Aplicação de Criptografia Homomórfica na Mineração de Dados em Fluxos de Roteadores de Borda na Internet

Felipe Martins Fernandes de Assis

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Luís Felipe M. de Moraes, Evandro Luiz Cardoso Macedo

Rio de Janeiro

Dezembro de 2023

APLICAÇÃO DE CRIPTOGRAFIA HOMOMÓRFICA NA MINERAÇÃO DE DADOS EM FLUXOS DE ROTEADORES DE BORDA NA INTERNET

Felipe Martins Fernandes de Assis

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO

Autor:

gov.br Documento assinado digitalmente
FELIPE MARTINS FERNANDES DE ASSIS
Data: 09/01/2024 22:20:35-0300
Verifique em https://validar.iti.gov.br

Felipe Martins Fernandes de Assis

Orientador:

gov.br Documento assinado digitalmente
LUIS FELIPE MAGALHAES DE MORAES
Data: 10/01/2024 15:14:17-0300
Verifique em https://validar.iti.gov.br

Prof. Luís Felipe Magalhães de Moraes, Ph. D.

Orientador:

gov.br Documento assinado digitalmente
EVANDRO LUIZ CARDOSO MACEDO
Data: 10/01/2024 07:59:24-0300
Verifique em https://validar.iti.gov.br

Prof. Evandro Luiz Cardoso Macedo, D.Sc.

Examinador:

Prof. Daniel Ratton Figueiredo, Ph. D.

Examinador:

Prof. Flávio Luis de Mello, D.Sc.

Rio de Janeiro

Dezembro de 2023

Declaração de Autoria e de Direitos

Eu, Felipe Martins Fernandes de Assis CPF 02573851265, autor da monografia *Aplicação de Criptografia Homomórfica na Mineração de Dados em Fluxos de Roteadores de Borda na Internet*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.

5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

7. Por ser verdade, firmo a presente declaração.

gov.br Documento assinado digitalmente
FELIPE MARTINS FERNANDES DE ASSIS
Data: 09/01/2024 22:21:56-0300
Verifique em https://validar.iti.gov.br

Felipe Martins Fernandes de Assis

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

DEDICATÓRIA

Dedico este trabalho aos meus amigos e familiares, em especial à minha mãe que sempre prezou pelos meus estudos.

AGRADECIMENTO

Primeiramente agradeço a Deus, por me dar a oportunidade de viver e me acompanhar desde o meu nascimento. Também agradeço a meus pais por me criarem e me darem muito amor e carinho.

Agradeço a meus avós por cuidarem de mim por toda a minha vida, sempre me dando apoio e suporte. Agradeço a todos os meus amigos e familiares, tanto os que podem estar presentes quanto os que já se foram, por serem uma fonte inesgotável de amor e companhia.

Agradeço ao Colégio Pedro II e à Universidade Federal do Rio de Janeiro que foram a raiz da minha formação acadêmica. Agradeço aos meus orientadores, Evandro e Professor Luís Felipe, que me deram inúmeras oportunidades.

E mais uma vez agradeço à minha mãe, de quem eu sinto saudades todos os dias.

RESUMO

Por conta da difusão das tecnologias da informação, a quantidade de dados gerada, transmitida e capturada aumenta expressivamente, o que permite que informações sejam obtidas através dos dados originais, assim como da combinação destes. O desenvolvimento das operações feitas com os dados para gerar mais informações e, posteriormente, conhecimento, é realizado na área de *Mineração de Dados*. Por outro lado, também cresce a preocupação e a conscientização sobre a privacidade de dados, indicando um caminho contrário à Mineração de Dados de forma indiscriminada. Neste sentido, a *Criptografia Homomórfica* se mostra uma ferramenta útil, pois permite a manipulação de dados encriptados, de forma a gerar informação sem que dados privados sejam expostos. Este trabalho propõe três diferentes métodos para a geração de Regras de Associação em um conjunto de dados distribuído entre n participantes. Tais métodos visam criar regras para associar tipos de dados que estão distribuídos em n partes, sem revelar o conteúdo de cada parte para as demais. Dados reais de fluxos de roteadores de borda da rede acadêmica Rede-Rio/FAPERJ foram utilizados para validar os métodos e compará-los entre si, tanto de maneira teórica quanto prática. Os resultados mostram a eficiência dos métodos propostos, assim como sua segurança em diferentes cenários.

Palavras-Chave: Mineração de Dados, Regras de Associação, Criptografia Homomórfica, Criptografia de Limiar, fluxos de rede, privacidade.

ABSTRACT

Due to the diffusion of information technologies, the amount of generated, transmitted and captured data increases expressively, which leverages information to be obtained by the original data, as well as its combination. The development of these operations on the data to generate more information and, then, knowledge, is accomplished by the area of *Data Mining*. On the other hand, there is also an increase in concern and conscientization about data privacy, indicating the opposite direction from indiscriminate Data Mining. In this sense, *Homomorphic Cryptography* presents itself as a useful tool, since it allows the manipulation of encrypted data in a way to generate information without exposing private data. This work proposes three different methods for Association Rules generation in a dataset shared by n participants. These methods for creating rules to associate types of data that are distributed in n parts without revealing the content of each share to the others. Real flow data from the Rede-Rio/FAPERJ academic network edge routers were used to validate the methods and compare them, in theoretical and pratical senses. The results show the efficiency of each proposed method, as well as security in different scenarios.

Key-words: Data Mining, Association Rules, Homomorphic Cryptography, Threshold Cryptography, network flow, privacy.

SIGLAS

BFV - Brakerski-Fan-Vercauteren

BGV - Brakerski-Gentry-Vaikuntanathan

CGGI - Chillotti-Gama-Georgieva-Izabachene

CKKS - Cheon-Kim-Kim-Song

DM - Ducas-Micciancio

FHE - Fully Homomorphic Encryption

SHE - Somewhat Homomorphic Encryption

RSA - Rivest-Shamir-Adleman

Sumário

1 Introdução **1**

2 Fundamentação Teórica **4**

2.1 Definições Matemáticas 4

2.2 Criptografia 6

2.2.1 Fundamentos da Criptografia 6

2.2.2 Criptografia Homomórfica 9

2.2.3 Brakerski-Fan-Vercauteren 11

2.2.4 Criptografia de Limiar 12

2.3 Mineração de Dados 13

3 Aplicações de Criptografia Homomórfica **17**

3.1 Trabalhos Relacionados 17

3.2 Mineração por Regra de Associação de Dois Participantes 22

3.3 Mineração por Regra de Associação de Múltiplos Participantes 24

4 Métodos de Geração Distribuída de Regras de Associação **27**

4.1 Método Padrão 29

4.2 Método Proposto 1: Sistema Homomórfico com Soma e Produto . . . 30

4.3 Método Proposto 2: Sistema Homomórfico de Limiar com Soma . . . 33

4.4 Método Proposto 3: Sistema Homomórfico de Limiar com Soma e Produto 37

5 Dados Utilizados **40**

6 Comparação Entre os Métodos **50**

6.1 Comparação Teórica 50

6.2 Comparação Prática 54

7 Conclusão **59**

7.1 Trabalhos Futuros 60

Bibliografia **62**

Appendices **68**

Apêndice A Complementos Matemáticos **69**

A.1 Operadores Booleanos 69

A.2 Conjuntos 70

A.3 Números Primos 70

A.4 Álgebra 71

Apêndice B Exemplos Numéricos **73**

Apêndice C Resumo dos Métodos **74**

C.1 Método Padrão 74

C.2 Método 1 74

C.3 Método 2 75

C.4 Método 3 76

Lista de Figuras

2.1 Esquema de encriptação e decriptação. 7
2.2 Esquema de encriptação e decriptação com chaves 8

4.1 Comunicações no Método Padrão. 30
4.2 Comunicações na primeira metade do Método 1. 32
4.3 Comunicações para a geração de chave pública no Método 2. 35
4.4 Comunicações para a decriptação no Método 2. 36

5.1 Topologia da Rede-Rio/FAPERJ 41
5.2 Histograma do número de pacotes com intervalos iguais. 44
5.3 Frequência do número de pacotes com intervalos variados. 45
5.4 Histograma do total de bytes com intervalos iguais. 46
5.5 Frequência do total de bytes com intervalos variados. 46

6.1 Dados após pré-processamento 58

Lista de Tabelas

2.1 Exemplo de compras em um mercado. 14
2.2 Exemplo de conjuntos de itens em um mercado. 16

5.1 Exemplo dos primeiros fluxos de um arquivo. 42
5.2 Dados após a primeira etapa de pré-processamento 48
5.3 Dados após a última etapa de pré-processamento 49

6.1 Comparação Teórica dos Métodos 53
6.2 Comparação Prática dos Métodos 56

Capítulo 1

Introdução

A evolução e a difusão dos dispositivos digitais e tecnologias de rede vem aumentando rapidamente o volume de dados gerados, ultrapassando o patamar de 95 zettabytes, com tendências de crescimento [1]. Com isso, há um miríade de dados que tem o potencial de lançar luz sobre informações até então não reveladas, o que permite extrair conhecimento de maneira sem precedentes. A área responsável por essa extração de conhecimento é chamada de Mineração de Dados e dispõe de diversas técnicas para alcançar este propósito.

Por outro lado, também cresce a preocupação com a privacidade de dados, como evidenciado até mesmo por artefatos legais como a Lei Geral de Proteção de Dados (LGPD)[2]. Tanto as pessoas, como as instituições, entendem que seus dados são valiosos e desejam ter controle sobre eles, seja por preocupação do uso malicioso de seus dados ou simplesmente para evitar que sejam utilizados indevidamente [3].

A Criptografia Homomórfica se apresenta como uma solução para a obtenção de informações de forma a evitar vazamento de dados, permitindo a manipulação de dados encriptados por terceiros por meio de um conjunto de operações. Ou seja, a Criptografia Homomórfica permite a utilização de dados que estão codificados de uma maneira que só agentes autorizados conseguem lê-los e aplicar operações básicas, como por exemplo soma e produto, para gerar informações que somente o agente autorizado do dado é capaz de descriptografar.

Uma das áreas nas quais a Criptografia Homomórfica pode atuar diretamente é no Aprendizado Federado [4], que se trata aprendizado distribuído, no qual participantes cooperam para construir um modelo sem a divulgação de seus dados, caracterizando um tipo de Computação de Múltiplas Partes [5]. Desta maneira, a Criptografia Homomórfica pode alavancar técnicas de Mineração de Dados em um ambiente distribuído, no qual participantes desejam cooperar sem revelar seus dados.

Este trabalho traz uma contribuição para solução do problema de computação de valores sigilosos entre partes que não desejam revelar suas informações individuais. Tal solução se baseia em uma técnica de Mineração de Dados que pode tomar proveito da estrutura da Criptografia Homomórfica. A técnica em questão é a geração de Regras de Associação, que se encaixa por depender apenas de operações fundamentais, como soma e produto, e apresentar uma base no contexto dado pelo trabalho de [6]. O problema base consiste então em um cenário no qual n participantes detêm diferentes partes de um mesmo conjunto de dados distribuído e desejam cooperar para criar Regras de Associação sobre este conjunto sem revelar suas partes individuais. Para isso, foram desenvolvidos três métodos diferentes.

Os métodos foram construídos para funcionar a partir do problema genérico, isto é, se encaixam em qualquer cenário que parte da necessidade de criação de Regras de Associação de um conjunto distribuído de forma a preservar a privacidade das partes. Apesar disso, foi escolhido o tema de fluxos de rede para a validação dos métodos. Fluxos de rede podem se beneficiar diretamente da aplicação, pois escondem diversos padrões que podem ser utilizados para previsão de tráfego [7], detecção de anomalias [8], entre outras coisas. Além disso, muitas vezes há a necessidade de cuidado na manipulação de dados de rede para manter a privacidade dos usuários que geraram tais fluxos. Desta forma, a aplicação direta deste trabalho ocorre quando detentores de diferentes roteadores de borda entendem que juntos têm características em comum, logo decidem cooperar sem revelar os dados referentes a cada fluxo a fim de descobrir padrões nesta base de dados conjuntos, para assim obter padrões frequentes que podem ser utilizados em detecção de anomalias, engenharia de tráfego e utras necessidades.

Os fluxos utilizados foram retirados de roteadores de borda da Rede-Rio/FAPERJ [9] e distribuídos em quatro conjuntos para atuar como diferentes participantes que desejam contribuir para criar Regras de Associação conjuntas, de forma a preservar a privacidade individual. Os conjuntos foram então distribuídos em quatro instâncias de containeres Docker e aplicados os três métodos propostos, assim como um método sem o uso de criptografia. Os resultados foram comparados, tanto no quesito prático, quanto teórico.

O restante do trabalho se organiza conforme o descrito a seguir. No Capítulo 2 são discutidas as fundamentações teóricas para se entender os aspectos matemáticos, criptográficos e de Mineração de Dados. No Capítulo 3 são apresentadas utilizações de Criptografia Homomórfica na Mineração de Dados, apresentando também a proposta principal deste trabalho. Os dados utilizados para experimentação são discutidos no Capítulo 5. O Capítulo 4 introduz os métodos propostos enquanto o Capítulo 6 realiza uma comparação entre tais métodos. Por fim, o Capítulo 7 conclui o trabalho com uma revisão do que foi feito e apresenta ideias para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Para entender o trabalho em sua totalidade, primeiramente são necessários alguns conceitos teóricos, que serão aplicados ao longo do trabalho. Neste capítulo, será apresentada a base de cada conceito necessário, desde a parte puramente matemática até a Mineração de Dados.

2.1 Definições Matemáticas

Começamos com uma revisão sobre conceitos básicos de álgebra que mais tarde serão aplicados em um contexto criptográfico. As definições apresentadas nesta seção são amplamente conhecidas e podem ser encontradas em livros de álgebra como [10]. Começamos por definir uma operação de forma genérica, de forma a se chegar em algo parecido com o que já conhecemos na informalidade.

Definição 2.1 (Operação Binária)**.** *Seja S um conjunto. Uma* ***operação binária*** *$*$ em S é uma função $* : S \times S \to S$.*

Definição 2.2 (Operação Associativa)**.** *Uma operação binária é dita* ***associativa*** *se $a * (b * c) = (a * b) * c$ para todos $a, b, c \in S$.*

Essas definições da operação binária estendem o que conhecemos de operações como a soma e o produto usual para um contexto mais abstrato, para resultados mais gerais serem obtidos em todas as operações que seguem um conjunto de propriedades. Nesse sentido, as duas próximas definições explicitam mais duas destas propriedades:

Definição 2.3 (Elemento Neutro)**.** *Um elemento $E \in S$ é chamado de* ***Identidade*** *(ou* ***Elemento Neutro****) de uma operação binária $*$ se $a * E = E * a = a$ para todo $a \in S$.*

Definição 2.4 (Inverso)**.** *Se uma operação binária $*$ tem um elemento neutro E e $a \in S$, então b é dito ser o* ***inverso*** *de a se $a * b = b * a = E$.*

Com estas quatro definições, é possível agora definir a estrutura de um grupo:

Definição 2.5 (Grupo)**.** *Um conjunto S juntamente a uma operação $*$ forma um* ***grupo*** *$(S, *)$ se satisfaz as seguintes propriedades:*

(i) ***Fechamento*** *Dados $a, b \in S$ temos que $a * b \in S$.*

(ii) ***Associatividade****: A operação $*$ é associativa em S como na Definição 2.2.*

(iii) ***Identidade:*** *S tem um Elemento Neutro para $*$ como na Definição 2.3.*

(iv) ***Inversos:*** *Todo elemento de S possui um inverso como na Definição 2.4.*

A definição de grupo é uma das mais importantes na álgebra, pois nos permite usar uma estrutura abstrata, porém familiar, para extrair propriedades comuns de diversos contextos que em primeiro momento se apresentam distintos. Como exemplos de grupos, podemos citar como exemplo:

- O conjunto dos Reais não nulos juntamente com a operação de multiplicação.
- O conjunto dos polinômios reais com a operação de soma.
- Um conjunto de funções de mesmo domínio e contradomínio com a operação de composição.
- O conjunto dos inteiros com a operação de adição.

Este último mostra o potencial dessa abstração. Ela nos permite, para certos casos, tratar exemplos mais robustos com a simplicidade dos números inteiros. Essa ideia de tratar as similaridades estruturais de conjuntos distintos leva à próxima definição.

Definição 2.6 (Homomorfismo)**.** *Sejam A e B dois grupos com operações $*$ e $+$ respectivamente. Uma função $f : A \to B$ é dita um* ***homomorfismo*** *se $f(x * y) = f(x) + f(y)$ para todo $x, y \in A$.*

Como o nome já sugere, o homomorfismo surge como uma maneira de ir de um grupo a outro mantendo a mesma forma, preservando as operações. Um exemplo trivial é $f(x) = e^x$, que transforma os Reais com a adição para os Reais positivos com produto. Isso é simples de ver porque $f(x + y) = e^{x+y} = e^x e^y = f(x)f(y)$

2.2 Criptografia

Segundo Fergunson e Schneier [11], "Criptografia é a arte e a ciência da encriptação. Ao menos, foi assim que começou. Atualmente, é bem mais amplo, cobrindo autenticação, assinaturas digitais, e muitas outras funções elementares de segurança.". Em termos simples, o núcleo da criptografia se encontra no uso de funções de encriptação e decriptação. A primeira busca pegar uma entrada e transformá-la em algo ininteligível, ou seja, que não tenha valor a não ser que seja traduzida de volta ao seu estado inicial, o que é feito pela decriptação. Em termos práticos, a situação usual de uso da criptografia dá-se quando deseja-se transmitir ou guardar uma informação sigilosa de maneira que só determinados agentes tenham acesso ao seu conteúdo.

2.2.1 Fundamentos da Criptografia

A Criptografia, além de suas aplicações práticas, tem sua fundamentação teórica matemática. Por isso, é importante a utilização da sutileza e da formalidade no momento de estudo deste tópico. Por este motivo, inicia-se esta seção com algumas definições básicas porém importantes, que podem ser encontradas na literatura, como em [12].

Definição 2.7 (Texto em claro)**.** *Um* ***texto em claro*** *(também chamado de* ***texto puro*** *se trata de um texto que pode ser compreendido (seja por homem ou máquina).*

Definição 2.8 (Encriptação e texto cifrado)**.** *O processo de disfarçar um texto em uma forma de esconder sua essência é chamado de* ***encriptação****. Um texto em*

Figura 2.1: Esquema de encriptação e decriptação. Imagem inspirada em [11].

claro que passou por encriptação é chamado de ***texto cifrado****.*

Definição 2.9 (Decriptação)**.** *O processo de transformar o texto cifrado em seu texto em claro tradicional é chamado de* ***decriptação****. De maneira mais formal, dado um texto em claro m e uma função de encriptação E, temos que a função de decriptação D é aquela que satisfaz a seguinte fórmula:*

$$D(E(m)) = m$$

Um esquema que ilustra o processo de decriptação pode ser visto na Figura 2.1.

É importante perceber que a decriptação se comporta como o processo inverso da encriptação e por isso é importante escolher-se um método que o processo de encriptação não ajude algum adversário a descobrir o processo de decriptação, pois, como já foi dito, o tema central da criptografia é a privacidade da informação. Veja como exemplo o seguinte sistema: seja o conjunto de todos os textos em claro igual aos números reais e seja $E(m) = a \cdot m$ para algum $a \in \mathbb{R}$, com $a \neq 0$. Se um adversário descobrir que essa é a função de encriptação, ele pode deduzir facilmente que $D(c) = \frac{c}{a}$, pois $D(E(m)) = D(a \cdot m) = \frac{a \cdot m}{a} = m$. Além disso, é interessante que se possa alterar levemente o processo de encriptação e decriptação em cada caso para garantir flexibilidade e mais segurança, no sentido de poder usar a ideia de uma encriptação para gerar encriptações similares. Esses motivos nos levam a definição de um sistema de criptografia por chaves.

Definição 2.10 (Criptossistema com chaves)**.** *Um* ***Criptossistema com chaves*** *é um sistema no qual existem duas chaves K_1 e K_2 de maneira que as funções de encriptação são E_{K_1} e D_{K_2}, ou seja, dependem das chaves. Da mesma forma, temos que para um texto em claro m:*

$$D_{K_2}(E_{K_1}(m)) = m$$

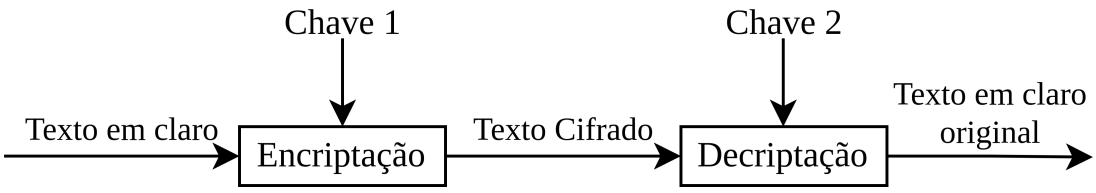

Figura 2.2: Esquema de encriptação e decriptação com chaves. Imagem inspirada em [11].

*Se $K_1 = K_2$, dizemos que o algoritmo é **simétrico**. Se $K_1 \neq K_2$, dizemos que o algoritmo é **assimétrico**, ou **de chave pública**. Neste último caso, é normal chamar K_1 de chave pública, usualmente denotada* $\mathbf{P}$ *(**P**ublic Key), e chamar K_2 de chave privada, usualmente denotada* $\mathbf{S}$ *(**S**ecret Key).*

As chaves são chamadas de pública e privada pois na maioria dos casos a chave pública não é um segredo, todos podem usar aquela chave para encriptar dados, mas a privada é, ou seja somente o dono da chave privada pode decriptar os dados. Um esquema que ilustra esse processo pode ser visto na Figura 2.2.

Esses conceitos podem ser muito abstratos para quem os não conhece. Por este motivo, são apresentados os seguintes exemplos:

Exemplo 2.1 (Exemplo de chave simétrica)**.** *Seja $\oplus$ o operador XOR como definido no Apêndice A. Seja m uma mensagem em texto puro representada como uma sequência binária, K uma chave também representada como uma sequência binária de mesmo tamanho que a de m. Seja $E_K(x) = x \oplus K$ e seja $D_K(x) = x \oplus K$, onde o a operação é feita bit a bit. Como $a \oplus b \oplus b = a$, temos que*

$$D_K(E_K(m) = D_K(m \oplus K) = m \oplus K \oplus K = m$$

Logo D_K transforma o texto cifrado $E_K(m)$ de volta ao texto puro original m.

Exemplo 2.2 (Exemplo de chave pública, RSA [13])**.** *O sistema RSA (**R**ivest–**S**hamir–**A**dleman) se constitui da seguinte forma: seja o número $n = pq$ onde p e q são primos e sejam d e e dois números tal que $de = 1 \mod \varphi(n)$, onde φ é a função*

totiente de Euler, como definido no Apêndice A. Temos e como a chave pública e d como a chave privada. Assim, $E(x) = x^e \mod n$ *e* $D(x) = x^d \mod n$*. Para um valor m, temos:*

$$\begin{aligned} D(E(m)) &= (m^e)^d \mod p \\ &= m^{ed} \mod p \\ &= m^{b\varphi(n)+1} \mod p \\ &= m^{b(p-1)(g-1)+1} \mod p \\ &= (m^{g-1})^{b(p-1)} m \mod p \\ &= 1^{b(p-1)} m \mod p \\ &= m \mod p \end{aligned}$$

Onde o número b aparece pois $ed-1$ *é múltiplo de* $\varphi(n)$*. A transformação de* m^{p-1} *para* 1 *ocorre pelo pequeno teorema de Fermat (Teorema A.5) e só funciona caso m não seja múltiplo de p. Se m for múltiplo de p, então* $m = 0 \mod p$ *o que torna o resultado acima trivial. O mesmo resultado vale para q. Se o resultado vale para p e q, o Teorema A.4 garante que vale para* $pq = n$*. Assim D recupera o valor original de m.*

É importante deixar claro que, apesar de serem exemplos de sistemas criptográficos, não apresentam segurança real. Segundo Ferguson e Schneier [11], o sistema que usa o XOR é facilmente decifrável. O segundo caso, o RSA, é um sistema amplamente utilizado, mas só com preenchimento aleatório para poder prover a segurança necessária [14]. Para o leitor que se interessar em algo mais prático, exemplos numéricos podem ser encontrados no Apêndice B.

2.2.2 Criptografia Homomórfica

Com as definições dos fundamentos de criptografia e de conceitos algébricos, pode-se agora definir o conceito principal utilizado neste trabalho:

Definição 2.11 (Criptossistema Homomórfico [14])**.** *Um criptossistema homomórfico é um criptossistema no qual o conjunto de possíveis textos em claro e o conjunto de*

possíveis textos cifrados são ambos grupos, dado que para toda chave e para quaisquer duas mensagens m_1 e m_2, temos:

$$D(E(m_1) \cdot E(m_2)) = m_1 \cdot m_2$$

Onde $\cdot$ representa a operação de cada grupo.

A definição acima pode ser estendida para um anel (Definição A.13) para suportar duas operações ao mesmo tempo. Para Henry [14], isso pode ser chamado de *Criptossistema algebricamente homomórfico*. Para ilustrar a ideia de criptossistema homomórfico, temos o seguinte exemplo [14]:

Exemplo 2.3 (Homomorfismo do RSA)**.** *Dado um criptossistema com e,d,n definidos de acordo com o Exemplo 2.2. Dados dois textos cifrados $c_1 = m_1^e$ e $c_2 = m_2^e$, temos:*

$$\begin{aligned} c_1 c_2 \mod n &= m_1^e m 2^e \mod n \\ &= (m_1 m_2)^e \mod n \end{aligned}$$

Então:

$$D(c_1 c_2) = D((m_1 m_2)^e) = m_1 m_2$$

Por último, se E é um homomorfismo, consequentemente D também será, pois

$$\begin{aligned} E(m_1 \cdot m_2) &= E(m_1) \cdot E(m_2) \\ D(E(m_1 \cdot m_2)) &= D(E(m_1) \cdot E(m_2)) \\ m_1 \cdot m_2 &= D(E(m_1) \cdot E(m_2)) \\ D(E(m_1) \cdot D(E(m_2) &= D(E(m_1) \cdot E(m_2)) \end{aligned}$$

É importante ressaltar que a Criptografia Homomórfica atualmente também é chamada de Encriptação Homomórfica, na qual a atenção é voltada a parte de encriptação. Também é interessante fazer a distinção do que é conhecido como Encriptação Completamente Homomórfica (*Fully Homomorphic Encryption* - FHE) e Encriptação Parcialmente Homomórfica (*Somewhat Homomorphic Encryption* -

SHE). A primeira se caracteriza por permitir operações arbitrárias de adição e subtração em textos cifrados, enquanto a segunda permite apenas algumas operações [15]. O esquema utilizado neste estudo é um exemplo de FHE.

2.2.3 Brakerski-Fan-Vercauteren

O esquema de criptografia utilizado neste trabalho é uma variante Fan-Vercauteren [16] do esquema Brakerski [17], cuja implementação provém de [18, 19] e se trata de uma Criptografia Completamente Homomórfica. A notação usada no texto a seguir é a mesma em toda a seção, que será dividida segundo as partes do Brakerski-Fan-Vercauteren (BFV).

Espaços de Texto Puro e Texto Cifrado: O espaço de texto puro é determinado por um inteiro $t \geq 2$, com uma mensagem neste espaço sendo um elemento de R_t, ou seja, um polinômio com grau menor ou igual a $n-1$ (com n potência de 2) e coeficientes em $\mathbb{Z}_t$. O espaço de texto cifrado se trata de R_q, isto é, dos polinômios de grau menor ou igual a $n-1$ e coeficientes em $\mathbb{Z}_q$, para um parâmetro q, produto de números q_i primos entre si (Definição A.8), com $i \in \{1, ..., k\}$.

Geração de Chaves Sejam χ_K uma distribuição uniforme sobre $\{-1, 0, 1\}^n$ e χ_e uma distribuição discreta gaussiana. Seja $s \leftarrow \chi_K$. A chave privada é $\mathbf{sk} = (1, s)$. Agora escolha um $a \in R_q$ e $e \leftarrow \chi_e$. Seja $b = [-(as + e)]_q$ e seja $\mathbf{pk = (b,a)}$. Além disso, denotamos $q_i^* = \frac{q}{q_i}$ e $\tilde{q}_i = [q_i^{*-1}]_{q_i}$. Também escolha um $\alpha_i \in R_q$ e $e_i \leftarrow \chi_e$ e tenha $\beta_i = [\tilde{q}_i q_i^* s^2 - \alpha_i s + e_i]_q$. A chave pública consiste de $\mathbf{pk}$ e dos vetores $W_i = (\beta_i, \alpha_i)$, para $i = 1, \ldots, k$.

Encriptação Para encriptar $m \in R_t$, escolhe-se $u \leftarrow \chi_K$ e $e_0', e_1' \leftarrow \chi_e$. O texto cifrado é $\mathbf{ct} = [u \cdot \mathbf{pk} + (e_0', e_1') + ((t/q)m, 0)]_q$.

Decriptação Dado um texto cifrado $\mathbf{ct} = (c_0, c_1)$, primeiro calcule $x = [\langle \mathbf{sk}, \mathbf{ct} \rangle]_q = [c_0 + c_1 s]_q$. A mensagem decriptada se trata de $m = [x \cdot t/q]_t$.

Adição homomórfica Dados dois textos cifrados $\mathbf{ct}^1, \mathbf{ct}^2$, calculamos $[\mathbf{ct}^1 + \mathbf{ct}^2]_q$.

Multiplicação homomórfica A multiplicação homomórfica ocorre em duas etapas, dados $\mathbf{ct}^i = (c_0^i, c_1^i)_{i=1,2}$:

1. **Tensorização** Calcular $c_0' = c_0^1 c_0^2$, $c_1' = c_0^1 c_1^2 + c_1^1 c_0^2$, $c_2' = c_1^1 c_1^2$ sem redução modular. Calcular então $c_i^* = [t/q \cdot c_i']_q$,para $i = 0, 1, 2$.

2. **Relinearização** Decompor c_2^* em componentes $c_{2,i} = [c_2^*]_{q_i}$. Tenha $\tilde{c}_0 = [\sum_{i=1}^k \beta_i c_{2,i}^*]_q, \tilde{c}_1 = [\sum_{i=1}^k \alpha_i c_{2,i}^*]_q$. Assim o produto é $\mathbf{ct}^3 = [(c_0^* + \tilde{c}_0, c_1^* + \tilde{c}_1)]_q$.

Primeiramente, nota-se que para funcionar q/t deve ser inteiro, ou seja, t deve ser múltiplo de q. Porém, se isso não for o caso, o esquema funciona se substituirmos q/t por seu inteiro mais próximo. A encriptação e a adição são bem diretas. A decriptação e o produto são menos intuitivos, mas podem ser checados que resultam no resultado desejado.

2.2.4 Criptografia de Limiar

Em algumas situações, é interessante distribuir um segredo que só pode ser acessado se houver a colaboração de distintos participantes. Para esse intuito, existe a Criptografia de Limiar.

Definição 2.12 (Criptografia de Limiar [14])**.** *Um criptossistema de chave pública é dito ser um criptossistema de limiar (t,l) se alguém pode criar uma instância deste sistema e distribuir l distintas partes secretas entre l participantes de maneira que quaisquer t ou mais participantes podem cooperar para decriptar uma mensagem. Um subconjunto de menos de t participantes não pode aprender nada sobre a mensagem.*

Note que é necessário $t \leq l$. Em muitas situações é interessante ter $t = l$ para impedir que parte do grupo de participantes aja contra o restante. Note que esse tipo de sistema se mostra de imediato como interessante para um sistema homomórfico. Isso ocorre pois há várias situações nas quais um grupo de participantes deseja colaborar a fim de descobrir uma informação consequente de seus dados individuais, sem necessariamente compartilhá-los. Isto se torna mais difícil em um sistema normal, pois se usa somente um par de chaves, o que pode facilitar que o membro que possui a chave privada leia seu texto puro caso consiga em mãos o respectivo texto cifrado.

Com o criptossistema de limiar, todos detêm parte da chave privada, assim, mesmo se outro participante conseguir o texto cifrado, não saberá seu significado sem a participação de outros. Henry [14] dá o exemplo de uma votação para ilustrar criptografia homomórfica de limiar. Assim, cada participante tem seu voto individual, mas só o resultado final pode ser visto por todos.

A ideia central da criptografia de limiar se resume em dividir uma chave privada em diversas partes. Um exemplo simples de como fazer isso é usando interpolação [20]. Note que dados n pontos distintos, existe um polinômio único de grau $n-1$ que passa por esses pontos. Se não houver o conhecimento de todos os pontos, não há como descobrir tal polinômio. Assim, valores deste polinômio em determinados pontos podem ser usados como partes de uma chave privada para reconstruir o polinômio, que pode agir como chave privada. Outros exemplos de como dividir a chave privada podem ser vistos em [21, 22, 23].

2.3 Mineração de Dados

Mineração de Dados (*Data Mining*) é a ciência de extrair conhecimento útil de grandes repositórios de dados [24]. Com o aumento significativo da quantidade de dados produzida diariamente, a Mineração de dados se mostra uma boa opção para a transformação de dados inicialmente desconexos em uma fonte útil de informação.

Uma das técnicas importantes, porém simples, da Mineração de dados é chamada de Mineração por Regra de Associação (*Association Rule Mining* [25]). Essa técnica se baseia na repetição de padrões de itens em conjuntos aparecendo frequentemente em determinadas operações. A ideia geral pode ser entendida no seguinte exemplo:

Exemplo 2.4 (Regra de Associação (Informal))**.** *Imagine um mercado que guarda as informações das compras de seus clientes em uma tabela, sendo essa a Tabela 2.1. Note que a tabela é pequena pois trata-se apenas de um exemplo ilustrativo. Note também que, em quase toda compra na qual uma banana está envolvida, o cliente também leva uma maçã. Só há um caso onde uma banana é comprada sem o acompanhamento de uma maçã. Isto, para um conjunto de dados maior, é um indício que pessoas que compram bananas estão mais dispostas a comprar maçãs.*

Tabela 2.1: Exemplo de compras em um mercado.

Id da Compra	**Itens da Compra**
1	banana, maçã, abacaxi
2	banana, maçã, abacate
3	banana, uva
4	abacaxi
5	maçã
6	maçã
7	banana, maçã

Note que o exemplo acima, apesar de lúdico, apresenta características simples, porém relevantes, de um conjunto de dados. Note que isso poderia ser aproveitado comercialmente pelo mercado, como agrupar maçãs e bananas afim de aumentar a compra de maçãs. Para seguir em frente, é necessário definir alguns termos.

Definição 2.13 (Transação)**.** *Uma transação (neste trabalho) trata de uma linha em uma tabela de dados. No Exemplo 2.4, se trata de um id de transação juntamente a um conjunto de itens.*

Definição 2.14 (Regra)**.** *Uma regra de associação de um conjunto S_1 e outro S_2 é dada como $S_1 \Rightarrow S_2$ quando a aparição do conjunto S_1 na transação implica que provavelmente o conjunto S_2 aparecerá.*

Note que o termo "aparecer" é vago. Mais precisamente, dizemos que o conjunto aparece na transação se ele é um subconjunto do conjunto total da transação. Por exemplo, se usarmos a tabela do Exemplo 2.4, o conjunto {maçã, banana} aparece em 3 compras, sendo ele precisamente o conjunto da compra de id 6.

Definição 2.15 (Suporte [6])**.** *O suporte de um conjunto S é dado por*

$$Suporte_S = \frac{\textit{número de transações onde o conjunto } S \textit{ aparece}}{\textit{número total de transações}}$$

Note a importância do Suporte. Se exigirmos um valor mínimo para o suporte, podemos excluir regras irrelevantes do conjunto de possíveis regras a serem analisadas. Veja no Exemplo 2.4. Temos que sempre que um abacate é comprado,

uma banana também é, mas uma transação desse tipo ocorre apenas uma vez, tendo suporte de 1/7, o menor valor não nulo possível para esse conjunto. Por isso, as regras {abacate} ⇒ {maçã} e {maçã} ⇒ {abacate} podem ser descartadas.

Definição 2.16 (Confiança [6]). *A confiança de um conjunto S_1 em relação a um conjunto S_2 é dada por*

$$Confiança_{S_1 \Rightarrow S_2} = \frac{Suporte_{S_1 \cup S_2}}{Suporte_{S_1}} = \frac{n^o \text{ de transações onde } S_1 \cup S_2 \text{ aparece}}{n^o \text{ de transações onde } S_1 \text{ aparece}}$$

A confiança age de maneira semelhante ao suporte, no sentido de ser um valor que deve ser atingido para a regra não ser descartada. Em seguida, um exemplo seguindo os dados do Exemplo 2.4 para a ilustração formal de como funciona a criação de regras de associação.

Exemplo 2.5 (Formação de Regras de Associação). *Sejam os dados da Tabela 2.1. Com isso, podemos construir a Tabela 2.2 contando em quantas transações cada conjunto aparece. Note que a tabela está incompleta para ficar mais clara, nela não constam conjuntos que não aparecem em nenhuma transação, como $\{uva, abacaxi\}$, apesar de serem possíveis.*

Agora é necessário escolher-se valores de corte para o suporte e a confiança. Escolhemos 2/7 para suporte e 7/10 para confiança. Note que só 3 conjuntos que tem suporte maior que 2/6, sendo eles {banana}, {maçã} e {banana, maçã}. Além disso, temos dois conjuntos com apenas um elemento cada, logo não podem ser divididos em dois conjuntos não nulos, não podendo assim ser separados em regras de associação. Logo, temos apenas as regras {banana} ⇒ {maçã} e {maçã} ⇒ {banana} a serem testadas.

Temos $Confiança_{\{banana\} \Rightarrow \{maçã\}} = 3/4$ e $Confiança_{\{maçã\} \Rightarrow \{banana\}} = 3/5$. Logo, somente a regra {banana} ⇒ {maçã} tem confiança maior que 7/10 e somente ela é aceita.

Por último, é importante notar que, apesar de simples e eficiente, esse processo está sujeito à interferência humana, pois devem-se ser escolhidos os parâmetros de corte do Suporte e da Confiança. Como é possível notar no exemplo, o valor de

Tabela 2.2: Exemplo de conjuntos de itens em um mercado.

Conjunto	**Ocorrências**
banana	4
maçã	5
abacaxi	1
abacate	1
uva	1
banana, maçã	3
banana, uva	1
banana, abacate	1
banana, abacaxi	1
maçã, abacaxi	1
maçã, abacate	1

corte para o Suporte mínimo não pode ser muito alto. Imagine um exemplo mais real de um mercado, onde há muitas mais transações e itens diferentes vendidos. É irreal esperar que qualquer item seja vendido em 1/6 das compras, por isso, deve-se escolher um valor bem menor. Por isso, é necessário um certo cuidado ao escolher os dois valores para atingir os resultados desejados.

Capítulo 3

Aplicações de Criptografia Homomórfica

Apesar das origens de implementação de alguma criptografia homomórfica datar de ao menos 1978 com a origem do RSA [13], o uso efetivo de operações homomórficas em dados criptografados se populariza em 2009 com o esquema proposto por Gentry [26]. Com isso, é possível observar que a utilização do conceito se trata de um fenômeno relativamente recente. Apesar disso, diversos trabalhos já foram produzidos sobre o assunto. Neste capítulo, serão apresentados brevemente alguns destes estudos, incluindo um dos mais relevantes da área e que inspirou o presente trabalho.

3.1 Trabalhos Relacionados

O estudo apresentado por Frikken *et al* [27] trabalha na união de conjuntos. Nota-se, que o trabalho é de 2007, anterior ao de Gentry, evidenciando que mesmo antes da "popularização" já havia estudos nesta área. A ideia da manipulação de conjuntos encriptados se dá no recurso de representar conjuntos como polinômios. Note que um polinômio mônico (i.e. com seu coeficiente de maior grau sendo 1) é unicamente caracterizado por suas raízes. Logo, é possível construir uma relação de 1 para 1 de conjuntos numéricos e polinômios. Seja, por exemplo, o conjunto $\{a_1, \ldots, a_n\}$. Este conjunto pode ser representado pelo polinômio $p(x) = (x - a_1) \ldots (x - a_n)$.

Por outro lado, um polinômio pode ser representado unicamente por seus coeficientes, podendo assim ser representado por uma tupla. Desta forma, como um polinômio é definido apenas por seus coeficientes e tem sua forma sendo composta por produtos e somas, pode-se estender a soma e o produto homomórficos de números para a sua versão polinomial, assim como a possibilidade de avaliar o polinômio em um certo ponto de maneira criptografada.

Com esses conceitos, Frikken *et al* [27] propõem o seguinte algoritmo para a união de dois conjuntos de participantes distintos de maneira criptografada:

1. O Participante 1, portador da chave pública p e da chave privada s codifica seu conjunto como o polinômio f_1 e envia p e $E_p(f_1)$ para o Participante 2.

2. Para cada valor a em seu conjunto, o Participante 2 escolhe uniformemente um parâmetro r e calcula a tupla $E_p(f_1(a) \cdot a \cdot r), E_p(f_1(a) \cdot r)$. Logo depois, permuta suas respostas e reenvia ao Participante 1.

3. O Participante 1 coloca seu conjunto na saída. Para cada tupla $(E_p(x), E_p(y))$ recebida, o Participante 1 decripta e descobre x e y. Se $x = y = 0$, essa tupla é desconsiderada. Caso contrário, $x \cdot y^{-1}$ é adicionado a saída.

Note que o método ignora a interseção, de forma que cada participante só descobrirá os elementos que o outro participante tem de diferente. Isso funciona, pois caso um elemento a esteja na interseção, então a é raiz de ambos os polinômios, logo $f_1(a) = 0$. Isso mostra o espírito da criptografia homomórfica, que é revelar parte da informação, mantendo-se o desejado escondido. Também note que isso só é possível pois os dois participantes estão dispostos a cooperar, logo a informação compartilhada é precisamente a desejada, nada mais, nada menos. Frikken no mesmo trabalho expande o algoritmo para mais participantes.

Outra aplicação dada para a criptografia homomórfica é dada por Yi *et al* [28]. Nesse trabalho, o objetivo é a implementação de um algorítmo de Recuperação Privada de Dados (*Private Information Retrival*). A ideia é a seguinte: imagine que um usuário queira recuperar o i-ésimo bit de uma sequência de dados armazenada em um banco de dados remoto, mas deseja fazer de modo com que o banco não

saiba o valor de i. Para solucionar esse problema, Yi *et al* usam o potencial da criptografia homomórfica. Primeiro, analisemos o algoritmo proposto em sua forma pura:

1. O usuário escolhe seu índice $i \in [1, n]$ ($i \in \mathbb{N}$) para sua consulta no banco de dados que é uma sequência binária $b_1 b_2 \dots b_n$. O usuário escreve i em binário como $i = \alpha_1 \dots \alpha_l$.

2. O servidor de banco de dados escreve cada índice $j \in [1, n]$ ($j \in \mathbb{N}$) do banco de dados em sua forma binária $\beta_{j,1} \dots \beta_{j,n}$e calcula

$$\gamma_j = \prod_{t=1}^{l} (\alpha_t \oplus \beta_{j,t} \oplus 1)$$

3. Calcula $R = \gamma_{k_1} \oplus \dots \oplus \gamma_{k_p}$, onde cada k_q é um índice no qual $b_{k_q} = 1$, e envia R para o usuário.

Alguns apontamentos devem ser feitos. Em cada termo do produto do passo 2, o termo é igual a 1 se e somente se $\alpha_t = \beta_{j,t}$. Como γ_i é o produto dos termos, só será igual a 1 se todos os termos também forem. Logo, somente $\gamma_i = 1$ e os outros são nulos. Assim, se $b_i = 1$, temos que $R = \gamma_i = 1$, pois todo γ_j com $i \neq j$ é nulo, então $R = 0 \oplus \dots \oplus 1 = 1$. Caso $b_i = 0$, então γ_i não fará parte do cálculo de R e terá o valor de $0 \oplus \dots \oplus 0$. Temos que $R = b_i$.

É claro que esse algoritmo não faz nada de especial em um primeiro momento. O servidor tem acesso ao valor i do usuário, então a recuperação de informação não é privada. Além disso, o processo é uma complicação da simples tarefa de recuperação de um bit. Isso muda quando há a introdução da criptografia homomórfica. Yi *et al* usam um sistema criptográfico que permite operar o Ou Exclusivo e o produto de maneira homomórfica. Assim, no passo 1, cada α_n é criptografado antes de ser enviado ao servidor, juntamente a sua chave pública. No passo 2, o servidor criptografa cada β_n, assim como obtém uma encriptação de 1 e calcula cada γ_j usando operações homomórficas. No passo 3, faz o mesmo para o cálculo de R e envia uma encriptação de R para o usuário, que detém a chave privada e pode acessar seu valor real. Assim o servidor entregou o valor do bit b_i sem saber o valor de i.

Também podemos ver o trabalho realizado por Drozdowski *et al*[29]. Nele, vemos a aplicação mais palpável descrita até o momento. A aplicação consiste em reconhecimento facial de forma encriptada por meio de operações homomórficas. A operação é simples e composta de 3 passos.

1. O cliente extrai biometrias registradas em um vetor $\mathbf{v} = (v_1, \ldots, v_n)$ e encripta em $\mathbf{p} = E(\mathbf{v})$. Após manda $\mathbf{p}$ ao servidor.

2. O servidor dispõe de diferentes encriptações $\mathbf{c_m}$ de informações biométricas. O servidor então calcula o vetor $\mathbf{R}$ de distâncias entre $\mathbf{p}$ e cada $\mathbf{c_m}$. A distância é calculada por $\sum_{n=0}^{s}(\mathbf{c_i} - \mathbf{p_i})^2$. Para um pré-determinado vetor de limiar $\mathbf{t}$, é calculado um vetor $\mathbf{d} = \mathbf{R} - \mathbf{t}$. Esse vetor é enviado a um terceiro agente confiável.

3. Esse terceiro agente decripta $\mathbf{d}$ e analisa cada uma de suas componentes. Se um dos componentes é positivo, é mandada uma decisão de *aceitar* o cliente. Caso contrário, é enviado um *rejeitar*.

Primeiramente, observe que o cliente envia somente informações encriptadas ao servidor. Note também que o servidor não tem nenhuma informação biométrica em claro. A distância é calculada por meio de operações homomórficas, ou seja, o servidor pode realiza-las. Após as operações, um terceiro participante que detêm a chave privada pode decriptar os valores de $\mathbf{d}$. Note que sabendo o valor de d não diz nada sobre o as informações do cliente. Por último, se ao menso um valor de $\mathbf{d}$ é positivo, então há uma correspondência com ao menos uma das informações biométricas do banco, logo a entrada é aceita. Caso contrário, não há correspondência no banco, então a entrada é rejeitada. Essa aplicação é útil para autenticar se uma pessoa com determinadas informações biométricas faz parte de um grupo de pessoas autorizadas.

Em termos mais diretamente ligados a Mineração de Dados em si, podemos citar o trabalho de [30]. Nele, os autores desenvolvem um método para realizar regressão linear em um banco de dados particionado verticalmente, isto é, as diferentes partições do banco armazenam diferentes atributos de um mesmo registro. Em um contexto regular de regressão linear, um atributo vetor de atributo Y pode ser relacionado a uma matriz de atributos X por um vetor de pesos β da seguinte forma:

$$\beta = (X^T X)^{-1} X^T Y$$

Os autores propõem uma situação no qual os atributos de X, isto é, as colunas, são divididas em dois participantes, por isso o nome partição vertical, e o vetor Y pertence a um terceiro participante. São dados os nomes A, B, C aos participantes e dito que desejam encontrar β. Ou seja, o problema é reduzido ao seguinte:

$$\beta = \left(\begin{pmatrix} V_A^T \\ V_B^T \end{pmatrix} V_A V_B \right)^{-1} \begin{pmatrix} V_A^T \\ V_B^T \end{pmatrix} V_C$$

Desta forma, o valor de β depende apenas de operações de soma e produto, que compõem a multiplicação de matrizes, e os valores de V_A, V_B, V_C. Sabendo disso, o que foi feito pelos autores pode ser resumido nos seguintes passos.

1. Um quarto participante gera uma chave pública e envia aos participantes.

2. Cada participante forma uma matriz encriptada com os valores de suas matrizes e enviam tais valores para um quinto participante, chamado de minerador.

3. O minerador utiliza das propriedades homomórficas e calcula β encriptado, enviando-o ao participante que criou as chaves.

4. O participante que criou as chaves decripta o valor recebido e envia aos demais participantes.

Em [31], a Criptografia Homomórfica é utilizada para realizar regressão logística. O objetivo consiste em achar o mínimo de uma função, que pode ser achada minimizando um vetor de pesos ω. Mas precisamente, os autores chegam a conclusão que o peso pode ser calculado iterativamente da seguinte forma:

$$\omega_{k+1} = \omega_k - \eta \left(\sum_i^n y_i \nabla L \times x_i + \nabla R \right)$$

Onde x_i, y_i pertencem ao dataset de treino. Desta forma, o passo iterativo é linearizado e pode ser calculado utilizando-se operações homomórficas. Desta forma, os autores conseguem distribuir este cálculo em diversos trabalhadores, com o resultado

sendo decriptografado pelo servidor, que depois atualiza os trabalhadores. Assim, os trabalhadores não sabem o valor dos dados, mas contribuem para a aceleração da regressão logística. Os autores também afirmam que isso pode ser usado para aprendizado federado, visto que os dados estão criptografados, podendo assim acontecer uma agregação de dados de múltiplas fontes sem a perda de suas privacidades. Um trabalho semelhante que obtém resultados de aprendizado federado é o de [32], que utiliza da Criptografia Homomórfica para calcular pesos de redes neurais na análise de imagens médicas.

3.2 Mineração por Regra de Associação de Dois Participantes

Está última aplicação merece mais atenção pois servirá de base para os métodos propostos. O trabalho foi desenvolvido por Kaosar, Paulet e Yi [6]. A ideia consiste em gerar regras de associação em um conjunto de dados distribuído entre dois participantes, sem que um saiba o conteúdo do conjunto do outro. Um exemplo prático que os autores apresentam se trata de dois hospitais que têm interesse em gerar regras de associação sobre os dados de ambos os hospitais, porém não podem fazê-lo por questões éticas/legais relacionadas a compartilhamento de informações de pacientes.

Kaosar *et al* utilizam os mesmos princípios para a criação de regras de associação introduzidos no Capítulo 2 para este trabalho. Nele, dois participantes A e B detêm um conjunto de dados distribuído, cada um contendo uma parte desse conjunto, e planejam descobrir regras de associação sobre esse conjunto de maneira que um não descubra o conteúdo do conjunto do outro. Para isso, é necessária uma maneira distribuída de calcular o Suporte e a Confiança de cada regra de maneira distribuída. Isso é feito da seguinte forma: sejam DB_A e DB_B os dois conjuntos de dados e sejam $|DB_A|$ e $|DB_B|$ os respectivos tamanhos, isto é, o número de transações de cada conjunto. Seja agora, para um conjunto S, c_A o número de ocorrências de S em A e c_B o número de ocorrências em B. Por último, seja $s/100$ um valor de Suporte mínimo desejado, com $0 < s < 100$. Temos:

$$Suporte_S = \frac{\text{número de transações onde o conjunto S aparece}}{\text{número total de transações}} = \frac{c_A + c_B}{|DB_A| + |DB_B|}$$

Precisamos saber se $Suporte_S$ é maior que $s/100$ ou não, ou seja:

$$\begin{gathered} Suporte_S \geq s/100 \\ \frac{c_A + c_B}{|DB_A| + |DB_B|} \geq s/100 \\ 100c_A + 100c_B \geq s|DB_A| + s|DB_B| \\ 100c_A - s|DB_A| \geq +s|DB_B| - 100c_B \end{gathered}$$

Note agora que um lado da inequação só depende de A e o outro só depende de B. Além disso, a primeira inequação é verdadeira, se e somente se, a última também for. Chamemos a última inequação de inequação do Suporte. O esquema para o cálculo de forma privada ocorre da seguinte forma:

1. A gera um sistema criptográfico homomórfico e envia a chave pública para B juntamente ao que foi calculado na parte esquerda da inequação do Suporte, de maneira encriptada.

2. B calcula a parte da direita da equação do Suporte e encripta com a chave pública de A.

3. B usa um mecanismo de comparação encriptada e manda o resultado de volta para A.

4. A utiliza sua chave privada para decriptar o resultado e ver se a comparação resultou em verdadeiro ou falso. Assim, descobre se o conjunto é frequente ou não e envia o resultado a B.

É importante entender o algoritmo acima, pois será importante para o resto do trabalho. Além disso, é fácil perceber que o mesmo processo acima pode ser usado para o cálculo da Confiança. Seja $S_1 \Rightarrow S_2$ a regra a ser avaliada. Além disso, sejam L_i e l_i as contagens de ocorrências de $S_1 \cup S_2$ e S_1 respectivamente em um

participante $i \in A, B$. Por fim, seja $c/100$ a Confiança mínima desejada, com $0 < c < 100$. Temos:

$$\text{Confiança}_{S_1 \Rightarrow S_2} = \frac{\text{nº de transações onde o conjunto } S_1 \cup S_2 \text{ aparece}}{\text{nº de transações onde o conjunto } S_1 \text{ aparece}} = \frac{l_A + l_B}{L_A + L_B}$$

Assim como para o Suporte, temos:

$$\begin{gathered}
\text{Confiança}_{S_1 \Rightarrow S_2} \geq c/100 \\
\frac{l_A + l_B}{L_A + L_B} \geq c/100 \\
100l_A + 100l_B \geq cL_A + cL_B \\
100l_A - cL_A \geq cL_B - 100l_B
\end{gathered}$$

Assim, podemos seguir os mesmos 4 passos estabelecidos anteriormente para o cálculo do Suporte de um conjunto para agora calcular a Confiança de determinada regra. É importante citar que o trabalho usa o algoritmo Apriori [33]. A vantagem desse algoritmo está na maneira que cria os conjuntos de itens. Nele, começa-se por conjuntos unitários e conjuntos maiores são feitos pela união de conjuntos menores. Nota-se que um conjunto de dois itens terá Suporte menor ou igual ao Suporte individual de cada um dos itens, pois cada vez que a dupla é contada, o item individual também é. Porém, é importante observar que nesse ambiente distribuído, isso acarreta também em um maior número de troca de mensagens. Com o cálculo distribuído e privado do Suporte e da Confiança, o resto do algoritmo segue como o original.

3.3 Mineração por Regra de Associação de Múltiplos Participantes

A ideia principal deste trabalho surge como uma extensão do trabalho realizado por Kosar, Paulet e Yi [6], explicado na seção anterior. Trata-se de lançar mão das propostas utilizadas para a criação de regras de associações para dois participantes e ampliá-las para o uso de n distintos participantes, com $n \in \mathbb{N}, n > 1$.

Deve-se primeiramente lembrar que a criação de regras de associação depende primariamente do cálculo de duas medidas: Suporte e Confiança. Na seção anterior, foram usadas manipulações algébricas para separar os cálculos em partes dependentes de cada participante. Para fazer o mesmo em um caso genérico, deve-se antes definir notação sobre o que está sendo falado. Seja i um índice de um dos participantes e seja $i \in 1, \ldots, n$, com n sendo o número de participantes. Para seguir a mesma notação do Capítulo 3, seja $|DB_i|$ o número de transações presentes no conjunto de dados do participante i. Ao mesmo tempo, para certo conjunto de itens S, seja c_i o número de vezes que S aparece no conjunto de dados do participante i. Da mesma forma, seja $s/100$ o Suporte mínimo a ser aceito. Com isso, temos o seguinte:

$$Suporte_S = \frac{\text{número de transações onde o conjunto S aparece}}{\text{número total de transações}} = \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} |DB_i|}$$

Precisamos saber se $Suporte_S$ é maior que $s/100$, ou seja:

$$\begin{gathered} Suporte_S \geq s/100 \\ \frac{\sum_{i=1}^{n} c_i}{\sum_{i=1}^{n} |DB_i|} \geq s/100 \\ 100 \sum_{i=1}^{n} c_i \geq s \sum_{i=1}^{n} |DB_i| \\ \sum_{i=1}^{n} 100c_i - s|DB_i| \geq 0 \end{gathered}$$

Como esta inequação é importante, será dado um nome para futura referência.

Definição 3.1 (Inequação do Suporte Distribuído)**.** *A inequação* $\sum_{i=1}^{n} 100c_i - s|DB_i| \geq 0$ *é chamada de Inequação do Suporte Distribuído e deve ser satisfeita segundo as condições previamente estabelecidas para o conjunto* S *ter Suporte mínimo.*

Dessa forma, temos que cada elemento no somatório depende apenas do participante i e a combinação de tais elementos é feita a partir da soma, uma operação que é homomórfica em muitos sistemas. Temos assim um bom candidato para o cálculo

do Suporte de maneira distribuída e privada. O mesmo deve ser feito para a Confiança. Seguindo a notação do Capítulo 3, seja $S_1 \Rightarrow S_2$ uma regra a ser avaliada. Também sejam l_i e L_i as contagens de ocorrências de $S_1 \cup S_2$ e S_1 respectivamente em um participante i, e seja $c/100$ a Confiança mínima desejada. Dessa forma:

$$\text{Confiança}_{S_1 \Rightarrow S_2} = \frac{\text{nº de transações onde o conjunto } S_1 \cup S_2 \text{ aparece}}{\text{nº de transações onde o conjunto } S_1 \text{ aparece}} = \frac{\sum_{i=1}^{n} l_i}{\sum_{i=1}^{n} L_i}$$

Para a Confiança ser mínima, temos:

$$\begin{aligned} \text{Confiança}_{S_1 \Rightarrow S_2} &\geq c/100 \\ \frac{\sum_{i=1}^{n} l_i}{\sum_{i=1}^{n} L_i} &\geq c/100 \\ 100 \sum_{i=1}^{n} l_i &\geq c \sum_{i=1}^{n} L_i \\ \sum_{i=1}^{n} 100 l_i - c L_i &\geq 0 \end{aligned}$$

De maneira semelhante ao Suporte, convém explicitar essa inequação.

Definição 3.2 (Inequação da Confiança Distribuída)**.** *A inequação* $\sum_{i=1}^{n} 100 l_i - c L_i \geq 0$ *é chamada de Inequação da Confiança Distribuída e deve ser satisfeita segundo as condições previamente estabelecidas para a regra* $S_1 \Rightarrow S_2$ *ter Confiança mínima.*

Assim, temos o mesmo tipo de trabalho que foi feito no Suporte, agora na Confiança, tendo as mesmas vantagens e características. Com isso, temos técnicas para podermos unir à criptografia homomórfica e gerar regras de associação.

Capítulo 4

Métodos de Geração Distribuída de Regras de Associação

Neste capítulo, serão apresentados os métodos desenvolvidos para o problema de criação de regras de associação de maneira privada para n participantes e suas implementações. Serão apresentados três métodos, cada um com seu conjunto de vantagem e desvantagens, assim como uma versão padrão, sem a preocupação com privacidade, para fins de comparação. Para simplificação, todos os métodos foram implementados para um ambiente de quatro participantes, mas funcionam para qualquer ambiente com dois ou mais participantes.

Para a simulação de múltiplos participantes, foi montado um ambiente utilizando a ferramenta de virtualização Docker [34]. Cada participante se trata então de um contêiner Docker em uma rede virtual, cada um com uma das fatias dos dados divididos. A comunicação entre as instâncias de contêineres foi feita através de um script Python inspirado em outro obtido da web [35]. Esta comunicaão funciona por meio de *sockets* TCP, onde para cada transferência de arquivo, uma das instâncias age como servidor, esperando que outra instância, o cliente, mande o arquivo por meio de uma porta previamente selecionada.

As ferramentas criptográficas utilizadas foram providas pela biblioteca OpenFHE [36]. Ela se trata de uma biblioteca open-source escrita em C++ que disponibiliza primitivas criptograficas que permitem operações homomórficas. Além do BFV já apresentado, dispões de outros métodos, como Brakerski-Gentry-Vaikuntanathan

(BGV)[37] para números inteiros, Cheon-Kim-Kim-Song (CKKS)[38] para números reais e Ducas-Micciancio (DM)[39] e Chillotti-Gama-Georgieva-Izabachene(CGGI) [40] para booleanos.

Para comparar diferentes métodos no sentido de etapas de comunicação, é criada a seguinte definição:

Definição 4.1 (Passo)**.** *Em um método distribuído neste trabalho, um Passo se trata de uma etapa da comunicação realizada entre diferentes membros, com sua conclusão sendo necessária para o começo da próxima comunicação entre quaisquer membros.*

Exemplo 4.1. *(Esquema com 3 Passos) Sejam A, B e C 3 participantes em algum evento que necessite de comunicação. Nele, a comunicação é feita da seguinte forma: A manda dados para B e C. B processa os dados recebidos e manda o resultado para C. A envia novos dados a C. Note que são 3 Passos envolvidos:*

1. *A envia dados a B e a C.*
2. *B envia dados a C.*
3. *A envia dados a C.*

Note que para cada passo acontecer, é necessário que o anterior esteja completo. Mesmo A não atuando no Passo 2, deve esperar que este acabe para enviar os dados para C no Passo 3. Note também que apesar de ocorrerem 2 comunicações no Passo 1 , o Passo é somente um, pois as comunicações ocorrem logicamente em paralelo. Em questões práticas, A enviará para B e só depois para ou vice versa, mas essa ordem não faz diferenças práticas e é vista como paralela.

Antes de prosseguir sobre os métodos em si, é importante discutir suas entradas. Temos 3 arquivos principais para cada participante: *out_key*, com os conjuntos, *out_val*, com as frequências dos conjuntos e *out_val_pre*, com os valores da parcela correspondente da Inequação do Suporte Distribuído (Definição 3.1) dos conjuntos. Mais informações sobre os arquivos são discutidos no Capítulo 5.

4.1 Método Padrão

Para fins de comparação, foi desenvolvido também um Método Padrão sem o uso de criptografia, ou seja, sem privacidade entre as bases de dados. Desta forma, é possível calcular as Regras de Associação de forma direta.

O processo se resume a dois Passos de comunicação. No primeiro, um membro escolhido recebe os dados das frequências dos conjuntos dos outros participantes, isto é, *out_val*. Os valores de cada conjunto são então somados para formar o valor total da frequência de cada conjunto, tornando o problema que era distribuído em uma situação agora com uma base de dados unificada. Com isso, é possível tratar o problema como uma situação clássica de geração de Regras de Associação. Portanto, é possível calcular o Suporte de cada conjunto conforme a Definição 2.15 e compará-lo com um valor de suporte mínimo para encontrar quais conjuntos são considerados frequentes.

O processo para a etapa de cálculo da Confiança é semelhante. Primeiramente, note que os valores dos arquivos seguem uma ordem de tamanho de conjuntos. Por exemplo, em *out_key*, temos que as linhas correspondentes a conjuntos com 2 itens aparecem somente após serem listados os conjuntos com 1 item. Desta forma, para cada conjunto frequente descoberto na etapa do suporte localiza-se as entradas anteriores no arquivo de frequência que são seus subconjuntos. Para cada subconjunto encontrado, a Confiança é calculada de acordo com a Definição 2.16. Os pares formados por um conjunto e seu respectivo subconjunto que tem confiança maior que um valor mínimo são considerados como Regras de Associação válidas.

Para encontrar os subconjuntos dos conjuntos frequentes, é realizada uma busca linear no arquivo com o nome dos conjuntos. Como o arquivo é ordenado, somente é necessária a busca nos conjuntos anteriores ao conjunto frequente investigado. No pior caso possível, onde todos os conjuntos são frequentes, é necessária uma busca para todos os conjuntos. Se assumirmos n conjuntos possíveis na base, para a procura dos subconjuntos do último item da lista, é necessária uma busca por $n-1$ conjuntos. Para o penúltimo, $n-2$ conjuntos, e assim por diante. No total, seriam $(n-1)+(n-2)+\cdots+1 = \frac{(n-1)(n-2)}{2}$. A complexidade do pior caso é quadrática,

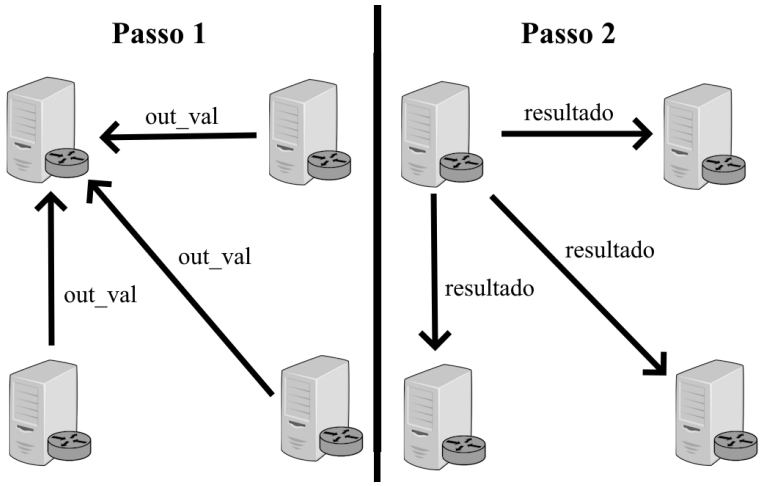

Figura 4.1: Comunicações no Método Padrão Ilustrado com 4 participantes.

ou seja, $O(n^2)$. Isto aparenta ser um problema, mas no caso prático se deseja que poucos conjuntos sejam frequentes, então escolhe-se um suporte mínimo grande o suficiente para que isso ocorra. Ou seja, poucos conjuntos são investigados para o cálculo de seus subconjuntos.

O Passo 2 consiste então do retorno das regras geradas pelo membro que realizou os cálculos aos demais membros. O esquema ilustrando os Passos de comunicação pode ser visto na figura 4.1, com o exemplo de 4 participantes.

4.2 Método Proposto 1: Sistema Homomórfico com Soma e Produto

O primeiro método desenvolvido é composto de 7 passos de comunicação. Assim como no método padrão, um dos integrantes é escolhido para ser o responsável para os cálculos necessários para a geração das Regras de Associação. Por conveniência, chamemos este responsável de *Operador*. De maneira semelhante, definimos o *Contextualizador*, que é o responsável por criar o contexto criptográfico que será utilizado no método. Essa terminologia será também utilizada nos seguintes métodos propostos.

O processo começa com o membro *Contextualizador* gerando o contexto criptográfico, isto é, a instância do sistema BFV utilizado assim como um par de público-privado de chaves. O Passo 1 consiste no envio do contexto criptográfico e da chave pública para os demais participantes.

Após isso, todos os participantes encriptam seus respectivos arquivos *out_val_pre* usando a chave pública recebida e enviam os textos cifrados ao *Operador* no Passo 2. O *Operador* pode então calcular o valor do lado esquerdo da Inequação do Suporte Distribuído (Definição 3.1) no espaço encriptado para cada um dos conjuntos criptografados. Para questões de segurança, cada valor calculado é multiplicado por um inteiro aleatório maior que 0. Essa nova sequência de valores cifrados é devolvida ao *Contextualizador* no que consiste o Passo 3.

Por fim, o *Contextualizador* em posse da chave privada decripta os valores recebidos do *Operador* e os compara individualmente com 0. Note que multiplicar o lado esquerdo da Inequação do Suporte Distribuído por um inteiro maior que 0 não muda seu sinal, isto é, o valor obtido após o produto realizado pelo *Operador* é maior que 0 se e somente se a parte esquerda da Inequação do Suporte Distribuído for maior que 0. Assim, o *Contextualizador* descobre os índices dos valores decriptografados maiores que 0 e descobre os conjuntos com suporte mínimo. Como Passo 4, envia esses índices para os demais participantes. O processo desta primeira metade do método pode ser visto na Figura 4.2.

Para cada índice recebido, os participantes descobrem os conjuntos de suporte mínimo checando sua posição em *out_key* e, de maneira semelhante ao método padrão, criam um arquivo com todas as regras de associação possíveis a partir destes conjuntos. Além disso, criam um outro arquivo, seguindo a mesma ordem deste primeiro, com os valores de sua parcela da Inequação da Confiança Distribuída (Definição 3.2), semelhante ao que foi feito para *out_val_pre*.

Desta forma, o resto da segunda metade do método segue semelhante a primeira. O Passo 5 consiste nos participantes enviando os valores criptografados ao *Operador* que os soma e multiplica por um inteiro positivo, mandando os resultados para o

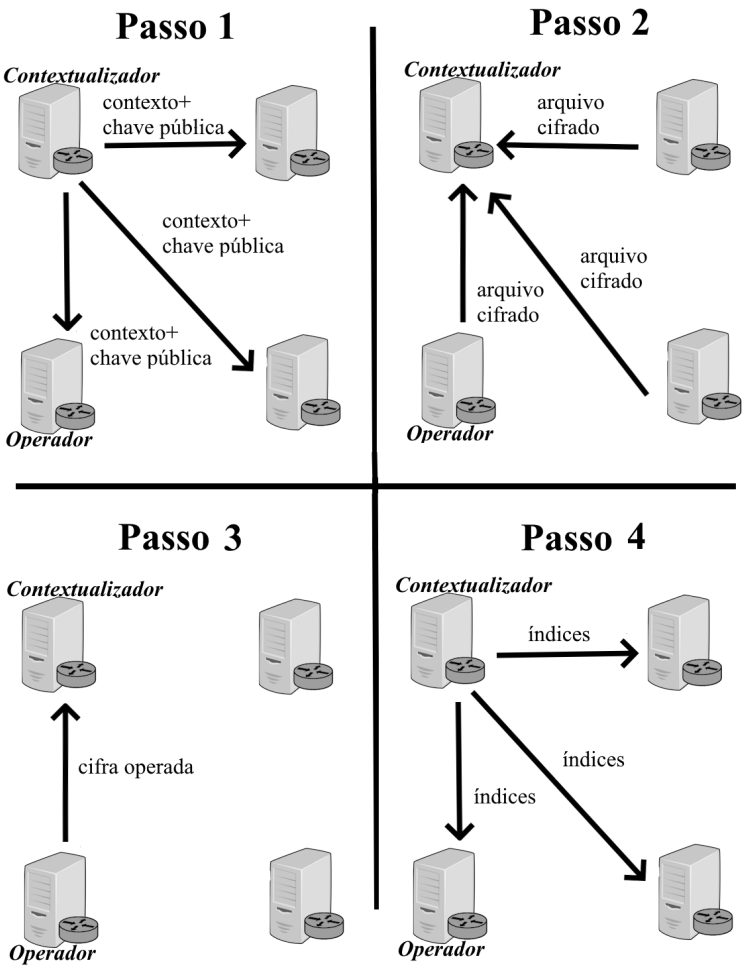

Figura 4.2: Comunicações na primeira metade do Método 1 ilustrado com 4 participantes.

Contextualizador no Passo 6. O *Contextualizador* então compara os valores com 0 e os valores que passam pelo teste são os referentes aos que satisfazem a Inequação da Confiança Distribuída, ou seja, às regras que tem confiança mínima. Por fim, o Passo 7 consiste em enviar os índices das regras de associação válidas para os demais participantes. Esses Passos são ilustrados da mesma forma que na primeira metade, com exceção da criação do contexto.

É importante notar o funcionamento do método. Caso todos os participantes executem suas funções de forma correta, o único participante que terá acesso a algum texto em claro que não seja o seu original será o *Contextualizador*. Porém, o texto em claro será a soma dos textos de todos, multiplicado por um número aleatório, ou seja, um dado que não diz nada sobre os dados individuais. Além disso, o único que tem acesso aos dados encriptados dos outros participantes é o *Operador*, que não possui a chave privada. Ou seja, a única maneira de um participante descobrir informações de outro ocorre caso o *Operador* e o *Contextualizador* cooperem, com o *Operador* enviando dados sigilosos encriptados dos outros participantes para o *Contextualizador* decriptografar.

4.3 Método Proposto 2: Sistema Homomórfico de Limiar com Soma

O segundo método surge com a ideia de superar o problema encontrado no método anterior no caso de cooperação maliciosa. A mudança central está na utilização de criptografia de limiar ao invés do tradicional método de apenas uma chave privada.

A biblioteca utilizada, OpenFHE, diferencia a geração de chaves para criptografia de limiar dependendo das operações a serem utilizadas. Como visto na Seção 2.2.3 que a chave pública do BFV consiste em duas partes, uma delas sendo necessária somente para a multiplicação. Assim, é possível gerar somente uma chave pública mais simples para o caso de ser só utilizada a soma. Assim, o Método 2 se propõe em utilizar somente a soma para a simplicidade da geração de chaves. O método mais completo utilizando o produto é o proposto no Método 3.

Da mesma forma que o método anterior, é necessário definir um *Operador* e um *Contextualizador*. Porém, desta vez, o *Contextualizador* somente começa o processo de criação da instância do sistema e não é o responsável pela decriptação. Mesmo assim, gera o contexto e um par de chaves.

O Passo 1 se trata do envio do contexto para todos os participantes juntamente à chave pública para um segundo participante. Esse segundo participante usa a chave pública para gerar um novo par de chaves de limiar com base na chave pública recebida. Após isso, o Passo 2 consiste neste segundo participante enviando sua chave pública para um terceiro. Este terceiro repete o processo executado pelo segundo, isto é, gerar um novo par e propagar a chave pública para um quarto participante. O processo se repete até o último participante receber uma chave pública e poder gerar seu próprio par. Note que o Passo 1 se trata da comunicação do *Contextualizador* com um segundo participante. Após isso, ocorrem $n-2$ Passos para geração de chaves intermediárias, com n sendo o número de participantes, pois todos os participantes exceto dois, o *Contextualizador* e o último, tem o mesmo processo de gerar um par de chaves com base na chave pública anterior e propagar uma nova chave pública. Assim, o próximo Passo, ou seja, o Passo de número n consiste no último participante enviar sua chave pública para os demais. Esta é a chave que será utilizada para a encriptação. Todo o esquema é ilustrado na Figura 4.3.

O resto do método segue como o anterior, com pequenas alterações. Primeiramente, o *Operador* e o *Contextualizador* podem ser o mesmo participante, pois o *Contextualizador* não possui mais o poder de decriptografar por si só, logo pode receber todos os textos cifrados. Em segundo lugar, a etapa realizada pelo operador não conta com a operação de produto por um inteiro positivo aleatório, pois esta operação não é permitida pelo contexto criado.

Desta forma, o Passo $n+1$ consiste no envio de textos cifrados para o *Operador* assim como foi feito no Passo 2 do Método 1. O Passo $n+2$ consiste no envio do resultado do *Operador*, assim como no Passo 3 do método anterior, porém agora não enviando ao *Contextualizador*, mas sim a todos os participantes. Cada participante

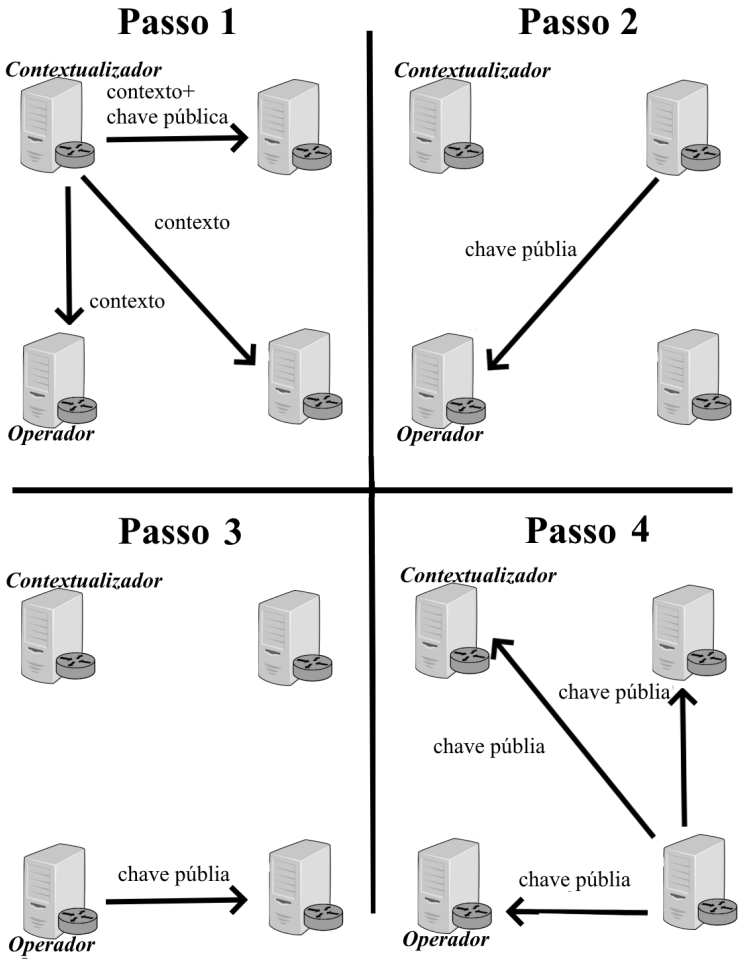

Figura 4.3: Comunicações para a geração de chave pública no Método 2 ilustrado com 4 participantes.

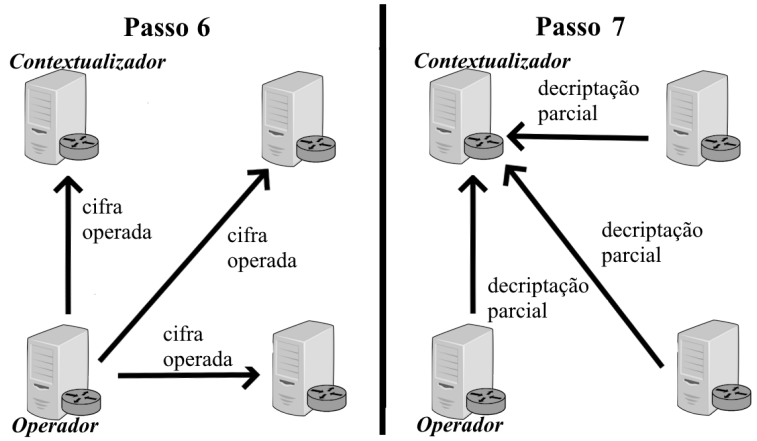

Figura 4.4: Comunicações para adecriptação no Método 2 ilustrado com 4 participantes.

faz a decriptação parcial do texto cifrado utilizando sua chave privada e enviam o resultado ao *Contextualizador*, no que consiste o passo $n + 3$. A primeira metade termina com o *Contextualizador* unindo as decriptações parciais para obter a decriptação completa e comparando os valores a 0, ou seja, verificando se respeitam a Inequação do Suporte Distribuído. Esta etapa de decriptação pode ser vista na Figura 4.4. O Passo $n+4$ então é semelhante ao Passo 4, no qual o *Contextualizador* envia os índices dos conjuntos frequentes o suficiente para os demais participantes.

A segunda metade ocorre de maneira similar à primeira. Da mesma forma que o método anterior, cada participante encripta e envia os valores da sua parcela da Inequação da Confiança Distribuída para o *Operador* no Passo $n + 5$. O Passo $n+6$ consiste no envio da soma dos valores das parcelas criptografadas encontradas pelo *Operador* para os demais participantes. Assim, cada participante decripta parcialmente o arquivo com a soma e envia de volta ao *Contextualizador* no Passo $n + 7$. Finalmente, o *Contextualizador* junta os valores parcialmente decriptados e compara-os com 0, enviando os índices dos valores positivos referentes às regras válidas de volta aos demais participantes, constituindo o Passo $n + 8$.

Este segundo método resolve o problema principal do primeiro, mas traz consigo um empecilho. Note que o *Contextualizador* tem acesso aos valores do lado esquerdo da Inequação do Suporte Distribuído e o lado esquerdo da Inequação da Confiança

Distribuída. Este valor em si não diz nada, mas note o que pode ser feito de existir mais informações. Seja v um dos valores obtidos pela *Contextualizaor* e seja I o índice de um dos participantes o qual se deseja descobrir informações sigilosas. Temos:

$$v = \sum_{i=1}^{n} 100c_i - s|DB_i|$$

$$v = (\sum_{i=1}^{I-1} 100c_i - s|DB_i|) + (100c_I - s|DB_I|) + (\sum_{i=I+1}^{n} 100c_i - s|DB_i|)$$

$$v - (\sum_{i=1}^{I-1} 100c_i - s|DB_i|) - (\sum_{i=I+1}^{n} 100c_i - s|DB_i|) = (100c_I - s|DB_I|)$$

Assim, se as informações sobre os demais participantes for conhecida, é possível descobrir o valor de um participante específico. Note que a informação obtida pode não ser tão relevante, pois é algo que foi gerado a partir da combinação de dois dados diferentes do participante alvo. Mas também note que se for descoberto o valor de DB_I, mesmo que por meios externos, é possível descobrir a frequência de todos os conjuntos de itens. De outra forma, se for conhecida a frequência de apenas um conjunto de itens de I, é possível descobrir o valor de DB_I. Dependendo da aplicação, pode ser esperado que algum participante tenha algum conjunto de itens com frequência 0, assim o conjunto com o menor valor obtido na equação acima descreverá que conjunto é este. Utilizando o ataque descrito acima e o conhecimento de um conjunto com frequência 0, é possível descobrir todas as frequências de I.

4.4 Método Proposto 3: Sistema Homomórfico de Limiar com Soma e Produto

Como discutido no método anterior, o terceiro método se manifesta como uma união dos métodos anteriores, ou seja, utiliza criptografia que ao mesmo tempo é de limiar e também comporta o uso do produto.

Este método é praticamente idêntico ao anterior no sentido de execução após o criptossistema estar bem estabelecido. A única diferença está no fato do *Operador*

multiplicar seu resultado por um inteiro positivo, assim como faz no Método 1. Por isso, o foco nesta seção será na etapa de estabelecimento do criptossistema.

Diferentemente do caso anterior, a maneira que a biblioteca OpenFHE implementa a geração da chave para o produto limiar ocorre em duas etapas. Ou seja, a geração de chaves que no método anterior acontecia em n etapas, desta vez ocorre em aproximadamente $2n$ etapas. Ocorre da seguinte forma: o *Contextualizador* instancia o criptocontexto e o envia aos demais participantes e envia a chave pública para um segundo participante como o Passo 1, idêntico ao do método anterior. Deve-se notar que esta chave pública é composta de uma chave de encriptação e uma chave para a realização do produto. O segundo participante cria um par de chave pública-privada assim como no método anterior, mas também manipula a chave para a realização do produto recebida. Essa chave pública do par gerado funciona como a chave de encriptação e juntamente com a chave para a realização do produto, formam a nova chave pública em desenvolvimento. O envio desta nova chave a um terceiro participante consiste no Passo 2. Os passos subsequentes ocorrem no mesmo estilo do Passo 2, assim como no método anterior. Assim, ao chegar no último participante, foram executados $n - 1$ passos (lembre que no método anterior, o n-ésimo passo ocorre quando o último participante devolve a chave pública completa).

Com a primeira parte completa, o último participante pode operar sobre a chave pública para finalizar a etapa de geração da chave de encriptação, assim como ocorreu no método anterior, e iniciar a segunda etapa de geração da chave do produto. Após terminar o processo, o Passo n consiste no envio para da chave de encriptação finalizada juntamente a chave para realização do produto incompleta para outro participante. Os próximos passos ocorrem de maneira semelhante à primeira parte, na qual um participante recebe uma chave, opera sobre ela e envia para o próximo. Desta forma, somente após o Passo $2n - 2$ é chegada a vez do último participante restante operar a chave para realização do produto e finalizá-la. Com a posse desta chave, duas opções são possíveis. Em um caso geral, resta o Passo $2n - 1$, que consiste em enviar está chave para o *Operador*. Porém, se este último participante for o *Operador*, isto não é necessário, pois somente ele precisa da chave para a realização do produto e todos os participantes já estão munidos da chave de encriptação, pois

seu envio é parte da segunda metade da geração de chaves. Por completude, serão considerados $2n - 1$ Passos para a geração do contexto e suas chaves.

Com isto, o resto do processo ocorre da mesma maneira que o método anterior, necessitando no total de $2n - 1 + 8 = 2n + 7$ Passos. Note que este método não tem os problemas dos métodos anteriores. É impossível decriptografar qualquer texto sem o apoio conjunto de todos os participantes, diferentemente do método 1. Além disso, se um dos participantes for alvo de um ataque conjunto dos demais, o valor decriptografado está multiplicado por um inteiro positivo aleatório, ou seja, é uma informação que não pode ser usada para inferir informações sobre este participantes. A única situação onde pode ocorrer vazamento de informações é o caso do *Operador* ser malicioso e não executar o algorítimo de forma correta, isto é, não executar a parte de produto por um inteiro positivo aleatório. Um resumo dos três métodos apresentados pode ser visto no Apêndice C.

Capítulo 5

Dados Utilizados

O conjunto de dados utilizado neste trabalho se trata de fluxos extraídos de roteadores de borda da Rede-Rio/FAPERJ [9]. A Rede-Rio/FAPERJ é uma instituição que provê conectividade através de um *backbone IP* situada no estado do Rio de Janeiro que interliga instituições de ciência, tecnologia e educação, tendo capacidade de transmissão em seu núcleo de até 10 Gbps. A topologia da Rede-Rio/FAPERJ pode ser vista na Figura 5.1.

Os fluxos são disponibilizados pela plataforma IPTraf [41], desenvolvida para servir como fonte de consulta sobre dos fluxos de rede. Munida de um Módulo coletor, a plataforma conta com a coleta de fluxos de um roteador de borda da Rede-Rio/FAPERJ através da ferramenta NFDUMP que posteriormente são enviados ao Laboratório Ravel da COPPE/UFRJ[42]. Cada fluxo é processado a fim de extrair as seguintes características: endereço de origem, endereço de destino, porta de origem, porta de destino, protocolo, flags tcp, número de pacotes, número de bytes e horário de início. A plataforma IPTraf gerou trabalhos de ferramentas de detecção de anomalias em fluxos baseadas em Redes Neurais[43] e Séries Temporais[8].

Diariamente, são coletados mais de 10GB de fluxos. Esses fluxos são divididos em intervalos de 5 minutos, ou seja, distribuídos em 288 arquivos que representam janelas de tempo. Para este trabalho, foram separados 20GB de dados de fluxos extraídos no dia 10 de dezembro de 2022, das 00:00h às 19:45h, totalizando 26988784 fluxos distintos. Note que o intervalo foi escolhido devido a quantia de dados, em um dia mais movimentado que a média.

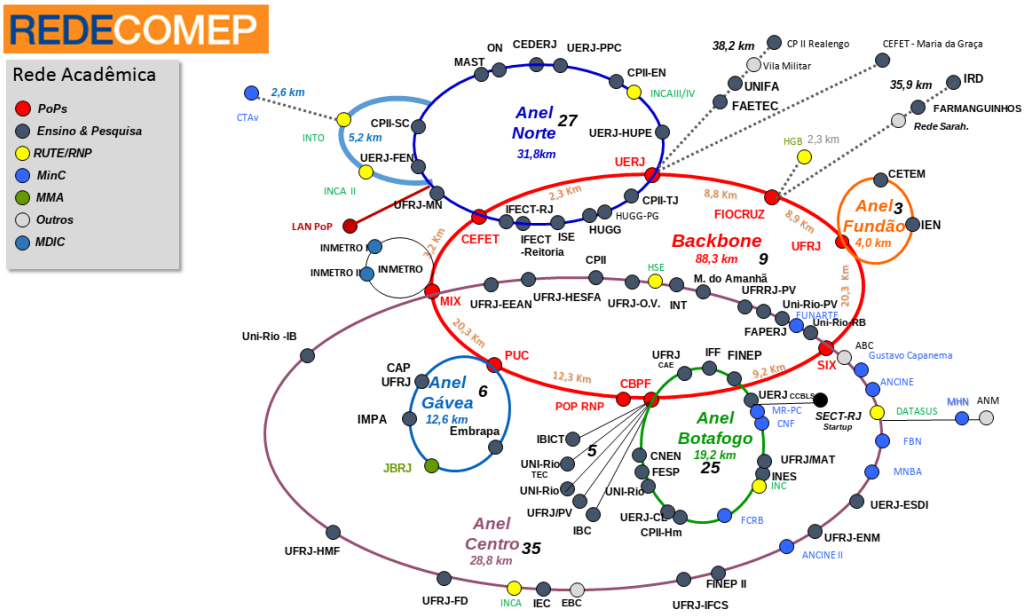

Figura 5.1: Topologia da Rede-Rio/FAPERJ.

Um fator importante a se atentar é o fato de que o trabalho requer uma base de dados distribuída, mas só há o acesso a um roteador de borda. Para suprir essa necessidade, esses 20GB de fluxos foram divididos em quatro subconjuntos, a fim de simular quatro agentes distintos que desejam colaborar. Os diferentes subconjuntos foram formados com tamanhos distintos com o intuito de gerar uma diferenciação das partes de forma a se aproximar de um cenário real. No total, foram separados 238 arquivos de intervalos de 5 minutos em proporções próximas a 40%, 30%, 20% e 10%. Para isso, um script em Python foi desenvolvido e a divisão dos arquivos ficou da seguinte forma: o maior conjunto com 109 intervalos, o segundo maior com 67 intervalos, e os outros dois com 42 e 20 intervalos. Eles foram chamados de A, B, C e D e têm aproximadamente 46%, 28%, 18% e 8% do total de intervalos.

Agora, é importante definir como esses fluxos serão trabalhados. Como foi citado anteriormente, cada fluxo é processado para se encaixar no formato *(endereço de origem, endereço de destino, porta de origem, porta de destino, protocolo, flags tcp, número de pacotes, número de bytes, horário de início)*. É importante lembrar que o objetivo é chegar em regras de associação e notar as diferenças no domínio do problema quando comparado com o Exemplo 2.5. Primeiramente, é possível ver que, no problema do Exemplo 2.5, os conjuntos nas transações apresentam tamanho

variável por se tratarem de compras. Em contraste, no problema da geração de regras de associação das características de fluxos, os conjuntos tem tamanho fixo. Veja bem, o fluxo é descrito por 9 campos, sendo que nenhum campo pode ser faltante ou duplicado. A outra diferença principal está na quantidade de itens possíveis para um conjunto. Enquanto em um exemplo de mercado temos uma quantidade menor de itens, no exemplo dos fluxos em sua forma bruta, temos uma enorme quantidade. Podemos pensar por exemplo na porta de origem, que há 65535 opções diferentes, sendo isso se tratar de somente um dos campos. Para resolver esse segundo problema, é necessário um pré-processamento dos arquivos de fluxo antes que possam ser analisados. Um exemplo de arquivo de fluxos para determinado intervalo de 5 minutos pode ser visto na Tabela 5.1.

Tabela 5.1: Exemplo dos primeiros fluxos de um arquivo.

Endereço de origem	**Endereço de destino**	**Porta de origem**	**Porta de destino**	**Protocolo**	**Flags tcp**	**Número de pacotes**	**Número de bytes**	**Horário de início**
20.201.67.116	146.164.123.218	443	53761	6	1	11	14240	2022-12-09 23:51:33.218
185.196.220.70	152.92.245.110	53550	22	6	1	1	40	2022-12-09 23:51:33.220
192.241.206.121	161.79.45.201	55198	7443	6	1	1	40	2022-12-09 23:51:33.220
71.6.146.130	147.65.123.165	24858	666	6	1	1	44	2022-12-09 23:51:33.220
45.143.200.102	147.65.250.212	52773	6669	6	1	1	40	2022-12-09 23:51:33.220
185.196.220.70	152.92.114.8	58388	22	6	1	1	40	2022-12-09 23:51:33.220
201.38.120.122	200.20.186.94	123	123	17	1	2	152	2022-12-09 23:51:33.220
104.156.155.12	157.86.146.104	54847	8877	6	1	1	40	2022-12-09 23:51:33.221
64.62.197.45	157.86.105.148	51491	53	17	1	1	70	2022-12-09 23:51:33.221
108.61.100.50	157.86.68.245	27015	14334	17	1	1	67	2022-12-09 23:51:33.221
27.124.32.183	139.82.112.28	33918	7548	6	1	1	44	2022-12-09 23:51:33.221
108.158.147.35	139.82.108.212	443	56102	6	1	2	250	2022-12-09 23:51:33.221
167.94.145.30	139.82.176.92	2459	14930	6	1	1	44	2022-12-09 23:51:33.221
4.204.205.157	200.20.0.16	47010	465	6	1	2	92	2022-12-09 23:51:33.222

Para a realização de uma primeira etapa de pré-processamento, serão analisados campo a campo será visto o que pode ser transformado em cada um. Antes disso, é importante lembrar o objetivo deste trabalho, que é comparar diferentes métodos para a criação de regras de associação distribuídas de maneira privada. Ou seja, o conjunto de dados escolhido é importante para a validação da ideia, mas não apresenta requisitos para moldar o formato das regras a serem geradas. Mais especificamente, não há nenhum fim inerente ao problema apresentado, logo o pré-processamento usado aqui não é voltado com alguma certa aplicação em mente, sendo possíveis outros tratamentos nos dados em outros casos. Por exemplo, pode-se

agrupar os endereços de origem por prefixo no intuito de classificar características destes prefixos, o que não é feito neste trabalho.

Os dois primeiros campos são referentes ao endereço de origem e de destino do fluxo. Note que há mais de 3 bilhões de endereços públicos. Obviamente, a maioria não aparecerá nestes fluxos, mas isso mostra a grande variedade possível para cada uma das colunas. Logo, seria necessário um agrupamento para sua utilização neste trabalho. Por este feito requerer uma maior subjetividade, isto é, esse modo de agrupamento depender bastante da aplicação, serão descartadas esses dois campos, pois ainda se tem dados o suficiente para o proposto nos outros campos.

Os dois campos seguintes são referentes às portas de entrada e saída dos fluxos. Da mesma maneira que o campo anterior, há muitas opções. Para ser preciso, há 65535 portas diferentes, então se mostra necessária a necessidade de alguma forma de agrupamento. Seguindo o trabalho [43], uma das formas úteis de separação de portas pode ser feita da seguinte maneira: Portas Baixas, sendo as portas de números 1 a 1023, Portas do Servidor, sendo as portas de números 1024 a 49151, e Portas Dinâmicas, sendo as portas de números 49152 a 65535. Essas portas se mostram úteis em um contexto prático: esse trabalho define um dos métodos de detecção de anomalia integrado ao IPTraf, ou seja, é uma separação que é aplicada diretamente a dados provenientes desse roteador de borda.

Em seguida, temos o tipo de protocolo. Da maneira que o IPTraf processa os fluxos, um valor binário, diferenciando apenas o protocolo TCP do resto. Logo, com apenas dois valores, não há a necessidade de aplicar algum tratamento prévio a esta coluna de dados.

De maneira similar, o campo de flags não apresenta grande variação. O IPTraf simplesmente separa os flags em 4 tipos. Com esse baixo valor, não há a necessidade de algum tratamento maior. Os 4 tipos ocorrem da seguinte forma:

1. Não há flags.

2. Apresenta somente flags ACK, Reset e SYN ou somente flags ACK e Reset.

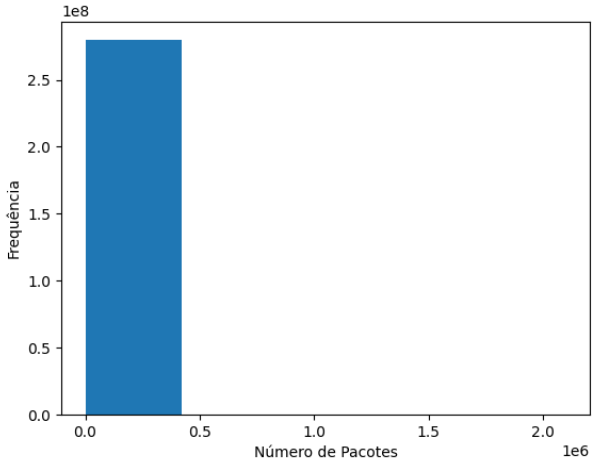

Figura 5.2: Histograma do número de pacotes com 5 intervalos de mesmo tamanho.

3. Apresenta somente flag de Reset ou somente de Reset e FIN.

4. Não se encaixa em nenhum dos anteriores.

Em relação ao número de pacotes, não há alguma divisão trivial sem pensar em alguma aplicação específica. A distribuição do número de pacotes pode ser estudada através de um histograma, como na Figura 5.2. Como pode ser visto, a distribuição ingênua em 5 grupos de tamanhos iguais resulta na grande maioria dos pacotes estarem agrupados no primeiro grupo, com os outros nem mesmo visíveis na imagem. Como esse agrupamento não se mostra adequado, será utilizado a divisão inerente ao nosso sistema decimal e serão divididos nos intervalos $[0, 1]$, $(1, 10]$, $(10, 100]$ e $(100, \infty)$. Essa distribuição pode ser vista na Figura 5.3. Nota-se primeiramente que os 4 agrupamentos criados não tem tamanho similar, mas isso é esperado, pois o ponto do trabalho depende de diferentes frequências para gerar regras de associação. Porém, o importante notar que cada grupo detêm uma frequência mínima. Assim, esse será o agrupamento utilizado.

Para o número de bytes, a situação é similar. Se usarmos uma divisão ingênua de tamanho igual para cada intervalo, temos o resultado da Figura 5.4. O resultado é similar ao encontrado no caso do número de pacotes, então convém adotar-se de uma estratégia similar, utilizando-se da divisão usual do sistema decimal. No entanto, diferentemente do número de pacotes, não é reservado um intervalo para

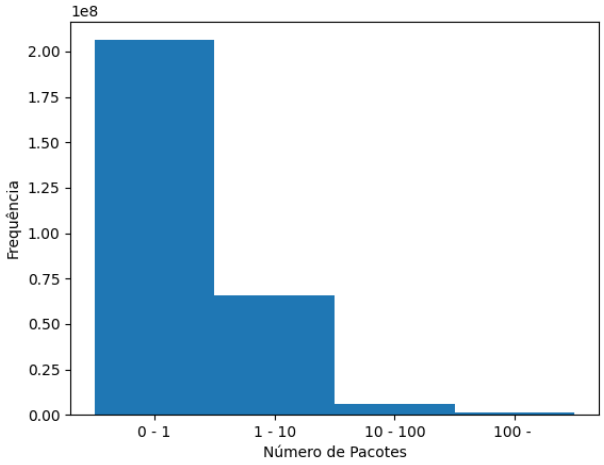

Figura 5.3: Frequência do número de pacotes com 4 intervalos de diferentes tamanhos.

o valor 1, pois não se espera ter um fluxo com somente 1 byte, também sendo assim para menos de 10 bytes. Além disso, espera-se que o valor de mais de 100 bytes seja mais comum. Por esses motivos, os intervalos escolhidos foram $[0, 100]$, $(100, 1000]$ e $(1000, \infty)$. Essa separação pode ser vista na Figura 5.5. Ela contém características semelhantes ao agrupamento utilizado para o número de pacotes e por isso é considerado satisfatório.

O último campo restante é referente a data de início do fluxo. Essa informação pode ser muito relevantes em determinados contextos. No caso deste conjunto de dados, todos os fluxos foram retirados de um mesmo dia, ou seja, para uma aplicação real, poderiam levar a resultados enviesados. Por exemplo, se considerarmos a data completa, só temos resultados de um dia, logo todos formariam um só grupo. Se considerarmos o horário apenas, poderia estar enviesado por ser um dia com tráfego anormal. Por estes motivos, esse campo será descartado.

Com esse pré-processamento, foram selecionados seis dos nove campos iniciais. Além disso, campos que haviam um número muito grande de valores foram agrupados em categorias. Agora, basta notar que é necessária alguma forma de expressar diferentes valores ou grupos em cada coluna como elementos a serem disponíveis em um conjunto. Para isso, as colunas foram numeradas de 0 a 5 e os possíveis valores de uma coluna são da forma número da coluna_número da categoria. Ou seja, isso

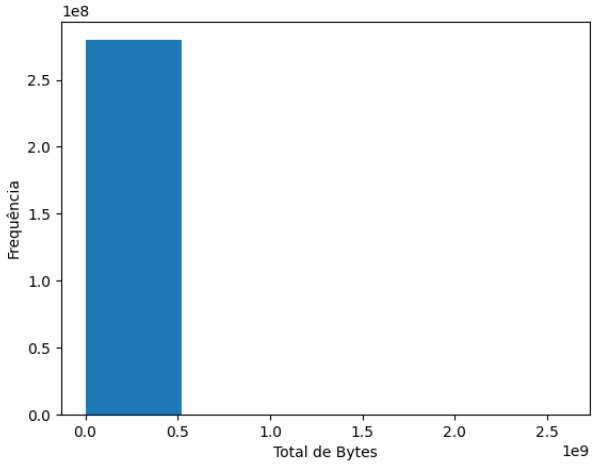

Figura 5.4: Histograma do total de bytes com 5 intervalos de mesmo tamanho.

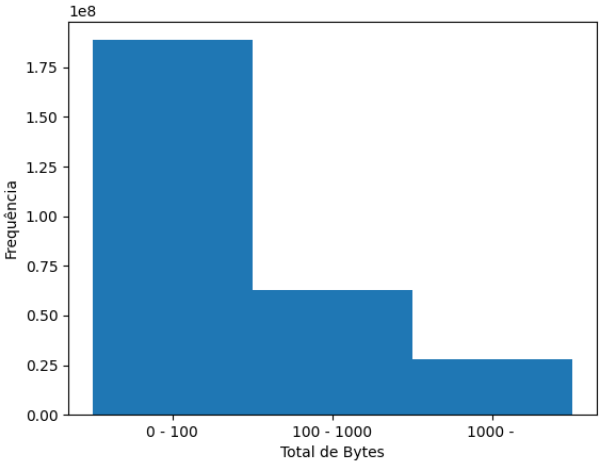

Figura 5.5: Frequência do total de bytes com 3 intervalos de diferentes tamanhos.

ocorre da seguinte forma:

0. Porta de origem, com o valor de 0_0 para Portas Baixas, 0_1 para Portas de Servidor e 0_2 para Portas Dinâmicas.

1. Porta de destino, com o valor de 1_0 para Portas Baixas, 1_1 para Portas de Servidor e 1_2 para Portas Dinâmicas.

2. Protocolo, com o valor 2_0 para TCP e 2_1 para os demais.

3. Flags, com o valor 3_0 para quando não há flags, 3_1 quando somente apresenta somente flags ACK, Reset e SYN ou somente flags ACK e Reset, 3_2 quando apresenta somente flag de Reset ou somente de Reset e FIN e 3_3 caso contrário.

4. Número de pacotes, com o valor 4_0 para 1 pacote, 4_1 para caso o número de pacotes esteja no intervalo $(1, 10]$, 4_2 para o intervalo $(10, 100]$ e 4_3 para os demais casos.

5. Total de bytes, com o valor 5_0 para caso o total de bytes se encontrasse no intervalo $[0, 100]$, 5_1 caso o total estivesse em $(100, 1000]$ e 5_2 para os demais casos.

Desta forma, a base de dados de cada um dos interessados a participar se mostra como conjuntos de exatamente seis elementos representando os fluxos. Como exemplo, o começo de uma dessas bases de dados após a primeira etapa de pré-processamento pode ser vista na Tabela 5.2.

Após isso, é essencial a contagem da frequência de cada conjunto de itens para posteriormente ser realizada a criação de regras de associação. Para isso, dois arquivos são criados, um chamado de *out_key*, com os nomes de cada um dos conjuntos, e um outro chamado *out_val*, com os valores da frequência de um conjunto. Note que os dois arquivos devem seguir a mesma ordem. Por exemplo, se olharmos a quinta linha de *out_val* veremos a frequência do conjunto cujo nome pode ser encontrado na quinta linha de *out_key*. Um exemplo pode ser visto na Figura 6.1. É interessante notar que até mesmo os conjuntos que não aparecem na base são representados com frequência nula. Isso acontece pois não se sabe o que ocorre com as bases de dados

Tabela 5.2: Dados após a primeira etapa de pré-processamento

Porta de origem	**Porta de destino**	**Protocolo**	**Flags**	**Número de pacotes**	**Total de bytes**
0_1	1_0	2_0	3_0	4_0	5_2
0_1	1_1	2_0	3_0	4_0	5_0
0_2	1_0	2_0	3_0	4_1	5_1
0_2	1_0	2_0	3_0	4_0	5_0
0_1	1_0	2_0	3_0	4_1	5_0
0_0	1_2	2_0	3_0	4_0	5_0
0_2	1_0	2_0	3_0	4_0	5_0
0_2	1_0	2_0	3_0	4_1	5_1
0_0	1_1	2_1	3_0	4_0	5_0
0_0	1_1	2_1	3_0	4_1	5_1
0_0	1_0	2_1	3_0	4_0	5_0
0_0	1_2	2_0	3_0	4_0	5_1
0_0	1_1	2_0	3_0	4_0	5_0

dos outros participantes, logo é necessário um valor para ser operado no contexto criptográfico.

Para a criação dos arquivos *out_key* e *out_val* é necessário separar o conjunto de cada transação em seus subconjuntos. Isto é necessário pois desta forma se obtém todos os conjuntos que de fato aparecem na transação. Por este motivo, o processo de construção se torna demorado, apesar de linear no número de transações na base. Por este motivo, esta etapa deve ser realizada com antecedência. Caso estes arquivos já estejam formados, atualiza-los progressivamente com a adição de novas transações para evitar tempo gasto para atualizações futuras.

Por último, pode ser observado que o valor de uma das parcelas da soma da Inequação do Suporte Distribuído, proposta na Definição 3.1, não depende dos outros participantes, só dos valores pertencentes ao participante juntamente a o valor s de suporte mínimo que pode ser combinado pelos participantes de antemão. Assim, é possível gerar mais um arquivo prévio a operação conjunta de geração de Regras de Associação, desta vez com o valor da parcela da soma da Inequação do Suporte Distribuído correspondente a um participante. Note que este arquivo segue a mesma ordem que *out_key* e *out_val*. Este arquivo é chamado de *out_val_pre*. Um exemplo do conteúdo destes arquivos pode ser visto na Tabela 5.3.

Tabela 5.3: Dados após a última etapa de pré-processamento

out_key	**out_val**	**out_val_pre**
0_0	5637603	-2491386800
0_1	49530795	-1102067600
0_2	35934544	-2461692700
1_0	56266064	-428540700
1_1	44501861	-1604961000
1_2	20335017	-4021645400
2_0	92684939	3213346800
2_1	28418003	-3213346800
3_0	121102942	6055147100
3_1	0	-6055147100
3_2	0	-6055147100
3_3	0	-6055147100
4_0	89169499	2861802800
4_1	28701590	-3184988100
4_2	2626907	-5792456400
4_3	604946	-5994652500
5_0	81661783	2111031200
5_1	27295485	-3325598600
5_2	12145674	-4840579700
0_0,1_0	9993996	-5055747500
0_0,1_1	13750508	-4680096300
0_0,1_2	11893099	-4865837200
0_0,2_0	18921563	-4162990800
0_0,2_1	16716040	-4383543100
0_0,3_0	35637603	-2491386800
0_0,3_1	0	-6055147100
0_0,3_2	0	-6055147100
0_0,3_3	0	-6055147100
0_0,4_0	23619276	-3693219500
0_0,4_1	10589293	-4996217800
0_0,4_2	1183221	-5936825000
0_0,4_3	245813	-6030565800

Capítulo 6

Comparação Entre os Métodos

Agora que todos os métodos foram apresentados, é possível compará-los com maior precisão. Neste capítulo, serão explicitadas as diferenças teóricas entre os algoritmos propostos assim como os resultados de uma avaliação prática.

6.1 Comparação Teórica

Primeiramente, é necessário explicitar que todos os métodos convergem para o mesmo resultado. Isto é, nenhum dos métodos produz regras melhores ou mais eficientes que os demais dada uma mesma entrada. Isto ocorre pelo fato de todos os métodos utilizarem o mesmo princípio, com a saída dependente somente dos dados de entrada e dos valores de suporte mínimo e confiança mínima, definidos pelos participantes.

Antes de continuar, é importante distinguir dois tipos de atacantes [5]:

Definição 6.1 (Adversário Honesto-mas-curioso)**.** *Um adversário é dito honesto-mas-curioso, ou semi-honesto, quando tenta descobrir a maior parte de informações possíveis seguindo o protocolo corretamente. Note que esse processo de descobrimento pode envolver participantes diferentes juntando suas informações.*

Definição 6.2 (Adversário Malicioso)**.** *Um adversário malicioso é aquele que pode desviar do protocolo em uma tentativa de violar a segurança. Note que um adversário malicioso tem todos os poderes de um adversário honesto-mas-curioso.*

Todos os métodos apresentados, com inclusão do padrão, apresentam resultados auditáveis. Isto é, mesmo que o participante responsável por enviar o resultado aos demais transmita regras incorretas para os demais, seja de forma proposital ou não, o processo pode ser refeito para atestar que a saída produzida é realmente aquela, com mudanças dos responsáveis por cada etapa para garantir que nenhum cargo utilizado de maneira incorreta por algum dos participantes. Isto é possível pois dada uma entrada, a saída é sempre a mesma, caso o processo seja executado corretamente. Se algum participante abusar de seu papel e enviar regras incoerentes, o processo pode ser refeito com troca de papéis para comparar ambas as saídas. Se estiverem diferentes, uma delas está incorreta e algum participante operou incorretamente.

Neste trabalho, o adversário será dado como bem-sucedido se conseguir alguma informação que não seja as regras de informação conjuntas de todos os participantes ou se conseguir gerar um resultado errado para este processo. Assim, todos os métodos propostos, com exceção do padrão, apresentam o mesmo grau de segurança caso seja executado da maneira correta, isto é, nenhum dos participantes tem, em algum momento, dados dos demais. O primeiro problema surge no caso de dois participantes cooperarem de maneira honesta-mas-curiosa. O *Operador* e o *Contextualizador* podem trabalhar juntos no Método 1 para descobrir os dados criptografados pelos demais participantes. Isso ocorre pois o *Operador* tem acesso ao texto cifrado e o *Contextualizador* tem acesso a chave privada que os decripta. Se o *Contextualizador* enviar a chave privada ao *Operador* ou o *Operador* enviar as cifras ao *Contextualizador*, os textos em claro são facilmente obtidos pela dupla. Este problema não ocorre nos métodos de Limiar pois a chave privada se encontra distribuída, ou seja, não está inteiramente na posse do *Contextualizador*.

É importante lembrar que o texto descriptografado por um possível atacante não se trata dos dados em si de um participante, mas sim uma informação gerada por esses dados para a obtenção do suporte e da confiança. Mas como explicado no Capítulo 4, essa informação pode ser usada para descobrir os dados originais.

Outro problema surge quando o *Operador* e mais $n-2$ se unem de forma honesta-mas-curiosa. Desta forma, pode ocorrer o que foi explicitado no Capítulo 4, isto é,

o *Operador* usar seus valores decriptados dos resultados das operações juntamente com os valores dos demais $n-2$ participantes para assim descobrir o valor do texto cifrado do participante restante. Esse caso ocorre mais evidentemente no Método 2, no qual o *Operador* tem acesso aos resultados em claro. Isso também acontece no Método 1 caso um dos $n-2$ participantes seja o *Contextualizador*, pois nessa situação, retornamos ao caso já citado. O Método padrão também sofre com este problema, pois o *Operador* tem acesso a todos os textos em claro.

Quando se leva em conta um adversário malicioso, há mais preocupações. Como já foram considerados os casos onde os participantes trabalham com o que recebem honestamente, resta considerar meios de participantes obterem o que não é devido pelo protocolo. Ou seja, um adversário agir como um ponto intermediário entre dois participantes, realizando um ataque de tipo *Man In The Middle* [44]. Como os dados já estão criptografados, naturalmente os integrantes não irão recriptografar para o envio de forma segura. Porém, diferente da comunicação tradicional, o destinatário da mensagem não é o detentor da chave privada. Isso pode ocasionar problemas no Método 1, caso o *Contextualizador* seja malicioso e intercepte a mensagem entre um participante e o *Operador*. Como o *Contextualizador* é o detentor da chave privada, este pode facilmente decodificar a mensagem, obtendo assim seu conteúdo original. Isso não ocorre nos outros dois métodos pois a chave privada é distribuída. Se considerarmos o caso onde o adversário malicioso pode, além de interceptar mensagens, trocá-las no meio do caminho.

Além disso, se há um participante malicioso, este pode enviar dados incoerentes com sua base de dados para assim gerar regras que não são realidade para este conjunto distribuído. Nesta situação, o atacante corrompe a sua parte da entrada para a saída se tornar incorreta. Assim, o atacante não descobre informações privadas, mas gera um resultado incorreto. Nota-se que este problema não está exclusivamente nos métodos desenvolvidos, pois também pode ocorrer na versão padrão. Se o participante malicioso for o *Operador*, este pode enviar a entrada de um dos participantes para a etapa de decriptação, assim obtendo o texto em claro deste participante ao invés do texto operado. Note que assim, o participante corrompido, não terá acesso a este texto em claro, pois estará em posse do *Contextualizador*, porém receberá

regras de associação geradas sobre a entrada de um só participante, ou seja, saberá regras de associação que caracterizam somente aquele participante, adquirindo assim informação sigilosa. Além disso, se o *Contextualizador* e o *Operador* forem o mesmo participante, o que pode ocorrer em uma situação honesta-mas-curiosa, o atacante terá acesso ao texto em claro. Isso pode ser evitado caso haja uma comparação entre arquivo recebido pelo responsável pela decriptação (que no caso do Método 2 e do Método 3, são vários responsáveis) e os arquivos originais, assim evitando que um dos arquivos originais seja descriptografado. O que o atacante pode fazer, no Método 1 e no Método 3, é alterar este arquivo inicial multiplicando-o por inteiros aleatórios, assim burlando esta comparação e obtendo Regras de Associação de um só participante. Além disso, o oposto pode ser feito no caso do *Operador* ser o único participante não malicioso. Neste caso, os participantes podem enviar contagens nulas para todos os conjuntos, de forma com que a frequência somada dos conjuntos estudados seja igual à frequência do *Operador*, fazendo que as regras geradas pelo processo sejam exclusivamente referentes aos dados dos *Operador*.

Tabela 6.1: Comparação Teórica dos Métodos

Método	**Padrão**	**1**	**2**	**3**
Auditável	✓	✓	✓	✓
Vulnerável com participantes semi-honestos	✓	X	X	X
Vulnerável com coordenação do *Contextualizador* e do *Operador* semi-honestos para leitura de textos em claro	✓	✓	X	X
Vulnerável com coordenação do *Operador* e mais n-2 participantes para leitura de textos em claro de um participante	✓	Caso o *Contextualizador* coopere	✓	X
Vulnerável a envio incorreto de dados	✓	✓	✓	✓
Vulnerável ao *Contextualizador* malicioso interceptar comunicações para leitura de textos em claro	✓	✓	X	X
Vulnerável ao *Operador* malicioso criar regras sobre um participante específico	✓	✓	✓	✓
Vulnerável a todos os participantes se unindo maliciosamente para criar regras sobre o *Operador*	✓	✓	✓	✓
Vulnerável a envios de chaves incorretas x	✓	✓	✓	✓

Por último, é necessária a análise para o caso de troca maliciosa de chaves, isto é, quando a chave correta utilizada no processo é trocada por outra. No caso de limiar, o último participante responsável pela geração da parte da chave pública que faz a encriptação pode, ao invés de mandar a chave gerada coletivamente por todos, enviar uma chave pública própria, de um par gerado por apenas este membro. Desta forma, pode colaborar com o *Operador* e conseguir o dado de todos os participantes, se assemelhando ao caso honesto mas curioso do Método 1. Alternativamente, considerando um ataque de tipo *Man In The Middle*, o *Operador* pode interceptar o envio da chave pública e trocá-la por uma própria, caso este envio não seja realizado de forma segura. Um resumo desta comparação pode ser visto na Tabela 6.1.

6.2 Comparação Prática

Para a comparação prática, o conjunto de dados foi dividido conforme indicado no Capítulo 5, em instâncias Docker. Todos os testes foram executados em um computador Acer Nitro 5, com 24 GB de RAM, processador Ryzen 7 4800H e Ubuntu 22.04 como sistema operacional. Cada método foi executado 100 vezes e medidas foram tomadas. Foi também acrescentado um atraso para a comunicação entre os participantes, a fim de se aproximar ao ambiente real, onde há um tempo maior de transferência de arquivos. O atraso de comunicação foi oriundo de uma distribuição normal com média 2 segundos e desvio padrão 0.5, ambos em segundos. Os resultados das medições podem ser vistos na Tabela 6.2. O significado de cada métrica é explicado a seguir:

Memória Máxima Média Usada Pelo *Contextualizador* (kB): Se trata da média do uso máximo de memória pelo *Contextualizador* dos 100 testes para o determinado método. Isto é, foi registrado o uso máximo de memória para o *Contextualizador* executar o método para as 100 repetições e foi tomada a média aritmética disso.

Tempo Médio de Comunicação do *Contextualizador* (s): Tempo médio utilizado pelo *Contextualizador* recebendo e enviando arquivos.

Tempo Médio de Processamento do *Contextualizador* (s): Tempo médio utilizado pelo *Contextualizador* para a execução das etapas de processamento. Por exemplo, encriptação, decriptação, etc.

Memória Máxima Média Usada Pelo *Contextualizador* (kB): Se trata da média do uso máximo de memória pelo *Operador* dos 100 testes para o determinado método. Semelhante ao feito para o *Contextualizador*.

Tempo Médio de Comunicação do *Operador* (s): Tempo médio utilizado pelo *Operador* recebendo e enviando arquivos.

Tempo Médio de Processamento do *Operador* (s): Tempo médio utilizado pelo *Operador* para a execução das etapas de processamento. Por exemplo, soma homomórfica e produto homomórfico.

Memória Máxima dos Participantes (kB): Média do uso máximo de memória por todos os quatro participantes nos 100 testes para o determinado método. Isto é, foi registrado o uso máximo de memória para o cada participante em um método para as 100 repetições e foi tomada a média disso.

Tempo Médio de Comunicação dos Participantes (s): Tempo médio utilizado pelos participantes para o etapas de comunicação, ou seja, enviando e recebendo arquivos.

Tempo Médio de Processamento dos Participantes (s): Tempo médio utilizado pelos participantes para o etapas de processamento, como decriptação, soma, etc.

Memória Máxima dos Participantes (kB): É o máximo de memória pelos participantes nos 100 testes para o determinado método.

Tempo Médio de Comunicação dos Participantes (s): Tempo máximo utilizado pelos participantes para o etapas de comunicação, isto é, receber e enviar arquivos.

Tempo Máximo de Processamento dos Participantes (s): Tempo máximo utilizado pelos participantes para o etapas de processamento, como encriptação, soma, etc.

Tabela 6.2: Comparação Prática dos Métodos

Métrica	Padrão	1	2	3
Memória Máxima Média Usada Pelo *Contextualizador* (kB)	32684.6	33907.36	34162.68	42080.12
Tempo Médio de Comunicação do *Contextualizador* (s)	12.31099049	45.11720003	71.2764336	99.28849761
Tempo Médio de Processamento do *Contextualizador* (s)	0.02082859131	0.7058775036	0.9844731435	1.375868527
Memória Máxima Média Usada Pelo Operador (kB)	32684.6	35128.6	34162.68	42080.12
Tempo Médio de Comunicação do Operador (s)	12.31099049	40.62299581	71.2764336	99.28849761
Tempo Médio de Processamento do Operador (s)	0.02082859131	0.5537480113	0.9844731435	1.375868527
Memória Máxima Média dos Participantes (kB)	32817.45	8477.179471	25569.84911	10520.78226
Tempo Médio de Comunicação dos Participantes (s)	10.44626296	43.3150367	69.79884519	97.47033287
Tempo Médio de Processamento dos Participantes (s)	0.005207147828	0.4381100562	0.5304994295	0.948154002
Memória Máxima dos Participantes (kB)	33320	34300	34548	42716
Tempo de Comunicação Máximo dos Participantes (s)	14.74249871	51.0247299	78.27570284	107.335783
Tempo de Processamento Máximo dos Participantes (s)	0.052720822	0.938938914	1.129721441	1.545615101

Primeiro, deve se observar que os valores para o *Operador* e para o *Contextualizador* são os mesmos para os métodos 2 e 3. Isso ocorre pois foi escolhido o mesmo participante para realizar os dois papéis. Como dito anteriormente, isto só é possível nesses dois Métodos. Esta escolha foi tomada por facilitar a implementação, reduzir as etapas de comunicação e permitir estudar um participante com o máximo de processamento possível.

Um fator importante para analisar estes valores se trata do contexto no qual os métodos propostos são utilizados. Regras de Associação procuram descobrir características de um certo conjunto de dados. Espera-se que as características sejam inerente ao estado do conjunto de dados e que este conjunto não altere sua essência muito rapidamente. Por este motivo, não há uma necessidade tão grande de velocidade. Uma loja *online* pode, por exemplo, necessitar que regras sejam desenvolvidas mensalmente, para se adequarem a padrões de consumo. No ambiente de Redes de Computadores ao qual esse trabalho se propõe, a janela de repetição da criação de Regras pode ser muito variada, como de uma semana para procurar maiores padrões de rede ou como 5 minutos para caracterizar a rede em um momento específico. De qualquer forma, no caso mediano não há a necessidade de agilidade

na ordem de dezenas de segundos. Desta forma, é possível perceber que todos os métodos apresentaram um desempenho aceitável, visto que o pior caso de todos demorou menos de 2 minutos.

É importante perceber que a grande maioria do tempo gasto se trata na comunicação, com as partes referentes a operações criptográficas tomando pouco tempo relativo. Como a comunicação é inevitável em um método distribuído, pode se tomar este fator como um gargalo. Desta forma, para a obtenção de um outro método mais eficiente na questão de tempo gasto, seria necessária a redução do número de comunicações realizadas. Se a prioridade é velocidade, pode ser utilizado o Método 1 ou o Método 2 por serem mais velozes que o Método 3.

Há uma diferença do *Contextualizador* e do *Operador* em relação aos outros participantes, mas de pequeno impacto quando comparado ao tempo de comunicação e ao nível de urgência. Também há uma diferença significativa na memória utilizada. Porém, da mesma forma que nos tempos, o máximo de memória utilizada não passa dos 42716 kB, valor relativamente baixo para um processo deste tamanho.

Outro ponto relevante a ser abordado é o tamanho dos arquivos gerados, como os textos cifrados e as chaves. Os textos encriptados em todos os métodos possuíam tamanhos menores que 700kB, enquanto as chaves podiam chegar a até 3.3 MB no caso da chave responsável pelo produto. Em geral, devido novamente a falta de urgência para aplicação desse tipo de computação e por não estar sob restrições de armazenamento, os tamanhos dos arquivos são considerados relativamente pequenos.

Sobre os parâmetros para o método em si, foram utilizados os valores de 50% tanto para o Suporte quanto para a Confiança. A saída pode ser vista na Figura 6.1. Pode-se observar que as regras encontradas são coerentes. Visto por exemplo que um fluxo com um pacote tende a ter menos de 100 bytes e os pacotes de menos de 100 bytes tendem a ter um pacote somente. Porém, o que mostra a corretude em si é o fato da saída de todos os métodos em si convergirem ao mesmo valor encontrado no Método Padrão.

```
3_0 => 2_0
2_0 => 3_0
4_0 => 2_0
2_0 => 4_0
5_0 => 2_0
2_0 => 5_0
4_0 => 3_0
3_0 => 4_0
5_0 => 3_0
3_0 => 5_0
5_0 => 4_0
4_0 => 5_0
3_0,4_0 => 2_0
2_0,4_0 => 3_0
2_0,3_0 => 4_0
4_0 => 2_0,3_0
3_0 => 2_0,4_0
2_0 => 3_0,4_0
3_0,5_0 => 2_0
2_0,5_0 => 3_0
2_0,3_0 => 5_0
5_0 => 2_0,3_0
3_0 => 2_0,5_0
2_0 => 3_0,5_0
4_0,5_0 => 3_0
3_0,5_0 => 4_0
3_0,4_0 => 5_0
5_0 => 3_0,4_0
4_0 => 3_0,5_0
3_0 => 4_0,5_0
```

(a) Saída codificada dos métodos.

```
Sem flags => TCP
TCP => Sem flags
1 pacote => TCP
TCP => 1 pacote
Menos de 100 bytes => TCP
TCP => Menos de 100 bytes
1 pacote => Sem flags
Sem flags => 1 pacote
Menos de 100 bytes => Sem flags
Sem flags => Menos de 100 bytes
Menos de 100 bytes => 1 pacote
1 pacote => Menos de 100 bytes
Sem flags,1 pacote => TCP
TCP,1 pacote => Sem flags
TCP,Sem flags => 1 pacote
1 pacote => TCP,Sem flags
Sem flags => TCP,1 pacote
TCP => Sem flags,1 pacote
Sem flags,Menos de 100 bytes => TCP
TCP,Menos de 100 bytes => Sem flags
TCP,Sem flags => Menos de 100 bytes
Menos de 100 bytes => TCP,Sem flags
Sem flags => TCP,Menos de 100 bytes
TCP => Sem flags,Menos de 100 bytes
1 pacote,Menos de 100 bytes => Sem flags
Sem flags,Menos de 100 bytes => 1 pacote
Sem flags,1 pacote => Menos de 100 bytes
Menos de 100 bytes => Sem flags,1 pacote
1 pacote => Sem flags,Menos de 100 bytes
Sem flags => 1 pacote,Menos de 100 bytes
```

(b) Saída traduzida dos métodos.

Figura 6.1: Saídas dos métodos de acordo com o que foi especificado no Capítulo 5. Ambos estão na forma (*Conjunto => Outro Conjunto*)

Capítulo 7

Conclusão

Este trabalho apresentou métodos de Mineração de Dados por meio da Criptografia Homomórfica para resolver o conflito entre entre as oportunidades geradas pela crescente quantidade de dados sendo criados e pelo avanço da preocupação com a privacidade.

A técnica estudada foi a de criação de Regras de Associação em um ambiente distribuído, seguindo e expandindo a ideia proposta em [6]. Desta forma, foram criados três métodos diferentes para um cenário no qual diferentes participantes detêm diferentes partes de um conjunto de dados distribuído e desejam descobrir padrões neste conjunto sem revelar suas partes.

Os métodos propostos foram implementados e testados, cada um tendo suas vantagens e desvantagens, com a comparação abrangendo questões de segurança e eficiência. Além disso, os métodos foram comparados com uma versão que não tem como premissa a garantia da privacidade.

Para a validação, foram utilizados fluxos reais de roteadores de borda, com o mesmo conjunto de regras sendo gerado em todos os métodos, estando de acordo com a base que não leva em consideração o aspecto de privacidade.

Em resumo, foram criados com sucesso diferentes métodos para resolver o problema de geração de Regras de Associação em um ambiente distribuído, de acordo com o proposto.

7.1 Trabalhos Futuros

Como trabalhos futuros, há muitas possibilidades. O principal melhoramento que pode ser implementado se trata na parte de comparação entre os valores obtidos pelas operações homomórficas e o número 0, no que consistem a Inequação do Suporte Distribuído e a Inequação da Confiança Distribuída. Já existem mecanismos que permitem a comparação de inteiros no domínio cifrado. Tais mecanismos não foram utilizados por não estarem disponíveis na biblioteca OpenFHE, que foi escolhida por sua robustez e sua Criptografia de Limiar. Porém, os criadores no presente momento já investigam métodos para a troca de criptossistema, com um de seus sistemas já tendo comparação numérica.

Outra proposta relacionada à anterior é a mudança do modo de implementação. Há outros criptossistemas e outras bibliotecas que podem ser utilizadas para a implementações. Tais variações podem prover diferenças práticas e teóricas na execução dos métodos. Uma ideia promissora é o uso de alguma forma de Encriptação Parcialmente Homomórfica, já que o número de operações utilizado neste trabalho é limitado e isso diminuiria a complexidade da criptografia utilizada.

Um desafio que pode ser investigado é a questão de tamanho do domínio e *overflows*. No momento de se configurar o criptossistema, é necessária a configuração do tamanho máximo de seus valores para que não ocorra um *overflow*. Isso se mostrou desafiador no caso dos fluxos, pois as contagens são numerosas, da ordem de milhões de ocorrências. Se não houver certo controle prévio, o método pode falhar ou até mesmo apresentar respostas incorretas. A solução para esta questão pode residir no fato que a geração depende apenas dos valores de Suporte e Confiança e funciona de maneira aproximada. Por exemplo, suponha que o Suporte mínimo seja 60/100 e um conjunto atinja 59.8/100. Rigorosamente, este conjunto não seria aprovado, mas semanticamente, este conjunto está perto o suficiente de ser considerado frequente de forma que possa ser considerado como tal. Logo, dependendo da precisão desejada, este conjunto pode ser considerado como frequente o suficiente. Com isso, pode se dividir os valores das inequações propostas por um valor fixo dependendo do caso, por exemplo 10000 para assim diminuir sua ordem de grandeza e diminuir

sua chance de *overflow*. Note que a argumentação sobre aproximação é necessária pois a divisão proposta é entre inteiros com saída inteira, pois o BFV só trabalha com inteiros.

Por fim, é importante apontar que o estudo de Regras de Associação não se limita somente aos conceitos apresentados. Podemos citar, por exemplo, o conceito de Convicção [45] que é utilizado no lugar da confiança, tendo suas próprias vantagens e desvantagens. Neste sentido, se mostra útil a investigação futura de outros conceitos de geração de Regras de Associação para o contexto distribuído alavancados por Criptografia Homomórfica.

Referências Bibliográficas

[1] TAYLOR, P., "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025", 2022.

[2] BRASIL, "Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD)", Disponível em: `https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm`, 2020, Acesso em: 18 jan. 2023.

[3] CLIFTON, C., KANTARCIOGLU, M., VAIDYA, J., "Defining privacy for data mining". In: *National science foundation workshop on next generation data mining*, v. 1, p. 1, Citeseer, 2002.

[4] ZHANG, C., XIE, Y., BAI, H., YU, B., LI, W., GAO, Y., "A survey on federated learning", *Knowledge-Based Systems*, v. 216, pp. 106775, 2021.

[5] EVANS, D., KOLESNIKOV, V., ROSULEK, M., OTHERS, "A pragmatic introduction to secure multi-party computation", *Foundations and Trends® in Privacy and Security*, v. 2, n. 2-3, pp. 70–246, 2018.

[6] KAOSAR, M. G., PAULET, R., YI, X., "Fully homomorphic encryption based two-party association rule mining", *Data & Knowledge Engineering*, v. 76, pp. 1–15, 2012.

[7] MACEDO, E. L. C., *Previsão de Tráfego em Enlaces de Redes Utilizando Séries Temporais*. Dissertação de mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2015.

[8] SILVA, V. L. P. D., *Identificacão de anomalias em fluxos de rede utilizando o método de previsão em séries temporais de Holt-Winters*. Dissertação de mestrado, COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2015.

[9] REDERIO, "RedeRio de Computadores/FAPERJ", Disponível em `https://rederio.br/`, 2023, Acessado em Maio de 2023.

[10] BEACHY, J. A., BLAIR, W. D., *Abstract algebra*. Waveland Press, 2019.

[11] FERGUSON, N., SCHNEIER, B., *Practical cryptography*, v. 141. Wiley New York, 2003.

[12] SCHNEIER, B., *Applied Cryptography: Protocols, Algorthms, and Source Code in C.2nd edition*. john wiley & sons, 1996.

[13] RIVEST, R. L., SHAMIR, A., ADLEMAN, L., "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, v. 21, n. 2, pp. 120–126, 1978.

[14] HENRY, K. J., *The theory and applications of homomorphic cryptography*. M.Sc. dissertation, University of Waterloo, 2008.

[15] COSTA, L. A., *Estudo da Criptografia Completamente Homomórfica Aplicada na Mineração de Dados*. M.Sc. dissertation, Universidade Federal de Pernambuco, 2014.

[16] FAN, J., VERCAUTEREN, F., "Somewhat practical fully homomorphic encryption", *Cryptology ePrint Archive*, , 2012.

[17] BRAKERSKI, Z., "Fully homomorphic encryption without modulus switching from classical GapSVP". In: *Annual Cryptology Conference*, pp. 868–886, Springer, 2012.

[18] HALEVI, S., POLYAKOV, Y., SHOUP, V., "An improved RNS variant of the BFV homomorphic encryption scheme". In: *Topics in Cryptology–CT-RSA 2019: The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4–8, 2019, Proceedings*, pp. 83–105, Springer, 2019.

[19] BAJARD, J.-C., EYNARD, J., HASAN, M. A., ZUCCA, V., "A full RNS variant of FV like somewhat homomorphic encryption schemes". In: *Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, pp. 423–442, Springer, 2017.

[20] SHAMIR, A., "How to share a secret", *Communications of the ACM*, v. 22, n. 11, pp. 612–613, 1979.

[21] ASHAROV, G., JAIN, A., LÓPEZ-ALT, A., TROMER, E., VAIKUNTANATHAN, V., WICHS, D., "Multiparty computation with low communication, computation and interaction via threshold FHE". In: *Advances in Cryptology–EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings 31*, pp. 483–501, Springer, 2012.

[22] FRANKLIN, M., HABER, S., "Joint Encryption and Message-Efficient Secure Computation.", *Journal of Cryptology*, v. 9, n. 4, 1996.

[23] CRAMER, R., DAMGÅRD, I., NIELSEN, J. B., "Multiparty computation from threshold homomorphic encryption". In: *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*, pp. 280–300, Springer, 2001.

[24] CHAKRABARTI, S., ESTER, M., FAYYAD, U., GEHRKE, J., HAN, J., MORISHITA, S., PIATETSKY-SHAPIRO, G., WANG, W., "Data mining curriculum: A proposal (Version 1.0)", *Intensive working group of ACM SIGKDD curriculum committee*, v. 140, pp. 1–10, 2006.

[25] AGRAWAL, R., IMIELIŃSKI, T., SWAMI, A., "Mining association rules between sets of items in large databases". In: *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pp. 207–216, 1993.

[26] GENTRY, C., *A fully homomorphic encryption scheme*. Stanford university, 2009.

[27] FRIKKEN, K., "Privacy-preserving set union". In: *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings 5*, pp. 237–252, Springer, 2007.

[28] YI, X., KAOSAR, M. G., PAULET, R., BERTINO, E., "Single-database private information retrieval from fully homomorphic encryption", *IEEE Transactions on Knowledge and Data Engineering*, v. 25, n. 5, pp. 1125–1134, 2012.

[29] DROZDOWSKI, P., BUCHMANN, N., RATHGEB, C., MARGRAF, M., BUSCH, C., "On the application of homomorphic encryption to face identification". In: *2019 international conference of the biometrics special interest group (biosig)*, pp. 1–5, IEEE, 2019.

[30] FANG, W., ZHOU, C., YANG, B., "Privacy preserving linear regression modeling of distributed databases", *Optimization Letters*, v. 7, pp. 807–818, 2013.

[31] LI, J., HUANG, H., "Faster secure data mining via distributed homomorphic encryption". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2706–2714, 2020.

[32] RIEYAN, S. A., NEWS, M. R. K., RAHMAN, A. M., KHAN, S. A., ZAARIF, S. T. J., ALAM, M. G. R., HASSAN, M. M., IANNI, M., FORTINO, G., "An advanced data fabric architecture leveraging homomorphic encryption and federated learning", *Information Fusion*, v. 102, pp. 102004, 2024.

[33] AGARWAL, R., SRIKANT, R., OTHERS, "Fast algorithms for mining association rules". In: *Proc. of the 20th VLDB Conference*, v. 487, p. 499, 1994.

[34] DOCKER, "Make better, secure software from the start", Disponível em `https://www.docker.com/`, 2023, Acessado em Novembro de 2023.

[35] TOMAR, N., "File Transfer using TCP Socket in Python3", Disponível em `https://idiotdeveloper.com/file-transfer-using-tcp-socket-in-python3/`, 2021, Acessado em Janeiro de 2023.

[36] BADAWI, A. A., BATES, J., BERGAMASCHI, F., *et al.*, "OpenFHE: Open-Source Fully Homomorphic Encryption Library", Cryptology ePrint Archive, Paper 2022/915, 2022, https://eprint.iacr.org/2022/915.

[37] BRAKERSKI, Z., GENTRY, C., VAIKUNTANATHAN, V., "Fully Homomorphic Encryption without Bootstrapping", Cryptology ePrint Archive, Paper 2011/277, 2011, https://eprint.iacr.org/2011/277.

[38] CHEON, J. H., KIM, A., KIM, M., SONG, Y., "Homomorphic encryption for arithmetic of approximate numbers". In: *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pp. 409–437, Springer, 2017.

[39] DUCAS, L., MICCIANCIO, D., "FHEW", Disponível em: https://github.com/lducas/FHEW, 2017, Acesso em: 5 set. 2023.

[40] CHILLOTTI, I., GAMA, N., GEORGIEVA, M., IZABACHENE, M., "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds". In: *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pp. 3–33, Springer, 2016.

[41] DE ASSIS, F. M., COUTINHO, M. A., DA SILVA FILHO, J. B., MACEDO, E. L., DE MORAES, L. F., "IPTraf: Coleta e Detecção de Anomalias em Fluxos de Rede". In: *Anais do XXVI Workshop de Gerência e Operação de Redes e Serviços*, pp. 96–109, SBC, 2021.

[42] RAVEL, "Laboratório de Redes de Alta Velocidade", Disponível em https://www.ravel.ufrj.br/, 2023, Acessado em Dezembro de 2023.

[43] DA SILVA FILHO, J. B., "Detecção de anomalias em fluxos de redes de computadores utilizando técnicas de redes neurais e estimadores lineares", 2015.

[44] CONTI, M., DRAGONI, N., LESYK, V., "A survey of man in the middle attacks", *IEEE Communications Surveys & Tutorials*, v. 18, n. 3, pp. 2027–2051, 2016.

[45] BRIN, S., MOTWANI, R., ULLMAN, J. D., TSUR, S., "Dynamic itemset counting and implication rules for market basket data". In: *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pp. 255–264, 1997.

Appendices

Apêndice A

Complementos Matemáticos

A.1 Operadores Booleanos

Definição A.1 (Ou Exclusivo)**.** *Dados dois valores booleanos a e b (i.e. $a, b \in 0, 1$, a operação $\oplus$, conhecida como Ou Exclusivo (XOR, do inglês Exclusive Or) é definida como*

$$a \oplus b = 1 \quad se \quad a \neq b$$

$$a \oplus b = 0 \quad se \quad a = b$$

Teorema A.1. *O Ou Exlcusivo é associativo e comutativo.*

Demonstração. A prova consiste em testar todos os casos. Será omitida por ser simples porém extensa. □

Teorema A.2. *Dados os booleanos a e b, temos que $a \oplus b \oplus b = a$*

Demonstração. Temos $a \oplus b \oplus b = a \oplus 0$. Para as duas possibilidades de a:

$$0 \oplus 0 = 0$$

$$1 \oplus 0 = 1$$

□

A.2 Conjuntos

Definição A.2 (Naturais)**.** *O conjunto dos naturais é* $\mathbb{N} = 0, 1, 2, \ldots$ *. Definições mais precisas podem ser encontradas em livros de análise.*

Definição A.3 (Inteiros)**.** *O conjunto dos inteiros é formado por* $\mathbb{N}$ *e seus opostos, ou seja, por todo* $n \in \mathbb{N}$ *e seu* $-n$ *equivalente.*

Definição A.4 (União)**.** *Dados dois conjuntos* A *e* B*, sua união é dada por*

$$A \cup B = \{x \mid x \in A \text{ ou } x \in B\}$$

Definição A.5 (Subconjunto)**.** *Um conjunto* A *é dito subconjunto de outro conjunto* B *se todo elemento pertencente a* A *pertence a* B*.*

Definição A.6 (Produto Cartesiano)**.** *Dados dois conjuntos* A *e* B*, o produto cartesiano de* A *e* B *, denotado por* $A \times B$*, se trata do conjunto cujos elementos são da forma* (a, b) *com* $a \in A$ *e* $b \in B$*, chamados de tuplas.* A^n *denota o produto cartesiano de* A *consigo mesmo* n *vezes. Exemplo:* $A^3 = A \times A \times A$*.*

A.3 Números Primos

Definição A.7 (Divisibilidade)**.** *É dito que* a *divide* b*, ou* $a \mid b$*, se* $b = ac$ *para algum* c *inteiro.*

Definição A.8 (Máximo Divisor Comum)**.** *O máximo divisor comum de dois números* a*,* b*, denotado por mdc(a,b), é um número* d*, tal que* $d \mid a$ *e* $d \mid b$ *e que, dado qualquer número* c *tal que* $c \mid a$ *e* $c \mid b$*, temos que* $c \mid d$*. Se mdc(a,b) = 1, dizemos que* a *e* b *são primos entre si.*

Definição A.9 (Número Primo)**.** *Um número* p *é dito primo se tem exatos dois divisores distintos. Mais precisamente, um número* p *é primo se, para todo* $a \in \{2, 3, ..., p-2, p-1\}$ *temos que* $p \neq ab$ *para* b *inteiro.*

Teorema A.3. *Dados dois primos* p *e* q*, temos que existem inteiros* x *e* y *tal que* $px + qy = 1$*.*

Demonstração. Pode ser encontrada em [10] □

Teorema A.4. *Dados dois primos p e q, se $p \mid a$ e $q \mid a$, então $pq \mid a$*

Demonstração. Primeiramente, note que se, para algum c, tivermos $p \mid qc$, então $p \mid c$. Isso ocorre, pois pelo Teorema A.3, temos:

$$px + qy = 1$$
$$pxc + qyc = c$$

Logo c é a soma de pxc, divisível por p, e $y(qc)$, que por hipótese é divisível por p, logo $p \mid c$. Agora, se $p \mid a$, $a = pk$ para algum inteiro k. Da mesma forma, $q \mid pk$. Pela parte anterior, $q \mid k$, logo $k = ql$. Assim temps $a = pk = pql$, logo $pq \mid a$. □

Definição A.10 (Congruência módulo n)**.** *Dados dois inteiros a e b, dizemos que são congruentes módulo n, denotado $a = b \mod n$, se $n \mid (a - b)$*

Definição A.11 ($\mathbb{Z}_n$)**.** *Seja n um inteiro e $\mathbb{Z}$ o conjunto dos inteiros. $\mathbb{Z}_n$ representa os inteiros módulo n, ou seja, $0, 1, 2, \ldots, n - 1$*

Teorema A.5 (Fermat)**.** *Seja p primo e a um inteiro com $mdc(a, p) = 1$. Temos*

$$a^{p-1} = 1 \mod p$$

Demonstração. Pode ser encontrada em [10]. □

Definição A.12 (Função Totiente de Euler)**.** *Dado um número n com decomposição prima $n = p_1^{k_1} \ldots p_m^{k_m}$. Definimos a função totiente, denotada por φ, como $\varphi(n) = (p_1 - 1)p_1^{k_1 - 1} \ldots (p_m - 1)p_m^{k_m - 1}$.*

A.4 Álgebra

Definição A.13 (Anel)**.** *Seja R um conjunto com duas operações binárias definidas, que serão chamadas de adição e multiplicação (denotadas por $+$ e $\cdot$ respectivamente). R é um anel com respeito a essas operações, se*

(i) *R é um grupo abeliano em relação a adição. Isto é, dados a,b $\in R$, temos $a + b = b + a$*

(ii) *Multiplicação é associativa.*

(iii) *R tem um elemento neutro da multiplicação.*

(iv) *Segue a propriedade associativa, isto é, dados a,b,c* $\in R$*, segue que*

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$(a + b) \cdot c = a \cdot c + b \cdot c$$

Apêndice B

Exemplos Numéricos

Exemplo B.1 (Criptografia com XOR)**.** *Seja* 1000001 *o caractere "A" em ASCII. Desejamos criptografar e descriptografar esse caractere usando XOR. Seja* 0111001 *a chave escolhida (note que ambos tem que ter o mesmo tamanho). Temos que* $1000001 \oplus 0111001 = 1111000$ *que é o valor criptografado gerado. Note que isso é igual a "x" em ASCII. Para recuperar o valor original, fazemos* $1111000 \oplus 111001 = 1000001$*, o texto em claro original.*

Exemplo B.2 (Criptografia RSA)**.** *Seja* $p = 11$ *e* $q = 17$ *como no exemplo 2.2. Assim* $n = pq = 187$ *e* $\varphi(n) = 160$*. Seja* $e = 7$ *e* $d = 23$*. Para uma mensagem* $m = 2$*, temos:* $E_K(2) = 2^7 \mod 187 = 128 \mod 187$*. Por outro lado,* $D_K(128) = 128^2 3 \mod 187 = 2$*, a mensagem original, como esperado.*

Apêndice C

Resumo dos Métodos

Resumo dos métodos propostos na Seção 4. Neste apêndice "P". significa Passo.

C.1 Método Padrão

- **P. 1**: Todos os participantes enviam seus dados para o *Operador*. O *Operador* então calcula as regras de associação de acordo com o descrito na Seção 2.3.
- **P. 2**: O *Operador* envia os resultados para os demais participantes.

C.2 Método 1

- **P. 1**: O *Contextualizador* cria o contexto criptográfico e um par de chaves. O contexto e a chave pública são enviadas aos demais participantes.
- **P. 2**: Os participantes encriptam seus arquivos *out_val_pre* e enviam ao *Operador*
- **P. 3**: O *Operador* soma os valores para cada conjunto e multiplica cada valor resultante por um inteiro positivo aleatório. O *Operador* envia o resultado do produto para o *Contextualizador*.
- **P. 4**: O *Contextualizador* decripta os valores recebidos e observa os valores positivos. Então envia os índices dos conjuntos cujos valores recebidos são positivos para os demais participantes.

- **P. 5**: Os participantes desmembram os conjuntos referentes aos índices recebidos em dois subconjuntos complementares para formar Regras de Associação. Após isso, calculam seus valores referentes a sua parcela da Inequação da Confiança Distribuída, os encriptam e os enviam ao *Operador*.

- **P. 6**: O *Operador* soma os valores para cada regra e multiplica cada valor resultante por um inteiro positivo aleatório. O *Operador* envia o resultado do produto para o *Contextualizador*.

- **P. 7**: O *Contextualizador* decripta os valores recebidos e observa os valores positivos. Então envia os índices das regras cujos valores recebidos são positivos para os demais participantes.

C.3 Método 2

- **P. 1**: O *Contextualizador* cria o contexto criptográfico e um par de chaves. O contexto é enviado a todos os outros $n-1$ participantes e a chave pública é enviada a um segundo participante.

- **P. 2 a n-1**: O participante que recebeu a chave pública do participante anterior gera seu próprio par de chaves de acordo com a chave recebida. Envia então a nova chave pública gerada ao próximo participante.

- **P. n**: O último participante recebe a chave pública do anterior e gera seu próprio par de chaves. Ele então envia a chave pública final aos demais participantes.

- **P. n+1**: Os participantes encriptam seus arquivos *out_val_pre* e enviam ao *Operador*

- **P. n+2**: O *Operador* soma os valores para cada conjunto. O *Operador* envia o resultado da soma para os demais participantes.

- **P. n+3**: Cada participante decripta parcialmente o arquivo recebido e envia ao *Operador*.

- **P. n+4**: O *Operador* decripta os valores recebidos e observa os valores positivos. Então envia os índices dos conjuntos cujos valores recebidos são positivos para os demais participantes.

- **P. n+5**: Os participantes desmembram os conjuntos referentes aos índices recebidos em dois subconjuntos complementares para formar Regras de Associação. Após isso, calculam seus valores referentes a sua parcela da Inequação da Confiança Distribuída, os encriptam e os enviam ao *Operador*.

- **P. n+6**: O *Operador* soma os valores para cada regra. O *Operador* envia o resultado da soma para os demais participantes.

- **P. n+7**: Cada participante decripta parcialmente o arquivo recebido e envia ao *Operador*.

- **P. n+8**: O *Operador* decripta os valores recebidos e observa os valores positivos. Então envia os índices das regras cujos valores recebidos são positivos para os demais participantes.

C.4 Método 3

- **P. 1** O *Contextualizador* cria o contexto criptográfico e um par de chaves. O contexto é enviado a todos os outros $n-1$ participantes e a chave pública é enviada a um segundo participante.

- **P. 2 a n-1** O participante que recebeu a chave pública do participante anterior gera seu próprio par de chaves de acordo com a chave recebida. Envia então a nova chave pública gerada ao próximo participante. Note que essa chave pública consiste em uma chave de encriptação e uma de produto.

- **P. n** O último participante gera seu próprio par de chave privada e de encriptação e começa a segunda etapa de geração da chave do produto. Ele então envia a chave de encriptação finalizada assim como a chave do produto incompleta para outro participante.

- **P. n+1** a 2n-2 O participante opera sobre a chave de produto incompleta no que consiste a sua segunda etapa de geração. A chave incompleta é enviada a outro participante que ainda não participou da segunda etapa.

- **P. 2n-1** O último participante termina a chave do produto e a envia ao *Operador*.

- **P. 2n**: Os participantes encriptam seus arquivos *out_val_pre* e enviam ao *Operador*

- **P. 2n+1** O *Operador* soma os valores para cada conjunto e multiplica cada valor resultante por um inteiro positivo aleatório. O *Operador* envia o resultado da soma para os demais participantes.

- **P. 2n+2** Cada participante decripta parcialmente o arquivo recebido e envia ao *Operador*.

- **P. 2n+3** O *Operador* decripta os valores recebidos e observa os valores positivos. Então envia os índices dos conjuntos cujos valores recebidos são positivos para os demais participantes.

- **P. 2n+4** Os participantes desmembram os conjuntos referentes aos índices recebidos em dois subconjuntos complementares para formar Regras de Associação. Após isso, calculam seus valores referentes a sua parcela da Inequação da Confiança Distribuída, os encriptam e os enviam ao *Operador*.

- **P. 2n+5** O *Operador* soma os valores para cada regra e multiplica cada valor resultante por um inteiro positivo aleatório. O *Operador* envia o resultado da soma para os demais participantes.

- **P. 2n+6** Cada participante decripta parcialmente o arquivo recebido e envia ao *Operador*.

- **P. 2n+7** O *Operador* decripta os valores recebidos e observa os valores positivos. Então envia os índices das regras cujos valores recebidos são positivos para os demais participantes.