir o espaço necessário para o armazenamento da Medida.

Para compreender o processo de compactação da Medida, observe inicialmente que os elementos do vetor M_E são números que variam de 0 a 255 e não se repetem. A partir de M_E são gerados dois novos vetores de *bits*, M_G e M_K, da seguinte forma:

1. $M_G = M_E \bmod 32$
2. $M_K = M_E \,/\, 32$

O Algoritmo 9 descreve o processo de compactação da Medida. Inicialmente, são criados os vetores M_G e M_K (linhas 2 a 5). A fim de ilustrar o processo de compactação da Medida, considere novamente o Exemplo 1. A seguir, apresentamos o vetor M_E, descrito anteriormente, além dos vetores M_G e M_K correspondentes.

---

1 M_E = { 13 , 55 , 216 , 63 , 20 ,188 ,148 , 5 , 50 , 35 , 212 , 59 , 102 , 81 , 72 , 240 , 23 , 85 , 98 , 12 , 166 ,206 , 126 , 47 , 54 , 192 , 18 , 8 , 151 ,246 , 136 , 97 , 48 , 146 , 27 ,121 , 190 , 204 , 220 , 68 , 208 ,124 ,125 ,107 ,169 ,113 ,112 ,118 , 232 ,197 , 10 , 83 ,165 ,231 , 178 , 155 ,243 ,171 , 33 ,182 , 42 , 149 , 64 ,167 , 71 , 134 , 79 , 74 ,117 , 235 ,127 ,129 ,245 ,145 ,139 ,154 , 66 , 65 , 38 , 215 , 123 , 1 ,161 ,101 , 252 ,248 ,177 , 17 , 53 , 36 ,238 ,170 , 90 ,217 ,115 ,230 , 198 , 29 , 80 , 21 , 67 , 6 ,183 ,202 , 57 ,244 , 0 ,209 , 45 ,253 ,254 ,163 , 255 , 140 ,236 , 201 , 119 , 95 ,157 , 31 , 86 ,103 ,173 ,162 , 40 ,228 ,214 , 22 , 241 ,135 ,226 , 73 ,150 ,250 , 26 ,141 ,180 , 24 , 2 ,199 ,203 ,186 , 44 ,224 , 15 , 58 , 39 ,191 , 91 ,132 ,116 , 32 ,218 , 78 ,249 , 89 , 16 ,229 ,205 , 4 , 92 , 7 ,223 , 88 , 75 ,122 ,221 , 25 , 3 , 56 , 62 , 76 ,114 , 77 ,187 , 93 , 111 ,184 , 14 ,207 , 87 , 30 ,110 ,227 ,147 ,222 ,168 ,164 ,189 , 49 ,195 ,200 , 181 ,233 ,251 ,109 ,142 ,193 ,158 ,120 ,143 ,211 ,130 ,176 , 96 , 19 ,239 , 94 , 61 ,104 ,106 ,172 ,194 , 41 ,247 , 43 , 9 ,225 , 46 ,210 , 51 , 69 , 60 ,108 , 99 ,137 ,213 ,174 ,152 , 11 , 28 , 37 ,242 , 34 ,156 ,105 ,219 ,144 ,179 ,185 , 196 ,175 ,237 ,128 , 52 ,160 , 84 , 70 ,100 , 82 ,131 ,138 ,133 ,159 ,153 ,234 }

2 M_G = { 13, 23, 24, 31, 20, 28, 20, 5 , 18, 3 , 20, 27, 6 , 17, 8 , 16, 23, 21, 2 , 12, 6 , 14, 30, 15, 22, 0 , 18, 8 , 23, 22, 8 , 1 , 16, 18, 27, 25, 30, 12, 28, 4 , 16, 28, 29, 11, 9 , 17, 16, 22, 8 , 5 , 10, 19, 5 , 7 , 18, 27, 19, 11, 1 , 22, 10, 21, 0 , 7 , 7 , 6 , 15, 10, 21, 11, 31, 1 , 21, 17, 11, 26, 2 , 1 , 6 , 23, 27, 1 , 1 , 5 , 28, 24, 17, 17, 21, 4 , 14, 10, 26, 25, 19, 6 , 6 , 29, 16, 21, 3 , 6 , 23, 10, 25, 20, 0 , 17, 13, 29, 30, 3 , 31, 12, 12, 9 , 23, 31, 29, 31, 22, 7 , 13, 2 , 8 , 4 , 22, 22, 17, 7 , 2 , 9 , 22, 26, 26, 13, 20, 24, 2 , 7 , 11, 26, 12, 0 , 15, 26, 7 , 31, 27, 4 , 20, 0 , 26, 14, 25, 25, 16, 5 , 13, 4 , 28, 7 , 31, 24, 11, 26, 29, 25, 3 , 24, 30, 12, 18, 13, 27, 29, 15, 24, 14, 15, 23, 30, 14, 3 , 19, 30, 8 , 4 , 29, 17, 3 , 8 , 21, 9 , 27, 13, 14, 1 , 30, 24, 15, 19, 2 , 16, 0 , 19, 15, 30, 29, 8 , 10, 12, 2 , 9 , 23, 11, 9 , 1 , 14, 18, 19, 5 , 28, 12, 3 , 9 , 21, 14, 24, 11, 28, 5 , 18, 2 , 28, 9 , 27, 16, 19, 25, 4 , 15, 13, 0 , 20, 0 , 20, 6 , 4 , 18, 3 , 10, 5 , 31, 25, 10 }

3 M_K = { 0, 1, 6, 1, 0, 5, 4, 0, 1, 1, 6, 1, 3, 2, 2, 7, 0, 2, 3, 0, 5, 6, 3, 1, 1, 6, 0, 0, 4, 7, 4, 3, 1, 4, 0, 3, 5, 6, 6, 2, 6, 3, 3, 3, 5, 3, 3, 3, 7, 6, 0, 2, 5, 7, 5, 4, 7, 5, 1, 5, 1, 4, 2, 5, 2, 4, 2, 2, 3, 7, 3, 4, 7, 4, 4, 4, 2, 2, 1, 6, 3, 0, 5, 3, 7, 7, 5, 0, 1, 1, 7, 5, 2, 6, 3, 7, 6, 0, 2, 0, 2, 0, 5, 6, 1, 7, 0, 6, 1, 7, 7, 5, 7, 4, 7, 6, 3, 2, 4, 0, 2, 3, 5, 5, 1, 7, 6, 0, 7, 4, 7, 2, 4, 7, 0, 4, 5, 0, 0, 6, 6, 5, 1, 7, 0, 1, 1, 5, 2, 4, 3, 1, 6, 2, 7, 2, 0, 7, 6, 0, 2, 0, 6, 2, 2, 3, 6, 0, 0, 1, 1, 2, 3, 2, 5, 2, 3, 5, 0, 6, 2, 0, 3, 7, 4, 6, 5, 5, 5, 1, 6, 6, 5, 7, 7, 3, 4, 6, 4, 3, 4, 6, 4, 5, 3, 0, 7, 2, 1, 3, 3, 5, 6, 1, 7, 1, 0, 7, 1, 6, 1, 2, 1, 3, 3, 4, 6, 5, 4, 0, 0, 1, 7, 1, 4, 3, 6, 4, 5, 5, 6, 5, 7, 4, 1, 5, 2, 2, 3, 2, 4, 4, 4, 4, 4, 7 }

Vale destacar que o vetor M_G irá armazenar 256 valores. Mais precisamente, o vetor M_G irá armazenar somente 32 valores distintos (variando do 0 ao 31), sendo que cada um desses valores irá ocorrer exatamente 8 vezes. Assim, pode-se afirmar que M_G irá conter 32 grupos com valores de 0 a 31, onde cada grupo terá 8 elementos. Ora, para armazenar valores na faixa de 0

a 31 são necessários somente 5 bits. Além disso, o vetor M_K irá armazenar 256 valores. Mais precisamente, o vetor M_K irá armazenar somente 8 valores distintos (variando do 0 ao 7), sendo que cada um desses valores irá ocorrer exatamente 32 vezes. Ora, para armazenar valores na faixa de 0 a 7 são necessários somente 3 bits. Logo, para armazenar M_K são necessários 768 bits (3 bits para cada elemento). Este vetor será compactado para 512 bits para economizar espaço de armazenamento em disco. Para isso, será utilizada a codificação Lehmer (BARBAY; NAVARRO, 2009), que determina o índice (ou posição lexicográfica) de uma dada permutação.

Observe agora que, para cada um dos 32 grupos existentes em M_G existe uma permutação de oito números entre 0 e 7 em M_K. A codificação Lehmer utiliza um índice para representar cada permutação diferente de 8 elementos (de 0 a 7). Neste caso, o índice varia de 1 a 40.320 (8!). Para representar números entre 1 e 40.320 são necessários 16 bits. Desta forma, se ao invés de armazenar M_k, que utiliza 768 bits, armazenássemos 32 índices (um índice Lehmer com tamanho de 16 bits, seriam necessários apenas 512 bits, o que proporcionaria uma economia de 256 bits. A Tabela 12 ilustra um trecho da codificação de Lehmer para permutações de 8 números (de 0 a 7).

Tabela 12 – Codigo Lehmer para Combinações de 8 Números

Combinações de 8 Elementos	Código Lehmer
0,1,2,3,4,5,6,7	1
0,1,2,3,4,5,7,6	2
0,1,2,3,4,6,5,7	3
0,1,2,3,4,6,7,5	4
0,1,2,3,4,7,6,5	5
...	...
1,3,7,2,0,5,6,4	7194
1,3,7,2,0,5,4,6	7195
...	...
6,2,0,7,1,3,4,5	31552
6,2,0,7,1,3,5,4	31553
...	...
7,0,1,2,3,6,5,4	40317
7,0,1,2,3,6,4,5	40318
7,0,1,2,3,4,6,5	40319
7,0,1,2,3,4,5,6	40320

A ideia principal é utilizar uma tabela *hash* contendo duas colunas, chave e valor, onde na coluna chave teremos os valores sequenciais de 1 a 40.320 e na coluna valor armazenaremos todas as permutações de 8 elementos (Tabela 12). Desta forma, em vez de representar M_K como um vetor de 256 números, de 0 a 7, pode-se utilizar apenas os 32 índices correspondentes da codificação Lehmer. A Tabela 13 ilustra como ficaria os valores dos elementos de M_K dos

grupos 0 e 31 após o processo de compactação.

Tabela 13 – M_K Após Compactação Utilizando a Codificação Lehmer

Grupo	Código Lehmer
0	31552
31	7194

Algoritmo 9: (M_G,M_K) ← Fa(M_E)

```
   Entrada  : M_E
   Saída    : M_G, M_K
 1 início
 2     cria_vetor_de_bytes(Temp[256]);
 3     preenche_vetor(Temp[256], 0);
 4     p ← 0;
 5     para i = 0 to 255 faça
 6         M_G[i] ← MOD32(M_E[i]);
 7         p ← Temp[M_G[i]];
 8         M_K[M_G[i]][p] ← INT(M_E[i] ÷ 32);
 9         Temp[M_G[i]] ← p + 1;
10         se Temp[M_G[i]] > 7 então
11             Temp[M_G[i]] ← 0;
12         fim se
13     fim para
14     n ← 0;
15     para g = 0 to 31 faça
16         M_K[g] ← LehmerCodeEncrypt(M_K[g][8]);
17     fim para
18     Retorna (M_G,M_K)
19 fim
```

Dois algoritmos foram desenvolvidos para realizar a conversão dos valores das permutações do vetor $M_K(iOBJ)$ em índices da codificação Lehmer e vice-versa: numero-para-indice() e indice-para-numero(). O primeiro algoritmo recebe como entrada um permutação de 8 números de $M_G(iOBJ)$ e retorna um índice *key*, que é um número entre 0 e 40319. O segundo algoritmo recebe como entrada o valor do índice *key* e retorna a permutação de 8 números correspondente. Os algoritmos LehmerCodeEncrypt() e LehmerCodeDecrypt() desenvolvidos são apresentados no Apêndice D.

Vale destacar ainda que, a partir dos valores dos códigos Lehmer, exemplificados na Tabela 13, é possível computar facilmente os 8 elementos do vetor M_K. Adicionalmente, dados M_K e M_G é possível computar rapidamente o vetor M_E. Os algoritmos que descrevem essas transformações são apresentados e discutidos em detalhes na Seção 6.4.4.

6.4.2.6 *Etapa 6: Geração dos Arquivos de Saída*

Nesta etapa, os três arquivos, **Fq.bin**, **Fs.bin** e **Fm.bin**, que representam as características da Qualidade, Quantidade e Medida do arquivo de entrada **F**, são criados. Para criar o arquivo **Fq.bin**, os elementos dos vetores Q_E de todos os iOBJs que formam o arquivo **F** = $\{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$ são concatenados. Para criar o arquivo **Fs.bin**, os elementos dos vetores S_E de todos os iOBJs que formam o arquivo **F** = $\{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$ são concatenados. Para criar o arquivo **Fm.bin**, realiza-se a operação de ou-exclusivo (*XOR*) entre ($M_K \oplus Q_E \oplus S_E$) e depois concatena-se os vetores M_G com M_K para todos os iOBJs do arquivo **F** = $\{iOBJ_1, iOBJ_2, \cdots, iOBJ_n\}$. É importante registrar que essas operações *XOR* são necessárias para esconder as informações dos grupos da Medida (M_K) e prevenir a recomposição do vetor M_E. A Figura 23 apresenta o detalhamento da operação de ou-exclusivo entre os elementos de M_K, Q_E e S_E.

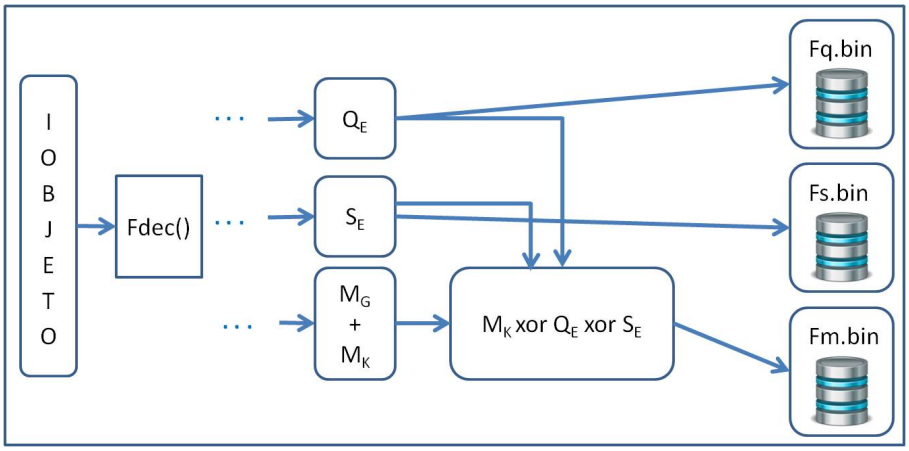

Figura 23 – Detalhe da Operação de ou-exclusivo para Anonimização da Medida

6.4.3 *Fase 3: Dispersão*

A fase de Dispersão transfere os arquivos **Fq.bin**, **Fs.bin** e **Fm.bin**, os quais representam, respectivamente, as características anonimizadas da Qualidade (Q), da Quantidade (S) e da Medida (M) dos objetos de informação que compõem o arquivo **F** para provedores de armazenamento na nuvem. A segurança desta técnica depende do isolamento dos dados de Q, S e M entre si. Na nuvem existem várias formas de atender a esta condição. Por exemplo, o uso de nuvens distintas para armazenamento dos arquivos, a utilização de sistemas de autenticação e autorização distintos para cada servidor onde os arquivos forem armazenados, ou ainda o uso

de chaves de identificação distintas para os arquivos, o quê, dependendo do volume de dados armazenados nos servidores, dificultaria bastante a tarefa de um atacante de reconstruir o arquivo **F**, mesmo que este consiga reunir os 3 arquivos.

Para acrescentar maior disponibilidade e integridade aos dados de Q, S e M armazenados nos provedores de nuvem, propõe-se utilizar o algoritmo denominado Rabin *Information Dispersal Algorithm* (IDA) (RABIN, 1989a), o qual possibilita dispersar os arquivos em **n** blocos distintos, de forma que **(m<n)** partes são suficientes para reconstruir o arquivo, mas **(m-1)** partes não possibilitam a sua reconstrução.

O processo de dispersão utilizando o algoritmo de Rabin é realizado de acordo com as seguintes etapas:

1. Os arquivos das características dos objetos de informação são vistos como uma sequência de **S** símbolos, onde **S** = sequência de valores decimais dos bytes do arquivo mod **p**, onde **p** é um número primo maior que 256. Os arquivos são fragmentados em **S/m** blocos de **m** bytes.
2. Para cada bloco, deriva-se **n** símbolos codificados cada um por uma combinação linear de **m** símbolos.
3. Derivar n pacotes, com o i-ésimo pacote contendo o i-ésimo byte derivado para cada uma das **N / m** partes (Figura 24).

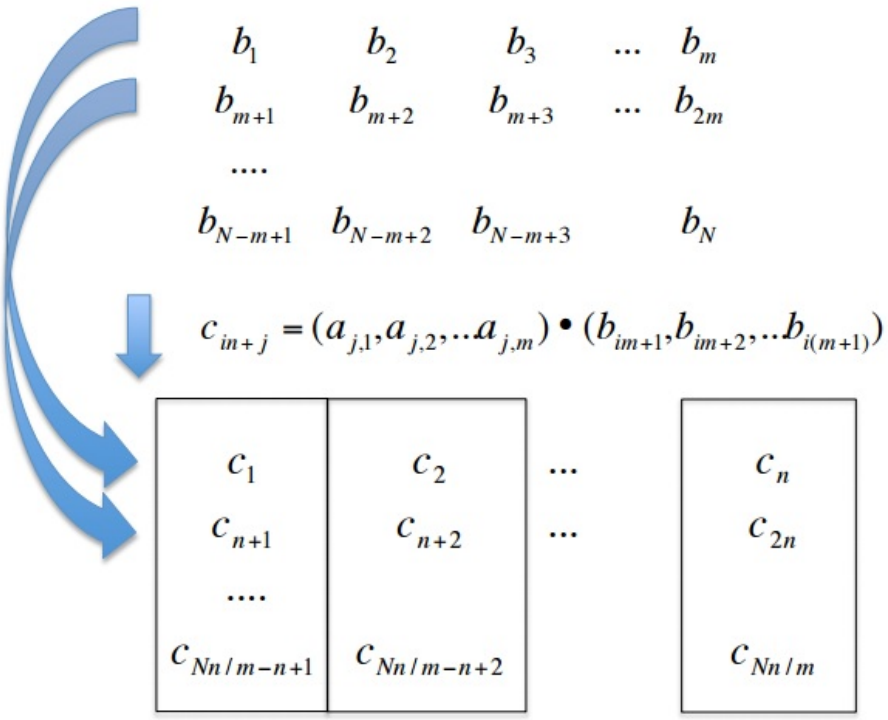

Figura 24 – Codificação Rabin

4. Dados **m** blocos, com **S** símbolos, pode-se inverter uma matriz para obter os arquivos originais da Qualidade, Quantidade e Medida (Figura 25).

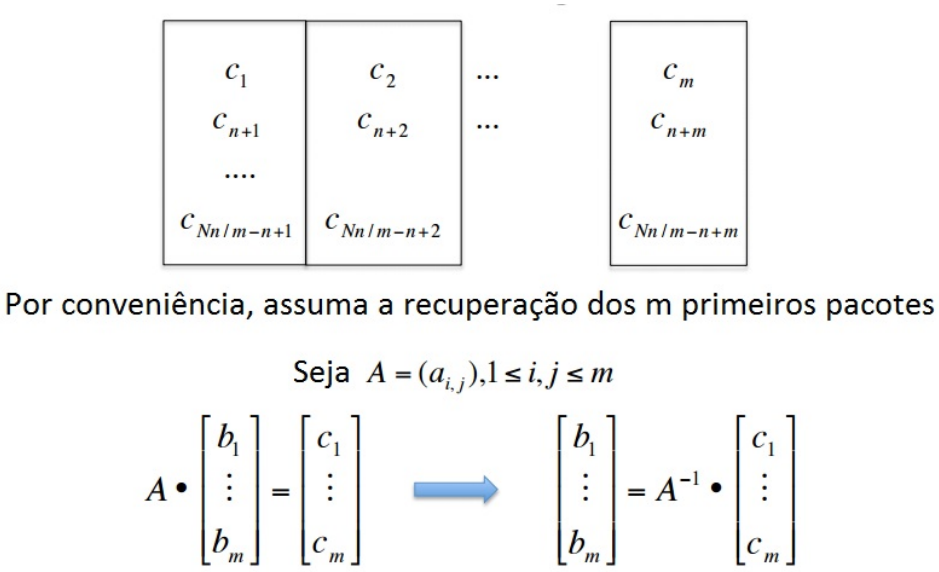

Figura 25 – Codificação Rabin

Com o uso da técnica de Rabin, consegue-se proteção adicional contra atacantes externos ao provedor, que teriam que invadir pelo menos **m** máquinas para conseguirem obter o arquivo referente à Medida, Qualidade ou Quantidade armazenado no provedor. Além disto, os arquivos armazenados na nuvem poderiam ser recuperados mesmo em caso de falha ou corrupção de arquivos intencional ou não intencional de **(n-m)** máquinas.

Os 3 arquivos decompostos podem ser armazenadas na nuvem de várias formas, considerando as esferas de influência dos atores envolvidos. Por exemplo, a Tabela 14 apresenta algumas opções de armazenamento, considerando o usuário do dado e 3 provedores de nuvem (CSP1, CSP2 e CSP3) como possíveis atores envolvidos. Os algoritmos utilizados nos experimentos deste trabalho foram criados considerando o cenário 3 da Tabela 14, onde os arquivos das características de Qualidade, Quantidade e Medida são armazenados em 3 nuvens distintas.

6.4.4 Processo de Reconstrução do Objetos de Informação

As etapas do processo de reconstrução dos objetos de informação armazenados na nuvem, a partir de suas características, ocorrem de forma reversa às operações de criação e armazenamentos dos mesmos. Todos os algoritmos da estratégia QSM-EXTRACTION, mostrados na Figura 26, executam em tempo O(n). Estes algoritmos estão descritos no Apêndice E. A seguir, são descritas as etapas desse processo:

Tabela 14 – Formas de Armazenamento de Q, S e M na Nuvem

Cenários	Proprietário do dado	Provedor de Nuvem	Grau de Confidencialidade
1	Não armazena nada	Armazena Q, S e M em CSP_1	Baixo: CSP_1 pode reconstruir os iobjetos a partir de Q, S e M
2	Não armazena nada	Armazena Q e S em CSP_1 e M em CSP_2	Médio: CSP_1 e CSP_2 podem reconstruir os iobjetos se cooperarem
3	Não armazena nada	Armazena Q em CSP_1, S em CSP_2 e M em CSP_3	Alto: CSP_1, CSP_2 e CSP_3 podem reconstruir os iobjetos se cooperarem
4	Armazena Q e S	Armazena M em CSP_1	Muito Alto: CSP_1 não consegue reconstruir os iobjetos
5	Armazena Q ou S	Armazena M em CSP_1 e (Q ou S) em CSP_2	Muito Alto: CSP_1 e CSP_2 não conseguem reconstruir os iobjetos mesmo se cooperarem

1. Recuperação do arquivo **Fq.bin** armazenado no provedor 1;
2. Recuperação do arquivo **Fs.bin** armazenado no provedor 2;
3. Recuperação do arquivo **Fm.bin** armazenado no provedor 3;
4. Descompactação das informações da medida contidas no arquivo **Fm.bin**;
5. Deanonimização da Medida;
6. Deanonimização da Qualidade;
7. Deanonimização da Quantidade;
8. Recomposição dos objetos de informação utilizando as informações da Qualidade, Quantidade e Medida;
9. Recomposição do arquivo original **F** por meio da concatenação dos objetos de informação.

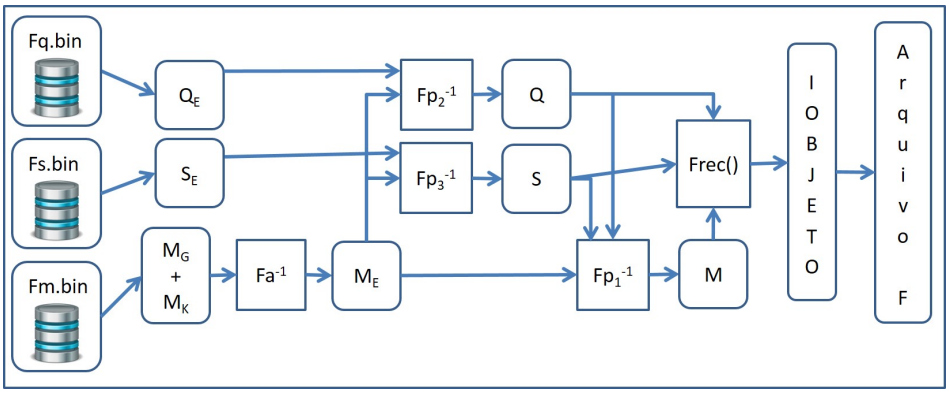

Figura 26 – Visão Geral do Processo de Reconstrução do Arquivo **F** Original

As três primeiras etapas do processo podem ocorrer de forma sequencial ou em paralelo, porém a quarta etapa só inicia após o recebimento dos três arquivos: **Fq.bin**, **Fs.bin**

e **Fm.bin**. O tamanho do arquivo **Fm.bin** equivale a 87,5% do tamanho do arquivo original **F**, enquanto os arquivos **Fq.bin** e **Fs.bin** equivalem a 12,5% do tamanho de **F**.

Os arquivos **Fq.bin** e **Fs.bin** são lidos de forma sequencial e processados em fragmentos de 256 bits. O arquivo **Fm.bin** é lido de forma sequencial e processado em fragmentos de 224 *bytes*, sendo 160 *bytes* correspondentes ao vetor M_G e 64 *bytes* referentes ao vetor M_K. As seguintes operações são realizadas:

1. A operação *XOR* entre os elementos de M_K, Q_E e S_E é realizada para descriptografar o valor de M_K: $M_K \oplus Q_E \oplus S_E$.
2. A função inversa Fa^{-1} recebe como entrada os elementos de M_K que são convertidos de códigos Lehmer para conjuntos de 8 elementos.
3. A função inversa Fa^{-1} recompõe o vetor M_E a partir dos vetores M_G e M_K.
4. A função inversa Fp_2^{-1} recebe como entrada M_E e Q_E e produz como saída o vetor Q_{BA}.
5. A função inversa Fp_3^{-1} recebe como entrada M_E e S_E e produz como saída o vetor S_{BA}.
6. A função Fp_1^{-1} utiliza os vetores Q_{BA} e S_{BA} para desembaralhar o vetor M_E, gerando M.
7. Por último, a função Frec utiliza as propriedade da Medida (M), Qualidade (Q_{BA}) e Quantidade(S_{BA}) para recriar o objeto de informação.

O Algoritmo 10 da função Fa^{-1}, que recupera as informações do vetor M_E, a partir dos vetores M_G e M_K, é apresentado a seguir:

Implementação de (M_E) $\leftarrow$ Fa^{-1}(M_G,M_K) **(Algoritmo 10)**.

Algoritmo 10: (M_E) $\leftarrow$ Fa^{-1}(M_G,M_K)

```
   Saída      : M_E
   Entrada    : M_G, M_K
 1 início
 2     cria_vetor_de_bytes(Temp[256]);
 3     preenche_vetor(Temp[256], 0);
 4     n ← 0;
 5     para g = 0 to 31 faça
 6         M_K[g][8] ← LehmerCodeDecrypt(M_K[g]);
 7     fim para
 8     p ← 0;
 9     para i = 0 to 255 faça
10         p ← Temp[M_G[i]];
11         M_E[i] ← (M_K[i][p] × 32) + M_G[i];
12         Temp[M_G[i]] ← p + 1;
13         se Temp[M_G[i]] > 7 então
14             Temp[M_G[i]] ← 0;
15         fim se
16     fim para
17     Retorna M_E
18 fim
```

A próxima etapa do processo de recuperação do arquivo **F** é a deanonimização dos vetores da Qualidade e da Quantidade. O algoritmo 11 deanonimiza o vetor Q_{BA}, a partir das informações da Medida anonimizada M_E e das informações da Qualidade anonimizada Q_E.

Implementação de $(Q_{BA}) \leftarrow Fp_2^{-1}(Q_E, M_E)$ **(Algoritmo 11)**.

Algoritmo 11: $(Q_{BA}) \leftarrow Fp_2^{-1}(Q_E, M_E)$

```
   Entrada     : Q_E, M_E
   Saída       : Q_BA
1  início
2      para i = 0 to 255 faça
3          Q_BA[i] ← Q_E[M_E[i]];
4      fim para
5      Retorna Q_BA;
6  fim
```

Em seguida as informações do vetor S_{BA} são recuperadas, utilizando-se o mesmo procedimento. O algoritmo 12 deanonimiza o vetor Q_{BA}, a partir das informações da Medida anonimizada M_E e das informações da Quantidade anonimizada S_E.

Implementação de $(S_{BA}) \leftarrow Fp_3^{-1}(S_E, M_E)$ **(Algoritmo 12)**.

Algoritmo 12: $(S_{BA}) \leftarrow Fp_3^{-1}(S_E, M_E)$

```
   Entrada     : S_E, M_E
   Saída       : S_BA
1  início
2      para i = 0 to 255 faça
3          S_BA[i] ← S_E[M_E[i]];
4      fim para
5      Retorna S_BA;
6  fim
```

A etapa seguinte deve recuperar as informações da Medida, a partir das informações da Qualidade e da Quantidade. Os vetores Q_{BA}, S_{BA} usados para embaralhar os elementos de M_{ALL} são agora utilizados novamente para a operação inversa de desembaralhamento. O Algoritmo 13 descreve o processo reverso.

Implementação de $(\mathrm{M}_{ALL}) \leftarrow \mathrm{Fp}_1^{-1}(\mathrm{M}_E, \mathrm{Q}, \mathrm{Q}_{BA}, \mathrm{S}_{BA})$ **(Algoritmo 13)**.

Algoritmo 13: $(\mathrm{M}_{ALL}) \leftarrow \mathrm{Fp}_1^{-1}(\mathrm{M}_E, \mathrm{Q}, \mathrm{Q}_{BA}, \mathrm{S}_{BA})$

```
   Entrada  : Q, Q_BA, S_BA, M_E
   Saída    : M_ALL
1  início
2      cria_vetor_de_bytes(Q_T[259]);
3      cria_vetor_de_bytes(M_E[256]);
4      n ← tamanho(Q);
5      para i = 0 to 255 faça
6          M_ALL[i] ← M_E[i];
7      fim para
8      cont ← 0;
9      para i = 0 to 255 faça
10         se Q_BA[i] = 1 então
11             Q[cont] ← i;
12             cont ← cont + 1;
13         fim se
14     fim para
15     para i = 0 to 259 faça
16         Q_T[i] ← Q[n];
17         n ← n − 1;
18         se n < 0 então
19             n ← tamanho(Q);
20         fim se
21     fim para
22     n ← 0;
23     para i = 255 to 0 faça
24         se Q_BA[i] = 0 então
25             n ← Q_T[i] ⊕ Q_T[i + 1];
26         fim se
27         senão
28             se S_BA[i] = 0 então
29                 n ← Q_T[i] ⊕ Q_T[i + 1] ⊕ Q_T[i + 2];
30             fim se
31             senão
32                 n ← Q_T[i] ⊕ Q_T[i + 1] ⊕ Q_T[i + 2] ⊕ Q_T[i + 3];
33             fim se
34         fim se
35         Temp ← M_ALL[i];
36         M_ALL[i] ← M_ALL[n];
37         M_ALL[n] ← Temp;
38     fim para
39     Retorna M_ALL(iOBJ)
40 fim
```

O Algoritmo 17, apresentado no Apêndice C, descreve as etapas do processo de

recuperação dos objetos de informação armazenados na nuvem. Uma versão simplificada da função Frec() é apresentada a seguir no Algoritmo 14:

Implementação de (iOBJ) ← Frec(M_{BA}, Q, S_{BA}) **(Algoritmo 14)**. O Algoritmo recebe como entrada as informações da Medida, Qualidade e Quantidade dos objetos de informação do arquivo original **F**. Os arquivos são processados sequencialmente. Cada objeto de informação é remontado byte a byte pelo Algoritmo 14, na ordem do menor para o maior byte. As informações sobre a existência de um determinado byte no objeto de informação (qualidade) e a extensão desse byte (quantidade) guiam o processo de montagem do objeto de informação nas posições (medida) corretas. O Algoritmo percorre o vetor da Qualidade Q_{BA} da posição 0 até a posição 255, identificando os bytes existentes no objeto de informação e se ocupam uma ou várias posições dentro dele. Caso haja mais de uma ocorrência de um determinado byte *a* no objeto de informação, todas as posições desse byte são processadas (linhas 11 a 15). Caso o byte *a* apenas ocorra uma única vez, o mesmo é armazenado na posição original dentro do objeto de informação (linha 8). No final, o Algoritmo 14 retorna os objetos de informação que serão juntados para formar o arquivo original **F**.

Algoritmo 14: (iOBJ) ← Frec(M_{BA}, Q, S_{BA})

```
   Entrada  : M, Q_BA, S_BA
   Saída    : iOBJ
 1 início
 2     i = 0;
 3     n = 0 ;
 4     pos = 0;
 5     para a = 0 to 255 faça
 6         se Q_BA[a] ≠ 0 então
 7             pos ← M[i];
 8             iOBJ[pos] ← CodASCII(Q_BA) ;
 9             i ← i + 1;
10             se S_BA[n] = 1 então
11                 Enquanto M[i] ⩾ pos AND i ⩽ 254 faça
12                     pos ← M[i];
13                     iOBJ[pos] ← CodASCII(Q_BA) ;
14                     i ← i + 1;
15                 fim-enquanto
16                 pos ← M[i];
17                 iOBJ[pos] ← CodASCII(Q_BA) ;
18                 n ← n + 1;
19             fim se
20             n ← n + 1;
21         fim se
22     fim para
23     Retorna iOBJ
24 fim
```

As informações do arquivo original **F** são recuperadas sem perda, por meio do

processamento das suas características de Qualidade, Quantidade e Medida. Nesta estratégia, ao invés de armazenar o arquivo **F** na nuvem, o quê se armazena são as suas características.

6.5 Segurança da Estratégia QSM-EXTRACTION

A estratégia QSM-EXTRACTION armazena as características anonimizadas do arquivo original **F** (**Fq.bin**, **Fs.bin** e **Fm.bin**) em servidores de nuvem, que possuem domínios fisicamente e administrativamente diferentes. De acordo com (RESCH; PLANK, 2011), a dispersão física de arquivos em servidores de armazenamento, feita com a escolha cuidadosa do número de servidores e quantidade de fragmentos necessários para recompor o arquivo, reduz as chances de um atacante obter as partes necessárias para restaurar o arquivo e é suficiente para manter o sistema seguro.

Para analisar o nível de segurança da estratégia do QSM-EXTRACTION, será considerado um cenário hipotético em que o arquivo Fq.bin está armazenado em um provedor de nuvem CSP_1. Fs.bin está armazenado em outro provedor CSP_2 e Fm.bin está armazenado em um outro provedor diferente dos dois primeiros CSP_3. Em seguida, será analisado o esforço que um atacante interno terá para restaurar o arquivo original **F** a partir de um dos arquivos **Fq.bin**, **Fs.bin** ou **Fm.bin**. Neste caso, pressupõe-se que o atacante possui acesso irrestrito ao servidor onde um dos arquivos está armazenado e pode recuperar um desses arquivos.

6.5.1 Modelagem do Adversário

O modelo de adversário proposto é aquele que considera que o proprietário do dado **P** é honesto, o cliente **C** pode ser malicioso e o provedor de serviço de nuvem **S** é “honesto-curioso"e não conspira com outros clientes (TANG *et al.*, 2016). Especificamente, o provedor de nuvem age como “honesto"executando corretamente os protocolos, mas é “curioso"no sentido de que tem interesse em obter informações adicionais sobre os dados armazenados (incluindo índices) e fluxo de mensagens recebidas durante o protocolo. Estas premissas são comuns na literatura (ver (JUNG *et al.*, 2015), (NING *et al.*, 2014)) e são bem justificadas em um ambiente de nuvem, uma vez que é de grande importância para prestadores de serviços manter uma boa aceitação de seus serviços, o quê os desencoraja a agir com má conduta, ao mesmo tempo em que eles podem ter interesse em coletar passivamente informações sensíveis, tendo em vista o valor econômico dos dados pessoais.

6.5.2 Confidencialidade da Qualidade

De acordo com Shannon, um sistema secreto é definido abstratamente como um conjunto de transformações de um espaço (o conjunto de mensagens possíveis) para um segundo espaço (o conjunto de possíveis criptogramas), por criptografia com um chave específica. As transformações devem ser reversíveis (SHANNON, 1949).

Nesse caso, um sistema secreto é um conjunto de transformações de um conjunto de elementos em outro, em que há uma organização natural de operações que produzem um terceiro sistema a partir dos dois primeiros. A relação entre estas operações é chamada de "operação produto"e corresponde à encriptação da mensagem com o primeiro sistema secreto **R** e encriptar o criptograma resultante com o segundo sistema secreto **S**, onde as chaves de **R** e **S** são escolhidas de forma aleatória e independente. Essa operação total é um sistema secreto, no qual as transformações representam todos os produtos de transformações de **S** em transformações de **R**. O espaço probabilístico no final é o produto das probabilidades das duas transformações. Está comprovado que um sistema secreto com estas operações combinadas forma essencialmente um sistema algébrico linear associativo com um elemento unitário. Esta é uma variedade de sistema algébrico que tem sido extensivamente estudada pelos matemáticos (SHANNON, 1949).

As operações realizadas que aumentam a confidencialidade da estratégia QSM-EXTRACTION, relacionadas às informações da Qualidade, são as seguintes:

1. Extrair e ordenar em ordem crescente os *bytes* diversos do objeto de informação e representar essas informações utilizando um vetor de 256 bits (sistema R_1).
2. Permutar os bits do vetor da Qualidade (Q_{BA}) utilizando como chave de permutação os elementos da Medida M_E (sistema S_1).

O resultado dessas operações é T_1 que representa uma operação produto: $T_1 = R_1 S_1$. R_1 tem chaves de 256-bits, que são escolhidas com probabilidade $p_1\ p_2\ p_3 ... p_{256}$.

S_1 possui chaves de 256-*bytes* que são escolhidas com probabilidade $p_1^{'}\ p_2^{'}\ p_3^{'} ... p_{256}^{'}$. Como cada *byte* pode estar ou não presente no iOBJ, a probabilidade de um atacante acertar se um *byte* existe no iOBJ é $p_i = 1/2$ para $1 \leqslant i \leqslant 256$.

Assim, como qualquer *byte* pode ocupar qualquer posição do 1º ao 256º no iOBJ, a probabilidade do atacante acertar qual é a posição do *byte* no iOBJ é $p_j^{'} = 1/n$, para $1 \leqslant j \leqslant 256$, onde **n** é o número de posições disponíveis em iOBJ.

$P(R_1)$ representa a probabilidade de um atacante que não possui o arquivo **Fq.bin** conseguir encontrar o vetor Q_{BA}. $P(S_1)$ representa a probabilidade de um atacante que possui o

arquivo **Fq.bin** conseguir encontrar o vetor Q_{BA}. Neste caso, o atacante conhece a quantidade de bits 1 e 0 de Q_E, mas como estes bits estão embaralhados, o valor real de Q_{BA} é uma das permutações com repetição das quantidades dos bits 0 e 1 de Q_E.

Os experimentos realizados demonstraram que no caso de iOBJs de arquivos compactados, a quantidade de bits 1 equivale aproximadamente a 2/3 do total de bits e a quantidade de bits 0 equivale aproximadamente a 1/3 do total de bits. Considerando a quantidade de bits 1 do vetor Q_E = 170 e a quantidade de bits 0 do vetor Q_E = 86, a probabilidade de um atacante encontrar o valor correto de Q_{BA} é aproximadamente $1/10^{69}$. Este cálculo refere-se apenas a um iOBJ:

$$P(R_1) = \prod_{i=1}^{256} p_i = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \cdots \times \frac{1}{2} = \frac{1}{2^{256}} \approx \frac{1}{10^{77}} = 10^{-77} \tag{6.1}$$

$$P(S_1) = \frac{1}{P_{256}^{170,86}} = \frac{1}{\frac{256!}{170! \times 86!}} = \frac{1}{\frac{10^{506}}{10^{437}}} \approx \frac{1}{2^{230}} \approx \frac{1}{10^{69}} = 10^{-69} \tag{6.2}$$

Apesar de $P(S_1) > P(R_1)$, conclui-se que o fato do provedor CSP_1 possuir o arquivo **Fq.bin** não melhora significativamente suas chances de descobrir o valor dos dados da característica Qualidade do iOBJ.

6.5.3 Confidencialidade da Quantidade

As operações que aumentam a confidencialidade da estratégia QSM-EXTRACTION, relacionadas às informações da Quantidade, são as seguintes:

1. Extrair a informação das frequências de ocorrência dos *bytes* diversos no iOBJ e representar estas informações utilizando um vetor de 256 bits (sistema R_2).
2. Permutar os elementos do vetor da Quantidade (S_{BA}), utilizando como vetor de ordenação os elementos da Medida M_E (sistema S_2).

   Isso produz como resultado uma operação T_2 que representa o produto $T_2 = R_2S_2$.

   R_2 tem chaves de **q**-bits, onde **q** é o tamanho de **Q**. Estas chaves são escolhidas com probabilidade p_1, p_2, p_3, ..., p_q. Onde q tem valor médio de 170 bits.

Os experimentos realizados demonstraram que no caso de iOBJs de arquivos compactados, a quantidade de bits que ocorrem apenas uma vez no iobjeto equivale, em média, a 50% dos bits de q. Assim sendo, pode-se considerar 85 bits 0 e 85 bits 1 nos 170 bits iniciais de (S_{BA}).

S_2 possui chaves de q-*bits* com probabilidades de ocorrência igual a p'_1, p'_2, p'_3,..., p'_q. Como a frequência de ocorrência dos *bytes* no iobjeto pode ser igual a 1 ou maior que 1, então p_i = 1/2 para $1 \leqslant i \leqslant q$.

$P(R_2)$ representa a probabilidade de um atacante que não possui o arquivo **Fs.bin**, e acerta uma estimativa da quantidade de bytes no iobjeto (exemplo: 170 bytes diversos), conseguir encontrar o vetor S_{BA}. $P(S_2)$ representa a probabilidade de um atacante que possui o arquivo **Fs.bin** conseguir encontrar o vetor S_{BA}, a partir do vetor S_E . Este cálculo refere-se apenas a um objeto de informação:

$$P(R_2) = \prod_{i=1}^{q} p_i = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \cdots \times \frac{1}{2} = \frac{1}{2^q} = \frac{1}{2^{170}} \tag{6.3}$$

$$P(S_2) = \frac{1}{P_{170}^{85,85}} = \frac{1}{\frac{170!}{85! \times 85!}} = \frac{1}{\frac{10^{306}}{10^{128} \times 10^{128}}} \approx \frac{1}{10^{49}} \approx \frac{1}{2^{165}} \tag{6.4}$$

Apesar de $P(S_2) > P(R_2)$, conclui-se que o fato do provedor CSP_2 possuir o arquivo Fs.bin não melhora significativamente suas chances de descobrir o valor dos dados da característica Quantidade do iOBJ.

6.5.4 Confidencialidade da Medida

Os dados permutados da Medida estão armazenados nos vetores M_G e M_K dentro do arquivo Fm.bin. O vetor M_K está criptografado por uma operação XOR com Q_E e S_E. Para o provedor CSP_3 obter a informação da Medida, ele teria que obter as informações da Qualidade e da Quantidade (Q_E e S_E), que estão armazenadas nos provedores CSP_1 e CSP_2 respectivamente.

As operações realizadas para aumentar a confidencialidade da estratégia QSM-EXTRACTION, relacionadas às informações da Medida, são as seguintes:

1. Extrair e ordenar em ordem crescente as informações das posições dos grupos de *bytes* da Qualidade do iOBJ (sistema R_3).
2. Permutar os elementos de M_{BA} utilizando Q_E e S_E como chaves de ordenação para gerar o vetor M_E (sistema S_3).
3. Anonimizar os elementos de M_E gerando os vetores M_G e M_K.
4. Criptografar os 64 *bytes* de M_G, por meio de operação XOR, com uma chave **K** formada pelos 32 *bytes* de Q_{BA} e 32 *bytes* de S_{BA} (sistema V_3).

Obtemos com esta operação T_3 que pode ser descrito como o produto $T_3 = R_3S_3V_3$.

R_3 e S_3 têm chaves de 256-*byte*s, que são escolhidas respectivamente com probabilidades $p_1\ p_2\ p_3...p_{256}$ e $p'_1\ p'_2\ p'_3...p'_{256}$

V_3 tem uma chave **K** com probabilidade $p_1 = 1/2^{512}$.

$P(R_3)$ representa a probabilidade de um atacante que não possui o arquivo **Fm.bin** conseguir encontrar o vetor M_{BA}. $P(S_3)$ x $P(V_3)$ representa a probabilidade de um atacante que possui o arquivo **Fm.bin** conseguir encontrar o vetor M_{BA}. Este cálculo refere-se apenas a um objeto de informação:

$P(R_3) = P(S_3) = 1/256! \approx 1/10^{506}$

$P(S_3)$ x $P(V_3) = 1/10^{506}$ X $1/2^{512} = \approx 1/10^{154}$ x $1/10^{506} \approx 1/10^{660}$

Como $P(S_3)$ x $P(V_3) < P(R_3)$, conclui-se que o fato do provedor CSP_3 possuir o arquivo Fm.bin não melhora suas chances de descobrir o valor dos dados da característica de Medida do objeto de informação.

O grau de confidencialidade da estratégia QSM-EXTRACTION é baseado na dificuldade do atacante reconstruir o arquivo original **F** a partir dos arquivos de suas características (**Fq.bin**, **Fs.bin** ou **Fm.bin**). Além disto, a estratégia proposta é flexível e pode ser utilizada sozinha ou em conjunto com outras abordagens (tais como os algoritmos de criptografia AES, DES ou 3-DES) para aumentar o nível de confidencialidade dos dados.

6.5.5 Integridade da Estratégia QSM-EXTRACTION

As características de Qualidade, Quantidade e Medida do objeto de informação possuem um conjunto de restrições que permitem detectar violações de integridade dos arquivos **Fq.bin**, **Fs.bin** e **Fm.bin**. Essas restrições são descritas a seguir:

1. Os elementos da Qualidade e da Quantidade são representadas por bits. Cada bit 1 da Qualidade está relacionado a um bit da Quantidade, que ao assumir o valor 0 está associado a um único byte na Medida ou ao assumir o valor 1 está associado a um conjunto de bytes na Medida. Uma alteração em um bit da Qualidade ou da Quantidade podem causar alterações na leitura dos elementos da Medida. Essas alterações podem ser detectadas na recomposição do objeto de informação, indicando erro nos arquivos **Fq.bin** ou **Fs.bin** ou em ambos.
2. Os bytes da Medida são todos diferentes entre si, isto permite detectar uma alteração do valor desde um único bit até uma quantidade maior de bits. Sendo que quanto maior for a

quantidade de bits alterados, maior a probabilidade de detecção de erro no valor de um elemento da Medida. A ocorrência de dois bytes iguais em um iobjeto da Medida indica erro de integridade no arquivo **Fm.bin**.

3. Os elementos da Medida são decompostos em dois vetores Q_G e Q_K. Existe uma relação entre os elementos destes dois vetores, sendo que uma alteração em um elemento de um vetor Q_G pode ser detectado, pois devem existir 32 elementos com 8 ocorrências cada um no vetor Q_G
4. As quantidades de bits 0 b(0) e 1 b(1) nos 160 bytes iniciais do objeto de informação gravados em **Fm.bin**, que são os elementos de Q_G, são duas constantes b(0) = 176x8 = 1408 e b(1) = 80 x 8 = 640. Valores diferentes de 640 bits 0 nos primeiros 160 bytes dos iobjetos de **Fm.bin** indica erro de integridade no arquivo **Fm.bin**.
5. Os 64 bytes finais do objeto de informação gravados em **Fm.bin** representam números que variam de 0 a 40319. Uma alteração nos bits desta parte do arquivo que produza números maiores do que 40319 indica erro de integridade no arquivo **Fm.bin**.
6. A quantidade de bits 1 no arquivo **Fq.bin** deve ser menor ou igual a quantidade de bits 1 do arquivo Fs.bin, o contrário indica erro de integridade no arquivo **Fq.bin**.

6.6 Considerações Gerais

Nesta seção, discute-se alguns aspectos e cenários relacionados ao funcionamento da estratégia QSM-EXTRACTION. Diferentes casos analisados, tais como: alterações realizadas pelo usuário no arquivo original; alterações efetuadas, de forma intencional ou não, pelo provedor nos arquivos que armazenam as informações da qualidade, quantidade e medida; dentre outros. Adicionalmente, examina-se diferentes questões relacionadas a integridade e disponibilidade.

Inicialmente, considere o cenário onde um usuário deseja realizar uma alteração em um arquivo que, anteriormente, foi armazenado na nuvem utilizando-se a estratégia QSM-EXTRACTION. Neste caso, será necessário, primeiramente, recuperar os arquivos **Fq.bin**, **Fs.bin** e **Fm.bin** (os quais armazenam, respectivamente, as características de qualidade, quantidade e medida). Ou seja, é preciso transferir os arquivos **Fq.bin**, **Fs.bin** e **Fm.bin** dos provedores de serviços em nuvem para o cliente ou para uma terceira parte confiável. Em seguida, a partir desses três arquivos, o arquivo original é recomposto e apresentado ao cliente, o qual poderá realizar as alterações desejadas. Após a realização dessas alterações, o arquivo alterado deve passar pelas etapas de fragmentação, decomposição e dispersão, o que irá gerar novamente os

arquivos **Fq.bin**, **Fs.bin** e **Fm.bin**, os quais serão enviados para os provedores, atualizando os arquivos anteriores.

Vale destacar que, neste caso, a estratégia QSM-EXTRACTION assemelha-se a abordagem de criptografia. Mais precisamente, para se atualizar um arquivo que foi, anteriormente, criptografado e armazenado na nuvem, é necessário, inicialmente, recuperar o arquivo criptografado (transferindo-o para o cliente ou para uma terceira parte confiável). Em seguida, deve-se descriptografar o arquivo, utilizando a chave adequada, para, em seguida, apresentar o arquivo ao usuário que deseja alterá-lo. Após a alteração, o arquivo é novamente criptografado e enviado para o provedor do serviço de armazenamento em nuvem (sobrepondo o arquivo criptografado anterior, por exemplo).

Em se tratando da abordagem da fragmentação, caso a alteração envolva atributos de um mesmo fragmento, será necessário apenas atualizar o fragmento. Porém, se a alteração envolve atributos de fragmentos diferentes, será necessário executar alguma estratégia de reescrita, com a finalidade de transformar esta atualização em um conjunto de atualizações, uma para cada fragmento envolvido na atualização original, ou ainda transferir os fragmentos envolvidos na atualização para o cliente ou para uma terceira parte confiável.

Ainda para este primeiro cenário, caso utilize-se a abordagem de fragmentação e criptografia, o processo de alteração dos dados armazenados na nuvem pelo cliente poderá envolver a transferência de dados (fragmentos) dos provedores para o cliente ou para uma terceira parte confiável, a descriptografia dos atributos criptografados e a reescrita da atualização desejada (caso envolva atributos de fragmentos diferentes).

No segundo cenário, considere o caso onde o conteúdo de um dos arquivos produzidos pela estratégia QSM-EXTRACTION, **Fq.bin**, **Fs.bin** ou **Fm.bin**, e armazenado na nuvem, foi alterado, de forma proposital, por um atacante, ou acidental, por falha de equipamento ou sistema.

Neste caso, o processo de recomposição dos objetos de informação que compõem o arquivo original utilizado no QSM-EXTRACTION irá detectar essa alteração, conforme discutido na Seção 6.5.5. Contudo, não será possível reconstruir o arquivo original. Por outro lado, vale destacar que uma das premissas assumidas por este trabalho é que os provedores de nuvem garantem a integridade e a disponibilidade dos dados armazenados. Normalmente a nuvem é uma arquitetura distribuída com várias máquinas virtuais (instâncias) em uma ou várias zonas de disponibilidade, que oferece serviços de aplicações distribuídas, balanceamento

de carga, replicação de dados e tolerância a falhas. Mesmo assim, caso a premissa da garantia de disponibilidade não seja atendida, a estratégia QSM-EXTRACTION pode ser utilizada em conjunto com algoritmos de dispersão de informações (*Information Dispersal Algorithm - IDA*) para adicionar esta funcionalidade aos dados armazenados na nuvem. Por exemplo, os arquivos **Fq.bin**, **Fs.bin** e **Fm.bin** poderiam ser fragmentados em 3 partes cada um (n = 3), de forma que 2 partes (m = 2) sejam suficientes para recuperação dos arquivos originais. Neste caso os fragmentos seriam dispersos para 3 nuvens distintas conforme esquema da Tabela 15. Caso uma das nuvens fique indisponível, os arquivos podem ser recompostos a partir dos fragmentos armazenados nas outras duas nuvens ativas. A mesma abordagem pode ser utilizada no caso de um dos provedores corromper, de forma proposital ou não, os arquivos sob sua responsabilidade. Desta forma, apesar desta falha, seria possível reconstruir o arquivo original.

Tabela 15 – Estratégia QSM-EXTRACTION com garantia de disponibilidade

Arquivos	Fq.bin	Fs.bin	Fm.bin
Fragmentos gerados	Fq1, Fq2, Fq3	Fs1, Fs2, Fs3	Fm1, Fm2, Fm3
Nuvens	CSP 1	CSP 2	CSP 3
Fragmentos dispersados nas nuvens	Fq1, Fs2, Fm3	Fq2, Fs3, Fm1	Fq3, Fs1, Fm2

Outro aspecto a ser considerado na avaliação de uma estratégia de confidencialidade de dados armazenados na nuvem é a relação entre segurança e utilidade dos dados. Neste trabalho, definimos como utilidade a facilidade com que o dado armazenado na nuvem pode ser utilizado para responder consultas. A criptografia homomórfica possibilita a realização de consultas simples sobre os dados criptografados, mas a maior parte dos criptosistemas não atribui nenhuma utilidade aos dados criptografados. A abordagem da fragmentação, por sua vez, permite consultas sobre os atributos de um mesmo fragmento. Porém, caso a consulta envolva atributos armazenados em diferentes fragmentos, esta deve ser executada no cliente ou em uma TPC, o que envolve a transferência dos fragmentos para a máquina onde será executada a consulta. Na abordagem mista, que combina a utilização da fragmentação e da criptografia, a utilidade dos dados armazenados é maior do que na criptografia e menor do que na fragmentação. A variação da utilidade depende da quantidade de atributos que são criptografados.

A estratégia QSM-EXTRACTION pressupõe que o provedor é honesto, porém curioso (semi-honesto). Neste contexto, não faz sentido guardar informações pessoais (tais

como, fotos, cartas ou senhas) sem algum tipo de proteção. A estratégia QSM-EXTRACTION proporciona um nível elevado de confidencialidade aos arquivos armazenados na nuvem. Apenas quando reunidos é que os arquivos da Qualidade, Quantidade e Medida podem ser utilizados para recompor o arquivo original. Por outro lado, a utilidade dos dados armazenados nos provedores é mínima. Contudo, esta propriedade pode ser interessante para cenários onde o cliente não confia inteiramente no provedor ou deseja armazenar dados sensíveis.

6.7 Conclusão

As vantagens da estratégia proposta neste trabalho sobre as demais técnicas voltadas para garantir a confidencialidade dos dados armazenados na nuvem, são as seguintes:

- A estratégia QSM-EXTRACTION permite a recuperação dos dados anonimizados, sem perda da informação.
- Garante alto grau de confidencialidade sem a utilização de chaves criptográficas.
- A estratégia QSM-EXTRACTION pode ser aplicada a qualquer formato de dado.
- Não há limitação máxima para o tamanho do arquivo a ser anonimizado.
- A solução proposta suporta expurgo de dados da nuvem, pois os arquivos disponibilizados em provedores distintos não revelam informações sobre os dados originais. Caso o usuário deixe a nuvem, os dados podem ser considerados automaticamente expurgados.
- Algoritmos que compõem a estratégia QSM-EXTRACTION executam em tempo linear O(n), com desempenho melhor do que algoritmos tradicionais de criptografia simétrica, tais como Triple-DES e AES, para recuperação do arquivo armazenado na nuvem.
- Em caso de arquivos de entrada no formato texto, o tamanho dos arquivos armazenados é menor ou igual ao tamanho do arquivo original.

As desvantagens da estratégia proposta neste trabalho em relação as técnicas de fragmentação e de criptografia são as seguintes:

- Aumento do tamanho do espaço necessário para armazenamento dos arquivos das características dos objetos de informação (**Fq.bin**, **Fs.bin** e **Fm.bin**) em 12,5% do tamanho original do arquivo **F**.
- A estratégia QSM-EXTRACTION foi projetada para ser aplicável apenas a dados pessoais em bancos de dados NoSQL, enquanto as demais estratégias podem ser aplicadas a qualquer tipo de dado e a bancos de dados SQL e NoSQL.
- Os arquivos pessoais devem ter tamanho de, no mínimo, 256 bytes, enquanto nas outras

abordagens, não há este limite inferior para o tamanho do arquivo a ser criptografado ou fragmentado.

O próximo capítulo trata de questões referentes as propostas experimentais para demonstrar a eficiência dos algoritmos da estratégia QSM-EXTRACTION.

7 AVALIAÇÃO EXPERIMENTAL

7.1 Introdução

Neste capítulo são apresentados os experimentos realizados com a finalidade de avaliar o desempenho da estratégia QSM-EXTRACTION. Inicialmente, os principais algoritmos de criptografia simétrica (DES, 3DES e AES) foram comparados com a estratégia proposta.

Para executar estes experimentos, foi utilizada uma infraestrutura de nuvem privada gerida pelo OpenStack (*Openstack Cloud Operating System*). A Figura 27 ilustra a arquitetura utilizada nos experimentos, a qual contém quatro máquinas virtuais, sendo um nó de processamento, chamado de Terceira Parte Confiável (TPC), e três nós de armazenamento de dados, denominados VM1 , VM2 e VM3. A Terceira Parte Confiável (TPC) atende as requisições de armazenamento e recuperação de arquivos enviadas pelo cliente (usuários e/ou aplicações), ou seja, executa as tarefas de decomposição e recomposição dos arquivos solicitados pelo cliente. Os nós de armazenamento de dados (VM1 , VM2 e VM3) emulam três diferentes provedores de serviços de nuvem. Assume-se que esses provedores são honestos, porém curiosos, isto é, executam corretamente os protocolos, mas têm interesse em obter informações extra sobre os dados armazenados.

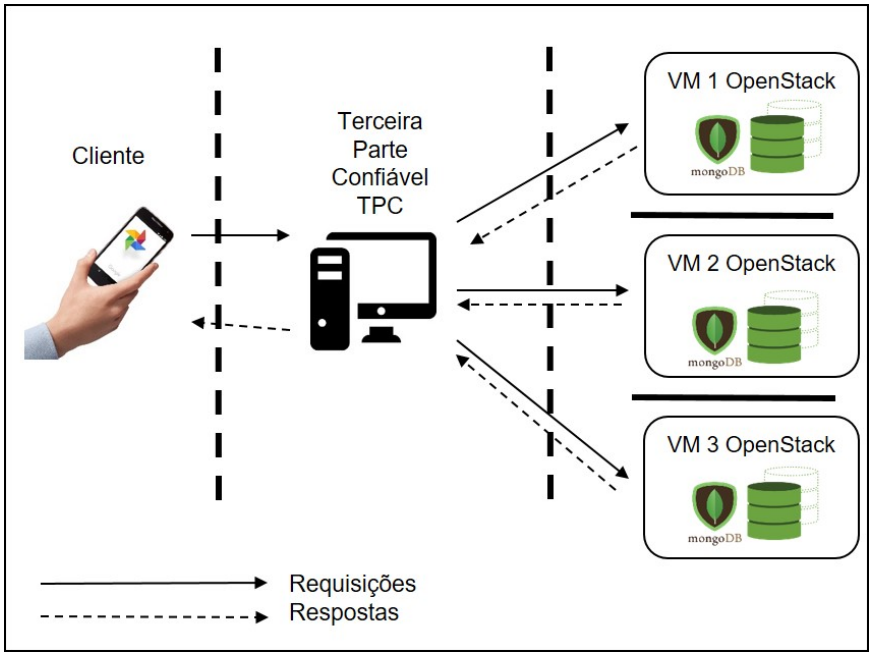

Figura 27 – Arquitetura dos Experimentos Realizados.

Como Terceira Parte Confiável (TPC) foi utilizada uma máquina virtual Windows Server 2008 com processador Intel Xeon E5-2407 CPU @ 2.20GHz, 4GB de RAM e 40GB

de armazenamento. Vale destacar ainda que, esta máquina virtual foi usada para executar os algoritmos de criptografia AES, DES e TRIPLE-DES, bem como os algoritmos da estratégia QSM-EXTRACTION. Cada um dos três nós de processamento (VM1, VM2 e VM3) tinha a seguinte configuração: Sistema Operacional Ubuntu 14.04, processador Intel Xeon E5-2407 CPU 2.20 GHz, 4 GB de RAM e 50 GB de armazenamento. Todas essas quatro máquinas virtuais estavam sob a gerência do OpenStack.

Neste cenário, um cliente que necessita armazenar um arquivo **F**, envia uma requisição para a TPC armazenar um arquivo nas nuvens, juntamente com o arquivo a ser armazenado. Em seguida, a TPC extrai as características de Qualidade, Quantidade e Medida dos objetos de informação que compõem o arquivo do cliente e as armazena em três nuvens administrativamente e fisicamente distintas. O processo de recuperação do arquivo por parte do cliente ocorre de forma inversa: O cliente envia uma solicitação de requisição do arquivo ao TPC que recupera os 3 arquivos contendo as características de Qualidade, Quantidade e Medida, reconstrói e envia o arquivo solicitado ao cliente.

Cada um dos nós de armazenamento (VM1 , VM2 e VM3) possui uma instância do MongoDB. Em cada uma dessas instâncias são armazenados os arquivos contendo, respectivamente, as características de Qualidade, Quantidade e Medida, dos arquivos enviados pelo cliente. O MongoDB é um banco de dados de código aberto, de alta performance, sem esquemas, orientado a documentos, escrito na linguagem de programação C++. Além de ser orientado a documentos, O MongoBD é formado por um conjunto de documentos JSON. Muitas aplicações podem, dessa forma, modelar dados de modo mais natural, pois estes podem ser aninhados em hierarquias complexas. O MongoDB foi escolhido por ser um dos bancos de dados mais utilizados em ambientes de computação em nuvem e por existirem oportunidades para melhoria da confidencialidade dos dados nele armazenados. O MongoDB suporta comunicação utilizando protocolo binário de baixo nível, mas não fornece um método para criptografar dados automaticamente. Isso significa que qualquer atacante com acesso ao sistema de arquivos pode extrair diretamente as informações dos arquivos (OKMAN *et al.*, 2011).

Para realizar a avaliação da eficiência da estratégia QSM-EXTRACTION, foi utilizado uma coleção de documentos criados sinteticamente, com diferentes tamanhos. Cada um dos arquivos gerados possui quatro partes (ou atributos), que possuem o mesmo tamanho. Estes atributos são: currículo vitae (A1), texto do artigo (A2), foto do autor (A3) e texto de avaliação do artigo (A4).

Os algoritmos que compõem a estratégia QSM-EXTRACTION foram implementados em linguagem C++ e são descritos a seguir (Tabela 16):

Tabela 16 – Algoritmos de QSM-EXTRACTION

Algoritmo	Entrada	Funcionalidade	Saída
Frag	Arquivo F	fragmentação do arquivo F em iobjetos	Coleção de iobjetos
Fdec	iobjeto	Extração das características do iobjeto	Qualidade, Quantidade e Medida
Fp_1	Qualidade, Quantidade, Medida	Permutação dos elementos da Medida	Medida anonimizada
Fp_2	Qualidade, Medida anonimizada	Permutação dos elementos da Qualidade	Qualidade anonimizada
Fp_3	Quantidade, Medida anonimizada	Permutação dos elementos da Quantidade	Quantidade anonimizada
Fa	Medida anonimizada	Compactação da Medida	Medida_Mod32, Resto_Mod32
Fa^{-1}	Medida Mod32, Resto_Mod32	Descompacta a Medida	Medida anonimizada
Fp_2^{-1}	Medida anonimizada, Qualidade anonimizada	Deanonimiza a Qualidade	Qualidade
Fp_3^{-1}	Medida anonimizada, Quantidade anonimizada	Deanonimiza a Quantidade	Quantidade
Fp_1^{-1}	Medida anonimizada, Qualidade, Quantidade	Deanonimização da Medida	Medida
Frec	Qualidade, Quantidade, Medida	Recomposição do iobjeto a partir de suas características	iobjeto

7.2 Resultados Experimentais

Com a finalidade de avaliar a eficiência da estratégia QSM-EXTRACTION, foram utilizadas duas métricas distintas: i) tempo de entrada e ii) tempo de saída. O tempo de entrada é definido como sendo o tempo total gasto para decompor um arquivo **F** e gerar os 3 arquivos contendo as características de qualidade (Fq.bin), quantidade (Fs.bin) e medida (Fm.bin) de **F**. Esses 3 arquivos serão enviados para os provedores de serviços de armazenamento na nuvem. O tempo de saída é definido como sendo o tempo total gasto para recompor o arquivo de entrada **F**, a partir dos três arquivos que contêm suas características (Fq.bin, Fs.bin e Fm.bin). É importante

destacar que o tempo gasto no processo de comunicação, para enviar e receber dados entre o cliente, a TPC e os nós de processamento (que simulam os provedores de nuvem) não compõe nem o tempo de entrada e nem o tempo de saída. Foram realizados testes com diferentes tamanhos de arquivos: 2^{18}, 2^{20} e 2^{22} bytes. Para cada tamanho de arquivo distinto, o processo foi repetido 10 vezes.

A estratégia QSM-EXTRACTION foi avaliada a partir de quatro cenários distintos, que serão apresentados a seguir. O primeiro cenário teve por finalidade analisar o desempenho dos algoritmos de criptografia simétrica mais conhecidos (DES, 3DES e AES). O segundo cenário compara o desempenho da estratégia QSM-EXTRACTION com as técnicas de criptografia, fragmentação e a combinação destas últimas. O terceiro cenário tem por objetivo analisar a utilização combinada da estratégia QSM-EXTRACTION com as abordagens de criptografia e fragmentação. Por fim, o quarto cenário foi utilizado para avaliar a combinação da criptografia com a estratégia QSM-EXTRACTION com a finalidade de aumentar a confidencialidade dos dados armazenados.

7.2.1 Cenário 1: Análise dos Algoritmos de Criptografia Simétrica

O primeiro cenário foi utilizado com o objetivo de comparar os algoritmos de criptografia simétrica mais populares: AES, DES e 3-DES. Esses algoritmos foram implementados em Java, utilizando a classe *Cryptix* como provedor dos métodos de criptografia e descriptografia. A classe *Cryptix* faz parte do JCE (*Java Cryptography Extension*) e pode prover os três algoritmos AES, DES e 3DES. Usando o Cryptix, em basicamente três operações:

1. Obtém a classe *Cipher* usando o método *getInstance* e passando o algoritmo selecionado.
2. Inicializa a classe *Cipher* para encriptação e decriptação usando o método *init* e passando a chave.
3. encripta ou decripta o dado

É importante ressaltar que, neste cenário, o Tempo de Entrada corresponde ao Tempo de Encriptação (o tempo gasto para criptografar um arquivo F) e o Tempo de Saída corresponde ao Tempo de Decriptação (tempo necessário para descriptografar um arquivo F).

Nesse cenário, a TPC recebe um arquivo **F** do cliente, criptografa-o, gerando um novo arquivo F_e e envia F_e para a VM1. A Figura 28 ilustra o tempo de criptografia para os algoritmos AES, DES e 3-DES. Em seguida, a TPC recebe o arquivo criptografado F_e da VM1 e o descriptografa, gerando o arquivo original F e envia F para o cliente. A Figura 29 ilustra o

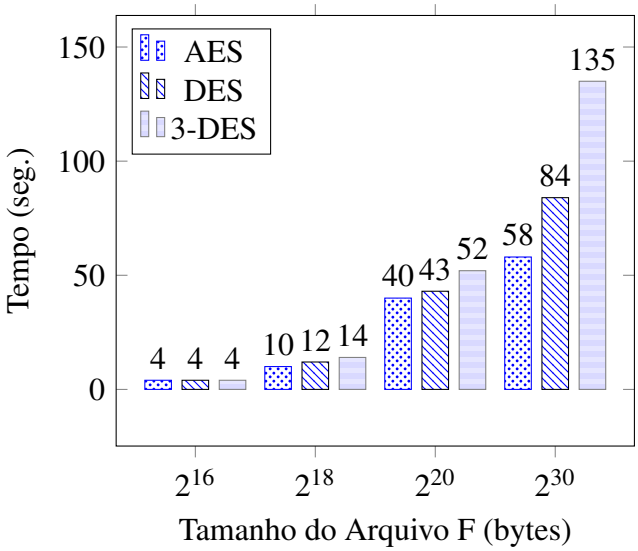

Figura 28 – Cenário 1: Tempo de Encriptação

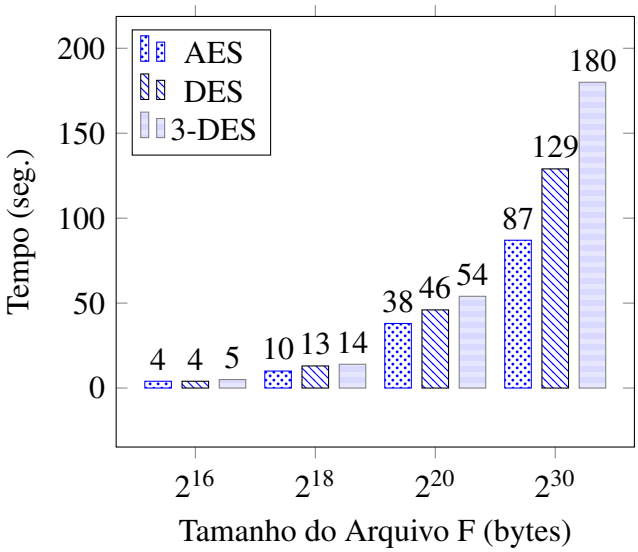

Figura 29 – Cenário 1: Tempo de Decriptação

tempo de desencriptação para os algoritmos AES, DES e 3-DES. Observe que, para arquivos com tamanho de 2^{16} bytes, esses três algoritmos apresentaram o mesmo tempo de criptografia (4 segundos), enquanto AES e DES apresentaram o mesmo tempo de descriptografia. No entanto, para arquivos com tamanhos de 2^{18}, 2^{20} e 2^{30} bytes, o AES supera o DES e o 3-DES, tanto para criptografia quanto para descriptografia. A análise do somatório do tempo das operações de encriptação e decriptação dos algoritmos, constata que o desempenho do AES supera o DES para os arquivos maiores do que 2^{16} bytes, conforme mostrado na Figura 30.

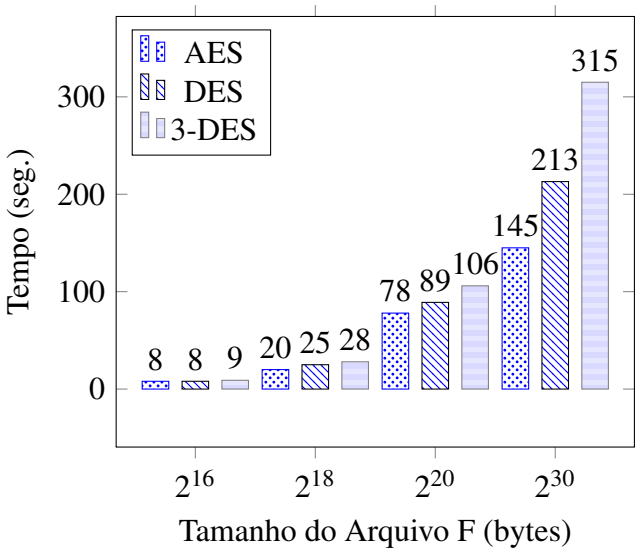

Figura 30 – Cenário 1: Tempo Total de Encriptação e Decriptação

7.2.2 Cenário 2: Análise das Principais Abordagens de Confidencialidade de Dados

O segundo cenário teve por finalidade possibilitar a comparação da estratégia QSM-EXTRACTION com as três principais abordagens para garantir a confidencialidade dos dados em ambientes de nuvem (SAMARATI; VIMERCATI, 2010; JOSEPH *et al.*, 2013):

a) criptografia de dados;

b) fragmentação;

c) combinação de criptografia e fragmentação.

Para executar as abordagens (b) fragmentação e (c) combinação de criptografia e fragmentação, foi necessário definir quais atributos são sensíveis, além de identificar as associações sensíveis entre atributos. Além disso, foi necessário separar os atributos com associação sensível em fragmentos diferentes, o quê é um problema NP-difícil (SAMARATI; VIMERCATI, 2010; JOSEPH *et al.*, 2013).

Nesse cenário, assume-se que cada documento tem quatro atributos: curriculum vitae (A1), texto do artigo (A2), foto do autor (A3) e avaliação do artigo (A4). Além disso, supõe-se que existe um conjunto C de restrições de associação sensíveis, as quais são apresentadas na Tabela 17.

Tabela 17 – Conjunto de Restrições entre Atributos Sensíveis

Conjunto	Atributos
C_1	{A1-curriculum vitae}
C_2	{A3-foto do autor}
C_3	{A2-texto do artigo, A4-avaliação do artigo}
C_4	{A1-curriculum vitae, A3-foto do autor}

Os atributos A_1 e A_3 são considerados sensíveis e devem ser criptografados na abordagem (c). A restrição C_3 = {A2, A4} indica que há uma associação sensível entre A2 e A4. A restrição C_4 = {A1, A3} indica que há uma associação sensível entre A1 e A3, e esses atributos devem estar em fragmentos distintos e serem armazenados em servidores diferentes na nuvem. Com base no conjunto *C* de restrições de confidencialidade, foi gerado um conjunto *P* de fragmentos de dados, os quais são descritos a seguir:

- a abordagem (b), fragmentação, formou os fragmentos P_1 = {A1}, P_2 = {A2, A3} e P_3 = {A4}.
- a abordagem (c), combinação de criptografia e fragmentação, produziu os fragmentos P_4={A1,A4} e P_5 = {A2,A3}; Estando A1 e A3 criptografados.

É importante enfatizar ainda que o tempo necessário para definir o conjunto de fragmentos (esquema de fragmentos) com a finalidade de dividir os atributos com associação sensível, o qual é um problema NP-difícil, não foi considerado neste experimento. Além disso, para a abordagem (a), criptografia de dados, foi utilizado o algoritmo AES, uma vez que este apresentou melhores resultados no primeiro cenário de testes.

Neste experimento, consideramos o desempenho com relação às seguintes métricas: (i) Tempo de Entrada e (ii) Tempo de Saída. Essas métricas mudam um pouco de acordo com a abordagem de confidencialidade de dados utilizada. A seguir são apresentadas as características de cada abordagem utilizada neste cenário.

O Tempo de Entrada é computado da seguinte forma:

- Abordagem (a), criptografia de dados: tempo para criptografar um arquivo **F** usando AES, gerando um arquivo F_e. O arquivo F_e será enviado para VM1. O tempo de transmissão do arquivo pela rede não é computado.
- Abordagem (b), fragmentação: o tempo para gerar P_1, P_2 e P_3. Onde, P_1 será enviado para VM1, P_2 para VM2 e P_3 para VM3. O tempo de transmissão dos fragmentos pela rede não é computado.
- Abordagem (c), combinação de criptografia e fragmentação: tempo para criptografar A_1 e A_3, mais o tempo para gerar P_4 e P_5 . P_4 será enviado para a VM1, enquanto P_5 será enviado para a VM2. O tempo de envio dos fragmentos para as máquinas virtuais não é computado.
- Estratégia QSM-EXTRACTION: tempo para decompor um arquivo F em $F_q.bin$, $F_s.bin$ e $Fm.bin$. Onde, $F_q.bin$ será enviado para VM1, $F_s.bin$ para VM2 e $F_m.bin$ para VM3;

O Tempo de Saída é computado da seguinte forma:

- Abordagem (a), criptografia de dados: tempo para descriptografar um arquivo F_e usando AES, gerando o arquivo original F;
- Abordagem (b), fragmentação: o tempo para juntar os fragmentos P_1, P_2 e P_3;
- Abordagem (c), combinação de criptografia e fragmentação: tempo para descriptografar A_1 e A_3, mais o tempo para juntar os fragmentos P_4 e P_5, remontando assim o arquivo F;
- Estratégia do QSM-EXTRACTION: tempo para reconstruir um arquivo F a partir dos três arquivos que contêm suas características, $F_q.bin$, $F_s.bin$ e $F_m.bin$.

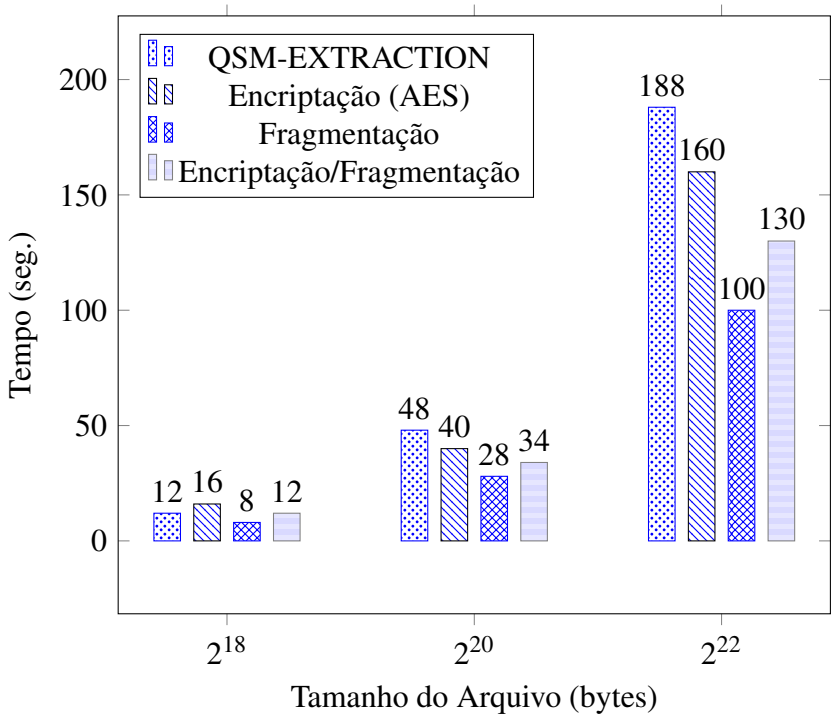

Figura 31 – Cenário 2: Tempo de Entrada

A Figura 31 ilustra o tempo de entrada para as quatro abordagens avaliadas. Observe que a estratégia QSM-EXTRACTION tem um desempenho ligeiramente inferior a criptografia de dados (AES). A abordagem de fragmentação supera as demais abordagens, para todos os tamanhos de arquivo. Por outro lado, as abordagens Fragmentação e Encriptação/Fragmentação precisam definir o conjunto de fragmentos (esquema de fragmentação) para separar os atributos com associação sensível. No entanto, como usamos um exemplo fixo, o tempo necessário para definir o esquema de fragmentação não foi computado. Em parte, isso explica os melhores resultados obtidos por essas duas abordagens.

A Figura 32 mostra o tempo de saída para as quatro abordagens avaliadas. Observe que a estratégia QSM-EXTRACTION supera todas as outras abordagens, para todos os tamanhos

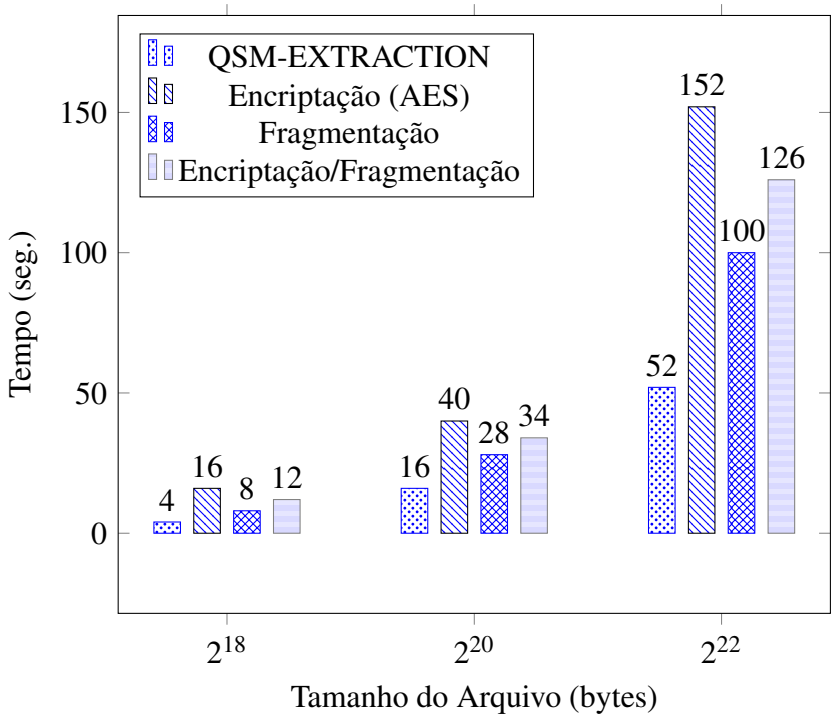

Figura 32 – Cenário 2: Tempo de Saída

de arquivo. É importante destacar que, para o tamanho de arquivo de 2^{22} bytes, a estratégia QSM-EXTRACTION é 88s mais lenta do que a abordagem de Fragmentação na fase de entrada, ou seja, para processar um arquivo **F** antes de enviá-lo para os provedores de serviços de armazenamento em nuvem. No entanto, para o mesmo tamanho de arquivo, a estratégia QSM-EXTRACTION é 48s mais rápida do que a abordagem de Fragmentação. Assim, para um ciclo completo de escrita e leitura, a estratégia QSM-EXTRACTION é apenas 40s mais lenta do que a abordagem de Fragmentação. Assim, se o usuário grava F uma vez e lê F duas vezes, a estratégia QSM-EXTRACTION é 8s mais rápida do que a abordagem de Fragmentação. Desta forma, a estratégia QSM-EXTRACTION supera todas as outras abordagens em cenários onde o número de leituras é pelo menos duas vezes maior do que o número de escritas, o que é esperado em bancos de dados reais e ambientes de armazenamento em nuvem.

A Figura 33 mostra o tempo de entrada e saída para as quatro abordagens avaliadas no Cenário 2. Observa-se que a estratégia QSM-EXTRACTION é mais rápida do que a abordagem de encriptação (AES) e Encriptação/Fragmentação para arquivos de tamanho 2^{18} e 2^{20}. Para arquivos de tamanho 2^{22}, a estratégia QSM-EXTRACTION continua sendo mais rápida do que a estratégia de encriptação (AES).

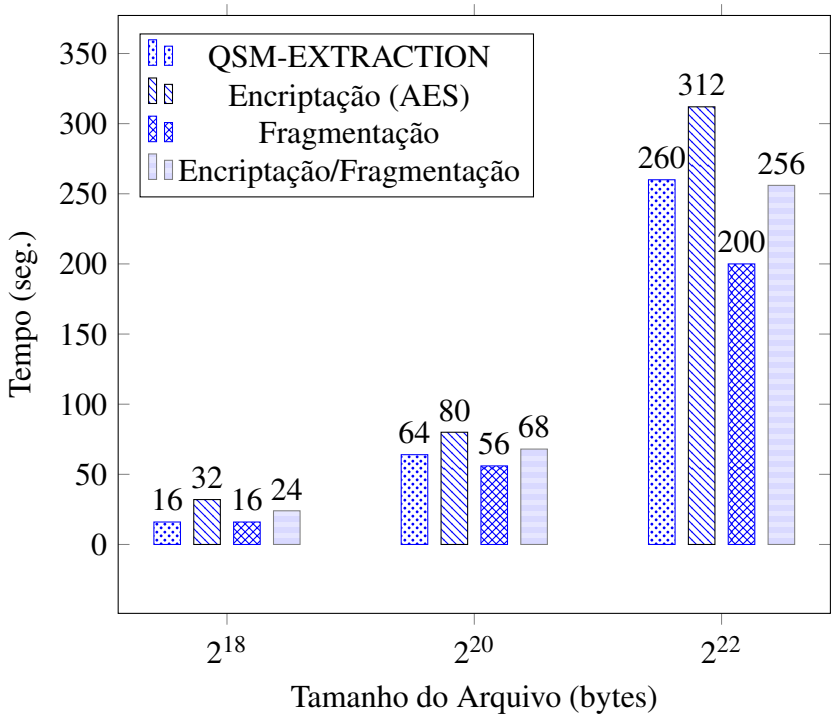

Figura 33 – Cenário 2: Tempo de Entrada e Saída

7.2.3 Cenário 3: Análise da Utilização da Estratégia QSM-EXTRACTION em Conjunto com Demais Abordagens

Neste terceiro cenário, avaliamos a utilização da estratégia QSM-EXTRACTION em conjunto com as demais abordagens para confidencialidade de dados: criptografia, fragmentação e criptografia/fragmentação. Neste sentido, vale destacar que a abordagem QSM-EXTRACTION é flexível e pode ser utilizada em conjunto com as demais abordagens de confidencialidade de dados, a fim de aumentar o nível de proteção dos dados.

A Figura 34 ilustra o tempo de entrada para a utilização da estratégia QSM-EXTRACTION em conjunto com as demais abordagens. A seguir será descrito como esse uso combinado pode ser realizado e qual a melhoria obtida na confidencialidade.

1. Encriptação + QSM-EXTRACTION: Os quatro atributos são criptografados, com a utilização do algoritmo AES. Em seguida as características de qualidade, quantidade e medida dos dados criptografados são extraídas e distribuídas para três provedores de armazenamento em nuvem. O aumento da confidencialidade ocorre pela dificuldade dos provedores reconstituírem os dados originais, mesmo com a posse da chave criptográfica.
2. Fragmentação + QSM-EXTRACTION: as características de qualidade, quantidade e medida dos fragmentos (P_1, P_2 e P_3) dos documentos são extraídas e enviadas para três provedores de armazenamento em nuvem. Neste caso, a confidencialidade aumenta, pois

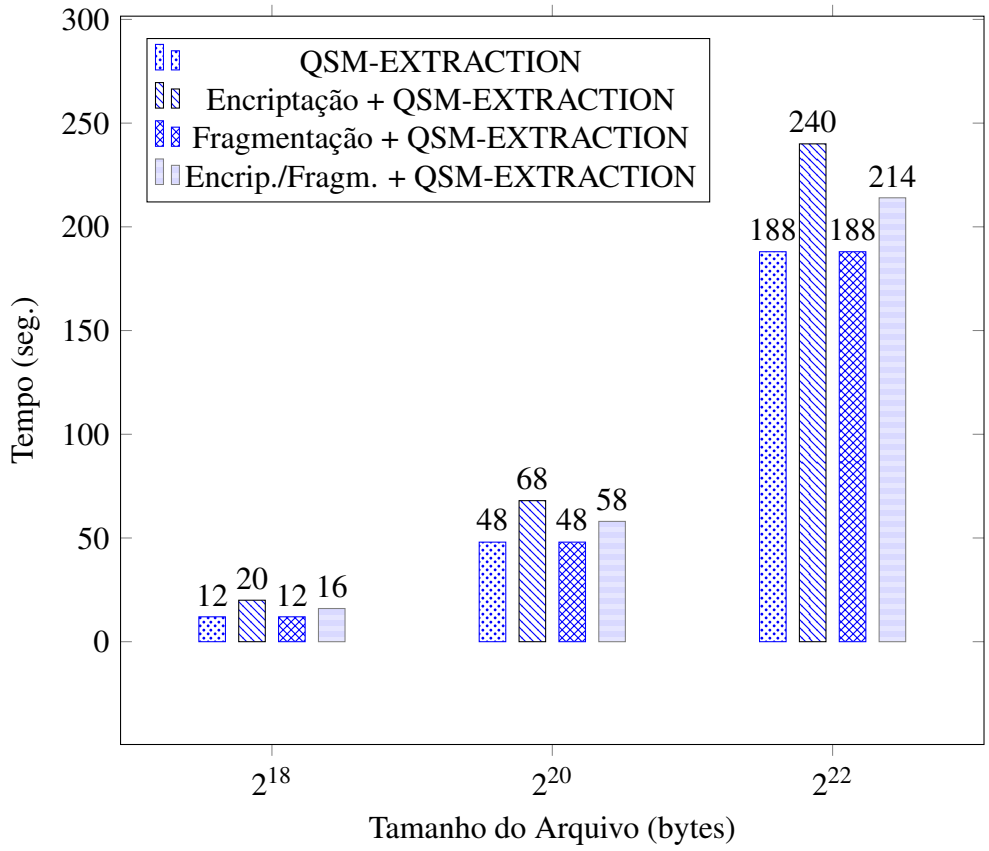

Figura 34 – Cenário 3: Tempo de Entrada

agora, os provedores não terão mais o conhecimento dos conteúdos dos fragmentos que armazenam.

3. Encriptação/Fragmentação + QSM-EXTRACTION: Os atributos sensíveis (A_1 e A_3 são criptografados, em seguida o processo de fragmentação (gerando os fragmentos P_4 e P_5), e por fim, executa-se a extração das características dos fragmentos. Neste caso, os provedores de armazenamento de dados não terão mais o conhecimentos dos fragmentos armazenados e não poderão descriptografar os atributos criptografados, mesmo se conhecerem a chave de criptografia.

Na Figura 34, a primeira coluna mostra o tempo de entrada para a estratégia QSM-EXTRACTION; a segunda coluna representa o tempo de entrada ao se aplicar a abordagem de criptografia de dados e, em seguida, a QSM-EXTRACTION; a terceira coluna indica o tempo de entrada ao aplicar a abordagem Fragmentação e, em seguida, a QSM-EXTRACTION; finalmente, a quarta coluna denota o tempo de entrada ao se aplicar a abordagem Encriptação/Fragmentação/QSM-EXTRACTION.

Na Figura 35, a primeira coluna mostra o tempo de saída para a estratégia QSM-EXTRACTION; a segunda coluna representa o tempo de saída ao se aplicar a abordagem de

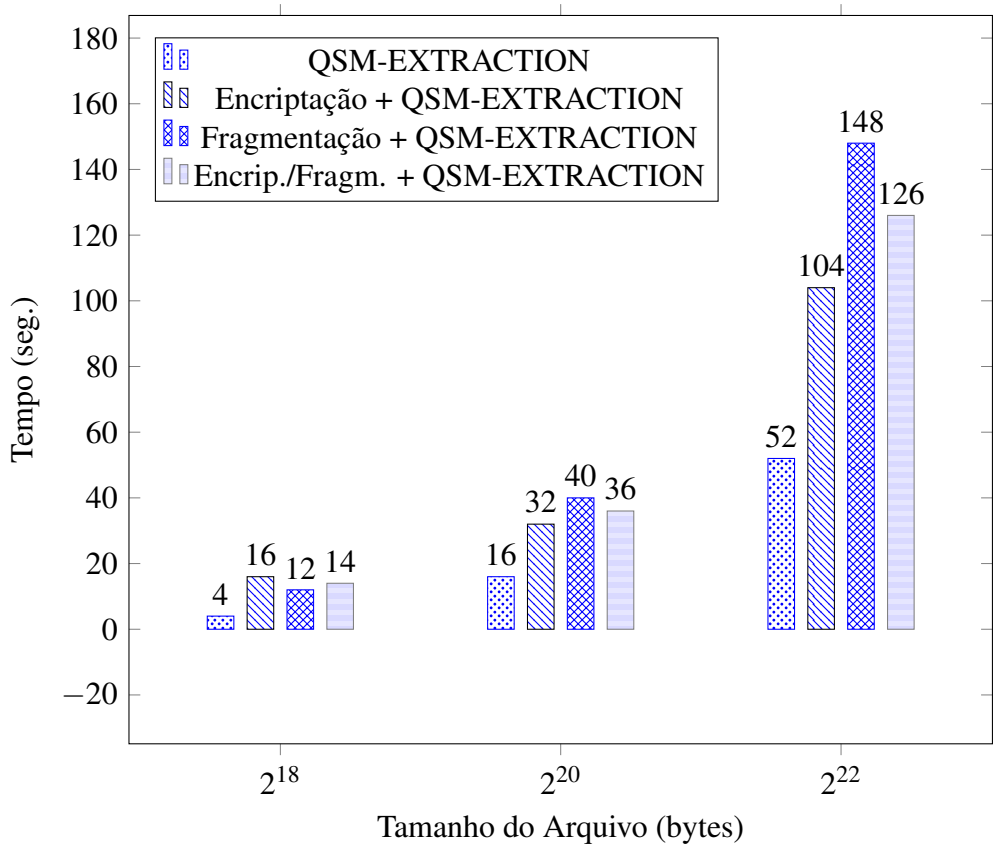

Figura 35 – Cenário 3: Tempo de Saída

criptografia de dados e, em seguida, a QSM-EXTRACTION; a terceira coluna indica o tempo de saída ao aplicar a abordagem Fragmentação e, em seguida, a QSM-EXTRACTION; finalmente, a quarta coluna denota o tempo de saída ao se aplicar a abordagem Encriptação/Fragmentação/QSM-EXTRACTION.

A análise dos tempos de entrada e saída, ilustrados nas Figuras 36 e 33, demonstra que o acréscimo de tempo devido a adição da estratégia QSM-EXTRACTION à abordagem de encriptação aumenta 13% (de 32 para 36 segundos) e 10% (de 312 para 344 segundos) para documentos de tamanho 2^{18} e 2^{22} respectivamente.

Desta forma, podemos argumentar que QSM-EXTRACTION é uma estratégia flexível, no sentido de que pode ser utilizada em conjunto com as abordagens previamente existentes, a fim de melhorar o nível de confidencialidade dos dados. Os resultados apresentados na Figura 34 mostram que esta estratégia proposta proporciona um pequeno aumento no tempo de entrada, ao mesmo tempo que proporciona um alto ganho na confidencialidade dos dados. A Figura 35 mostra que a sobrecarga de tempo de saída tem um comportamento semelhante ao tempo de entrada.

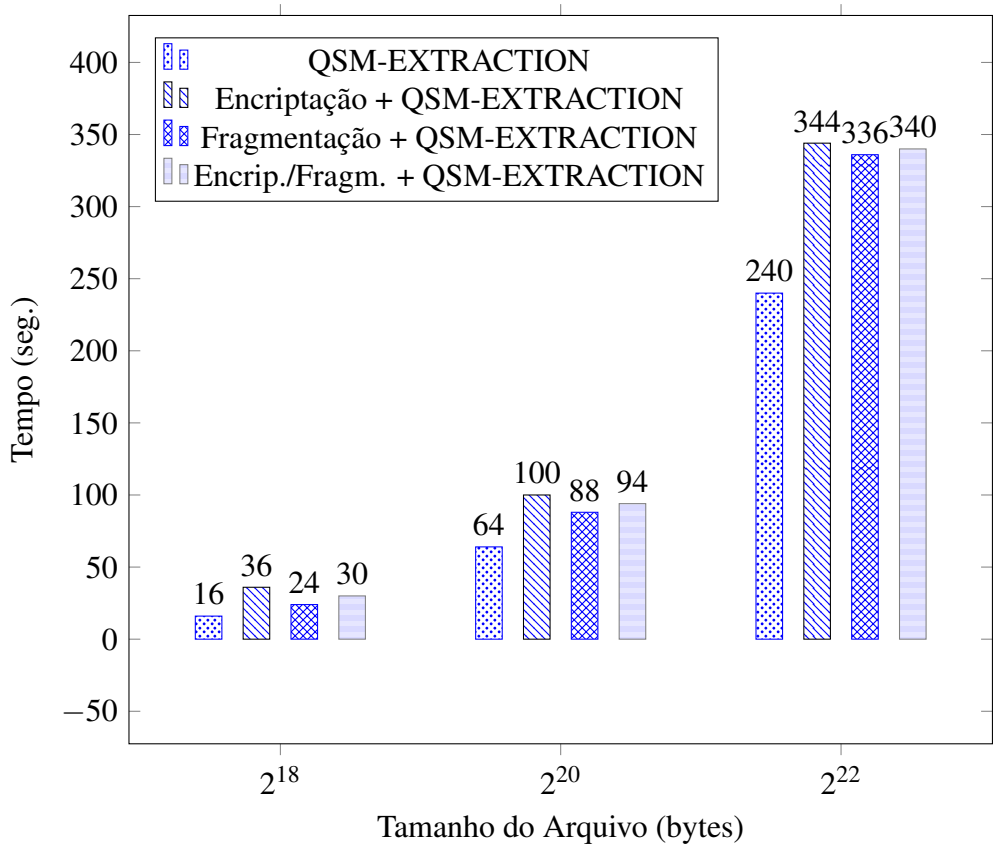

Figura 36 – Cenário 3: Tempo de Entrada e Saída

7.2.4 Cenário 4: Melhorando a Confidencialidade de Dados na Estratégia QSM-EXTRACTION

No quarto cenário, avaliamos 3 abordagens para melhorar a confidencialidade dos dados na estratégia QSM-EXTRACTION. A primeira abordagem é criptografar os arquivos $F_q.bin$ e $F_s.bin$, a segunda abordagem e a de criptografar apenas o arquivo $F_q.bin$ e a terceira abordagem é a de criptografar Apenas o arquivo $F_s.bin$. A Figura 37 compara o tempo de entrada da estratégia QSM-EXTRACTION com as 3 abordagens propostas. O uso de criptografia aplicada apenas aos arquivos de características que possuem tamanhos menores, no caso os arquivos $F_q.bin$ e $F_s.bin$, reduz a quantidade de dados que precisa ser criptografada para aumentar a confidencialidade. Criptografando apenas o arquivo $F_q.bin$, o acréscimo de tempo é de 33% para arquivos de 2^{18} bytes (de 12 para 16 segundos), 17% para arquivos de 2^{20} bytes(de 48 para 56 segundos) e 6% para arquivos de 2^{22} bytes (de 188 para 212 segundos).

As Figura 38 compara o tempo de saída da estratégia QSM-EXTRACTION com as 3 abordagens propostas. Observe que a estratégia de criptografar apenas o arquivo $F_q.bin$ fornece uma sobrecarga baixa (acréscimo de 64s para 76s para arquivos com 2^{20} *bytes*, conforme mostrado na Tabela 18), enquanto aumenta substancialmente o nível de confidencialidade de

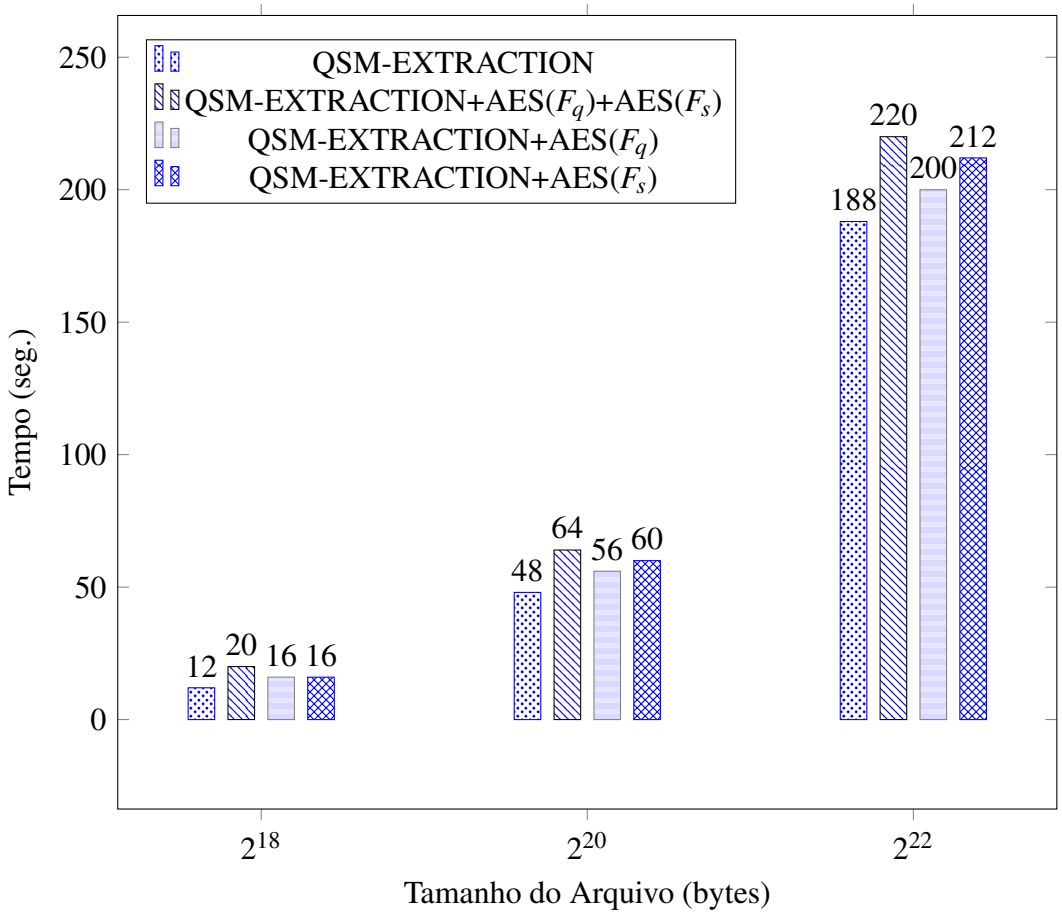

Figura 37 – Cenário 4: Tempo de Entrada

dados da estratégia QSM-EXTRACTION.

Tabela 18 – Comparação entre as Abordagens Utilizadas para Aumentar a Confidencialidade da Estratégia QSM-EXTRACTION (Considerando Arquivos de $2^{20} bytes$)

Estratégia	Tempo de Entrada	Tempo de Saída	Tempo de Entrada + Saída
QSM-EXTRACTION	48	16	**64**
QSM-EXTRACTION + AES(F_q) + AES(F_s)	64	28	92
QSM-EXTRACTION + AES(F_q)	56	20	**76**
QSM-EXTRACTION + AES(F_s)	60	20	80

7.2.5 Considerações sobre Custo de Armazenamento

Na estratégia QSM-EXTRACTION, um arquivo F é decomposto em três arquivos (Fq.bin, Fs.bin e Fm.bin), os quais são dispersos utilizando-se três provedores de nuvem diferentes. Os resultados experimentais mostraram que o tamanho desses arquivos representa,

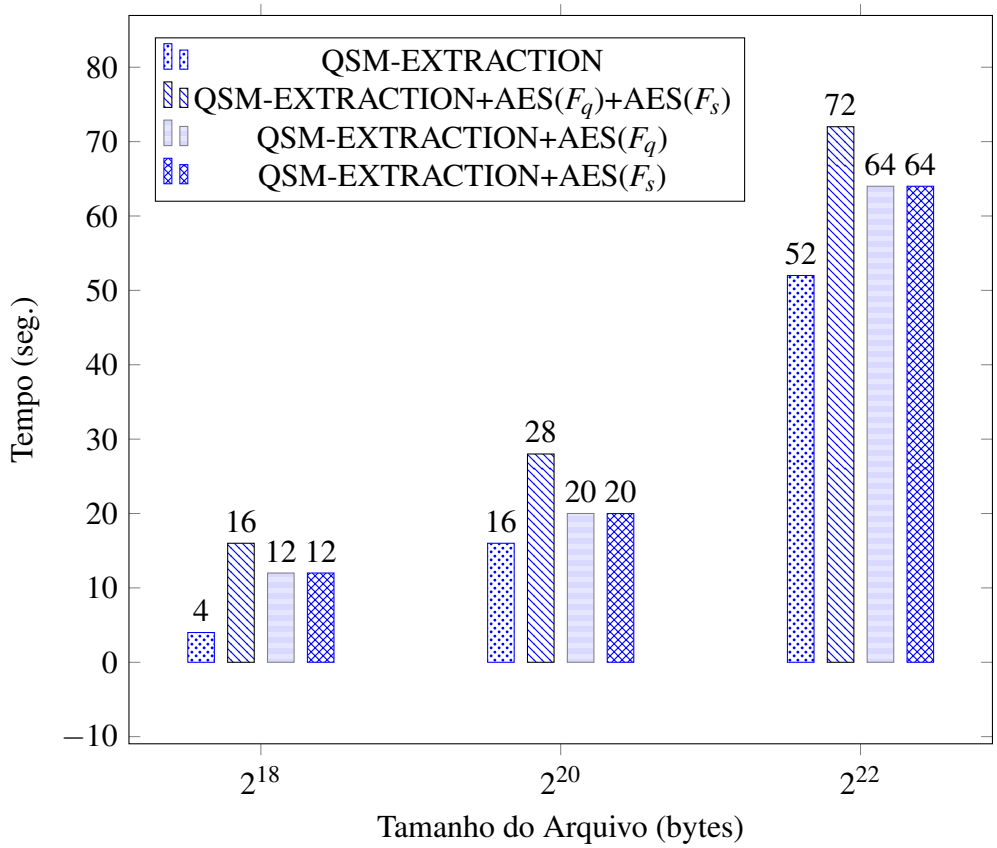

Figura 38 – Cenário 4: Tempo de Saída

respectivamente, 12,5 %, 12,5 % e 87,5 % do tamanho original do arquivo. Assim, adicionando-se esses valores, a estratégia proposta fornece uma sobrecarga de 12,5 % na utilização do espaço em disco. Esta desvantagem é minimizada uma vez que os serviços de armazenamento em nuvem são projetados para armazenar uma grande quantidade de dados.

7.2.6 Análise de Tráfego de Rede

Nos experimentos anteriores, desprezou-se o tempo gasto na comunicação entre a TPC e os três nós de armazenamento. Porém, como observado na sub-seção anterior, os tamanhos dos três arquivos produzidos pela estratégia QSM-EXTRACTION (Fq.bin, Fs.bin e Fm.bin), somados, proporciona um aumento de 12,5 % na utilização do espaço em disco, se compararmos ao tamanho do arquivo original. Por este motivo, tornou-se fundamental avaliar a sobrecarga proporcionada pela estratégia QSM-EXTRACTION sobre o tráfego na rede de comunicação. Neste sentido, um experimento específico foi realizado. Neste experimento foram utilizados arquivos de diferentes tamanhos: 1 GB, 10 GB, 20 GB, 40 GB e 80 GB.

A estratégia QSM-EXTRACTION foi comparada com o algoritmo AES, uma vez que este apresentou os melhores resultados entre os algoritmos de criptografia simétrica (AES,

DES e 3-DES). Para a estratégia QSM-EXTRACTION foi calculado o tempo gasto para enviar os três arquivos **Fq.bin**, **Fs.bin** e **Fm.bin** da TPC para os três nós de armazenamento (VM1, VM2 e VM3) e o tempo para receber de volta esses arquivos, utilizando em ambos os casos uma transmissão de dados em paralelo. Já para o algoritmo AES foi calculado o tempo gasto para enviar o arquivo criptografado da TPC para a máquina virtual VM1 e o tempo necessário para receber de volta este arquivo. Para cada tamanho de arquivo distinto, foram utilizadas 10 cópias do arquivo e foi feito o cálculo do tempo médio gasto para envio e recebimento. As Figuras 39 e 40 ilustram os resultados obtidos neste experimento. A analise dessas figuras permite concluir que a estratégia QSM-EXTRACTION não adiciona impacto significativo no tráfego da rede, em comparação com o algoritmo AES. Adicionalmente, vale destacar que a estratégia QSM-EXTRACTION não requer a criação e manutenção de infra-estruturas de gerenciamento de chaves públicas, o que reduz custos de implementação e manutenção.

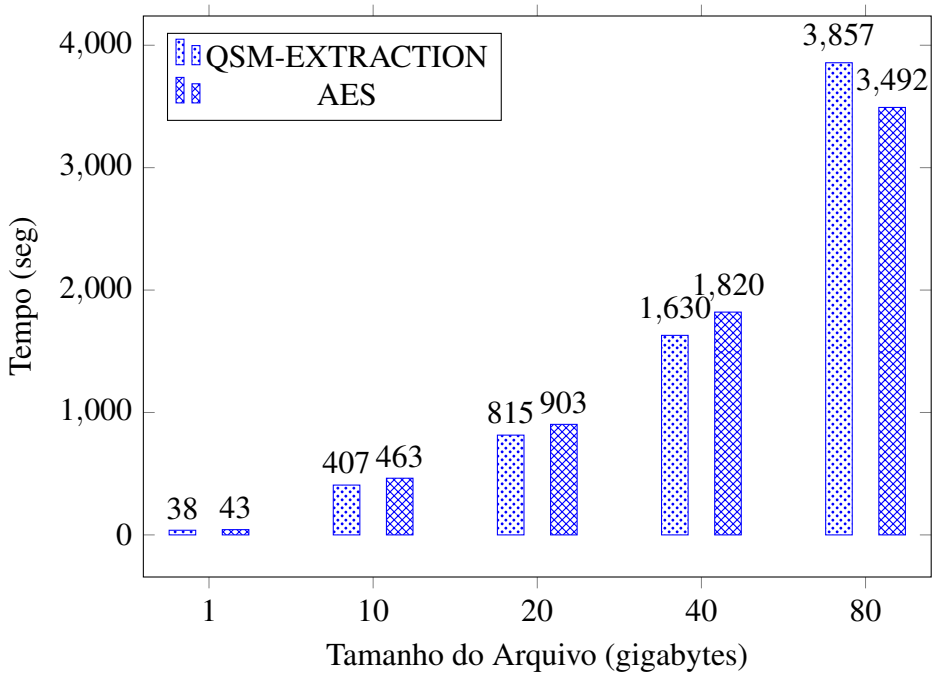

Figura 39 – Tempo de Transmissão

7.3 Conclusão

Neste capítulo, apresentamos os experimentos realizados e os resultados obtidos, com a finalidade de demonstrar a eficiência das ideias que norteiam a estratégia proposta nesta tese. Os resultados experimentais nos dão suporte para a comparação das nossas contribuições face aos trabalhos existentes na literatura relacionada.

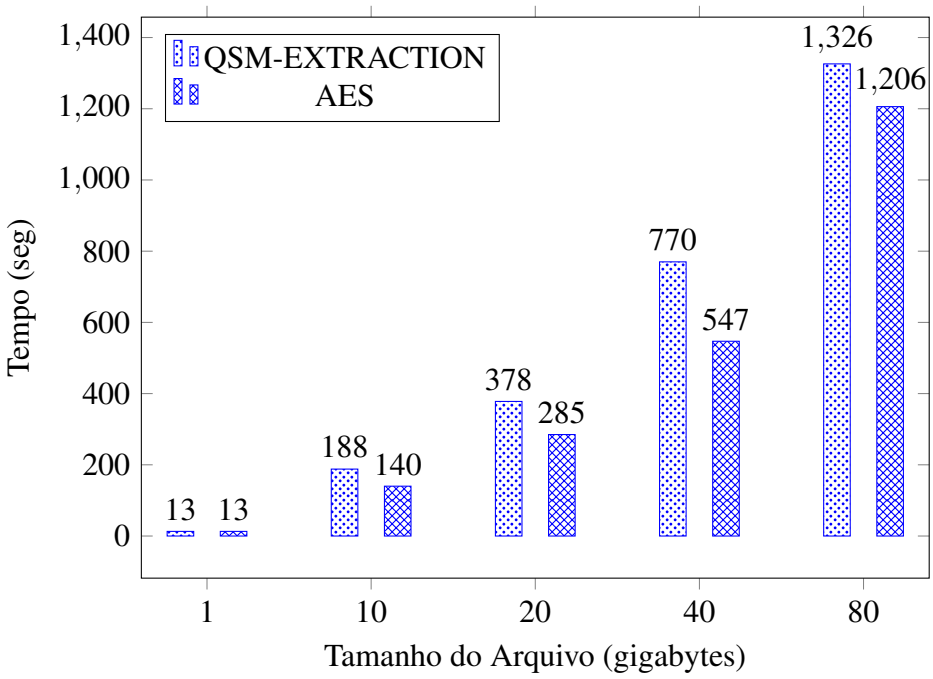

Figura 40 – Tempo de Recepção

8 TRABALHOS RELACIONADOS

8.1 Introdução

O armazenamento de dados na nuvem tem se tornado bastante popular, principalmente devido aos serviços online de arquivamento, backup, compartilhamento de dados e sincronização de dispositivos. Algumas vantagens destes serviços de armazenamento são a ubiquidade de acesso aos dados, a escalabilidade e o modelo de cobrança pelo uso. Contudo, apesar das vantagens, existem preocupações em relação à segurança e privacidade dos dados. Aqui destacamos na segurança, a dimensão da confidencialidade, que implica que os dados dos clientes e as tarefas computacionais são mantidas confidenciais tanto para os provedores de nuvem quanto para outros clientes (XIAO; XIAO, 2013).

A seguir são discutidos alguns estudos relacionados com privacidade e segurança de dados armazenados na nuvem. Para (SWEENEY, 2002), a segurança computacional não implica em proteção à privacidade, pois embora mecanismos de controle de acesso e autenticação possam proteger as informações contra a divulgação, eles não tratam da propagação indireta da informação, nem de divulgações com base em inferências e correlações sobre informações extraídas de outras fontes.

De acordo com (SHIREY, 2000), a privacidade pode ser definida como o direito de uma determinada entidade (normalmente um indivíduo), agindo em seu próprio nome, determinar o grau de interação de suas informações com o contexto onde se encontra inserida, incluindo o grau de comprometimento/disposição em divulgar essas informações para outras entidades.

Nos últimos anos, várias propostas para assegurar a confidencialidade dos dados armazenados em ambiente de computação em nuvem tem sido apresentadas (RYAN, 2011; CHEN; ZHAO, 2012; NIMGAONKAR *et al.*, 2012; STEFANOV E. SHI, 2013; LI *et al.*, 2013),mais recentemente (YANG *et al.*, 2013; JUNG *et al.*, 2013; YEH, 2013).

Stefanov propõe uma solução de segurança para armazenamento de dados distribuídos na nuvem, que utiliza hardware confiável e implementação *ORAM* (*Oblivious RAM*), que foi originalmente proposta por Goldreich e Ostrovsky (STEFANOV E. SHI, 2013). *ORAM* é um criptosistema que permite um cliente acessar dados criptografados armazenados em servidores não confiáveis. Este sistema grava e regrava continuamente os dados nos servidores para remodelar os blocos de dados e esconder o padrão de acesso lógico dos dados. Esta solução de segurança tem sido utilizada para proteger a privacidade do usuário em uma grande quantidade

de serviços na nuvem, tais como propaganda comportamental online, serviços de localização, mapas, pesquisas na web, etc.

Li apresenta algumas técnicas para busca de dados encriptados com chave simétrica. Neste caso, considera-se o cenário da busca por dados armazenados de forma criptografada em provedores de nuvem semi-confiáveis, que são aqueles que na maioria do tempo comportam-se de forma esperada, executando os protocolos definidos, mas que podem tentar acessar informação privada nos dados armazenados. (LI *et al.*, 2013) define dois fatores negativos para buscas de dados criptografados:

- Usabilidade funcional: a maioria das técnicas de busca encriptada trata apenas com busca booleana de palavras, onde as consultas são expressas como fórmulas booleanas e os documentos encriptados que satisfazem à fórmula são retornados. Estas operações de busca são ainda muito restritivas e não são adequadas a aplicações reais de armazenamento de dados em larga escala.
- Eficiência: Esquemas com muitas funcionalidades de pesquisa requerem operações com chaves públicas, o que sobrecarrega os sistemas de armazenamento de dados em nuvem em termos de esforço computacional, largura de banda e latência de acesso.

(XIAO; XIAO, 2013) relaciona as características da nuvem que causam vulnerabilidades de segurança e privacidade, as quais são descritas a seguir:

- Máquinas virtuais de diferentes clientes compartilhando os mesmos recursos físicos (*hardware*) possibilitam ataque de canal lateral (*side-channel attack*), situação em que o atacante pode ler informações do cache da máquina e descobrir o conteúdo de chaves criptográficas de outros clientes.
- Perda de controle físico da máquina pelo cliente, que não pode se proteger contra ataques e acidentes. Por exemplo: os dados armazenados na nuvem podem ser alterados ou perdidos, sem que o cliente tenha nenhum tipo de controle sobre os eventos que ocasionaram a perda ou roubo desses dados.
- Subprovisionamento da largura de banda da rede, o que provocou o surgimento de um novo tipo de ataque de negação de serviço (*Denial of Service (DOS)* que se aproveita do fato da capacidade de rede do provedor de nuvem ser menor do que a quantidade de máquinas alocadas na mesma sub-rede (LIU *et al.*, 2012).

Estas soluções podem ser classificadas em dois grupos ou abordagens: *Statistical Disclosure Control (SDC)* e *Fragmentation-Redundancy-Scattering (FRS)*.

1. *Statistical Disclosure Control* (SDC): As técnicas, neste grupo, gerenciam a proteção dos dados publicados sem revelar informações confidenciais relacionadas com pessoas específicas as quais os dados se referem ou pertencem. A proteção fornecida pelas técnicas SDC provocam, em algum grau, a modificação nos dados publicados, dentro de limites de nenhuma modificação (máxima utilidade para os usuários e nenhuma proteção aos dados) e criptografia (proteção máxima para os dados e nenhuma utilidade para uso sem a chave criptográfica) (DOMINGO-FERRER, 2008). A abordagem de SDC é apropriada quando os dados armazenados na nuvem precisam ser públicos.
2. *Fragmentation-Redundancy-Scattering*(FRS): As técnicas, neste grupo, permitem a recuperação de dados destruídos, incorretos ou contaminados por falhas acidentais ou por intrusões, preservando a confidencialidade de dados sensíveis. Esta abordagem trata primeiramente de fragmentar as informações sensíveis, de tal forma que quaisquer fragmentos isolados não contenham informação significativa e, em seguida, distribuir os fragmentos entre os diferentes locais do sistema distribuído, de modo que uma intrusão em uma parte do sistema dê acesso apenas aos fragmentos não relacionados entre si. A abordagem FRS é apropriada quando os dados a serem armazenados na nuvem não precisam ser públicos.

A solução de confidencialidade de dados da estratégia QSM-EXTRACTION possui as seguintes características: permite a utilização de nuvens públicas; o serviço de privacidade e segurança é centralizado por meio de um servidor *proxy* que realiza a dispersão dos dados para a nuvem; grande flexibilidade para customização dos níveis de confidencialidade dos dados armazenados na nuvem. Podemos citar dois trabalhos semelhantes ao QSM-EXTRACTION que possuem estas características: o primeiro utiliza criptografia para garantir a confidencialidade (SEIGER *et al.*, 2011); o segundo é aplicável apenas a arquivos digitais de imagens (NOURIAN; MAHESWARAN, 2012).

As próximas seções apresentam as recentes pesquisas sobre o uso isolado e combinado das técnicas de criptografia e fragmentação para aumento da confidencialidade dos dados armazenados na nuvem.

8.2 Criptografia

Uma quantidade significativa de pesquisas foram recentemente dedicadas à investigação da confidencialidade de dados armazenados e processados em ambiente de computação em nuvem utilizando técnicas criptográficas (POPA *et al.*, 2011; NING *et al.*, 2014). A maior

parte destas pesquisas têm assumido que os dados devem estar inteiramente criptografados, concentrando-se no aprimoramento das técnicas de execução de consultas (CIRIANI *et al.*, 2010). Em (CESELLI *et al.*, 2005), os autores discutem diferentes estratégias para avaliação da inferência da exposição de dados criptografados acrescidos de informações de indexação, mostrando que mesmo um número limitado de índices pode favorecer muito a tarefa de um atacante que pretende violar a confidencialidade dos dados.

A criptografia é uma ferramenta útil para aumentar a confidencialidade de dados sensíveis. Entretanto, quando os dados são encriptados, a realização de consultas se torna um desafio. Assim, embora a encriptação de dados proporcione confidencialidade, as cifras produzidas são muito menos convenientes para uso do que os dados originais. Quando utilizada com bancos de dados relacionais, a criptografia cria dois grandes problemas. O primeiro problema é que os bancos relacionais requerem que os tipos de dados sejam definidos antes do seu armazenamento. O segundo problema é que consultas ou funções não podem ser executadas sobre dados criptografados. Não é possível avaliar faixas de datas ou fazer comparações de valores em dados criptografados. As estruturas de índice também não podem ser utilizadas. Adicionalmente, os métodos baseados em criptografia precisam incluir estratégias de geração e distribuição de chaves (TIAN; ZHANG, 2012). Porém, existem várias desvantagens relacionadas com a gestão de chaves criptográficas, tais como:

1. A necessidade de guardar as chaves pelo igual período de tempo em que os dados permanecerem criptografados.
2. A atribuição ou a revogação de chaves para o acesso aos dados por parte dos usuários.
3. A necessidade de manter múltiplas cópias encriptadas do mesmo arquivo, para acesso multi-usuário utilizando chave-pública.

As vantagens da estratégia QSM-EXTRACTION sobre as técnicas convencionais que utilizam criptografia para garantir confidencialidade dos dados armazenados na nuvem são as seguintes:

1. Não utilização de chaves criptográficas.
2. A técnica pode ser aplicada a qualquer formato de dado armazenado (dados e programas).
3. Não há limitação máxima para o tamanho do arquivo de dados, o quê habilita a técnica a ser usada com grandes volumes de dados (*big data*).
4. A solução suporta expurgo de dados da nuvem, pois os arquivos disponibilizados em provedores distintos não revelam informações sobre os dados originais. Caso o usuário

deixe a nuvem, os dados podem ser considerados automaticamente expurgados.

8.3 Fragmentação

A primeira abordagem que propõe o armazenamento de dados não criptografados, assegurando ao mesmo tempo uma série de restrições de confidencialidade foi apresentado em (AGGARWAL, 2005). Neste trabalho, os autores supõem que os dados sejam divididos em dois fragmentos, os quais serão armazenados em dois provedores de serviço de nuvem honestos-mas-curiosos, que nunca compartilharão informações entre si e recorre à encriptação quando estes dois fragmentos não são suficientes para impor restrições de confidencialidade.

Em (CIRIANI *et al.*, 2009; CIRIANI *et al.*, 2010), os autores resolvem o problema da confidencialidade, propondo uma solução que primeiro divide os dados a serem protegidos em vários (possivelmente mais de dois) fragmentos diferentes de modo a quebrar as associações sensíveis entre atributos e minimizar a quantidade de atributos representados apenas em formato criptografado. Esses fragmentos podem ser armazenados em diferentes servidores. A heurística proposta para descobrir esses fragmentos apresenta um tempo de custo de computação em tempo polinomial, ao mesmo tempo que é capaz de recuperar soluções próximas ao ideal. Uma solução próxima ao ideal é aquela em que a relação entre o custo **C** da heurística proposta e o custo $\mathbf{C}^*$ de uma solução que retorna todos os resultados possíveis (solução ótima) é menor do que um determinado parâmetro **p**. Caso contrário, não é possível realizar qualquer avaliação do resultado da heurística.

Em (XU *et al.*, 2015), os autores propõem um método eficiente baseado em pesquisa em grafo para o problema de fragmentação com restrições de confidencialidade, que obtém resultados próximos ao ideal. (SAYI *et al.*, 2012) estendeu o trabalho de Ciriani, propondo o uso de algoritmos de coloração de grafos para determinar quais dados deveriam ser armazenados na nuvem e quais deveriam ficar sobre a guarda do proprietário dos dados.

As vantagens da técnica do QSM-EXTRACTION sobre as técnicas que utilizam fragmentação para garantir confidencialidade dos dados armazenados na nuvem são as seguintes:

1. Fragmentação do arquivo de dados em apenas 3 partes que armazenam as informações da qualidade, quantidade e medida dos objetos de informação do arquivo, independente da quantidade de atributos existentes.
2. O embaralhamento dos dados das 3 partes garantem confidencialidade não revelando informações úteis sobre os dados originais, enquanto a fragmentação revela informações

sobre os dados dos fragmentos.

8.4 Criptografia e Fragmentação

O trabalho apresentado em (CIRIANI *et al.*, 2009) propõe um novo paradigma para preservação da confidencialidade dos dados sob a guarda de terceiros, que dispensa uso de criptografia, liberando assim o proprietário do dado do ônus de gestão de chaves. A ideia básica por trás deste mecanismo é envolver o proprietário do dados no armazenamento de atributos sensíveis. Assim sendo, para cada associação sensível, o proprietário deve armazenar localmente pelo menos um atributo. Os atributos restantes são armazenados, em formato não criptografado, do lado do servidor de nuvem. Com este processo de fragmentação, uma relação original R é então quebrada em dois fragmentos, chamados F_o e F_s, que são armazenados na infraestrutura de TI do proprietário do dado e no servidor de nuvem, respectivamente. (WIESE, 2010) faz uma extensão da abordagem de “apenas fragmentação vertical"e propõe utilizar fragmentação horizontal para filtrar linhas confidenciais para serem armazenadas de forma confidencial na infraestrutura do proprietário do dado. Em (REKATSINAS *et al.*, 2013), os autores apresentam SPARSI, um *framework* teórico para particionamento de dados sensíveis em múltiplos servidores adversários não coniventes. Eles introduzem o problema de particionamento de dados entre “respeitadores da privacidade", onde um conjunto de dados sensíveis deve ser dividido entre k partes não confiáveis (adversários). O objetivo é o de maximizar a utilidade derivada do particionamento e distribuição do conjunto de dados, enquanto minimiza a quantidade total de informação sensível divulgada. Solução de particionamento entre “respeitadores da privacidade"é, em geral, NP-difícil.

Em (SAMARATI; VIMERCATI, 2010) os autores discutem as principais questões a serem abordadas nos serviços de armazenamento em nuvem, que vão desde a confidencialidade dos dados até a utilidade dos dados. Eles mostram as principais direções de pesquisa que estão sendo investigadas para fornecer efetiva confidencialidade de armazenamento e consulta de dados. A pesquisa apresentada em (JOSEPH *et al.*, 2013) endereça algumas abordagens para garantir a confidencialidade dos dados em serviços de armazenamento em nuvem não confiáveis. Em (SAMARATI, 2014), os autores discutem os problemas da garantia adequada confidencialidade de dados e privacidade dos usuários na nuvem, e ilustram as possíveis soluções para eles.

8.5 Soluções de Gestão Confiável de Dados na Nuvem

Esta seção apresenta algumas soluções estudadas no âmbito deste trabalho que propõem resolver a maioria das questões relacionadas à confidencialidade de dados na nuvem. Estes sistemas de armazenamento de dados em nuvem atendem a objetivos de confidencialidade, integridade e tolerância a falhas. Existem problemas relacionados à confiança em terceiros que gerenciam a nuvem, à disponibilidade dos recursos, ao nível de segurança oferecido pela nuvem e, por último, à espionagem por parte do fornecedor do serviço ou até acesso aos dados em si (STRUMBUDAKIS, 2014).

1. **iDataGuard:** Este sistema apresenta um componente de *middleware* a ser executado no cliente ou em uma Terceira Parte Confiável - TPC. As aplicações exportam os seus dados em um conjunto de pastas para este *middleware* que por sua vez realiza o armazenamento em um conjunto de nuvens. O sistema se responsabiliza pelo gerenciamento dos dados nas nuvens, que não percebem a sua existência, visto que para elas, o sistema é apenas mais uma aplicação a ser atendida. Para alcançar este tipo de comunicação, o sistema usa adaptadores para cada uma das nuvens, criando assim uma camada de abstração, a fim de combater a heterogeneidade entre os vários fornecedores, dado que as operações agora suportadas pelo sistema limitam-se a inserções, atualizações, leituras e remoções de conteúdo.

   Os dados do sistema são criptografados por chaves geradas a partir de uma senha fornecida pelo usuário e em função das iterações do algoritmo, de modo a que não seja possível, para os atacantes, recriar a chave. Os objetos guardados nas diversas nuvens são compostos por vários atributos. Por exemplo: o conteúdo dos dados, o nome do dado, os metadados e um identificador gerado pelo módulo **CryptInd** (JAMMALAMADAKA *et al.*, 2013), para permitir indexação da pasta. Além de prover a confidencialidade dos dados, para prevenir ataques baseados na estrutura, é possível esconder a estrutura do sistema de pastas, recorrendo a especificação da estrutura dentro dos metadados que vão criptografados, como também os outros atributos, a exceção do identificador. O sistema também preserva a integridade dos diversos objetos, utilizando códigos de criptografia *Hash-based Message Authentication Code - HMAC*, os quais são calculados e guardados por parte dos servidores. A pesquisa sobre os conteúdos criptografados é uma das limitações do sistema. Para minimizar este problema os autores usam uma segunda abordagem, particionando várias vezes o índice de cada objeto, propondo assim a versão **CryptInd++**, permitindo pesquisas

por padrões. Por exemplo: nas consultas, a pesquisa pode ser feita procurando um subconjunto de caracteres de uma palavra para encontrar assim a palavra em questão, que fornece o identificador do conteúdo completo permitindo assim obter o conteúdo baseado numa pesquisa parcial. Por fim, existem algumas limitações relacionadas com a escalabilidade do sistema e pode haver casos onde a solução não seja viável, por exemplo quando um cliente executa um pedido de indexação completa da estrutura. A necessidade do uso de índices auxiliares não é o ideal, devido a necessidade de comunicação adicional, o quê causa aumento do tráfego na comunicação com as nuvens.

2. **Silverline:** Este sistema oferece serviços de “segurança como um serviço"para proteger dados de inquilinos em nuvens públicas, mesmo que o software ou serviços de um determinado inquilino sejam eles próprios inseguros. SilverLine aumenta o desempenho do gerenciador de máquinas virtuais do provedor de nuvem (Por exemplo, XenServer, VMware ESXi, etc.) para isolamento dos dados e da rede entre vários inquilinos de nuvem, e combina esse isolamento com modificações para sistemas operacionais convidados para obter isolamento de dados. Estes mecanismos protegem contra vazamentos de dados decorrentes de comprometimento, configuração incorreta ou ataques de inquilinos coresidentes na nuvem (RISTENPART *et al.*, 2009).

   Para garantir confidencialidade, os dados são encriptados ao máximo possível, o quê pode comprometer o desempenho do sistema. Para evitar isto, o sistema minimiza o acessos aos dados para os usuários, de modo a não expor a um único usuário toda a informação caso não seja necessário. Para tal, são definidos grupos de acesso a partir da analise das operações realizadas pelos usuários. Deste modo, os clientes só podem visualizar conteúdo para o qual lhes foi autorizado acesso, recorrendo unicamente às chaves que cada possuem, podendo existir clientes com mais que uma chave de acordo com o grupo de permissões nos quais se inserem.

   Silverline oferece vantagens para atualização de chaves criptográficas por não necessitar criptografar novamente os objetos guardados, situação que iria necessitar protocolos de comunicação e computação adicional, causando impacto indesejável no desempenho do sistema. A alternativa proposta quando um usuário é removido é, em vez de cifrar os dados de novo, o sistema gera uma nova chave para os dados armazenados, atualizando os clientes restantes do grupo com a nova chave. O ponto negativo desta solução é o perigo de, a longo prazo, criar inconsistências e provocar sobrecarga de gerenciamento.

3. ***HAIL (High Availability and Integrity Layer)*:** É um sistema distribuído criptográfico que utiliza um conjunto de ferramentas para permitir o uso de múltiplos serviços de armazenamento simultâneo na nuvem, possibilitando ao cliente consultar se uma pasta de dados armazenada está intacta e é recuperável (BOWERS *et al.*, 2009). Este sistema baseia-se em duas verificações básicas: prova de recuperação (*Proof of Recovery - POR*) e prova de posse (*Proof of Possesion – PDP*). Estas provas permitem que o cliente verifique a existência e integridade dos dados disponibilizados pelo sistema na nuvem.

   O sistema HAIL oferece garantias de recuperabilidade de dados, considerando o modelo de adversário como sendo o de um atacante que procura corromper ou controlar sucessivos servidores simultaneamente, mas nunca todos os servidores ao mesmo tempo, permitindo que as outras réplicas reconstruam os dados. Normalmente, os servidores não se comunicam diretamente, estando a comunicação limitada ao cliente-servidor, sem incrementar o custo da comunicação, por permitir computação do lado do servidor.

   Em relação ao mecanismo de detecção e correção de servidores defeituosos, o cliente fica encarregado de se comunicar com os outros servidores cujos fragmentos de dados não estão corrompidos para atualizar os servidores em falha. Para este efeito o HAIL recorre a 3 elementos importantes:

   a) código de agregação: técnica de compressão das respostas para os clientes, pela combinação de diversos códigos *message authentication code-MACs* em um único código, sendo assim mais eficiente a verificação de vários blocos simultaneamente, em tempo real, sem a necessidade de ocupar espaço adicional em disco.
   b) Códigos de dispersão: distribuem os dados nos diversos servidores, mais concretamente usam código de correção de erros com proteção de integridade garantida pelos códigos MACs.
   c) Código de servidor: os blocos de dados encontram-se codificados com código de correção, protegidos contra corrupção de baixo nível dos conteúdos caso os testes de integridade falhem.

   As limitações do sistema HAIL são as seguintes: trata apenas com dados estáticos (i.e., os algoritmos não suportam atualizações e múltiplas versões dos dados) e requer que os servidores executem código.

4. **DepSky:** É um sistema para replicação de dados em várias nuvens que melhora a disponibilidade, integridade e confidencialidade da informação armazenada. As unidades

de dados (*data units*) no DepSky disponibilizadas pelos proprietários dos dados podem ser consultados no modo de leitura por um conjunto de usuários autorizados. A disponibilidade é garantida mesmo em caso de falhas por algoritmos de replicação para sistemas de quóruns bizantinos de disseminação, onde os dados armazenados em cada servidor são auto-verificáveis por meio de assinaturas digitais e resumos criptográficos. São apresentadas duas versões do sistema: DepSky-A, para garantir integridade e disponibilidade dos dados, e a versão DepSky-CA que utiliza partilha de chaves a fim de prover confidencialidade aos dados armazenados.

O sistema garante capacidade de recuperação para um conjunto de (**3f + 1**) para **f** servidores defeituosos ou com dados corrompidos. Para garantir a integridade dos dados, os mesmos são assinados digitalmente sendo guardadas as assinaturas nos próprios servidores. No sistema DepSky-CA, o sistema permite que os dados estejam distribuídos pelos diversos servidores de forma que seja necessário obter **f + 1** partes para reconstruir o segredo, sendo assim, para um atacante com **f** partes, é impossível recriar os dados originais.

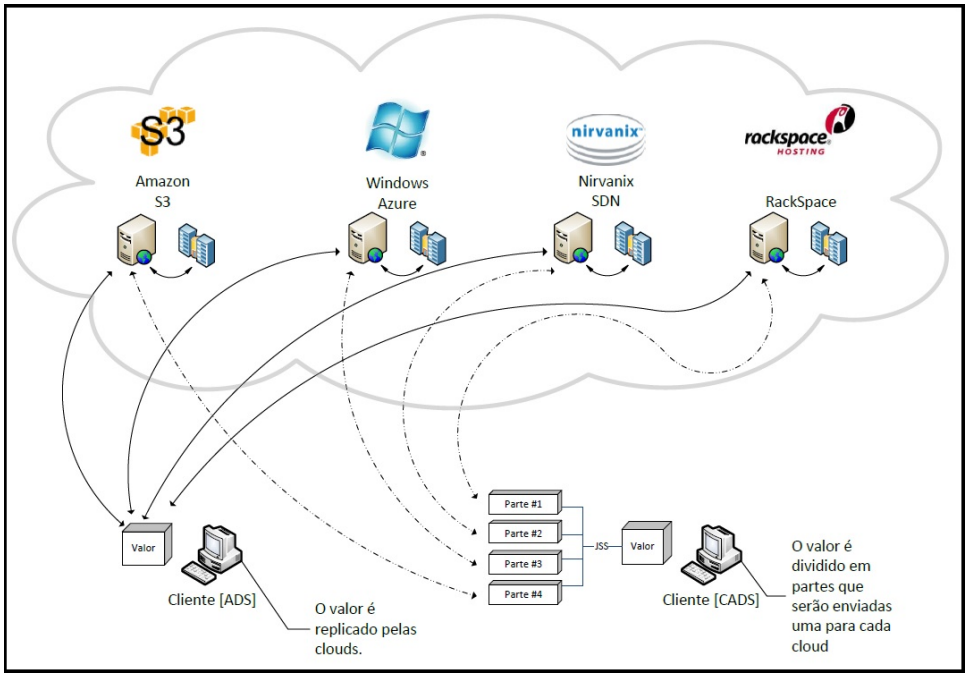

Figura 41 – Distribuição de dados nas nuvens (STRUMBUDAKIS, 2014)

A figura 41 mostra um exemplo de dispersão de dados por várias nuvens. Os protocolos ADS (*Available DEPSKY*) e CADS (*Confidential & Available DEPSKY*) são utilizados nos clientes. A diferença entre estes protocolos é a informação enviada para as nuvens. No caso de um cliente usar o protocolo ADS, é enviada uma cópia da informação para todas as nuvens. Com o protocolo CADS a informação é dividida em partes, tantas quanto o

número de nuvens, e depois cada parte é enviada para sua nuvem. A informação é dividida de maneira que apenas seja necessário um determinado número de partes para reconstruir a informação original, e não a totalidade das partes.

Tabela 19 – Comparação entre os Sistemas de Armazenamento Confiável

Sistema	Confidencialidade	Integridade	Disponibilidade
iDATAGUARD	criptografia simétrica	criptografia HASH	usa camada de *middleware* em TPC
Silveline	criptografia simétrica	-	-
HAIL	fragmentação e dispersão -	usa TPC e Prova de Recuperação (PDP)	Prova de Posse (POR)
DepSky	partilha de chaves	assinaturas digitais	quóruns bizantinos
QSM-EXTRACTION	extração das características dos objetos de informação	restrições de integridade das características dos objetos de informação	Algoritmos de dispersão (*RABIN's IDA*)

A Tabela 19 apresenta uma comparação das características de confidencialidade, integridade e disponibilidade dos quatro sistemas de armazenamento confiável descritos nesta seção. Pode-se constatar que os sistemas *iDataguard*, *HAIL* e *DepSky* fornecem as 3 características de segurança. Enquanto *Silverline* fornece apenas confidencialidade. HAIL é o único que trabalha com dados não criptografados, por isso é considerado o que oferece menor confidencialidade aos dados armazenados. A estratégia QSM-EXTRACTION é flexível e poderia ser adaptada para uso no sistema *iDATAGUARD* em substituição à criptografia simétrica para garantir a confidencialidade. No sistema *Silverline*, o uso de QSM-EXTRACTION acrescentaria integridade ao sistema. A combinação de QSM-EXTRACTION com *HAIL* poderia minimizar a fragmentação e dispersão dos dados.

8.6 Conclusão

Este capítulo descreveu os trabalhos relacionados à proteção da confidencialidade, integridade e disponibilidade dos dados armazenados na nuvem que utilizam técnicas de criptografia, fragmentação isoladamente ou em conjunto. Foram destacadas as vantagens da abordagem da estratégia QSM-EXTRACTION em relação às técnicas de criptografia e fragmentação. O próximo capítulo expressa as conclusões deste trabalho e indica direções para trabalhos futuros.

9 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo destaca as principais conclusões que resultaram das pesquisas efetuadas durante a realização desta tese. Adicionalmente, discute-se possíveis extensões e aplicações das ideias aqui apresentadas.

9.1 Considerações Finais

As tecnologias disruptivas de informação e comunicação estarão produzindo nos próximos anos grandes transformações na forma como as informações serão produzidas e armazenadas na nuvem. Questões relacionadas à privacidade dos usuários e confidencialidade dos dados serão cada vez mais críticas para o sucesso da implantação destas tecnologias. Segundo pesquisa do *Global Agenda Council* (ESPINEL *et al.*, 2015), o primeiro *smartphone* implantável no corpo estará disponível em 2025, 90% da população mundial estará conectada na internet em 2023, haverá 1 trilhão de sensores conectados à Internet das Coisas em 2022 e 10% do PIB mundial estará armazenado na nuvem em 2027.

Neste contexto, o armazenamento de dados na nuvem tem um lado obscuro - ameaças à privacidade e confiança do consumidor, segurança de dados, pirataria, fraude, *cyberhacking* e *cibercrime* são, todos estes, riscos que precisam ser gerenciados. Assim sendo, a disseminação de boas práticas de uso da tecnologia são importantes para promover habilidades digitais e consciência sobre os riscos que acompanham os avanços tecnológicos, tais como questões de privacidade, segurança e fraude, entre outros. Neste capítulo final, apresentam-se os resultados alcançados decorrentes da realização desta pesquisa, assim como as principais limitações do trabalho e sugestões para estudos futuros.

9.2 Contribuições para a ciência da computação

Esta tese teve por finalidade propor uma nova estratégia para assegurar a confidencialidade de dados armazenados em ambientes de computação em nuvem e, consequentemente, garantir a privacidade do proprietário desses dados. A estratégia proposta, denominada QSM-EXTRACTION, inviabiliza que provedores de serviços em nuvem (“honesto-curiosos") tenham acesso ao conteúdo original dos dados dos clientes. Assim, por meio da confidencialidade dos dados armazenados em nuvem, a estratégia assegura a privacidade dos proprietários desses dados.

As principais contribuições desta tese foram:

1. Um estudo sobre as técnicas de anonimização, fragmentação e criptografia de dados;
2. Um estudo sobre as principais métricas e indicadores relacionados a privacidade;
3. Uma nova estratégia, denominada QSM-EXTRACTION, para assegurar a confidencialidade de dados em serviços de armazenamento em nuvem;
4. Definição do conceito de objeto de informação;
5. Definição das características de um objeto de informação, com base nos conceitos da Doutrina do Ser de Hegel;
6. Implementação da estratégia QSM-EXTRACTION;
7. Avaliação da estratégia QSM-EXTRACTION.

Vale destacar, ainda, que a estratégia QSM-EXTRACTION apresenta as seguintes propriedades:

1. Generalidade: é aplicável a vários formatos de arquivos (documentos texto, som, multimídia).
2. Flexibilidade: pode ser combinada com soluções de dispersão de arquivos (ex.: Rabin, Shamir, AONTRS, etc), fragmentação vertical e criptografia (simétrica ou assimétrica).
3. Simplicidade: a confidencialidade dos dados é obtida com o uso de operações simples de substituição e transposição, que são os blocos básicos de construção de todas as técnicas criptográficas.
4. Eficiência: aplica-se a ambientes de computação em nuvem, onde, em geral, as leituras são mais frequentes que as atualizações. Adicionalmente, a estratégia QSM-EXTRACTION não requer a definição de fragmentos, o que é um problema NP-difícil.
5. Gerenciabilidade: não requer o armazenamento e nem o gerenciamento de chaves criptográficas.
6. Utilidade: a recuperação das informações armazenadas na nuvem ocorre sem nenhuma perda de informação.

9.3 Limitações e recomendações para estudos futuros

Para a concepção da estratégia QSM-EXTRACTION foram assumidas algumas premissas:

- Os usuários da nuvem devem estar identificados e autenticados para terem acesso aos dados armazenados na nuvem;

- Os provedores de nuvem devem garantir disponibilidade e integridade dos dados;
- Os provedores de nuvem são considerados “honesto-curiosos", isto é, executam corretamente os protocolos de acesso aos dados, mas têm interesse de inferir e analisar dados (incluindo índices) e o fluxo de mensagens recebidas durante o protocolo de modo a aprender informações adicionais sobre os dados.

É importante ressaltar que estas premissas podem não ser viáveis em determinados cenários ou ambientes de computação em nuvem.

Devido à amplitude e complexidade do tema privacidade, esta tese não pretendeu ser um estudo exaustivo. Mas, ao contrário, ela teve um caráter exploratório, buscando abrir possibilidades para novas pesquisas e soluções que visem propor tecnologias que assegurem a privacidade dos usuários em um mundo cada vez mais conectado e digital. A partir dos estudos realizados e da estratégia proposta, é possível ter uma visão mais clara dos desafios existentes e das características que tais soluções devem apresentar. Neste sentido, a estratégia QSM-EXTRACTION pode ser um ponto de partida. A flexibilidade da estratégia proposta nesta tese permite sua utilização em conjunto com as técnicas tradicionais de criptografia e fragmentação. Desta forma, abre-se a possibilidade de investigar os diversos arranjos e combinações possíveis. Uma primeira evolução da estratégia QSM-EXTRACTION paraleliza os algoritmos relacionados a decomposição e recomposição dos objetos de informação utilizando o paradigma **map-reduce**. Esses algoritmos poderiam ser executados utilizando-se a Plataforma **Spark**. Acreditamos que a paralelização dos algoritmos propostos pode melhorar de forma significativa o desempenho da estratégia QSM-EXTRACTION. A seguir, discute-se alguns possíveis trabalhos futuros.

Em relação à proteção da privacidade dos proprietários de dados armazenados na nuvem, os seguintes direcionamentos podem ser dados para pesquisas adicionais envolvendo a estratégia QSM-EXTRACTION:

1. Desenvolvimento de soluções integradas entre sistemas de identificação e autorização e a estratégia QSM-EXTRACTION. Um exemplo disto, seria o uso de uma das características, a qualidade, por exemplo, como informação de identificação, substituindo a senha, que o usuário teria que apresentar para se identificar perante o sistema da nuvem. O provedor de nuvem armazenaria apenas a quantidade e a medida. Ao receber a informação da qualidade, o provedor recomporia o objeto de informação, que poderia ser utilizado como chave de autorização às funcionalidade do sistema.
2. Pesquisa para implementação da estratégia QSM-EXTRACTION no sistema de armaze-

namento de bancos de dados NoSQL (Not only SQL) , tais como o MongoDB, Hadoop, Casandra, Amazon SimpleDB e Hypertable.

3. Investigações sobre formas de auditoria de dados armazenados na nuvem utilizando a estratégia QSM-EXTRACTION. Por exemplo, existe uma quantidade de bits não utilizada na característica da Quantidade que poderia ser utilizada para guardar informações sobre a quantidade de alterações realizada no objeto de informação, possibilitando a verificação de alterações no conteúdo do arquivo apenas pela leitura dos dados da quantidade, que corresponde a cerca de 12,5 % do tamanho do arquivo original.

A adaptação e utilização da estratégia QSM-EXTRACTION em outras áreas da Ciência da Computação também pode ser tema de pesquisas futuras, como por exemplo:

1. Segurança de Dados: estudos sobre a adição de chave criptográfica a característica Qualidade para expandir o uso da estratégia QSM para outros tipos de dados, além de arquivos pessoais.
2. Lógica Matemática: estudos para testar formas alternativas de representação dos dados da característica Medida, ao invés de usar números para representar as posições dos bytes no objeto de informação.
3. Teoria da Informação: estudos para avaliar a estratégia QSM-EXTRACTION na computação quântica, utilizando *qubits* (bits quânticos) ao invés de bits para representar as característica do objeto de informação
4. Redes de Computadores: testes de transmissão das características de qualidade, quantidade e medida de modo a não permitir a identificação dessas característica nos pacotes (unidade de transferência de informação) de rede.
5. Inteligência Artificial: pesquisas utilizando *machine learning* para inferência do valor de uma das três características a partir do conhecimento de duas outras características. Por exemplo inferir a qualidade, a partir da quantidade e da medida.

REFERÊNCIAS

AGGARWAL, C. C. On k-anonymity and the curse of dimensionality. In: **Proceedings of the 31st international conference on Very large data bases**. [S.l.]: VLDB Endowment, 2005. p. 901–909.

AGGARWAL, C. C.; PHILIP, S. Y. A condensation approach to privacy preserving data mining. In: ____. **Advances in Database Technology-EDBT 2004**. [S.l.]: Springer, 2004. p. 183–199.

AGGARWAL, C. C.; PHILIP, S. Y. Privacy-preserving data mining: A survey. In: ____. **Handbook of Database Security**. [S.l.]: Springer, 2008. p. 431–460.

AGRAWAL, D.; AGGARWAL, C. C. **On the design and quantification of privacy preserving data mining algorithms**. [S.l.]: ACM, 2001. 247-255 p.

ALLIANCE, C. S. **The Treacherous 12 CSA's Cloud Computing Top Threats in 2016**. 2016.

ALON, N.; KAPLAN, H.; KRIVELEVICH, M.; MALKHI, D.; STERN, J. Scalable secure storage when half the system is faulty. In: SPRINGER. **International Colloquium on Automata, Languages, and Programming**. [S.l.], 2000. p. 576–587.

ASSANTE, M. America's critical infrastructure is vulnerable to cyber attacks. **Forbes. November**, v. 11, 2014.

BARBAY, J.; NAVARRO, G. Compressed representations of permutations, and applications. In: **26th International Symposium on Theoretical Aspects of Computer Science**. [S.l.: s.n.], 2009. p. 111–122.

BARDIN, J.; CALLAS, J.; CHAPUT, S.; FUSCO, P.; GILBERT, F.; HOFF, C.; HURST, D.; KUMARASWAMY, S.; LYNCH, L.; MATSUMOTO, S. *et al.* Security guidance for critical areas of focus in cloud computing. **Cloud Security Alliance**, p. 0–176, 2009.

BARKER, E.; CHEN, L.; REGENSCHEID, A.; SMID, M. Recommendation for pair wise key establishment schemes using integer factorization cryptography. **NIST Special Publication**, 2014.

BASSO, T.; MATSUNAGA, R.; MORAES, R.; ANTUNES, N. Challenges on anonymity, privacy, and big data. In: IEEE. **Dependable Computing (LADC), 2016 Seventh Latin-American Symposium on**. [S.l.], 2016. p. 164–171.

BAY, M. The ethics of unbreakable encryption: Rawlsian privacy and the san bernardino iphone. **First Monday**, v. 22, n. 2, 2017.

BEDI, R.; MAHAJAN, A. Identification of k-tuples using k-anonymity algorithm to the watermarking of social network database. **International Journal of Computer Applications**, Foundation of Computer Science, v. 142, n. 14, 2016.

BEZZI, M. Expressing privacy metrics as one-symbol information. In: **Proceedings of the 2010 EDBT/ICDT Workshops**. New York, NY, USA: ACM, 2010. (EDBT '10), p. 29:1–29:5. ISBN 978-1-60558-990-9. Disponível em: <http://doi.acm.org/10.1145/1754239.1754272>.

BIDOIT, N.; COLAZZO, D.; MALLA, N.; ULLIANA, F.; NOLè, M.; SARTIANI, C. Processing xml queries and updates on map/reduce clusters. In: **Proceedings of the 16th International Conference on Extending Database Technology**. New York, NY, USA: ACM, 2013. (EDBT '13), p. 745–748. ISBN 978-1-4503-1597-5. Disponível em: <http://doi.acm.org/10.1145/2452376.2452470>.

BLAKLEY, G. R. Safeguarding cryptographic keys. **Managing Requirements Knowledge, International Workshop on**, IEEE Computer Society, Los Alamitos, CA, USA, v. 00, p. 313, 1979.

BONEH, D.; SHACHAM, H. Fast variants of rsa. **CryptoBytes**, v. 5, n. 1, p. 1–9, 2002.

BORRETT, D. S.; SAMPSON, H.; CAVOUKIAN, A. Research ethics by design: A collaborative research design proposal. **Research Ethics**, SAGE Publications, p. 1747016116673135, 2016.

BOWERS, K. D.; JUELS, A.; OPREA, A. Hail: A high-availability and integrity layer for cloud storage. In: **Proceedings of the 16th ACM Conference on Computer and Communications Security**. New York, NY, USA: ACM, 2009. (CCS '09), p. 187–198. ISBN 978-1-60558-894-0. Disponível em: <http://doi.acm.org/10.1145/1653662.1653686>.

BUHRMAN, H.; CHRISTANDL, M.; HAYDEN, P.; LO, H.-K.; WEHNER, S. Security of quantum bit string commitment depends on the information measure. **Physical Review Letters**, APS, v. 97, n. 25, p. 250501, 2006.

BUYYA, R.; BROBERG, J.; GOSCINSKI, A. M. **Cloud computing: Principles and paradigms**. [S.l.]: John Wiley & Sons, 2010. v. 87.

CAMENISCH, J.; CRISPO, B.; FISCHER-HüBNER, S.; LEENES, R.; RUSSELLO, G. Privacy and identity management for life: 7th ifip wg 9.2, 9.6/11.7, 11.4, 11.6 international summer school, trento, italy, september 5-9, 2011. revised selected papers. In: **The 7th IFIP WG 9.2, 9.6/11.7, 11.4, 11.6 International Summer School on Privacy and Identity Management for Life**. [S.l.: s.n.], 2011.

CAO, J.; KARRAS, P. Publishing microdata with a robust privacy guarantee. **Proc. VLDB Endow.**, v. 5, n. 11, p. 1388–1399, 2012.

CAVOUKIAN, A. Privacy by design. **https://www.ipc.on.ca/wp-content/uploads/Resources/PbDReport.pdf**, p. 4–5, 2011.

CESELLI, A.; DAMIANI, E.; VIMERCATI, S. D. C. D.; JAJODIA, S.; PARABOSCHI, S.; SAMARATI, P. Modeling and assessing inference exposure in encrypted databases. **ACM Transactions on Information and System Security (TISSEC)**, v. 8, n. 1, p. 119–152, 2005.

CHEN, D.; ZHAO, H. Data security and privacy protection issues in cloud computing. In: IEEE. **Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on**. [S.l.], 2012. v. 1, p. 647–651.

CHOR, B.; KUSHILEVITZ, E.; GOLDREICH, O.; SUDAN, M. Private information retrieval. **Journal of the ACM (JACM)**, v. 45, n. 6, p. 965–981, 1998.

CHOUDHURY, A.; PATRA, A. An efficient framework for unconditionally secure multiparty computation. **IEEE Transactions on Information Theory**, v. 63, n. 1, p. 428–468, Jan 2017. ISSN 0018-9448.

CIRIANI, V.; VIMERCATI, S. D. C. D.; FORESTI, S.; JAJODIA, S.; PARABOSCHI, S.; SAMARATI, P. Keep a few: Outsourcing data while maintaining confidentiality. In: **Proceedings of the 14th European Conference on Research in Computer Security**. Springer-Verlag, 2009. (ESORICS'09), p. 440–455. ISBN 3-642-04443-3, 978-3-642-04443-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1813084.1813120>.

CIRIANI, V.; VIMERCATI, S. D. C. D.; FORESTI, S.; JAJODIA, S.; PARABOSCHI, S.; SAMARATI, P. Combining fragmentation and encryption to protect privacy in data storage. **ACM Transactions on Information and System Security (TISSEC)**, ACM, v. 13, n. 3, p. 22, 2010.

CLARKE, R. **Introduction to Dataveillance and Information Privacy, and Definitions of Terms**. 1999. Available from: http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html.

CLARKE, R. Privacy impact assessment: Its origins and development. **Computer law & security review**, v. 25, n. 2, p. 123–135, 2009. Elsevier.

CONSORTIUM, U. **The Unicode Standard, Version 3.0**. Addison-Wesley, 2000. (Programming languages, v. 1). ISBN 9780201616330. Disponível em: <https://books.google.com.br/books?id=-u5QAAAAMAAJ>.

CORREIA MIGUEL PUPO E SOUSA, P. J. **Segurança no software**. [S.l.]: Lisboa: Editora FCA, 2010.

DANEZIS, G.; DOMINGO-FERRER, J.; HANSEN, M.; HOEPMAN, J.; MÉTAYER, D. L.; TIRTEA, R.; SCHIFFNER, S. Privacy and data protection by design - from policy to engineering. **CoRR**, abs/1501.03726, 2015. Disponível em: <http://arxiv.org/abs/1501.03726>.

DEGELING, M.; LENTZSCH, C.; NOLTE, A.; HERRMANN, T.; LOSER, K. U. Privacy by socio-technical design: A collaborative approach for privacy friendly system design. In: **2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)**. [S.l.: s.n.], 2016. p. 502–505.

DELFS, H.; KNEBL, H. **Introduction to Cryptography: Principles and Applications**. [S.l.]: Springer Berlin Heidelberg, 2015. (Information Security and Cryptography). ISBN 9783662479742.

DETERMANN, L. **Determann's Field Guide to Data Privacy Law: International Corporate Compliance, Second Edition**. Edward Elgar Publishing, Incorporated, 2015. (Elgar Practical Guides). ISBN 9781783476893. Disponível em: <https://books.google.com.br/books?id=_oIZBgAAQBAJ>.

DIFFIE, W.; HELLMAN, M. New directions in cryptography. **IEEE Trans. Inf. Theor.**, IEEE Press, Piscataway, NJ, USA, v. 22, n. 6, p. 644–654, set. 2006. ISSN 0018-9448. Disponível em: <http://dx.doi.org/10.1109/TIT.1976.1055638>.

DOMINGO-FERRER, J. A survey of inference control methods for privacy-preserving data mining. In: ____. **Privacy-preserving data mining**. [S.l.]: Springer, 2008. p. 53–80.

DUBOVITSKAYA, A.; UROVI, V.; VASIRANI, M.; ABERER, K.; SCHUMACHER, M. I. A cloud-based ehealth architecture for privacy preserving data integration. In: SPRINGER. **IFIP International Information Security Conference**. [S.l.], 2015. p. 585–598.

DUNCAN, G. T.; KELLER-MCNULTY, S. A.; STOKES, S. L. Disclosure risk vs. data utility: The ru confidentiality map. In: **Chance**. [S.l.]: Citeseer, 2001.

ELLISON, C. Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure. **Computer Security Journal**, v. 16, n. 1, p. 1–7, 2000. Disponível em: <http://www.schneier.com/paper-pki.pdf>.

ENSSLIN, L.; ENSSLIN, S. R.; LACERDA, R. T. d. O.; TASCA, J. E. Proknow-c, knowledge development process-constructivist. **Processo técnico com patente de registro pendente junto ao INPI. Brasil**, v. 10, n. 4, p. 2015, 2010.

ESPINEL, V.; O'HALLORAN, D.; BRYNJOLFSSON, E.; O'SULLIVAN, D. *et al.* Survey report:"deep shift: Technology tipping points and societal impact.". In: **World Economic Forum, September**. [S.l.: s.n.], 2015.

EVFIMIEVSKI, A.; SRIKANT, R.; AGRAWAL, R.; GEHRKE, J. Privacy preserving mining of association rules. **Information Systems**, v. 29, n. 4, p. 343–364, 2004.

FIGUEIREDO, G.; BRAGANHOLO, V.; MATTOSO, M. Processing queries over distributed xml databases. **Journal of Information and Data Management**, v. 1, n. 3, p. 455, 2010.

FIPS, P. 46-3: Data encryption standard (des). **National Institute of Standards and Technology**, v. 25, n. 10, p. 1–22, 1999.

FRIEDMAN, W. F. **The index of coincidence and its applications in cryptanalysis**. [S.l.]: Aegean Park Press California, 1987.

FUGKEAW, S. Achieving privacy and security in multi-owner data outsourcing. In: **Seventh International Conference on Digital Information Management, ICDIM 2012, Macau, Macao, August 22-24, 2012**. [S.l.: s.n.], 2012. p. 239–244.

FUNG, B. C.; WANG, K.; FU, A. W.-C.; YU, P. S. **Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques**. [S.l.]: Chapman-Hall, 2010. 376 p.

FUNG, B. C. M.; KE, W.; YU, P. S. Anonymizing classification data for privacy preservation. **Knowledge and Data Engineering, IEEE Transactions on**, v. 19, n. 5, p. 711–725, 2007.

GAI, K.; QIU, M.; THURAISINGHAM, B.; TAO, L. Proactive attribute-based secure data schema for mobile cloud in financial industry. In: IEEE. **High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), 2015 IEEE 12th International Conferen on Embedded Software and Systems (ICESS), 2015 IEEE 17th International Conference on**. [S.l.], 2015. p. 1332–1337.

GARAY, J. A.; GENNARO, R.; JUTLA, C.; RABIN, T. Secure distributed storage and retrieval. **Theoretical Computer Science**, Elsevier, v. 243, n. 1, p. 363–389, 2000.

GAVISON, R. Privacy and the limits of law. **Yale Law Journal**, v. 89, p. 471, 1980.

GIESSING, S. Survey on methods for tabular data protection in argus. In: **Privacy in Statistical Databases**. [S.l.]: Springer, 2004. p. 1–13.

GLEICK, J. **The Information: A History, a Theory, a Flood**. Knopf Doubleday Publishing Group, 2011. ISBN 9780307379573. Disponível em: <https://books.google.com.br/books?id=617JSFW0D2kC>.

GOODSON, G. R.; WYLIE, J. J.; GANGER, G. R.; REITER, M. K. Efficient byzantine-tolerant erasure-coded storage. In: **Dependable Systems and Networks, 2004 International Conference on**. [S.l.: s.n.], 2004. p. 135–144.

GOPAL, R.; GARFINKEL, R.; GOES, P. Confidentiality via camouflage: The cvc approach to disclosure limitation when answering queries to databases. **Operations Research**, v. 50, n. 3, p. 501–516, 2002.

GOPAL, R. D.; GOES, P. B.; GARFINKEL, R. S. Interval protection of confidential information in a database. **INFORMS Journal on Computing**, v. 10, n. 3, p. 309–322, 1998.

GROBAUER, B.; WALLOSCHEK, T.; STOCKER, E. Understanding cloud computing vulnerabilities. **IEEE Security & Privacy**, IEEE, v. 9, n. 2, p. 50–57, 2011.

GRUSCHKA, N.; JENSEN, M. Attack surfaces: A taxonomy for attacks on cloud services. In: **Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on**. [S.l.: s.n.], 2010. p. 276–279.

GUERON, S.; FEGHALI, W. K.; GOPAL, V. **Architecture and instruction set for implementing advanced encryption standard (AES)**. [S.l.]: Google Patents, 2016. US Patent 9,230,120.

GUTTMAN, B.; ROBACK, E. A. **An introduction to computer security: the NIST handbook**. [S.l.]: DIANE Publishing, 1995.

HEGEL, G. W. F. **Enciclopédia das ciências filosóficas II-Filosofia da natureza**. [S.l.]: Edicoes Loyola, 1995. v. 2.

HERRMANN, T.; PRILLA, M.; NOLTE, A. Socio-technical process design—the case of coordinated service delivery for elderly people. In: **Blurring the Boundaries Through Digital Innovation**. [S.l.]: Springer, 2016. p. 217–229.

HON, W. K.; MILLARD, C.; WALDEN, I. The problem of 'personal data'in cloud computing: what information is regulated?—the cloud of unknowing. **International Data Privacy Law**, v. 1, n. 4, p. 211–228, 2011.

HOVEN, J. Van den; LOKHORST, G.-J.; POEL, I. Van de. Engineering and the problem of moral overload. **Science and engineering ethics**, v. 18, n. 1, p. 143–155, 2012. Springer.

HU, T. **A Prehistory of the Cloud**. MIT Press, 2015. 209 p. ISBN 9780262029513. Disponível em: <https://books.google.com.br/books?id=34hkCgAAQBAJ>.

II, R. M. T. **Domestic Drones and Privacy: a primer**. [S.l.]: Congressional Research Service, 2015. v. 43965.

IYENGAR, V. S. Transforming data to satisfy privacy constraints. In: **Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2002. (KDD '02), p. 279–288. ISBN 1-58113-567-X. Disponível em: <http://doi.acm.org/10.1145/775047.775089>.

JAMMALAMADAKA, R. C.; GAMBONI, R.; MEHROTRA, S.; SEAMONS, K.; VENKATASUBRAMANIAN, N. A middleware approach for outsourcing data securely. **computers & security**, Elsevier, v. 32, p. 252–266, 2013.

JANSEN, W.; GRANCE, T. Guidelines on security and privacy in public cloud computing. **NIST Special Publication**, p. 800–144, 2011.

JOSEPH, N. M.; DANIEL, E.; VASANTHI, N. A. Article: Survey on privacy-preserving methods for storage in cloud computing. **IJCA Proceedings on Amrita International Conference of Women in Computing - 2013**, AICWIC, n. 4, p. 1–4, jan. 2013. Full text available.

JUNG, T.; LI, X.-y.; WAN, Z.; WAN, M. Privacy preserving cloud data access with multi-authorities. In: **INFOCOM, 2013 Proceedings IEEE**. [S.l.: s.n.], 2013. p. 2625–2633.

JUNG, T.; LI, X.-Y.; WAN, Z.; WAN, M. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. **IEEE Transactions on Information Forensics and Security**, IEEE, v. 10, n. 1, p. 190–199, 2015.

KAHN, D. **The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet**. Scribner, 1996. ISBN 9781439103555. Disponível em: <https://books.google.com.br/books?id=SEH_rHkgaogC>.

KANTARCIOĞLU, M.; CLIFTON, C. Security issues in querying encrypted data. In: ____. **Data and Applications Security XIX: 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA, August 7-10, 2005. Proceedings**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 325–337. ISBN 978-3-540-31937-5. Disponível em: <http://dx.doi.org/10.1007/11535706_24>.

KEEN, A. **Digital Vertigo: How Today's Online Social Revolution Is Dividing, Diminishing, and Disorienting Us**. St. Martin's Press, 2012. ISBN 9780312624989. Disponível em: <https://books.google.com.br/books?id=9T-RgMe9a9wC>.

KLING, P.; ÖZSU, M. T.; DAUDJEE, K. Generating efficient execution plans for vertically partitioned xml databases. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 4, n. 1, p. 1–11, 2010.

KLING, P.; ÖZSU, M. T.; DAUDJEE, K. Scaling xml query processing: distribution, localization and pruning. **Distributed and Parallel Databases**, Springer, v. 29, n. 5-6, p. 445, 2011.

KOCHER, P. C. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In: SPRINGER. **Annual International Cryptology Conference**. [S.l.], 1996. p. 104–113.

KRAWCZYK, H. Distributed fingerprints and secure information dispersal. In: **Proceedings of the Twelfth Annual ACM Symposium on Principles of Distributed Computing**. New York, NY, USA: ACM, 1993. (PODC '93), p. 207–218. ISBN 0-89791-613-1. Disponível em: <http://doi.acm.org/10.1145/164051.164075>.

KRUTZ, R. L.; VINES, R. D. **Cloud security: A comprehensive guide to secure cloud computing**. [S.l.]: Wiley, 2010.

LANE, A. **Understanding and Selecting Data Masking Solutions: Creating Secure and Useful Data**. 2012. 33 p. Securosis, L.L.C. 515 E. Carefree Blvd. Suite 766 Phoenix, AZ.

LAURANT, C. Privacy international: Privacy and human rights 2003: an international survey of privacy laws and developments, electronic privacy information center (epic), privacy international. **Privacy International. http://www. privacyinternational. org/survey/phr2003**, 2003.

LEFEVRE, K.; DEWITT, D. J.; RAMAKRISHNAN, R. Incognito: Efficient full-domain k-anonymity. In: ACM. **Proceedings of the 2005 ACM SIGMOD international conference on Management of data**. [S.l.], 2005. p. 49–60.

LEFEVRE, K.; DEWITT, D. J.; RAMAKRISHNAN, R. Mondrian multidimensional k-anonymity. In: IEEE. **22nd International Conference on Data Engineering (ICDE'06)**. [S.l.], 2006. p. 25–25.

LI, M.; YU, S.; REN, K.; LOU, W.; HOU, Y. T. Toward privacy-assured and searchable cloud data storage services. **IEEE Network**, v. 27, n. 4, p. 56–62, July 2013. ISSN 0890-8044.

LI, N.; LI, T.; VENKATASUBRAMANIAN, S. t-closeness: Privacy beyond k-anonymity and l-diversity. In: **In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07)**. [S.l.: s.n.], 2007. p. 106–115.

LIANG, H.; YUAN, H. On the complexity of t-closeness anonymization and related problems. In: **Database Systems for Advanced Applications**. [S.l.]: Springer, 2013. p. 331–345.

LIU, J.; XIAO, Y.; LI, S.; LIANG, W.; CHEN, C. L. P. Cyber security and privacy issues in smart grids. **Communications Surveys & Tutorials, IEEE**, v. 14, n. 4, p. 981–997, 2012.

MELL, P. M.; GRANCE, T. **SP 800-145. The NIST Definition of Cloud Computing**. Gaithersburg, MD, United States, 2011.

MENEZES, A. J.; OORSCHOT, P. C. V.; VANSTONE, S. A. **Handbook of applied cryptography**. [S.l.]: CRC press, 1996.

MEYERSON, A.; WILLIAMS, R. On the complexity of optimal k-anonymity. In: ACM. **Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**. [S.l.], 2004. p. 223–228.

MIRANDA, F. B.; FARIAS, V. S. Princípios para o desenvolvimento de uma teoria da definição lexicográfica. **Alfa: Revista de Linguistica**, v. 55, n. 1, 2011.

MITCHELL, C. J. On the security of 2-key triple des. **IEEE Transactions on Information Theory**, v. 62, n. 11, p. 6260–6267, Nov 2016. ISSN 0018-9448.

MOHAMMED, N.; FUNG, B. C.; HUNG, P. C.; LEE, C.-k. Anonymizing healthcare data: A case study on the blood transfusion service. In: **Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2009. (KDD '09), p. 1285–1294. ISBN 978-1-60558-495-9. Disponível em: <http://doi.acm.org/10.1145/1557019.1557157>.

NIMGAONKAR, S.; KOTIKELA, S.; GOMATHISANKARAN, M. Ctrust: A framework for secure and trustworthy application execution in cloud computing. In: **Cyber Security (CyberSecurity), 2012 International Conference on**. [S.l.: s.n.], 2012. p. 24–31.

NING, C.; CONG, W.; MING, L.; KUI, R.; WENJING, L. Privacy-preserving multi-keyword ranked search over encrypted cloud data. **Parallel and Distributed Systems, IEEE Transactions on**, v. 25, n. 1, p. 222–233, 2014.

NOTARIO, N.; CRESPO, A.; MARTÍN, Y.-S.; ALAMO, J. M. D.; MÉTAYER, D. L.; ANTIGNAC, T.; KUNG, A.; KROENER, I.; WRIGHT, D. Pripare: integrating privacy best practices into a privacy engineering methodology. In: IEEE. **Security and Privacy Workshops (SPW), 2015 IEEE**. [S.l.], 2015. p. 151–158.

NOURIAN, A.; MAHESWARAN, M. Using segmentation for confidentiality aware image storage and retrieval on clouds. In: **Global Communications Conference (GLOBECOM), 2012 IEEE**. [S.l.: s.n.], 2012. p. 758–763.

OKMAN, L.; GAL-OZ, N.; GONEN, Y.; GUDES, E.; ABRAMOV, J. Security issues in nosql databases. In: **Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on**. [S.l.: s.n.], 2011. p. 541–547.

OLUMOFIN, F.; GOLDBERG, I. Revisiting the computational practicality of private information retrieval. In: ____. **Financial Cryptography and Data Security**. [S.l.]: Springer, 2012. p. 158–172.

PEARSON, S. **Privacy, Security and Trust in Cloud Computing**. [S.l.]: Springer, 2013. 3-42 p.

PFITZMANN, A.; KöHNTOPP, M. Anonymity, unobservability, and pseudonymity—a proposal for terminology. In: **Designing privacy enhancing technologies**. [S.l.]: Springer, 2005. p. 1–9.

PFLEEGER, C. P.; PFLEEGER, S. L. **Security in Computing (4th Edition)**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2006. ISBN 0132390779.

PIETERS, W. Explanation and trust: what to tell the user in security and ai? **Ethics and information technology**, v. 13, n. 1, p. 53–64, 2011. Springer.

POPA, R. A.; REDFIELD, C. M. S.; ZELDOVICH, N.; BALAKRISHNAN, H. Cryptdb: Protecting confidentiality with encrypted query processing. In: **Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles**. New York, NY, USA: ACM, 2011. (SOSP '11), p. 85–100. ISBN 978-1-4503-0977-6. Disponível em: <http://doi.acm.org/10.1145/2043556.2043566>.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8ª Edição**. [S.l.]: McGraw Hill Brasil, 2016.

QUEIROZ, M. J.; LINO, N. C.; MOTTA, G. Uma ontologia de domínio para preservação de privacidade em dados publicados pelo governo brasileiro. **XII Simpósio Brasileiro de Sistemas de Informação. Florianópolis, SC, Brasil**, 2016.

RABIN, M. O. Efficient dispersal of information for security, load balancing, and fault tolerance. **J. ACM**, ACM, New York, NY, USA, v. 36, n. 2, p. 335–348, abr. 1989. ISSN 0004-5411. Disponível em: <http://doi.acm.org/10.1145/62044.62050>.

RABIN, M. O. Efficient dispersal of information for security, load balancing, and fault tolerance. **Journal of the ACM (JACM)**, ACM, v. 36, n. 2, p. 335–348, 1989.

RAGHAVAN, B.; VISHWANATH, K.; RAMABHADRAN, S.; YOCUM, K.; SNOEREN, A. C. Cloud control with distributed rate limiting. **ACM SIGCOMM Computer Communication Review**, ACM, v. 37, n. 4, p. 337–348, 2007.

RAO, K. K.; YADAV, S. K. Implementation of cloud storage security mechanism using digital signature. **International Journal of Current Engineering and Technology**, p. 467–471, 2016.

REBOLLO-MONEDERO, D.; FORNE, J.; DOMINGO-FERRER, J. From t-closeness-like privacy to postrandomization via information theory. **Knowledge and Data Engineering, IEEE Transactions on**, v. 22, n. 11, p. 1623–1636, 2010.

REKATSINAS, T.; DESHPANDE, A.; MACHANAVAJJHALA, A. Sparsi: Partitioning sensitive data amongst multiple adversaries. **Proc. VLDB Endow.**, VLDB Endowment, v. 6, n. 13, p. 1594–1605, ago. 2013. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2536258.2536270>.

RESCH, J. K.; PLANK, J. S. Aont-rs: blending security and performance in dispersed storage systems. In: **Proceedings of FAST-2011: 9th Usenix Conference on File and Storage Technologies,February 2011**. [S.l.: s.n.], 2011.

RHEA, S.; WELLS, C.; EATON, P.; GEELS, D.; ZHAO, B.; WEATHERSPOON, H.; KUBIATOWICZ, J. Maintenance-free global data storage. **Internet Computing, IEEE**, v. 5, n. 5, p. 40–49, 2001. 1089-7801.

RISTENPART, T.; TROMER, E.; SHACHAM, H.; SAVAGE, S. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds. In: **Proceedings of the 16th ACM Conference on Computer and Communications Security**. New York, NY, USA: ACM, 2009. (CCS '09), p. 199–212. ISBN 978-1-60558-894-0. Disponível em: <http://doi.acm.org/10.1145/1653662.1653687>.

RIZVI, S. J.; HARITSA, J. R. Maintaining data privacy in association rule mining. In: **Proceedings of the 28th international conference on Very Large Data Bases**. [S.l.]: VLDB Endowment, 2002. p. 682–693.

RODRIGUES, C.; BRAGANHOLO, V.; MATTOSO, M. Virtual partitioning ad-hoc queries over distributed xml databases. **Journal of Information and Data Management**, v. 2, n. 3, p. 495, 2011.

RYAN, M. D. Cloud computing privacy concerns on our doorstep. **Commun. ACM**, ACM, New York, NY, USA, v. 54, n. 1, p. 36–38, jan. 2011. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1866739.1866751>.

SAMARATI, P. Protecting respondents identities in microdata release. **Knowledge and Data Engineering, IEEE Transactions on**, v. 13, n. 6, p. 1010–1027, 2001.

SAMARATI, P. Data security and privacy in the cloud. In: **Information Security Practice and Experience - 10th International Conference, ISPEC 2014, Fuzhou, China, May 5-8, 2014. Proceedings**. [s.n.], 2014. p. 28–41. Disponível em: <http://dx.doi.org/10.1007/978-3-319-06320-1_4>.

SAMARATI, P.; SWEENEY, L. **Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression**. [S.l.], 1998.

SAMARATI, P.; VIMERCATI, S. D. C. di. Data protection in outsourcing scenarios: Issues and directions. In: **Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security**. ACM, 2010. (ASIACCS '10), p. 1–14. ISBN 978-1-60558-936-7. Disponível em: <http://doi.acm.org/10.1145/1755688.1755690>.

SANTOS, E. **Avaliação de Consultas Executadas sobre Bases de Dados de Proveniência Distribuídas**. 2013. Monografia (Bacharelado em Engenharia de Computação e Informação) UFRJ -Universidade Federal do Rio de Janeiro.

SAYI, T. J. V. R. K.; KRISHNA, R. K. N. S.; MUKKAMALA, R.; BARUAH, P. K. Data outsourcing in cloud environments: A privacy preserving approach. In: **Information Technology: New Generations (ITNG), 2012 Ninth International Conference on**. [S.l.: s.n.], 2012. p. 361–366.

SCHNEIER, B. **Applied Cryptography: Protocols, Algorithms and Source Code in C**. Wiley, 2015. 784 p. ISBN 9781119096726. Disponível em: <https://books.google.com.br/books?id=VjC9BgAAQBAJ>.

Sedghi, S. **Towards provably secure efficiently searchable encryption**. Tese (Doutorado) — University of Twente, Enschede, the Netherlands, February 2012. IPA dissertation series ; no. 2012-5. Disponível em: <http://doc.utwente.nl/79737/>.

SEI, Y.; OHSUGA, A. Randomized addition of sensitive attributes for l-diversity. In: **2014 11th International Conference on Security and Cryptography (SECRYPT)**. [S.l.: s.n.], 2014. p. 1–11.

SEIGER, R.; GROSS, S.; SCHILL, A. Seccsie: a secure cloud storage integrator for enterprises. In: IEEE. **Commerce and Enterprise Computing (CEC), 2011 IEEE 13th Conference on**. [S.l.], 2011. p. 252–255.

SHAMIR, A. How to share a secret. **Commun. ACM**, ACM, New York, NY, USA, v. 22, n. 11, p. 612–613, nov. 1979. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/359168.359176>.

SHANNON, C. E. Communication theory of secrecy systems. **Bell system technical journal**, Wiley Online Library, v. 28, n. 4, p. 656–715, 1949.

SHANNON, C. E. A mathematical theory of communication. **SIGMOBILE Mob. Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 5, n. 1, p. 3–55, jan. 2001. ISSN 1559-1662. Disponível em: <http://doi.acm.org/10.1145/584091.584093>.

SHIREY, R. **Internet Security Glossary**. IETF, 2000. RFC 2828 (Informational). (Request for Comments, 2828). Obsoleted by RFC 4949. Disponível em: <http://www.ietf.org/rfc/rfc2828.txt>.

SIBSON, P.; ERVEN, C.; GODFREY, M.; MIKI, S.; YAMASHITA, T.; FUJIWARA, M.; SASAKI, M.; TERAI, H.; TANNER, M. G.; NATARAJAN, C. M. *et al.* Chip-based quantum key distribution. **Nature Communications**, Nature Publishing Group, v. 8, 2017. Disponível em: <http://doi.org/10.1038/ncomms13984>.

SIEWIOREK, D.; SWARZ, R. **Reliable computer systems: design and evaluation**. Digital Press, 1992. (Computer Technology Series). Disponível em: <https://books.google.com.br/books?id=94lQAAAAMAAJ>.

SILVA, T. L. da; BAIãO, F. A.; SAMPAIO, J. de O.; MATTOSO, M.; BRAGANHOLO, V. Recomendações para fragmentação horizontal de bases de dados xml. In: **SBBD (Short Papers)**. [S.l.: s.n.], 2012. p. 145–152.

SLAMANIG, D.; HANSER, C. On cloud storage and the cloud of clouds approach. In: **Internet Technology And Secured Transactions, 2012 International Conferece For**. [S.l.: s.n.], 2012. p. 649–655.

SOLOVE, D. **Understanding Privacy**. Harvard University Press, 2008. (Understanding privacy, v. 10). ISBN 9780674027725. Disponível em: <https://books.google.com.br/books?id=XU5-AAAAMAAJ>.

SPIEKERMANN, S.; CRANOR, L. F. Engineering privacy. **Software Engineering, IEEE Transactions on**, v. 35, n. 1, p. 67–82, 2009.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 6rd. ed. [S.l.]: Pearson Education, 2006.

STALLINGS, W. **Network Security Essentials: Applications and Standards**. 4th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0136108059, 9780136108054.

STEFANOV E. SHI, E. Oblivistore: High performance oblivious cloud storage. In: **Security and Privacy (SP), 2013 IEEE Symposium on**. [S.l.: s.n.], 2013. p. 253–267.

STORER, M. W.; GREENAN, K. M.; MILLER, E. L.; VORUGANTI, K. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In: **Proceedings of the 6th USENIX Conference on File and Storage Technologies**. [S.l.]: USENIX Association, 2008. p. 1.

STORER, M. W.; GREENAN, K. M.; MILLER, E. L.; VORUGANTI, K. Potshards—a secure, recoverable, long-term archival storage system. **ACM Transactions on Storage (TOS)**, ACM, v. 5, n. 2, p. 5, 2009.

STRUMBUDAKIS, A. C. d. S. **FairSky: gestão confiável e otimizada de dados em múltiplas nuvens de armazenamento na internet**. Tese (Doutorado) — Faculdade de Ciências e Tecnologia, 2014.

SUBASHINI, S.; KAVITHA, V. Review: A survey on security issues in service delivery models of cloud computing. **J. Netw. Comput. Appl.**, v. 34, n. 1, p. 1–11, 2011.

SUBBIAH, A.; BLOUGH, D. M. An approach for fault tolerant and secure data storage in collaborative work environments. In: **Proceedings of the 2005 ACM Workshop on Storage Security and Survivability**. New York, NY, USA: ACM, 2005. (StorageSS '05), p. 84–93. ISBN 1-59593-233-X. Disponível em: <http://doi.acm.org/10.1145/1103780.1103793>.

SUN, H.; JAFAR, S. A. The capacity of private information retrieval. **IEEE Transactions on Information Theory**, PP, n. 99, p. 1–1, 2017. ISSN 0018-9448.

SWEENEY, L. k-anonymity: A model for protecting privacy. **International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems**, v. 10, n. 05, p. 557–570, 2002.

TANENBAUM, A. S.; WOODHULL, A. S. **Sistemas Operacionais: Projetjos e Implementação**. [S.l.]: Bookman Editora, 2009.

TANG, J.; CUI, Y.; LI, Q.; REN, K.; LIU, J.; BUYYA, R. Ensuring security and privacy preservation for cloud data services. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 49, n. 1, p. 13:1–13:39, jun. 2016. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2906153>.

TESHOME, A.; RILLING, L.; MORIN, C. **Including Security Monitoring in Cloud SLA**. 2016.

TIAN, M.; ZHANG, Y. Analysis of cloud computing and its security. In: **International Symposium on Information Technology in Medicine and Education (ITME)**. [S.l.: s.n.], 2012. (TIME '12).

VENKATARAMANAN, N.; SHRIRAM, A. **Data Privacy: Principles and Practice**. CRC Press, 2016. ISBN 9781498721059. Disponível em: <https://books.google.com.br/books?id=iCANDgAAQBAJ>.

WACKS, R. **Privacy. A very short introduction**. Oxford ; New York: Oxford University Press, 2010. v. 221. XVI, 160 p. (Very short introductions, v. 221).

WANG, K.; FUNG, B. C.; PHILIP, S. Y. Handicapping attacker's confidence: an alternative to k-anonymization. **Knowledge and Information Systems**, Springer, v. 11, n. 3, p. 345–368, 2007.

WANG, K.; FUNG, B. C. M. Anonymizing sequential releases. In: **Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2006. p. 414–423. ISBN 1-59593-339-5. Disponível em: <http://doi.acm.org/10.1145/1150402.1150449>.

WANG, L.-J.; ZOU, K.-H.; SUN, W.; MAO, Y.; ZHU, Y.-X.; YIN, H.-L.; CHEN, Q.; ZHAO, Y.; ZHANG, F.; CHEN, T.-Y. *et al.* Long-distance copropagation of quantum key distribution and terabit classical optical data channels. **Physical Review A**, APS, v. 95, n. 1, p. 012301, 2017.

WARREN, S. D.; BRANDEIS, L. D. The right to privacy. **Harvard law review**, JSTOR, p. 193–220, 1890.

WESTIN, A. F. Privacy and freedom, atheneum. **New York**, v. 7, 1967.

WIESE, L. Horizontal fragmentation for data outsourcing with formula-based confidentiality constraints. In: ____. **Advances in Information and Computer Security**. [S.l.]: Springer, 2010. p. 101–116.

WILLENBORG, L.; WAAL, T. D. **Elements of statistical disclosure control**. [S.l.]: Springer Science & Business Media, 2012. v. 155.

WINKLER, J. R. V. **Securing the Cloud: Cloud Computer Security Techniques and Tactics**. [S.l.]: Syngress Publishing, 2011. ISBN 1597495921, 9781597495929.

XIA, Z.; WANG, X.; SUN, X.; WANG, Q. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. **IEEE Transactions on Parallel and Distributed Systems**, v. 27, n. 2, p. 340–352, Feb 2016. ISSN 1045-9219.

XIAO, X.; TAO, Y.; CHEN, M. Optimal random perturbation at multiple privacy levels. **Proc. VLDB Endow.**, Very Large Data Base Endowment Inc., v. 2, n. 1, p. 814–825, ago. 2009. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/1687627.1687719>.

XIAO, Z.; XIAO, Y. Security and privacy in cloud computing. **IEEE Communications Surveys Tutorials**, v. 15, n. 2, p. 843–859, Second 2013. ISSN 1553-877X.

XU, X.; XIONG, L.; LIU, J. Database fragmentation with confidentiality constraints: A graph search approach. In: **Proceedings of the 5th ACM Conference on Data and Application Security and Privacy**. ACM, 2015. (CODASPY '15), p. 263–270. ISBN 978-1-4503-3191-3. Disponível em: <http://doi.acm.org/10.1145/2699026.2699121>.

YANG, K.; JIA, X.; REN, K.; ZHANG, B.; XIE, R. Dac-macs: Effective data access control for multiauthority cloud storage systems. **IEEE Transactions on Information Forensics and Security**, v. 8, n. 11, p. 1790–1801, 2013.

YANG, K.; ZHANG, J.; ZHANG, W.; QIAO, D. A light-weight solution to preservation of access pattern privacy in un-trusted clouds. In: **Proceedings of the 16th European Conference on Research in Computer Security**. Berlin, Heidelberg: Springer-Verlag, 2011. (ESORICS'11), p. 528–547. ISBN 978-3-642-23821-5. Disponível em: <http://dl.acm.org/citation.cfm?id=2041225.2041263>.

YEH, C.-H. A secure shared group model of cloud storage. In: **Proceedings of the 2013 27th International Conference on Advanced Information Networking and Applications Workshops**. Washington, DC, USA: IEEE Computer Society, 2013. (WAINA '13), p. 663–667. ISBN 978-0-7695-4952-1. Disponível em: <http://dx.doi.org/10.1109/WAINA.2013.231>.

ZHANG, X.; DOU, W.; PEI, J.; NEPAL, S.; YANG, C.; LIU, C.; CHEN, J. Proximity-aware local-recoding anonymization with mapreduce for scalable big data privacy preservation in cloud. **IEEE transactions on computers**, IEEE, v. 64, n. 8, p. 2293–2307, 2015.

ZHANG, X.; LIU, Q.; LIU, D.; XIE, W. A survey on anonymity for privacy preserving data mining. In: CRC PRESS. **Information Science and Electronic Engineering: Proceedings of the 3rd International Conference of Electronic Engineering and Information Science (ICEEIS 2016), 4-5 January, 2016, Harbin, China**. [S.l.], 2016. p. 343–346.

APÊNDICE A – ALGORITMO DE FRAGMENTAÇÃO

Algoritmo 15: (Fq.bin,Fs.bin,Fm.bin) ← Frag(F))

Entrada : um arquivo digital F
Saída : três arquivos que contêm as propriedades Q, S e M do arquivo F

```
1  início
2      define SIZEBLOC 57344;
3      define QUANTBLOC 224;
4      k := 256;
5      Set(Q[k],0);
6      Set(S[k],0);
7      Set(M[k],0);
8      Set(iOBJ[k],0);
9      bloc:= Read(F,SIZEBLOC);
10     CreateFile(Fq.bin);
11     CreateFile(Fs.bin);
12     CreateFile(Fm.bin);
13     Enquanto size(bloc) > 0 faça
14         para b = 1 to QUANTBLOC faça
15             para n = 1 to k faça
16                 iOBJ[n] := bloc[i];
17                 i ++;
18             fim para
19             Dec(iOBJ);
20         fim para
21         bloc:= Read(F,SIZEBLOC);
22         se SIZEBLOC > size(bloc) então
23             QUANTBLOC := size(bloc) / k;
24         fim se
25     fim-enquanto
26     Append(Fq.bin,Q);
27     Append(Fs.bin,S);
28     Append(Fm.bin,M);
29 fim
```

APÊNDICE B – ALGORITMO DE DECOMPOSIÇÃO

Implementação de (Q,S,M) ← Fdec(iOBJ) (**Algoritmo 16**). Na primeira parte do algoritmo (linhas 12 a 17), os bytes do iOBJ são lidos um a um sequencialmente e as informações das posições dos bytes são armazenadas em um vetor bidimensional $M_{ALL}[256][256]$, onde a primeira dimensão do vetor representa o valor decimal do *byte* e a segunda dimensão representa a posição ocupada pelo *byte* no iOBJ.

A segunda parte do algoritmo (linhas 20 a 32) utiliza dois vetores de 256 bits ($Q_{BA}[256]$ e $S_{BA}[256]$) para guardar as informações da qualidade e quantidade. No início do algoritmo, os vetores $Q_{BA}[256]$ e $S_{BA}[256]$ são preenchidos com o valor 0 (linhas 3 e 4). A posição *byte* do vetor $\mathbf{Q}_{BA}$ ($Q_{BA}[byte]$) é preenchida com o valor 1 (linha 22) se o *byte* existir no iOBJ.

O vetor de *bits* $S_{BA}[256]$ é preenchido com o valor 1, caso a frequência de ocorrência do *byte* no iOBJ seja maior do que 1, (linha 25). Por exemplo, o valor 1 em $S_{BA}[20]$ representa a existência de várias ocorrências do 20^{o} *byte* do vetor $Q_{BA}[256]$ no iOBJ, enquanto o valor 0 em $S_{BA}[20]$ representa uma única ocorrência do 20^{o} *byte* do vetor $Q_{BA}[256]$ no iOBJ.

As posições dos bytes do iOBJ são armazenadas na matriz $\mathbf{M}_{ALL}$, na segunda dimensão da matriz em ordem crescente. Caso uma linha da matriz tenha mais de um elemento, os dois últimos elementos dessa linha serão colocados em ordem decrescente para indicar o final do grupo de posições do byte naquela linha. (linhas 26 A 28)

A matriz $\mathbf{M}_{ALL}$ é anonimizada (linha 33), gerando o vetor $\mathbf{M}_E$ que é utilizado em seguida para anonimizar os vetores da qualidade e quantidade ($Q_{BA}[256]$ e $S_{BA}[256]$) que serão embaralhados para esconder o padrão de ocorrência e frequência dos *bytes* do arquivo **F** (linhas 34 e 35). A rotina de anonimização dos vetores da qualidade e quantidade será descritas na seção 6.4.2.2 desse capítulo.

Por fim, o vetor da medida ($M_E[256]$) é decomposto em dois vetores: $\mathbf{M}_G$ e $\mathbf{M}_K$ (linha 36) e o vetor $\mathbf{M}_K$ é compactado com o objetivo de reduzir o espaço de armazenamento do arquivo **Fm.bin**. O conteúdo dos vetores $\mathbf{M}_G$ e $\mathbf{M}_K$ são concatenados e armazenados no vetor $\mathbf{M}_E$. O algoritmo retorna as informações das características do objeto de informação nos vetores $\mathbf{Q}_E$, $\mathbf{S}_E$ e $\mathbf{M}_E$.

Algoritmo 16: (Q,S,M) ← Fdec(iOBJ)

Entrada : IObj: iobjeto do arquivo F
Saída : três vetores $Q_E[256]$, $S_E[256]$ e $M_E[256]$

início
  $Set(Q[256],0)$;
  $Set(Q_{BA}[256],0)$;
  $Set(S_{BA}[256],0)$;
  $Set(M_{ALL}[256][256],0)$;
  $Set(M_E[256],0)$;
  $Set(M_G[256],0)$;
  $Set(M_K[256],0)$;
  $Set(freq[256],0)$;
  $byte$:= 0;
  $cont$:= 0;
  para *pos = 0 to 255* **faça**
    $byte$:= $IObj[pos]$;
    $freq[byte]$ ++;
    $cont$:= $freq[byte]$;
    $M_{ALL}[byte][cont]$:= pos;
  fim para
  $postemp$:= 0;;
  $cont2$:= 0;
  para *byte = 0 to 255* **faça**
    se $freq[byte] > 0$ **então**
      $Q_{BA}[byte]$:= 1;
      $Q[cont2]$:= byte;
      se $freq[byte] > 1$ **então**
        $S_{BA}[cont2]$:= 1;
        $postemp$:= $M_{ALL}[byte][freq[byte]]$;
        $M_{ALL}[byte][freq[byte]]$:= $M_{ALL}[byte][freq[byte]-1]$;
        $M_{ALL}[byte][freq[byte]-1]$:= $postemp$;
      fim se
      $cont2$ ++;
    fim se
  fim para
  $M_E := Fp_1(M_{ALL},Q,Q_{BA},S_{BA})$;
  $Q_E := Fp_2(Q_{BA},M_E)$;
  $S_E := Fp_3(S_{BA},M_E)$;
  $(M_G,M_K) := fa(M_E)$;
  $M_K \leftarrow M_K \oplus S_E \oplus Q_E$;
  $M_E := join(M_G,M_K)$;
fim

APÊNDICE C – ALGORITMO DE RECOMPOSIÇÃO

Implementação de (F) $\leftarrow$ Frec(Fm.bin, Fq.bin, Fs.bin) (**Algoritmo 17**). Este algoritmo implementa o processo de recomposição dos objetos de informação do arquivo **F**, a partir das características dos iobjetos que estão nos arquivos **Fq.bin**, **Fs.bin** e **F.bin** dispersos em 3 nuvens distintas.

O Algoritmo 17 recebe como entrada os arquivos **Fq.bin**, **Fs.bin** e **F.bin** e processa as informações da Qualidade, Quantidade e Medida para criar o arquivo **F**.

As seguintes operações são realizadas:

1. A operação *XOR* entre os elementos de M_K, Q_E e S_E é realizada para descriptografar o valor de MP_K: $M_K \oplus QE \oplus SE$.
2. A função inversa Fa^{-1} recebe como entrada os elementos de M_K que são convertidos de códigos Lehmer para conjuntos de 8 elementos.
3. A função inversa Fa^{-1} recompõe o vetor M_E a partir dos vetores M_G e M_K.
4. A função inversa Fp_2^{-1} recebe como entrada M_E e Q_E e produz como saída o vetor Q_{BA}.
5. A função inversa Fp_3^{-1} recebe como entrada M_E e S_E e produz como saída o vetor S_{BA}.
6. A função Fp_1^{-1} utiliza os vetores Q_{BA} e S_{BA} para desembaralhar o vetor M_E, gerando M.
7. Por último, a função Frec utiliza as propriedade da Medida (M), Qualidade (Q_{BA}) e Quantidade(S_{BA}) para recriar o objeto de informação.

Algoritmo 17: (F) ← Frec(Fm.bin, Fq.bin, Fs.bin)

Entrada : arquivos Fq.bin, Fs.bin, Fm.bin
Saída : arquivo F

```
1  início
2      CriarArquivo(F);
3      Q_E[256] = Read(Fq.bin, 256);
4      S_E[256] = Read(Fs.bin, 256);
5      M_E[256] = Read(Fm.bin, 256);
6      Enquanto não for fim de Q faça
7          (M_G, M_K) ← M_E;
8          M_K ← M_K ⊕ S_E ⊕ Q_E;
9          M_K ← LehmerCodeDecrypt(M_K);
10         M_E ← Fa^{-1}(M_G, M_K);
11         S_BA ← Fp_3^{-1}(S_E, M_E);
12         Q_BA ← Fp_2^{-1}(Q_E, M_E);
13         M ← Fp_1^{-1}(Q, S, M_E);
14         ordem = 0; cont = 0; pos = 0;
15         para bit = 0 to 255 faça
16             se Q_BA[bit] ≠ 0 então
17                 pos ← M[ordem];
18                 IObj[pos] ← CodASCII(Q[bit]);
19                 ordem ++;
20                 se S_BA[cont] = 1 então
21                     Enquanto M[ordem] ⩾ pos faça
22                         pos ← M[ordem];
23                         IObj[pos] ← CodASCII(Q[bit]);
24                         ordem ++;
25                     fim-enquanto
26                     pos ← M[ordem];
27                     IObj[pos] ← CodASCII(Q[bit]);
28                     ordem ++;
29                 fim se
30                 cont ++;
31             fim se
32         fim para
33         Gravar(F, IObj);
34         Q_E[256] = Read(Fq.bin, 256);
35         S_E[256] = Read(Fs.bin, 256);
36         M_E[256] = Read(Fm.bin, 256);
37     fim-enquanto
38 fim
```

APÊNDICE D – ALGORITMOS DE COMPRESSÃO

Algoritmo 18: ($M_K[64]$) ← LehmerCodeEncrypt($M_T[32][8]$)

Entrada : $M_T[32][8]$: Matriz de permutação de 8 números
Saída : $M_K[64]$: Matriz de Códigos Lehmer

início
  $cria\_vetor\_de\_bytes(G[8])$;
  $cria\_matriz\_de\_bytes(GNTI[8][8][8][8][8][8][8][8][1])$;
  $cria\_matriz\_de\_bytes(M_{K2}[64])$;
  para *n = 0 to 7* **faça**
    G[n] = n + 1;
  fim para
  Gera-G-NTI(G[8],0,8);
  a = 0;
  para *i = 0 to 32* **faça**
    num-fat =
    $GNTI[M_T[i][0]][M_T[i][1]][M_T[i][2]][M_T[i][3]][M_T[i][4]][M_T[i][5]][M_T[i][6]][M_T[i][7]][0]$;
    M_K[a] = num-fat;
    num-fat =
    $GNTI[M_T[i][0]][M_T[i][1]][M_T[i][2]][M_T[i][3]][M_T[i][4]][M_T[i][5]][M_T[i][6]][M_T[i][7]][1]$;
    M_K[a+1] = num-fat;
    a = a + 2;
  fim para
fim

Algoritmo 19: ($M_T[32][8]$) ← LehmerCodeDecrypt($M_K[64]$)

Entrada : $M_K[64]$: Matriz de Códigos Lehmer
Saída : $M_T[32][8]$:Matriz de permutação de 8 números

```
1  início
2      cria_vetor_de_bytes(G[8]);
3      cria_matriz_de_bytes(GITN[40320][8]);
4      preenche_matriz(GITN,0);
5      para n = 0 to 7 faça
6          G[n] = n + 1;
7      fim para
8      Gera-G-ITN(G[8],0,8);
9      a = 0;
10     para i = 0 to 32 faça
11         num1 = (255 X M_K[a]);
12         num2 = M_K[a+1];
13         num = num1 + num2;
14         M_T[i][1] = GITN[num][0];
15         M_T[i][2] = GITN[num][1];
16         M_T[i][3] = GITN[num][2];
17         M_T[i][4] = GITN[num][3];
18         M_T[i][5] = GITN[num][4];
19         M_T[i][6] = GITN[num][5];
20         M_T[i][7] = GITN[num][6];
21         M_T[i][8] = GITN[num][7];
22         a = a + 2;
23     fim para
24 fim
```

Algoritmo 20: (GNTI) ← Gera-G-NTI(G[8],p-num1, p-qt-num)

Entrada : G[8] : permutação de 8 números , primeiro número: p-num1, quantidade de números: p-qt-num

Saída : GNTI: Matriz de Conversão de Número para Índice

```
1  início
2      se (p-num1 = p-qt-num - 1);
3       então
4          n1 = G[0]-1; n2 = G[1]-1; n3 = G[2]-1; n4 = G[3]-1 ;
5          n5 = G[4]-1; n6 = G[5]-1; n7 = G[6]-1; n8 = G[7]-1 ;
6          se n ≤ 255 então
7              num1 = 0;
8              num2 = n;
9          fim se
10         senão
11             num1 = n ÷ 255;
12             num2 = mod255(n);
13         fim se
14         GNTI[n1][n2][n3][n4][n5][n6][n7][n8][0] = num1;
15         GNTI[n1][n2][n3][n4][n5][n6][n7][n8][1] = num2;
16         n = n + 1;
17     fim se
18     senão
19         Gera-G-NTI(G[8],p-num1 + 1, p-qt-num);
20         i = p-num1 + 1;
21         Enquanto (i ≤ p-qt-num) faça
22             aux = v[k];
23             v[k] = v[i];
24             v[i] = aux;
25             Gera-G-NTI(vG[8],p-num1 + 1, p-qt-num);
26             aux = G[p-num1];
27             G[p-num1] = G[i];
28             G[i] = aux;
29             i = i + 1;
30         fim-enquanto
31     fim se
32 fim
```

Algoritmo 21: (GITN) ← Gera-G-ITN(G[8],p-num1, p-qt-num)

Entrada : G[8] : permutação de 8 números , primeiro número: p-num1, quantidade de números: p-qt-num
Saída : GITN: Matriz de Conversão de Índice para Número

```
1  início
2      se (p-num1 = p-qt-num - 1);
3       então
4          GITN[n][0] = G[0]-1;
5          GITN[n][1] = G[1]-1;
6          GITN[n][2] = G[2]-1;
7          GITN[n][3] = G[3]-1;
8          GITN[n][4] = G[4]-1;
9          GITN[n][5] = G[5]-1;
10         GITN[n][6] = G[6]-1;
11         GITN[n][7] = G[7]-1;
12         n = n + 1;
13     fim se
14     senão
15         Gera-G-ITN(G[8],p-num1 + 1, p-qt-num);
16         i = p-num1 + 1;
17         Enquanto (i ≤ p-qt-num) faça
18             aux = v[k];
19             v[k] = v[i];
20             v[i] = aux;
21             Gera-G-ITN(G[8],p-num1 + 1, p-qt-num);
22             aux = G[p-num1];
23             G[p-num1] = G[i];
24             G[i] = aux;
25             i = i + 1;
26         fim-enquanto
27     fim se
28 fim
```

APÊNDICE E – TABELA ASCII

Tabela 20 – Tabela ASCII

Dec	Hexa	char		Dec	Hexa	char		Dec	Hexa	char
000*d*	00*h*	(nul)		023*d*	17*h*	(etb)		046*d*	2E*h*	.
001*d*	01*h*	(soh)		024*d*	18*h*	(can)		047*d*	2F*h*	/
002*d*	02*h*	(stx)		025*d*	19*h*	(em)		048*d*	30*h*	0
003*d*	03*h*	(etx)		026*d*	1A*h*	(eof)		049*d*	31*h*	1
004*d*	04*h*	(eot)		027*d*	1B*h*	(esc)		050*d*	32*h*	2
005*d*	05*h*	(enq)		028*d*	1C*h*	(fs)		051*d*	33*h*	3
006*d*	06*h*	(ack)		029*d*	1D*h*	(gs)		052*d*	34*h*	4
007*d*	07*h*	(bel)		030*d*	1E*h*	(rs)		053*d*	35*h*	5
008*d*	08*h*	(bs)		031*d*	1F*h*	(us)		054*d*	36*h*	6
009*d*	09*h*	(tab)		032*d*	20*h*	(espaço)		055*d*	37*h*	7
010*d*	0A*h*	(lf)		033*d*	21*h*	!		056*d*	38*h*	8
011*d*	0B*h*	(vt)		034*d*	22*h*	"		057*d*	39*h*	9
012*d*	0C*h*	(np)		035*d*	23*h*	#		058*d*	3A*h*	:
013*d*	0D*h*	(cr)		036*d*	24*h*	$		059*d*	3B*h*	;
014*d*	0E*h*	(so)		037*d*	25*h*	%		060*d*	3C*h*	<
015*d*	0F*h*	(si)		038*d*	26*h*	&		061*d*	3D*h*	=
016*d*	10*h*	(dle)		039*d*	27*h*	’		062*d*	3E*h*	>
017*d*	11*h*	(dc1)		040*d*	28*h*	(		063*d*	3F*h*	?
018*d*	12*h*	(dc2)		041*d*	29*h*	)		064*d*	40*h*	@
019*d*	13*h*	(dc3)		042*d*	2A*h*	*		065*d*	41*h*	A
020*d*	14*h*	(dc4)		043*d*	2B*h*	+		066*d*	42*h*	B
021*d*	15*h*	(nak)		044*d*	2C*h*	’		067*d*	43*h*	C
022*d*	16*h*	(syn)		045*d*	2D*h*	-		068*d*	44*h*	D
069*d*	45*h*	E		092*d*	5C*h*	\		115*d*	73*h*	s
070*d*	46*h*	F		093*d*	5D*h*	]		116*d*	74*h*	t
071*d*	47*h*	G		094*d*	5E*h*	^		117*d*	75*h*	u
072*d*	48*h*	H		095*d*	5F*h*	_		118*d*	76*h*	v
073*d*	49*h*	I		096*d*	60*h*	‘		119*d*	77*h*	w
074*d*	4A*h*	J		097*d*	61*h*	a		120*d*	78*h*	x
075*d*	4B*h*	K		098*d*	62*h*	b		121*d*	79*h*	y
076*d*	4C*h*	L		099*d*	63*h*	c		122*d*	7A*h*	z

Dec	Hexa	char		Dec	Hexa	char		Dec	Hexa	char
077*d*	4D*h*	M		100*d*	64*h*	d		123*d*	7B*h*	{
078*d*	4E*h*	N		101*d*	65*h*	e		124*d*	7C*h*	|
079*d*	4F*h*	O		102*d*	66*h*	f		125*d*	7D*h*	}
080*d*	50*h*	P		103*d*	67*h*	g		126*d*	7E*h*	~
081*d*	51*h*	Q		104*d*	68*h*	h		127*d*	7F*h*	DEL
082*d*	52*h*	R		105*d*	69*h*	i				
083*d*	53*h*	S		106*d*	6A*h*	j				
084*d*	54*h*	T		107*d*	6B*h*	k				
085*d*	55*h*	U		108*d*	6C*h*	l				
086*d*	56*h*	V		109*d*	6D*h*	m				
087*d*	57*h*	W		110*d*	6E*h*	n				
088*d*	58*h*	X		111*d*	6F*h*	o				
089*d*	59*h*	Y		112*d*	70*h*	p				
090*d*	5A*h*	Z		113*d*	71*h*	q				
091*d*	5B*h*	[		114*d*	72*h*	r				