

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



**CONSULTA A BASE DE DADOS CIFRADA EM COMPUTAÇÃO NAS
NUVENS**

HUMBERTO GUSTAVO DE MELO

Uberlândia - Minas Gerais

2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



HUMBERTO GUSTAVO DE MELO

CONSULTA A BASE DE DADOS CIFRADA EM COMPUTAÇÃO NAS NUVENS

Dissertação de Mestrado apresentada à Faculdade de Ciência da Computação da Universidade Federal de Uberlândia, Minas Gerais, como parte dos requisitos exigidos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados.

Orientador:

Prof. Dr. João Nunes de Souza

Uberlândia, Minas Gerais

2011

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Ciência da Computação a aceitação da dissertação intitulada “**Consulta a base de dados cifrada em computação nas nuvens**” por **Humberto Gustavo de Melo** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, 8 de Agosto de 2011

Orientador:

Prof. Dr. João Nunes de Souza
Universidade Federal de Uberlândia

Banca Examinadora:

Prof. Dr. Daniel Gomes Mesquita
Universidade Federal de Uberlândia

Prof. Dr. Pedro Frosi Rosa
Universidade Federal de Uberlândia

Prof. Dr. Edward David Moreno Ordóñez
Universidade Federal de Sergipe

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Data: Agosto de 2011

Autor: **Humberto Gustavo de Melo**
Título: **Consulta a base de dados cifrada em computação nas nuvens**
Faculdade: **Faculdade de Ciência da Computação**
Grau: **Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

Autor

O AUTOR RESERVA PARA SI QUALQUER OUTRO DIREITO DE PUBLICAÇÃO DESTE DOCUMENTO, NÃO PODENDO O MESMO SER IMPRESSO OU REPRODUZIDO, SEJA NA TOTALIDADE OU EM PARTES, SEM A PERMISSÃO ESCRITA DO AUTOR.

Dedicatória

Dedico este trabalho aos meus pais que investiram em minha profissionalização, aos meus irmãos Marcelo e Herick pelo apoio. Dedico em especial a minha esposa Anastácia que sempre esteve junto de mim e à quem atribuo as minhas vitórias.

Agradecimentos

Agradeço a Deus pela oportunidade de ingressar no mestrado, aos meus pais que investiram na minha profissionalização e capacitação. Aos meus irmãos Marcelo e Herick pelo apoio nos momentos difíceis. Aos professores João Nunes e Daniel Mesquita pelos ensinamentos em sala de aula que contribuíram diretamente para elaboração desse trabalho. Ao meu amigo Winicius companheiro de sala e viagens que contribuiu com ensinamentos, incentivos, auxílios e apoio nos momentos difíceis. A minha esposa Anastácia que sempre esteve junto de mim.

Resumo

Com a disseminação e a popularização da internet, cada vez mais pessoas e empresas estão utilizando os serviços em computação nas nuvens. Entre esses serviços, existe o armazenamento e hospedagem das bases de dados nas nuvens. Nesse contexto surge um problema, como assegurar a privacidade e integridade das bases de dados que estão expostas a ataques. O objetivo desta dissertação é propor um protocolo de criptografia para realizar consultas de forma segura nas bases de dados cifradas que estão armazenadas nesses servidores. A principal contribuição deste trabalho é permitir a verificação dos dados cifrados e armazenados nos bancos de dados dos servidores sem a necessidade de decifrá-los.

Diante desse problema, esta dissertação apresenta inicialmente as principais características do ambiente de computação nas nuvens. Em seguida são apresentados os conceitos de criptografia e assinatura digital necessários para a construção do protocolo proposto. Por fim, foi desenvolvido o modelo formal do protocolo e o seu protótipo.

Para validar o protocolo foram realizados cinco experimentos. Cada experimento é composto por uma base de dados cifrada e uma base de dados não cifrada. Nesse caso foram avaliadas as seguintes métricas: tempo de execução de uma consulta, espaço em disco ocupado pela base de dados e o tamanho da chave utilizada pelo protocolo. Os resultados obtidos demonstram que o desempenho das consultas nas bases de dados cifradas é muito inferior que as bases convencionais. Observou-se também que o espaço em disco necessário para armazenar os dados cifrados é muito maior que os dados não cifrados. Esses resultados estão diretamente relacionados com o tamanho da chave. Nos experimentos, o tamanho da chave utilizada foi de 1024 bits. Esse tamanho é o mínimo recomendado para que o protocolo de criptografia proposto seja seguro.

Diante das desvantagens apresentadas, conclui-se que em um ambiente real, por exemplo, onde existem centenas ou até milhares de tabelas, o protocolo de criptografia proposto deve ser utilizado em tabelas cuja segurança é crítica, como por exemplo, tabelas que armazenam: senhas, salários de funcionários, notas de alunos, entre outras.

Palavras chave: base de dados, criptografia, segurança, computação nas nuvens.

Abstract

With the dissemination and popularization of the Internet, more and more people and businesses are using cloud computing services. Among these services, there is a storage and hosting of databases in the clouds. In this context a problem arises, how to ensure the privacy and integrity of databases that are exposed to attacks. The goal of this dissertation is to propose an encryption protocol for consultations securely in encrypted databases that are stored on those servers. The main contribution of this work is to allow verification of the data encrypted and stored in databases of servers without having to decipher them.

Given this problem, this paper first presents the main features of cloud computing environment. The following are the concepts of encryption and digital signatures required for the construction of the proposed protocol. Finally, we developed the formal model of the protocol and its prototype.

To validate the protocol were carried out five experiments. Each experiment consists of an encrypted database and a database is not encrypted. In this case we evaluate the following metrics: execution time of a query, disk space occupied by the database and the key size used by the protocol. The results show that the performance of queries on encrypted databases is much less than the conventional bases. It was also noted that the disk space required to store the encrypted data is much larger than the unencrypted data. These results are directly related to the key size. In the experiments, the key size used was 1024 bits. This is the minimum recommended size for the encryption protocol proposed is secure.

Given the disadvantages presented, we conclude that in a real environment, for example, where there are hundreds or even thousands of tables, the proposed encryption protocol should be used on tables whose security is critical, for example, tables that store: passwords , employee salaries, student grades, among others.

Keywords: database, encryption, security, cloud computing.

Sumário

Lista de Figuras	x
Lista de Tabelas	xii
Lista de Abreviaturas e Siglas	xiii
1 Introdução	14
1.1 Contexto e Definição do Problema	15
1.2 Justificativa	16
1.3 Hipóteses	17
1.4 Organização do texto	17
2 Material e Metodologia	19
2.1 Considerações Finais	21
3 Fundamentação Teórica	22
3.1 Estado da Arte	22
3.2 Computação nas Nuvens	26
3.2.1 Modelos de Serviços	29
3.2.2 Segurança em Computação nas Nuvens	31
3.3 Fundamentos Matemáticos	32
3.3.1 Grupo Multiplicativo	32
3.3.2 Elemento Gerador	33
3.3.3 Logaritmo Discreto	34
3.3.4 Inverso Multiplicativo	35
3.4 Criptografia	37
3.4.1 Criptosistema El-Gamal	38
3.4.2 Assinatura digital	40
3.5 Considerações Finais	46
4 Contribuições, Experimentos e Resultados	47
4.1 Novo Protocolo Proposto	47

4.2	Modelo Relacional com Dados Cifrados	49
4.2.1	Operadores de Consulta	51
4.3	Considerações Finais	73
5	Resultados e Análises	74
5.1	Segurança do Protocolo	74
5.2	Descrição do Experimento	81
5.2.1	Arquitetura do protótipo	81
5.2.2	Implementação do protótipo	82
5.2.3	Esquema de Validação	87
5.3	Considerações Finais	90
6	Conclusão e Trabalhos Futuros	92
6.1	Análise crítica dos resultados	92
6.2	Trabalhos futuros	93
6.3	Considerações finais	93
	Referências Bibliográficas	95
A	Código Fonte	98
A.1	Classe Cripto_Servidor	98
A.2	Classe Cripto_Cliente	101
B	Representação Base64	104

Lista de Figuras

1.1	Ambiente de comunicação entre cliente e servidor [Anciaux et al. 2006]	15
3.1	Visão de uma nuvem computacional, adaptado de [Sousa et al. 2009].	26
3.2	Topologia da computação nas nuvens [Tujal 2010].	28
3.3	Arquitetura de serviços em computação nas nuvens de três camadas.	30
3.4	Resultado de $2^i \bmod 11$ e $3^i \bmod 11$	33
3.5	comunicação Alice e Bob em um criptosistema [Stinson 2005].	38
4.1	Relação <i>funcionario</i>	49
4.2	representação numérica das colunas alfanuméricas utilizadas no cálculo da cifragem da relação <i>funcionario</i>	50
4.3	Resultado da cifragem das colunas na relação <i>funcionario</i>	51
4.4	Relação $\text{funcionario}^C = (41415, 44586)$ no modelo relacional cifrado.	52
4.5	Resultado da projeção na relação funcionario^C	53
4.6	Protocolo de seleção com condição $A_j > q$	59
4.7	Cálculo de λ_{i3} para condição salário $> 800,00$	60
4.8	Resultado final consulta $\text{salario} > 800,00$ na relação <i>funcionario</i>	61
4.9	Protocolo de Inserção em R^C	62
4.10	Resultado da cifragem das colunas na relação <i>funcionario</i>	63
4.11	Relação funcionario^C após a inserção do novo funcionário	63
4.12	Protocolo de remoção com condição $A_j > q$	65
4.13	Cálculo de λ_{i3} para condição salário $< 600,00$	66
4.14	Relação (41415, 44586) resultante após a operação de remoção.	67
4.15	Protocolo operação de atualização para uma condição $A_j < q$	70
4.16	Cálculo de λ_{i3} para condição salário < 700	71
4.17	Resultado atualização dos salários nas tuplas 6 e 9.	72
4.18	Relação $\text{funcionario}^{C''}$ contendo novos salários.	72
4.19	Relação (41415, 44586) resultante após operação de atualização.	72
5.1	Cálculo de λ_{i3} para condição salário < 1000	77
5.2	Arquitetura protótipo.	81
5.3	Exemplo de criação do tipo de dados Cripto no <i>SQL Server 2008</i>	83

5.4	Exemplo de relação cifrada utilizando representação <i>Base64</i>	84
5.5	Tela do protótipo onde o cliente cria uma nova relação.	84
5.6	Inserção da relação <i>funcionarios^C</i> e seus atributos na relação TbProtocolo . . .	85
5.7	Interface da aplicação cliente para consultas SQL.	86
5.8	Resultado da consulta na interface do “módulo consulta no cliente”.	87
5.9	Relação <i>funcionarios</i>	88
5.10	Comparação entre tempo (ms) das consultas cifradas x não cifradas.	89
5.11	Comparação entre os tamanhos em (MB) das relações cifradas e não cifradas. .	90
B.1	Conversão de <i>byte</i> para <i>Base64</i> [Knudsen 1998]	104

Lista de Tabelas

3.1	Lista de algoritmos para cálculo do logaritmo discreto e suas respectivas complexidades [Shoup 2008].	35
5.1	Parâmetros para testes do protocolo	83
5.2	Relação nível de segurança com o tamanho da chave, tabela adaptada de [Ecrypt 2010].	90

Lista de Abreviaturas e Siglas

ASCII	American Standard Code For Information Interchange
AES	<i>Advanced Encryption Standard</i>
API	<i>Application Programming Interface</i>
DES	<i>Data Encryption Standard</i>
ECRYPT	<i>European Network of Excellence in Cryptology</i>
FPGA	Field Programmable Gate Array
MDC	Máximo Divisor Comum
RSA	Ron Shamir Adleman
SGBD	Sistema Gerenciador de Banco de dados
SQL	<i>Structured Query Language</i>
TI	Tecnologia da Informação

Capítulo 1

Introdução

Tradicionalmente a infraestrutura de tecnologia da informação nas empresas é caracterizada pelo uso de recursos e serviços proprietários, ou seja, as empresas adquirem e utilizam recursos próprios de infraestrutura, hardware, funcionários, entre outros. Porém esse contexto tem como principais desvantagens o alto custo de aquisição e manutenção de novas tecnologias, licenças e atualizações de softwares. Além disso, com o surgimento de novas tecnologias, o hardware adquirido torna-se obsoleto ao longo do tempo para clientes e empresas, além de depreciar-se rapidamente [Breitman e Viterbo 2010]

Com a crescente evolução das tecnologias de armazenamento e comunicação, muitas empresas e clientes têm procurado cada vez mais na computação nas nuvens¹ uma solução eficaz e barata para seus negócios. Nesse ambiente os softwares de aplicação e as bases de dados são armazenadas em *data centers*, onde existe uma estrutura física adequada para armazenamento e segurança física do local. Esse tipo de serviço traz muitas vantagens com relação à portabilidade, além de um custo menor de aquisição e manutenção de infraestrutura de tecnologia da informação [Wang et al. 2009].

Com o avanço e a popularização da computação nas nuvens, o custo desses serviços tendem a cair. Porém, esse ambiente apresenta desafios de segurança, tais como: roubo de informações sigilosas e garantia da integridade das informações nele armazenados [Wang et al. 2009].

Além do desafio de desenvolver sistemas seguros, o ambiente de computação nas nuvens apresenta riscos de segurança no qual esse ambiente está exposto. Logo, um problema de segurança nesses ambientes ocasiona grandes prejuízos financeiros, vazamento de informações, indisponibilidade de informação, entre outros [Novais et al. 2010].

Atualmente, a fornecedora de computação nas nuvens oferece alguns serviços de segurança tais como: *backup* de dados, políticas de segurança, antivírus, patch de segurança, controle de contingência, entre outros. Porém, esses serviços não são suficientes para evitar riscos de segurança. No capítulo 3 esses riscos são apresentados e detalhados. Além disso, o cliente é

¹É o modelo de computação em que os dados e aplicações ficam armazenados em servidores remotos localizados em diversas partes do mundo. Esses servidores são acessados através da internet [Sousa et al. 2009]. Um conceito mais detalhado é apresentado no capítulo 3.

responsável pela proteção dos seus dados críticos, mesmo que estes tenham sido armazenados na nuvem [Novais et al. 2010].

1.1 Contexto e Definição do Problema

A figura 1.1 mostra um ambiente de comunicação entre o cliente e o servidor de banco de dados, que é considerado como base para o protocolo proposto. Existem três entidades nesse contexto:

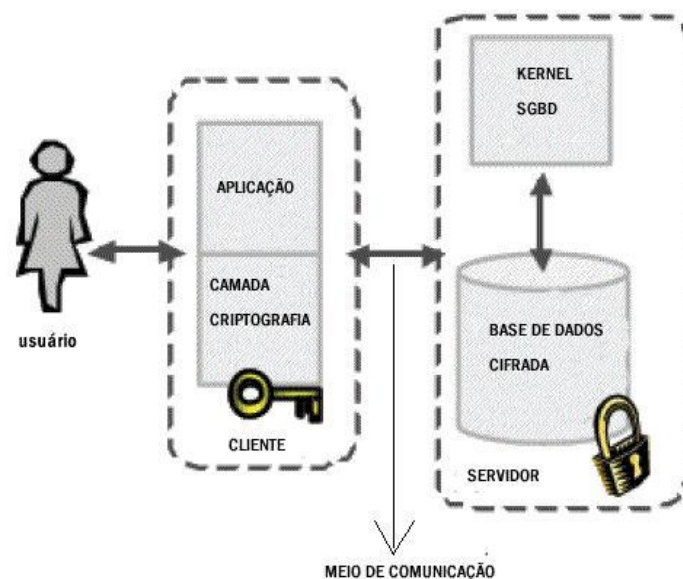


Figura 1.1: Ambiente de comunicação entre cliente e servidor [Anciaux et al. 2006]

A primeira é o cliente, cujo papel é inserir, atualizar, remover ou selecionar informações na base de dados. Os clientes podem ser usuários individuais ou empresas. Porém os usuários que utilizam esse ambiente, certamente, não devem ser leigos. Ou seja, são usuários que conhecem fundamentos de linguagem de consulta a base de dados e de criptografia. Além disso, a chave para cifrar as informações fica em segredo no cliente. E, também, a máquina do cliente é considerada segura contra invasões externas.

A segunda entidade é o servidor, responsável pelo armazenamento dos dados do cliente, exceto sua chave criptográfica, e processamento das consultas requisitadas por ele. Os servidores estão localizados em qualquer parte do mundo e são administrados por pessoas que os clientes não conhecem. Diante disso, os seus dados estão expostos a ataques por parte dos administradores desses servidores e dos administradores de banco de dados. Isso significa que a máquina do servidor não é segura contra invasões.

Finalmente, o meio entre o cliente e o servidor, por onde trafegam os dados também não é seguro. A partir desse contexto, existe então uma necessidade de garantir a segurança e a

privacidade das informações dos clientes. Ou seja, os clientes devem ter certeza que suas informações estão privadas, seguras e íntegras contra possíveis ataques por parte dos administradores dos servidores, de invasores no meio onde as informações trafegam e, enfim, de qualquer outra pessoa ou entidade.

Assim, para a segurança dos clientes, tais fatos exigem, por exemplo, que toda consulta a base de dados seja cifrada por eles antes de ser enviada ao servidor. Além disso, seus dados devem ser cifrados antes de armazená-los na base de dados do servidor.

Esta dissertação soluciona a cifragem de dados para comunicação da maneira usual. Isto é, antes da consulta ser enviada ao servidor, ela pode ser cifrada por algoritmos criptográficos. Mas, além de cifrar dados antes de sua transmissão ao servidor, é também necessário que a consulta à base de dados seja efetuada em um ambiente totalmente cifrado. Ou seja, que a consulta e os dados armazenados não sejam decifrados para que tal consulta seja executada na base de dados.

O principal objetivo do protocolo proposto neste trabalho é garantir a segurança e a privacidade das informações armazenadas nas nuvens contra possíveis ataques de pessoas não credenciadas através da manipulação de dados cifrados no servidor sem a necessidade de decifrá-los.

Além disso, como objetivos específicos destacam-se:

- Desenvolver um mecanismo seguro para acessar bases de dados cifradas;
- Propor um protocolo que realize consultas em base de dados cifradas com os operadores relacionais sem a necessidade de decifrar os dados no servidor;
- Construir protocolos de criptografia que realizem as operações de inserção, atualização e remoção dos dados armazenados em base de dados cifradas;
- Definir uma **álgebra relacional para dados cifrados**;

1.2 Justificativa

Atualmente, os principais problemas para o uso de base de dados em nuvem são: privacidade, integridade e confidencialidade dos dados. Nesse caso, somente é possível armazenar dados nesses serviços se os dados não são sigilosos ou o provedor de serviço é altamente confiável.

Diante deste contexto, surge a necessidade de desenvolver protocolos de criptografia que permita ao cliente ter certeza que seus dados estejam seguros contra pessoas não credenciadas.

Além disso, a construção de um protocolo seguro que manipule base de dados cifradas em computação em nuvem traz benefícios aos seus clientes, uma vez que eles não terão que desenvolver controles internos e nem investir em especialistas, estruturas físicas e softwares de segurança.

Esta dissertação propõe um protocolo para consulta cifrada em bases de dados também cifradas. Isto é, o servidor realiza as consultas sem a necessidade de decifrar os dados nele armazenados. E, então, as informações requisitadas pelo cliente são obtidas de forma segura.

Conforme foi pesquisado, não foi encontrado na literatura trabalho que solucione esse tipo de consulta como é definido no protocolo.

1.3 Hipóteses

Considerando o contexto explicitado na seção 1.1, levantou-se a hipótese de que através da combinação de algoritmos de chave assimétrica e manipulações matemáticas é possível resolver esse problema. Tal ponderação se deve ao aparato matemático que esses algoritmos possuem, diferentemente dos algoritmos de chave simétrica que são baseados em operações lógicas, permutações e transposições. Dentre os algoritmos assimétricos existentes, decidiu-se utilizar o criptosistema e o esquema de assinatura El-Gamal. Essa escolha deve-se principalmente ao elemento gerador presente nesses criptosistemas. Diferentemente do RSA que não possui esse elemento, por isso esse algoritmo não foi utilizado. A partir do elemento gerador e de suas propriedades foi possível construir um mecanismo de comparação entre dados cifrados. A prova e a demonstração dessas escolhas são apresentadas nos capítulos 3 e 4.

A partir da hipótese anterior, foi levantada uma segunda hipótese em relação ao desempenho do protocolo. Conforme apresentado anteriormente, o protocolo proposto é baseado em algoritmos de criptografia de chave pública, tal escolha implica em um protocolo com um custo computacional maior se comparado aos algoritmos de chave simétrica, logo, esse protocolo pode se tornar lento para grandes bases de dados cifradas. Nesse caso, espera-se que os resultados sejam satisfatórios para base de dados com poucos registros, ou seja, algumas dezenas de milhares de registros. Entretanto, para grandes bases de dados, ou seja, centenas de milhares ou até milhões de registros, espera-se que o desempenho seja bem inferior. Porém, essa desvantagem pode ser minimizada se o protocolo for executado em um ambiente de alto desempenho, como computação nas nuvens, por exemplo. O que tornaria esse protocolo também viável para grandes bases de dados. Os resultados e análises de desempenho são apresentadas no capítulo 5.

1.4 Organização do texto

Esta dissertação está estruturada como se segue:

O capítulo 1 mostra o contexto e a definição do problema, além disso, são apresentadas as justificativas e as hipóteses levantadas a partir do problema inicial.

No capítulo 2 são apresentados a metodologia e os meios materiais utilizados para a solução do problema inicial. Além disso, nesse capítulo é apresentado um resumo dos trabalhos cor-

relatos. Por fim, são descritos os passos utilizados para a construção dos protótipos e quais as métricas utilizadas para análise dos resultados e a avaliação das hipóteses levantadas.

Já no capítulo 3, é apresentado o estado da arte e as principais dificuldades encontradas na área de consulta a base de dados cifradas. Em seguida, esses trabalhos são analisados e comparados com a proposta desta dissertação. Além disso, são apresentados os conceitos sobre computação nas nuvens e em seguida, são introduzidos os fundamentos teóricos e matemáticos de criptografia e de assinatura digital utilizados como base para o desenvolvimento do protocolo. Nesse capítulo, também é justificado o porque da escolha do criptosistema e esquema de assinatura El-Gamal como fundamento.

No capítulo 4, são apresentadas as definições do novo protocolo e da **álgebra relacional para dados cifrados**. Em seguida no capítulo 5, a segurança do protocolo é analisada contra possíveis ataques no qual ele está exposto. Além disso, são apresentadas a arquitetura e os passos utilizados na construção do protótipo. Por fim, o protótipo é testado e os resultados são avaliados.

No capítulo 6, é realizada uma análise crítica dos resultados obtidos. Em seguida, são apresentadas sugestões para os trabalhos futuros, conclusões e contribuições desta dissertação.

O Apêndice A contém o código fonte do protótipo desenvolvido. Já o apêndice B apresenta uma breve definição da representação de dados *Base64*.

Capítulo 2

Material e Metodologia

Neste capítulo são destacados a metodologia e os meios materiais utilizados para o desenvolvimento da dissertação. Além disso, são apresentadas as principais dificuldades nos trabalhos já realizados na área de consultas em bases de dados cifradas que influenciaram na proposta desta dissertação. Por fim, é apresentada uma breve descrição da construção do protótipo e dos experimentos realizados e como eles podem demonstrar as hipóteses levantadas.

Para o desenvolvimento da solução do problema apresentado na seção 1.1 foram definidas as seguintes etapas:

- Referencial teórico;
- Estado da arte;
- Desenvolvimento do protótipo e ambiente de testes;
- Análise dos resultados, conclusões e trabalhos futuros;

Inicialmente foi elaborada uma pesquisa bibliográfica a partir de artigos publicados em congressos, *journals*, dissertações de mestrado e principalmente livros. Essa pesquisa consiste no levantamento de conceitos relacionados à computação nas nuvens, teoria dos números, criptografia, segurança e técnicas existentes na utilização de consultas em base de dados cifradas.

Após a definição da base teórica, foi realizado o levantamento dos trabalhos correlatos existentes na área. A partir do estado da arte, foram verificadas as principais técnicas utilizadas e as principais dificuldades encontradas. Após esse levantamento, verificou-se que a principal dificuldade para execução de consultas em base de dados cifradas está relacionada a consultas que utilizam os operadores $\{>, <\}$. Isto ocorre porque em criptografia os dados cifrados não correspondem necessariamente a mesma ordem da sua forma decifrada, ou seja, dados dois números x e y não cifrados e E_x e E_y os seus correspondentes cifrados, é possível existir valores para x e y tais que $x < y$, mas $E_x > E_y$. Em outras palavras, sabendo que $E_x > E_y$ não significa necessariamente que $x < y$. Nesse caso como não existe nenhuma relação direta entre a ordem dos dados não cifrados e cifrados, a comparação entre os dados cifrados torna-se difícil.

Em seguida, verificou-se também que os sistemas gerenciadores de banco de dados (SGBDs) comerciais atuais, possuem implementados diversos algoritmos de criptografia. Entretanto, não foi encontrado nenhum mecanismo que resolva a comparação dos dados cifrados utilizando os operadores $\{<, >\}$.

Além disso, nos trabalhos pesquisados os autores realizam a comparação entre os dados cifrados através de funções específicas definidas nesses trabalhos. Em [Hacigumus et al. 2002] por exemplo, os autores utilizam uma função secreta que mapeia a ordem dos dados. Já em [Wang et al. 2004], a ordem dos dados é obtida através de técnicas de metadados.

Já em [Evdokimov e Guenther 2007] os autores utilizam criptografia simétrica, números aleatórios e indistinguibilidade perfeita. O principal objetivo é procurar os dados por tags especiais que representam os dados. Em [Li et al. 2009] é proposto um mecanismo que busca palavras por similaridade baseados em lógica fuzzy. Por fim, em [Laur et al. 2011] é apresentado um protocolo para compartilhar uma base de dados, o objetivo principal é compartilhar os dados secretos com outras entidades confiáveis. Para atingir esse objetivo, os autores utilizam técnicas de compartilhamento de segredo.

Após o levantamento dos trabalhos correlatos, foram analisados os criptosistemas existentes, afins de encontrar a melhor solução para o problema da comparação de dados cifrados. Por fim, decidiu-se utilizar como fundamento o criptosistema El-Gamal e o seu o esquema de assinatura.

Em seguida, foi desenvolvido o modelo formal do protocolo e o seu protótipo. Primeiramente, foram definidos os requisitos não funcionais do protótipo, tais como: linguagem de programação, Sistema Gerenciador de Banco de Dados (SGBD) e a massa de dados utilizada nos testes. Para a geração dessa massa de dados, foram utilizados softwares geradores de dados para testes que podem ser obtidos em [Benjaminkeen 2001]. Todos esses requisitos são detalhados e apresentados no capítulo 4. Além disso, foram definidos os requisitos funcionais do protótipo, tais como: tamanho da chave utilizada, o valor da chave, elemento gerador, número primo utilizado, entre outros. Dentre esses parâmetros, a chave privada e o número primo, por exemplo, foram obtidos pelo aplicativo *BigCalculatorNumber* que pode ser acessado em [Reinhold 2000]. Esse software foi utilizado porque possui implementado os algoritmos de geração de números primos grandes, ou seja, números com mais de 1024 bits. Nos testes foi utilizada uma chave de 1024 bits e um número primo de 1025 bits. Já o número w foi obtido aleatoriamente. O elemento gerador foi obtido a partir do algoritmo apresentado em [Shoup 2008]. Esses parâmetros são detalhados no capítulo 5.

Por fim, foram definidas as métricas para a avaliação dos resultados, essas métricas são: tempo, tamanho da base de dados e tamanho da chave privada. A primeira refere-se ao tempo gasto pelo servidor para a realização da consulta. Já a segunda métrica, refere-se ao tamanho do espaço em disco utilizado para armazenamento dos dados cifrados. Em relação a terceira, essa escolha se deve pelo fato de que a segurança de um protocolo de criptografia está diretamente relacionada ao tamanho de sua chave. Nesse caso, para avaliar a segurança do protocolo em relação ao tamanho da chave, foi utilizado um resultado proposto pela ECRYPT (*European*

Network of Excellence for Cryptology) em [Ecrypt 2010], que relaciona o nível de segurança exigido por um ambiente e o tamanho da chave necessário para alcançar tal segurança. No capítulo 5 essa métrica é brevemente explicada e analisada no contexto dessa dissertação.

Para realizar os experimentos, foram criados cinco cenários distintos. Cada cenário é composto por uma base de dados cifrada e uma base de dados não cifrada. Os testes de desempenho foram realizados em uma relação *funcionarios* composta pelos seguintes campos: *codigo*, *nome*, *salario* e *estadocivil*. Além dessa, existe também uma relação *funcionarios^C* que corresponde a cifragem de *funcionarios*. No primeiro cenário, a relação *funcionarios* e *funcionarios^C* possui 400 registros cada. Já nos outros cenários são utilizados: 6.400, 10.050, 80.400 e 643.200 registros respectivamente. Esses registros foram obtidos através do software [Benjaminkeen 2001]. A partir desses dados, definiu-se a seguinte consulta para comparação dos resultados:

“Select * from *funcionarios* where *salario* > 3000”

A partir desses dados, foram realizadas as implementações em um computador com a seguinte configuração: Intel Pentium D 2,66 GHZ com 1 GB de memória.

Além disso, o protocolo foi implementado em um ambiente local para testar a sua correção. O protocolo foi desenvolvido na linguagem C# e de forma integrada ao SGBD *SQL Server 2008*. Para representar os números grandes (1024 bits ou superior) utilizados pelo protocolo foi utilizado o tipo de dados *BigInteger*.

A análise dos resultados desses experimentos foi dividida em duas etapas. A primeira consistiu na verificação dos resultados retornados pelas consultas cifradas e não cifradas, ou seja, analisar se os funcionários retornados tem o salário superior a 3000. Já a segunda análise comparou o desempenho do protocolo e o tamanho da base de dados em cada cenário. Além disso, foi construído um gráfico comparando as métricas obtidas em cada experimento. A partir desses resultados, constaram-se as hipóteses levantadas no capítulo 1, ou seja, o desempenho das consultas em dados cifrados são inferiores ao desempenho de uma consulta em uma relação não cifrada.

2.1 Considerações Finais

Neste capítulo foi apresentado os métodos utilizados para o desenvolvimento do protocolo e a verificação das hipóteses levantadas no capítulo 1. Essa metodologia foi elaborada a partir das principais dificuldades encontradas nos trabalhos correlatos sobre consulta a base de dados cifradas. A partir disso, o próximo capítulo apresenta as fundamentações teóricas utilizadas como base para a solução do problema abordado.

Capítulo 3

Fundamentação Teórica

Conforme as hipóteses apresentadas no capítulo 1, pressupõe-se que os algoritmos do protocolo proposto pode ter um alto custo computacional para consultas em grandes bases de dados cifradas, isso ocorre, porque esses algoritmos são baseados na criptografia de chave pública. Diante desse cenário, faz-se necessário a utilização de servidores de banco de dados de alto desempenho. A partir disso, verificou-se que todos esses requisitos são encontrados nos serviços de computação nas nuvens. Logo, o problema do desempenho pode ser minimizado com a hospedagem das bases de dados cifradas em computação nas nuvens.

Diante desse contexto, este capítulo apresenta os fundamentos de computação nas nuvens e criptografia utilizados como fundamento para o desenvolvimento do protocolo.

O capítulo está estruturado da seguinte forma: Inicialmente na seção 3.1 é mostrado o estado da arte e os principais trabalhos similares. Além disso, é realizado um comparativo dessas propostas com o protocolo desta dissertação. Na seção 3.2 são apresentados os conceitos teóricos e os principais riscos de segurança do ambiente em computação nas nuvens. Em seguida na seção 3.3 são introduzidos os fundamentos matemáticos necessários para a utilização dos algoritmos analisados no capítulo. Finalmente, na seção 3.4 é apresentada uma definição formal de criptografia e as notações utilizadas nos algoritmos. Em seguida, o criptosistema de chave pública El-Gamal é definido e analisado. Além disso, é introduzido o conceito de assinatura digital e o esquema de assinatura El-Gamal. Por fim, são apresentadas as justificativas pela escolha do El-Gamal como fundamentação teórica.

3.1 Estado da Arte

Uma abordagem de consulta em base de dados cifradas foi proposta em [Hacigumus et al. 2002]. Nessa abordagem é implementado um modelo relacional, no qual cada tabela é composta de n atributos e uma coluna adicional. Cada um dos atributos armazena a ordem daquele registro na tabela, esse armazenamento é realizado através de uma função de mapeamento que é responsável por preservar a ordem dos dados. De acordo com os autores essa função é secreta.

A coluna adicional contém a cifragem da concatenação de todos os atributos juntos, ou seja, utilizando uma função de cifragem $E(x)$ tal que: $E(x) = E(A_1 + A_2, ..., A_n)$. Nesse caso, $E(x)$ é o algoritmo utilizado para cifrar e decifrar os dados no cliente. O algoritmo utilizado pode ser o algoritmo RSA, DES, etc.

Diferentemente do protocolo proposto em [Hacigumus et al. 2002], no protocolo proposto neste trabalho o algoritmo utilizado é público e a ordem dos dados não estão explícitas nas tabelas. Nesse caso, a ordem dos dados pode ser utilizada pelo espião para decifrar ou descobrir os dados armazenados na tabela. No trabalho utilizado em [Hacigumus et al. 2002], é utilizada uma função de mapeamento para ordenar os dados que é considerada pelos autores como secreta, além disso, essa coluna indica nas colunas a ordem dos dados na tabela.

Outro trabalho similar é proposto em [Wang et al. 2004]. Nesse caso, a principal idéia é considerar os textos não cifrados como uma cadeia de caracteres e, através de uma função de *hash*, codificar os caracteres dois a dois. As codificações dos caracteres são então utilizadas, posteriormente, como indexadores para filtros desses registros na base de dados. A consulta ao banco de dados é dividida em duas fases: Fase 1: traduz as condições das consultas usando as regras de meta dados e executa a consulta cifrada sobre os índices cifrados. Em seguida são retornados para o cliente os registros que satisfazem as condições. Na fase 2, os dados cifrados são retornados para o cliente que decifra o resultado e em seguida executa a consulta SQL original para um refinamento desses dados. O resultado final obtido é refinado pelo cliente para eliminar os resultados indesejáveis.

No protocolo proposto nesta dissertação, a verificação dos dados cifrados é feita em uma única fase, diferentemente do artigo proposto em [Wang et al. 2004], no qual, os autores afirmam que é necessário realizar uma segunda consulta com os dados decifrados no cliente. Além disso, no protocolo desta dissertação a consulta é executada somente no servidor, ou seja, o cliente apenas decifra o resultado final, diferentemente do trabalho de [Wang et al. 2004] onde o cliente necessita realizar a consulta localmente sobre os dados decifrados.

Em [Silberschatz et al. 2006] são apresentados alguns mecanismos de criptografia em banco de dados comerciais. Dentre estes mecanismos destacam-se:

- **Criptografia em nível de bloco de dados:** Os blocos de dados são criptografados antes de serem armazenados no disco. A chave utilizada para cifragem é disponibilizada pelo usuário ao software de Gerenciamento de Banco de Dados (SGBD) que realiza todo processo de criptografia. No entanto, para extrair os dados cifrados do disco, inicialmente o SGBD decifra os blocos de dados utilizando a mesma chave disponibilizada pelo usuário ao SGBD.
- **Criptografia em nível de aplicação:** diversos SGBDs oferecem APIs para criptografar os dados. Nesse caso, os dados são criptografados pela aplicação antes de serem enviados ao servidor de banco de dados. Geralmente, é utilizada uma única chave para cifrar e decifrar todas as colunas e linhas das tabelas.

- **Sistemas desafio resposta:** O servidor de banco de dados escolhe aleatoriamente um número e cifra o mesmo utilizando a chave pública do cliente. Em seguida, o servidor envia o número cifrado para o cliente. Ao receber o desafio do servidor, o cliente decifra o número recebido utilizando sua chave privada e retorna o resultado para o servidor de banco de dados. Ao receber o resultado, ele verifica a resposta. Esse método é um mecanismo utilizado para autenticação do cliente no servidor de banco de dados.

Além dos métodos apresentados anteriormente, SGBDs comerciais como, por exemplo, *MySQL*, *Oracle* e *SqlServer* possuem, atualmente, implementados diversos algoritmos de criptografia tais como: DES, AES, MD5, SHA1, entre outros [Mysql 2011], [Oracle 2011] e [Microsoft 2011]. No entanto, para realizar uma consulta em uma base de dados cifrada, é necessário inicialmente decifrar os dados no servidor. Em seguida, esses dados são comparados utilizando, por exemplo, operadores $\{>, <\}$. As únicas exceções a essa regra são os operadores $\{=, \neq\}$. Através desses operadores, é possível verificar se dois dados cifrados são iguais ou diferentes sem a necessidade de decifrá-los no servidor.

Portanto, a principal desvantagem dos modelos pesquisados nos SGBDs comerciais, é a necessidade de decifrar os dados no servidor para realizar as comparações como, por exemplo, os operadores $\{>, <\}$. Diferentemente do protocolo proposto neste trabalho que possui mecanismos para a verificação dos dados cifrados, ou seja, a partir deste protocolo é possível decidir entre dois dados cifrados qual deles é o maior, por exemplo.

Em [Evdokimov e Guenther 2007] é proposto um modelo e um protocolo de criptografia que resolve o problema de consultas em base de dados cifrada. O protocolo é baseado em criptografia de chave simétrica, números aleatórios e indistinguibilidade perfeita.

Seja k a chave privada do cliente definida por:

$$k = (k_0, k_1, k_2)$$

O algoritmo de cifragem E é definido da seguinte forma:

1. O texto não cifrado x é cifrado por um algoritmo simétrico E e uma chave privada K_0 , ou seja, $c = E(x, k_0)$;
2. Utilizando uma função pseudoaleatória P calcula-se uma chave K_s definida por $K_s = P(x, k_1)$;
3. O texto x é cifrado pelo algoritmo pseudoaleatório P e uma chave k_2 , ou seja, $s = P(x, k_2)$;
4. Usando o texto cifrado s e a chave K_s a tag de busca t é gerada da seguinte forma: $t = E(K_s, s)$;
5. A saída do algoritmo é o par (t, c) .

Dado o texto cifrado (t, c) , a decifragem é definida por:

$$D(c, k_0) = x.$$

A partir das definições anteriores essa abordagem permite as seguintes operações nas bases de dados cifradas: Seleção com condições que utilizam o operador “=”, projeção, produto cartesiano, inserção, remoção, união e junção. A principal vantagem desse protocolo é o desempenho do algoritmo de seleção, por exemplo, a complexidade desse algoritmo é $O(n)$, onde n é o número de tuplas de uma relação.

Entretanto, essa proposta apresenta algumas limitações, por exemplo, não é possível realizar consultas nas bases de dados cifradas com condições que utilizam operadores $>$ e $<$. Diferentemente do protocolo proposto nesta dissertação.

Em [Li et al. 2009] é proposto um mecanismo de busca em dados cifrados armazenados em computação nas nuvens. Nesse caso, a principal contribuição é permitir buscas nos dados cifrados considerando erros de escrita dos usuários. Em outras palavras, o protocolo não realiza busca por palavras chaves exatas, mas sim por similaridade das palavras, nesse caso se o usuário busca uma palavra que contém erros, o sistema verifica dentre todos os dados cifrados nas nuvens quais são as palavras mais similares e em seguida retorna o resultado para o usuário.

O protocolo proposto nesse trabalho é baseado em técnicas de lógica fuzzy, algoritmos de distância de edição e busca por similaridade das palavras. Além disso, a criptografia dos dados é realizada por algoritmos de criptografia simétrica, o motivo de tal escolha é baseado na eficiência desses algoritmos. A busca no servidor de nuvem é realizada em três passos: Inicialmente o aplicativo do usuário constrói um conjunto de palavras similares a partir da palavra chave pesquisada. Em seguida, o protocolo cifra cada palavra com a chave privada do usuário e envia esse conjunto cifrado para a nuvem.

Ao receber esse conjunto, o servidor de nuvem constrói uma árvore de indexação das palavras cifradas. Em seguida, o servidor aplica nessa árvore algoritmos de busca para encontrar as palavras mais similares. Após a busca na árvore, o servidor retorna para o cliente, por exemplo, o conjunto de arquivos cifrados que contêm as palavras mais similares em relação à palavra chave informada pelo usuário. Ao receber essa lista do servidor, o usuário decifra os resultados retornados e a partir disso filtra as informações mais relevantes. A principal desvantagem desse protocolo é permitir apenas buscas em textos, diferentemente do protocolo proposto nesta dissertação que realiza buscas em campos textuais e numéricos e permite comparações do tipo $>$ ou $<$.

Em [Laur et al. 2011] é proposto um protocolo de criptografia para manipulação de base de dados compartilhada. O protocolo possui três entidades nesse contexto: Os proprietários dos dados que compartilham as bases de dados, os servidores que armazenam os dados e que realizam os cálculos que reconstroem os segredos e os clientes que realizam buscas nessas bases. Inicialmente os proprietários dos dados compartilham seus dados com os servidores utilizando técnicas de criptografia baseadas em compartilhamento de segredo. Esses segredos são utilizados pelo protocolo proposto para realizar pesquisas solicitadas pelos clientes. Os dados que não são compartilhados continuam protegidos e são mantidos em segredo. Por exemplo, suponha

que o proprietário da base de dados deseja compartilhar as seguintes informações: pacientes do sexo feminino com mais 65 anos e que possuem pressão alta. O servidor a partir desses predicados constrói um vetor indicador f que indica quais os registros que possuem esses predicados. Seja x o vetor com os dados da base de dados, durante o processo de filtragem o servidor usa x e f como entradas para algoritmos de construção de segredo que obtém como saída uma nova base de dados com os dados compartilhados denotada por y .

A principal vantagem desse trabalho é a eficiência desses protocolos $O(\log \log n)$. A principal desvantagem é que os servidores utilizados para a reconstrução dos segredos devem ser seguros e confiáveis.

Na próxima seção são introduzidos os conceitos e as principais características do ambiente de computação nas nuvens.

3.2 Computação nas Nuvens

Definição

Segundo [Sousa et al. 2009], o NIST (*National Institute of Standards and Technology*) apresenta a seguinte definição “*Computação em nuvem é um modelo que possibilita acesso, de modo conveniente e sob demanda, a um conjunto de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicações e serviços)*”.

A figura 3.1 mostra uma visão geral do ambiente em computação nas nuvens.

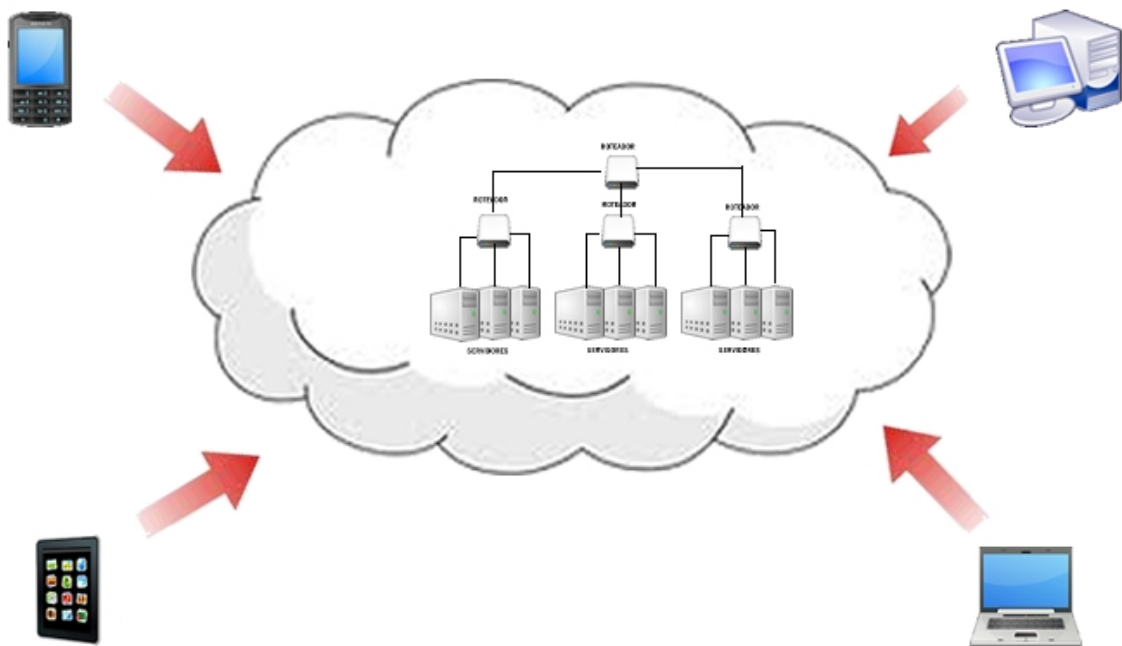


Figura 3.1: Visão de uma nuvem computacional, adaptado de [Sousa et al. 2009].

O cenário apresentado na figura 3.1 mostra basicamente como funciona esse ambiente.

Nesse cenário, as aplicações e os dados dos usuários são armazenados em servidores remotos espalhados em diversos locais do mundo acessados pela internet. Porém, essa estrutura é abstraída para o usuário que utiliza os recursos de diversos servidores como se fossem em um único servidor. Dessa idéia é que surge a metáfora de nuvem. Os usuários podem utilizar diversos dispositivos para acessar a nuvem, como por exemplo, computadores pessoais, celulares, *notebooks*, entre outros [Sousa et al. 2009].

As principais características da computação nas nuvens são [Sousa et al. 2009]:

- **Ampla acesso:** a computação nas nuvens possui mecanismos que abstraem os recursos de hardware da infraestrutura do provedor. Ou seja, o acesso aos serviços independe do sistema operacional ou do hardware. Além disso, não é necessário que as aplicações estejam instaladas nos computadores dos usuários para serem executadas, é necessário apenas que o usuário tenha acesso a internet e uma conta de usuário na nuvem.
- **Economia:** Devido ao alto custo de infraestrutura, licenças de softwares, funcionários e treinamentos, diversas empresas utilizam serviços de terceiros como uma solução mais barata. A principal vantagem desse modelo é permitir que as empresas transfiram para os terceiros todos os custos e serviços de infraestrutura de TI envolvidos. Outra vantagem do modelo em nuvem é permitir que empresas ampliem seus recursos computacionais de TI sem que sejam necessários grandes investimentos.
- **Segurança:** é de responsabilidade da fornecedora os serviços de *backups*, manutenção, autenticação, plano de contingência, entre outros.
- **Disponibilidade:** os usuários podem acessar seus serviços e recursos usando-se da internet a qualquer hora e de qualquer lugar.
- **Auto serviço sob demanda:** Os provedores de serviços monitoram e disponibilizam automaticamente aos clientes recursos de acordo com sua necessidade. Por exemplo, tempo de processamento no servidor para execução de determinado serviço ou um maior espaço de armazenamento na rede, etc.
- **Escalabilidade e elasticidade rápida:** permite facilmente a alocação de recursos de hardware em tempo de execução de acordo com a demanda do cliente. Essa alocação é automática, ou seja, o próprio sistema pode dispor temporariamente os recursos e em seguida retirá-los. Essa capacidade dos recursos de armazenamento é considerada teoricamente infinita [Breitman e Viterbo 2010].

A figura 3.2 mostra a topologia do ambiente em computação nas nuvens. Existem três entidades nesse contexto: [Tujal 2010]

- **Provedor da infraestrutura da nuvem:** Seu papel é gerenciar e administrar o ambiente de infraestrutura e hardware. Todos esses serviços são disponibilizados para os provedores de serviços.

- **Provedor de serviço:** Utiliza os recursos disponibilizados pelo provedor de infraestrutura para oferecer aos seus clientes serviços de software, armazenamento, segurança e ambiente de testes.
- **Consumidor de software como serviço:** São os clientes dos provedores de serviços, por exemplo, *Google* e *Microsoft*, que utilizam os serviços de aplicações *web*. Os clientes podem ser desde usuários comuns que armazenam seus documentos pessoais até empresas que terceirizam toda sua infraestrutura [Sousa et al. 2009].

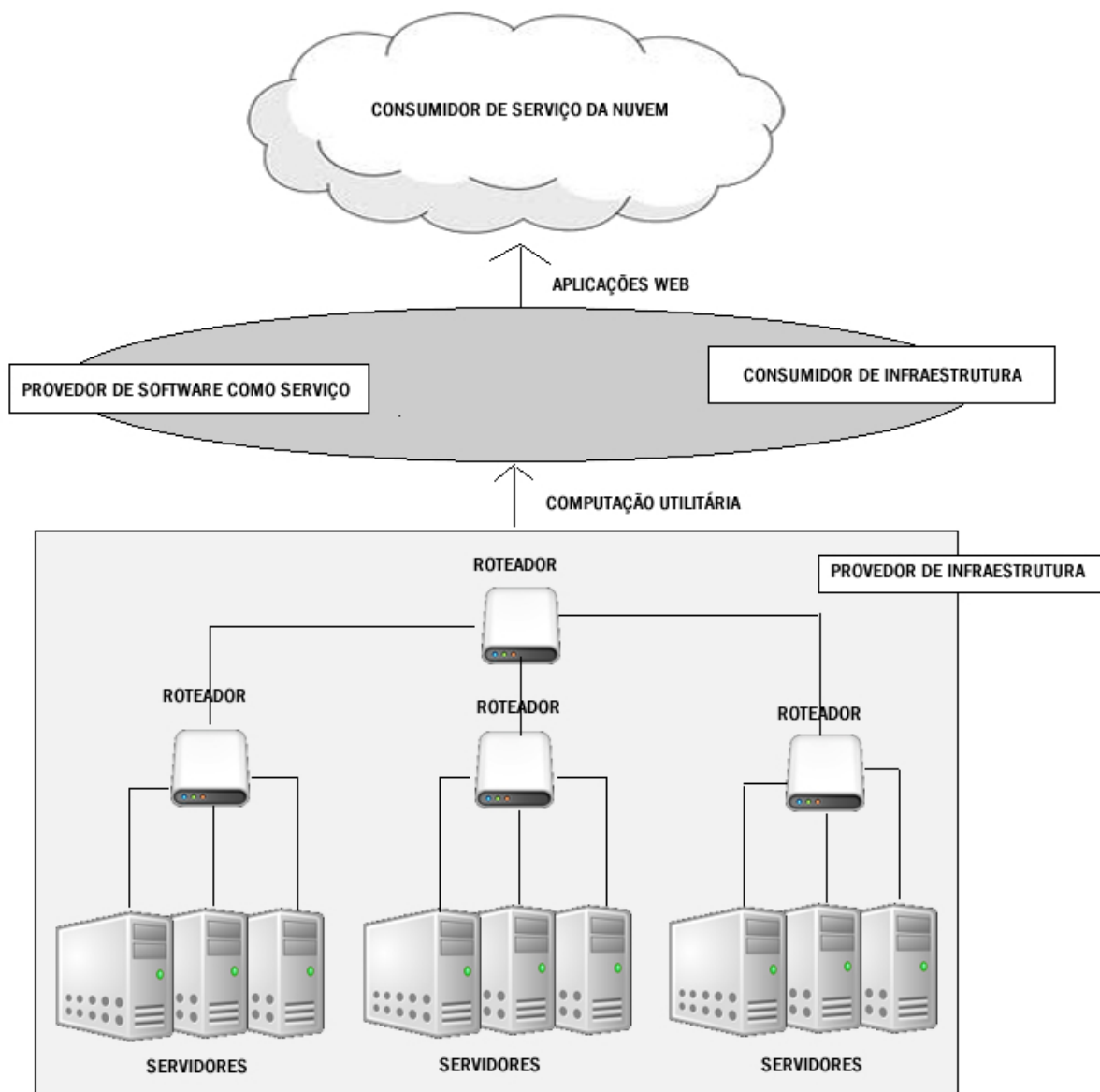


Figura 3.2: Topologia da computação nas nuvens [Tujal 2010].

Além da topologia, as nuvens são classificadas de acordo com o modelo de implantação. Esse modelo define o nível de segurança e acesso das informações, além dos recursos disponíveis na nuvem.

Os modelos de implantação são divididos basicamente em três grupos [Tujal 2010]:

- **Nuvem privada ou proprietária:** Nuvens restritas a um único cliente ou empresa, toda sua infraestrutura de servidores e equipamentos é de propriedade ou alugada do cliente ou empresa, ou seja, o cliente possui acesso e controle total sobre os serviços utilizados na nuvem. Entretanto, seus recursos são limitados ao parque tecnológico da fornecedora de serviço que estão localizados em *data centers* privados [Tujal 2010].
- **Nuvem pública:** A fornecedora disponibiliza e controla a infraestrutura da nuvem. Além disso, ela vende serviços ou aplicativos para usuários comuns ou empresas em geral e disponibiliza para os desenvolvedores e programadores diversas APIs para que eles possam criar funcionalidades e aplicativos nas nuvens. Nesse ambiente, existem aplicações com licenças proprietárias e código aberto. Para esse último caso, a nuvem é denominada como nuvem de código aberto [Tujal 2010].
- **Nuvem híbrida:** Esse modelo permite que as nuvens privadas expandam seus recursos utilizando recursos da nuvem pública. Além disso, esse modelo permite que os serviços e recursos não fiquem indisponíveis, pois as nuvens estão interligadas e permitem a portabilidade de serviços e informações entre elas [Tujal 2010].

3.2.1 Modelos de Serviços

Todos os recursos de computação nas nuvens são oferecidos como serviços. Cada serviço é definido conforme a necessidade do cliente. A figura 3.3 mostra a arquitetura de serviços em três camadas.

A primeira camada é caracterizada pelas aplicações destinadas aos usuários finais. A principal vantagem dessa camada é permitir que os usuários não adquiram licenças e nem precisam instalar os softwares nas suas máquinas locais, ou seja, os serviços são executados diretamente na nuvem. Além disso, todo o processamento desses aplicativos é executado na nuvem, isso possibilita então que dispositivos de pequeno poder computacional, como celulares, por exemplo, utilizem os softwares nas nuvens. Nesses casos, os únicos pré requisitos que os usuários precisam para acessar os serviços são: acesso à internet e possuir um cadastro em algum serviço na nuvem [Martins 2010].

Os custos desses serviços dependem da fornecedora e da aplicação que podem exigir que o usuário pague ou não um valor por esses serviços. Atualmente a *Google* e *IBM* disponibilizam várias soluções de software como serviço seus clientes.

A segunda camada inclui os seguintes serviços: desenvolvimento de aplicações, ambientes de testes e simulações. Os principais clientes desses serviços são os desenvolvedores de software. Porém, é necessário que os desenvolvedores utilizem linguagens de programação e ferramentas suportadas pela nuvem. Ao término do desenvolvimento, os aplicativos são transferidos para as nuvens e em seguida disponibilizá-los aos clientes. O acesso dos desenvolvedores a nuvem é limitado ao controle e gerenciamento do ambiente onde é executado o aplicativo [Martins 2010].

As camadas superiores de software como serviço (SaaS) e plataforma como serviço (PaaS) são analisadas e detalhadas em [Breitman e Viterbo 2010]. Dentro do contexto desta dissertação a camada inferior IaaS é apresentada a seguir.

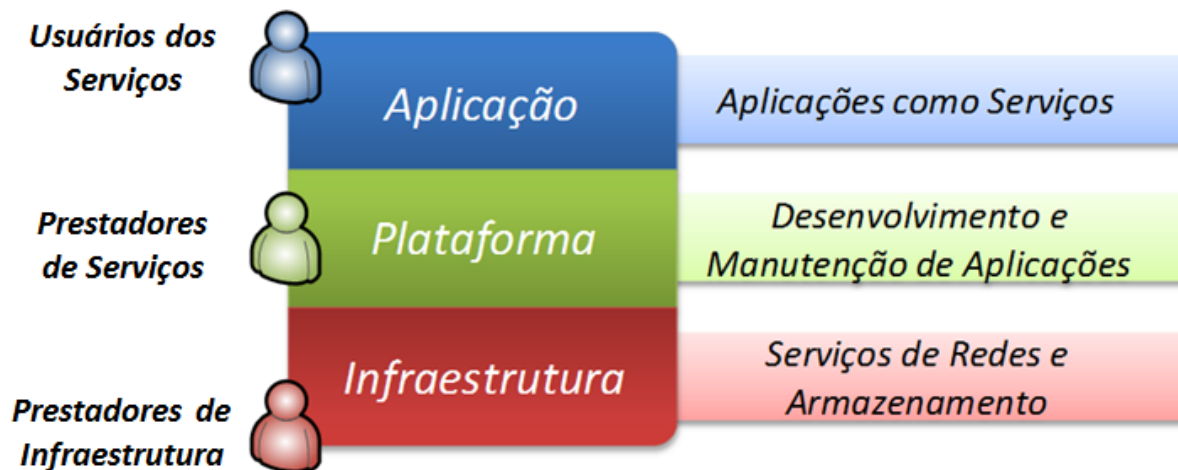


Figura 3.3: Arquitetura de serviços em computação nas nuvens de três camadas.

Infraestrutura como serviço (IaaS)

Nessa camada o cliente aluga da fornecedora toda a infraestrutura de TI composta por servidores, *firewalls*, roteadores, estrutura de *backup*, banco de dados, entre outros. Além disso, o cliente não administra ou controla os recursos de infraestrutura adquiridos, porém, ele tem controle sobre o sistema operacional, unidades de armazenamentos, aplicativos, plataforma e alguns recursos de rede como *firewall*, por exemplo. Além disso, os servidores alugados já possuem todos os softwares instalados e licenciados. Todas as atualizações desses softwares são de responsabilidade da fornecedora de IaaS [Breitman e Viterbo 2010]. Além disso, é de responsabilidade da fornecedora os serviços de *backups*, manutenção, plano de contingência, entre outros. As principais fornecedoras de IaaS são: *Force.com*, *Google app Engine*, *Bungee*, entre outras.

O IaaS é dividido em outros serviços, dentre estes é analisado a seguir a Base de dados como serviço (DaaS).

Base de dados como serviço (Daas)

É um modelo onde o provedor de serviço oferece uma base de dados física e lógica como serviço. Além disso, a base de dados nas nuvens apresenta as mesmas funcionalidades das bases de dados locais. A única diferença é que no DaaS a base de dados é hospedada remotamente.

A principal vantagem desse modelo em relação ao tradicional é transferir toda a responsabilidade da infraestrutura para a fornecedora de serviços, além desta, destacam-se [Cambiucci 2009]:

- Controle de *backup* e segurança dos dados;
- Instalação, atualização e configuração dos servidores e dos sistemas gerenciadores de banco de dados (SGBD);

- Redundância contra falhas de hardware;
- Acompanhamento e redimensionamento dos servidores;
- APIs utilizadas para comunicação com os SGBDs;
- Redução do custo de infraestrutura da base de dados.

No contexto em nuvem, o papel do administrador da base de dados passa a ser focado na regra de negócio da empresa, no desenvolvimento e melhoria das aplicações [Cambiucci 2009].

Na próxima seção são apresentados os principais riscos dos serviços de computação nas nuvens.

3.2.2 Segurança em Computação nas Nuvens

O principal problema enfrentado na utilização dos serviços de computação nas nuvens é a segurança. Clientes e empresas que contratam esses serviços exigem cada vez mais garantias de segurança e privacidade nesses ambientes. Em [Brodkin 2008] é apresentada uma análise dos riscos em computação nas nuvens. Nesse relatório, [Brodkin 2008] lista sete dos principais riscos de segurança na utilização dos serviços de computação nas nuvens:

1. **Acesso privilegiado de usuários:** Quando se utiliza uma infraestrutura própria, é possível controlar o acesso físico, lógico e de pessoas ao ambiente de tecnologia da informação. Entretanto, no ambiente de computação nas nuvens isso não é possível, pois os dados são armazenados e processados fora do ambiente interno da empresa.
2. **Integridade dos dados:** As empresas são responsáveis pela segurança e integridade de seus próprios dados, mesmo quando esses dados são armazenados nos provedores de serviços.
3. **Localização dos dados:** No serviço de computação nas nuvens não se sabe onde exatamente os dados das empresas estão armazenados. Além disso, os servidores são administrados por pessoas que as empresas não conhecem.
4. **Segregação dos dados:** Diversas empresas e clientes dividem um mesmo ambiente de computação nas nuvens, essa estrutura pode apresentar um risco de segurança no qual deve ser garantido pelo provedor de serviço.
5. **Recuperação dos dados:** Como os dados das empresas estão armazenados em diversos servidores espalhados pelo mundo, existe um risco de perda dos dados. Nesse caso, as empresas devem verificar com os fornecedores de serviços de computação nas nuvens quais são os mecanismos oferecidos por eles para recuperação dos dados em caso de perdas ou desastres.
6. **Apoio a investigação:** A investigação de serviços ilegais pode ser extremamente complexa ou até impossível. Principalmente porque os dados estão espalhados e muitas das vezes não se sabe exatamente onde a informação está armazenada.

7. **Disponibilidade e viabilidade a longo prazo:** O provedor de serviços de computação nas nuvens precisa garantir que os dados de seus clientes estejam sempre disponíveis. Além disso, caso o provedor de serviços encerre suas atividades é necessário que ele disponibilize de volta os dados de todos os seus clientes.

Diante dos cenários apresentados nesta seção, fica clara a necessidade da utilização de mecanismos seguros para garantir a autenticidade, integridade e a privacidade dos dados armazenados nas nuvens. O aumento dessa segurança pode ser obtida através da utilização de técnicas e algoritmos de criptografia.

Dado esse contexto, nas próximas seções são introduzidos os fundamentos matemáticos e uma breve introdução de criptografia e as notações utilizadas nos algoritmos. Em seguida, é apresentado o criptosistema de chave pública El-Gamal.

3.3 Fundamentos Matemáticos

3.3.1 Grupo Multiplicativo

Definição 3.1. (Grupo) Um grupo G representado por $(G, \bullet)$ é um conjunto de elementos definidos por uma operação binária indicada por $\bullet$ que satisfaz as seguintes propriedades [Stallings 2007]:

1. **Fechado:** Se x e $y \in G$, então $x \bullet y \in G$.
2. **Associativo:** Se $x \bullet (y \bullet s) = (x \bullet y) \bullet s$ para todo x, y e $s \in G$.
3. **Elemento identidade:** Existe um elemento $e \in G$, tal que $x \bullet e = e \bullet x = x$.
4. **Elemento inverso:** Para cada $x \in G$ existe um elemento $x' \in G$, então $x \bullet x' = x' \bullet x = e$.

Se a operação $\bullet$ é a soma, então o grupo $(G, +)$ é chamado de aditivo. Porém se a operação $\bullet$ é a multiplicação, então o grupo $(G, \cdot)$ é chamado de multiplicativo [Koblitz 1994].

O criptosistema El-Gamal utilizado como fundamento nessa dissertação utiliza como base o grupo multiplicativo $(\mathbb{Z}_p, \cdot)$, tal que $\mathbb{Z}_p$ é o conjunto dos inteiros não negativos definidos por:

$$\mathbb{Z}_p = \{0, 1, \dots, (p - 1)\}.$$

Além disso, o criptosistema El-Gamal é definido pelo conjunto de inteiros positivos não nulos indicados por $\mathbb{Z}_p^*$. Esse conjunto é definido por:

$$\mathbb{Z}_p^* = \{1, 2, \dots, (p - 1)\}$$

Exemplo 3.1. Seja $p = 11$, então o grupo $\mathbb{Z}_{11}$ e $\mathbb{Z}_{11}^*$ é dado por:

$$\mathbb{Z}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

e

$$\mathbb{Z}_{11}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Um estudo detalhado de grupos multiplicativos é apresentado em [Koblitz 1994] e [Shoup 2008].

3.3.2 Elemento Gerador

Dado um grupo multiplicativo, entre seus elementos pode existir um que é especial, a partir do qual todos os elementos do grupo podem ser gerados. Ou seja, se G é um grupo multiplicativo e $\alpha \in G$, tal que α é um elemento gerador de G , então todos os elementos de G podem ser gerados a partir de α [Scheinerman 2003].

Suponha, por exemplo, p um número primo e $\mathbb{Z}_p$ o grupo multiplicativo definido utilizando as operações modulares. Nesse caso, $\mathbb{Z}_p$ tem ordem p e existe um elemento gerador $\alpha \in \mathbb{Z}_p^*$, [Shoup 2008]:

Definição 3.2. (Elemento Gerador em $\mathbb{Z}_p^*$) Um elemento $\alpha \in (\mathbb{Z}_p^*, \cdot)$ é chamado de gerador, se e somente se, todo elemento de $(\mathbb{Z}_p^*, \cdot)$ pode ser gerado a partir de α , utilizando apenas a operação de multiplicação módulo p , denotada por “ $\cdot$ ”, ou seja,

$$\mathbb{Z}_p^* = (\alpha) = \{\alpha^i : 0 \leq i \leq p - 1\}.$$

Nesse caso, $\mathbb{Z}_p^* = (\alpha)$ é um grupo cíclico de ordem¹ p .

Exemplo 3.2. Seja $p = 11$ e $\alpha = 2$, a tabela 3.4 mostra todas as potências de α módulo 11, para todo inteiro positivo i , tal que:

$$\{2^i : 1 \leq i \leq 10\}$$

i	$2^i \bmod 11$	$3^i \bmod 11$
1	2	3
2	4	9
3	8	5
4	5	4
5	10	1
6	9	3
7	7	9
8	3	5
9	6	4
10	1	1

Figura 3.4: Resultado de $2^i \bmod 11$ e $3^i \bmod 11$.

¹Representa o número de elementos de um grupo multiplicativo [Shoup 2008].

Observe na figura 3.4 que todos os elementos de $\mathbb{Z}_{11}^*$ são gerados a partir de 2. Nesse caso, diz-se que o número 2 é um elemento gerador de $\mathbb{Z}_{11}^*$.

Suponha agora que $\alpha = 3$, a figura 3.4 apresenta o resultado de $3^i \bmod 11$, para todo i no intervalo $1 \leq i \leq 10$.

Através do resultado da figura 3.4, observa-se que não é possível obter todos os elementos de $\mathbb{Z}_{11}^*$ a partir de 3. Isso ocorre porque existe uma congruência² entre os expoentes, por exemplo, $3^1 \equiv 3^6 \bmod 11$, $3^2 \equiv 3^7 \bmod 11$, etc. Em outras palavras, a sequência se repete, quando os expoentes diferem em 5 ou múltiplo de 5 [Stallings 2007]. Nesse caso, o número 3 não é um gerador de $\mathbb{Z}_{11}^*$.

Observe, também, que um grupo cíclico de ordem p , pode conter vários elementos geradores [Stallings 2007]. No caso do grupo $\mathbb{Z}_{11}^*$, os geradores são 2, 6, 7, 8.

3.3.3 Logaritmo Discreto

A segurança do protocolo proposto neste trabalho se fundamenta na dificuldade em resolver o problema do **logaritmo discreto**. Define-se, então, a seguir o problema do **logaritmo discreto** em um grupo multiplicativo $\mathbb{Z}_p^*$ [Mao 2003].

Definição 3.3. (Logaritmo discreto) Seja p um número primo, e α um elemento gerador de $\mathbb{Z}_p^*$. [Mao 2003]

Logo, para um inteiro $\beta \in \mathbb{Z}_p^*$, existe um único inteiro k , tal que,

$$\alpha^k \equiv \beta \pmod{p} \text{ onde } 0 \leq k \leq p - 1$$

O expoente k é o **logaritmo discreto** de β na base α [Mao 2003].

Utilizando a notação modular, o valor de k é obtido por:

$$k \equiv \log_{\alpha} \beta \pmod{p}$$

Na definição anterior

$$k \equiv \log_{\alpha} \beta \pmod{p}$$

dados, α , k e p , é fácil encontrar o valor de β . Isso, porque existem vários algoritmos eficientes que determinam o resultado da exponenciação na aritmética modular, como, por exemplo, o algoritmo *square-and-multiply* [Stallings 2007].

Porém, dados α , β e p , encontrar o valor de k pode ser extremamente difícil. Segundo [Stallings 2007] esse cálculo equivale a dificuldade da fatoração de números primos grandes.

Dada essa dificuldade, existem diversas propostas de algoritmos que solucionam o **problema do logaritmo discreto**, porém nenhuma apresenta uma solução eficiente em tempo polinomial para primos na ordem de 1024 bits [Stinson 2005]. Em [Shoup 2008] são apresentados

²Dois inteiros x e y são congruentes módulos p , se e somente, se $(x \bmod p) = (y \bmod p)$. Isso é denotado por $x \equiv y \pmod{p}$ [Mao 2003].

diversos algoritmos que procuram solucionar esse problema. A tabela 3.1 apresenta, resumidamente, alguns algoritmos para o cálculo do **logaritmo discreto** em um grupo multiplicativo $\mathbb{Z}_p^*$ de ordem q , sendo q um divisor de $(p - 1)$.

Algoritmo	Complexidade
Força bruta	$O(q (p)^2)$
<i>Baby step Giant step method</i>	$O(q^{\frac{1}{2}} (q) (p)^2)$
RDL	$O(e (q) (p)^2)$

Tabela 3.1: Lista de algoritmos para cálculo do **logaritmo discreto** e suas respectivas complexidades [Shoup 2008].

Além dos algoritmos anteriores, existem algoritmos mais eficientes como o Pollard rho apresentado em [Pollard 1978]. Esse algoritmo tem aproximadamente complexidade $O(\sqrt{n})$.

Portanto, o uso de **logaritmos discretos** em criptografia baseia-se, principalmente, na dificuldade do seu cálculo [Stinson 2005]. Logo isso permite a construção de mecanismos de criptografia mais seguros.

3.3.4 Inverso Multiplicativo

Nos cálculos presentes no protocolo proposto neste trabalho aparece, com frequência, o cálculo do inverso multiplicativo em um grupo. Dado inteiro $a \in \mathbb{Z}_p^*$, seu inverso multiplicativo é denotado por a^{-1} .

Definição 3.4. (Inverso Multiplicativo) Dado $a \in \mathbb{Z}_p^*$, o inverso multiplicativo de a é um único elemento $a^{-1} \in \mathbb{Z}_p^*$ tal que

$$a * a^{-1} \equiv 1 \pmod{p}.$$

Uma importante propriedade dos grupos multiplicativos $\mathbb{Z}_p^*$ é a seguinte: dado um inteiro a , ele tem inverso multiplicativo em $\mathbb{Z}_p^*$ se, e somente se, o inteiro a é relativamente primo com p , ou seja, $\text{mdc}(p, a) = 1$. A prova dessa propriedade pode ser encontrada em [Stinson 2005].

Além disso, o inverso multiplicativo é obtido eficientemente pelo algoritmo de Euclides estendido apresentado em [Stinson 2005].

O valor de a^{-1} é obtido a partir da seguinte definição [Stinson 2005]:

Definição 3.5. (Cálculo do Inverso Multiplicativo) Dados os inteiros p e a , então existem inteiros r e s tais que [Stinson 2005]:

$$1 = \text{mdc}(p, a) = s * p + r * a$$

Reduzindo a equação anterior módulo p tem-se:

$$r * a \equiv 1 \pmod{p}$$

Nesse caso r é o inverso de a , denotado por a^{-1} , ou seja,

$$a^{-1} * a \equiv 1 \pmod{p}$$

Exemplo 3.3. Suponha $p = 11$ e $a = 3$,

Como o $\text{mdc}(11, 3) = 1$, logo p e a são primos entre si, portanto, existe um inverso multiplicativo de a .

A seguir é descrito os passos para o cálculo de a^{-1} .

O primeiro passo é calcular o resto da divisão³ entre 11 e 3, ou seja,

$$\frac{11}{3} = 3, \text{ resto} = 2$$

que corresponde a,

$$\begin{aligned} 11 &= 3 * 3 + 2 \\ 2 &= 11 - 3 * 3 \end{aligned} \tag{3.1}$$

Como o resto da divisão entre 11 e 3 é diferente de 1, logo é necessário repetir o processo anterior. Porém, nesse caso é calculado o resto da divisão entre o divisor e resto obtidos no passo anterior, ou seja, 3 e 2.

$$\frac{3}{2} = 1, \text{ resto} = 1.$$

que corresponde a:

$$\begin{aligned} 3 &= 1 * 2 + 1 \\ 1 &= 3 - 1 * 2 \end{aligned} \tag{3.2}$$

Como o resto da divisão anterior é igual a 1, o processo é parado.

O próximo passo consiste em substituir a equação 3.1 na equação 3.2, ou seja,

Na equação 3.2 tem-se:

$$1 = 3 * 1 - 1 * 2$$

Substituindo o número 2 da equação 3.2 pela equação 3.1 tem-se:

$$\begin{aligned} 1 &= 3 * 1 - 1 * (1 * 11 - 3 * 3) \\ 1 &= 3 * 1 - 1 * 11 + 3 * 3 \\ 1 &= 3 * 1 + 3 * 3 - 1 * 11 \end{aligned}$$

que equivale a

³O resto da divisão entre dois inteiros a e p , resulta em um quociente q_o e um resto inteiro r , tal que: $r = a - q_o * p$ [Shoup 2008].

$$1 = 4 * 3 - 1 * 11$$

Reduzindo o resultado anterior módulo 11, obtém-se

$$4 * 3 \equiv 1 \pmod{11}$$

Portanto pela definição 3.5 tem-se que o inverso de $a = 3$ módulo 11 é igual a $a^{-1} = 4$.

Agora suponha que $p = 4$ e $a = 2$, nesse caso como a e p não são primos entre si, ou seja, o $\gcd(4, 2) = 2$, logo não existe o inverso multiplicativo de 2 módulo 4.

3.4 Criptografia

A seguir é apresentada a definição formal de um criptosistema, a notação é utilizada segundo [Stinson 2005].

Criptosistema

Um criptosistema é uma quintupla (P, C, K, E, D) que satisfaz as seguintes condições:

1. P é um conjunto finito de textos não cifrados;
2. C é um conjunto finito de textos cifrados;
3. K é um conjunto finito de chaves possíveis;
4. Para cada $k \in K$, existe uma cifragem $e_k \in E$ e uma decifragem $d_k \in D$, tal que:

$$e_k : P \longrightarrow C$$

$$d_k : C \longrightarrow P$$

Dado um texto não cifrado $x \in P$, a cifragem de x é obtido por:

$$e_k(x) = y$$

Para $y \in C$, o texto original $x \in P$ é obtido por:

$$d_k(y) = d_k(e_k(x)) = x$$

O objetivo principal de um criptosistema é possibilitar a comunicação segura entre duas ou mais partes em um canal inseguro. Esse canal pode ser a internet ou linha telefônica, entre outros. Nos exemplos deste trabalho são utilizados os nomes de Alice e Bob para descrever o emissor e o receptor respectivamente, de um processo de comunicação entre duas entidades. Essa comunicação é realizada através de um canal inseguro que possui um espião, denotado por Oscar. O objetivo principal de Oscar é quebrar o sistema, ou seja, descobrir a chave secreta utilizada por Alice ou Bob. A figura 3.5 ilustra basicamente esse processo de comunicação [Stinson 2005].

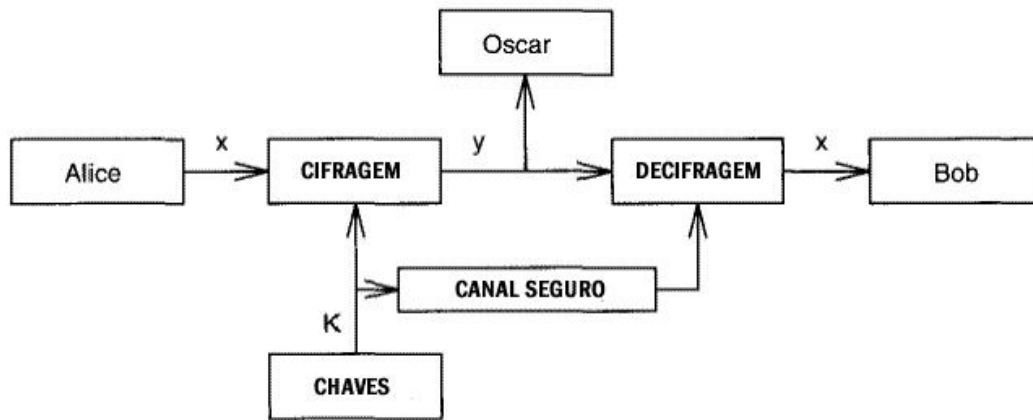


Figura 3.5: comunicação Alice e Bob em um criptosistema [Stinson 2005].

Em criptografia há dois modelos de uso de chaves: chave simétrica e assimétrica. No primeiro, o destinatário e o remetente utilizam a mesma chave para cifragem e decifragem dos dados. A principal vantagem desse modelo é o alto desempenho dos algoritmos. Entretanto, esse modelo apresenta como problema a dificuldade para a troca de chaves. Um estudo dos fundamentos da criptografia simétrica e dos algoritmos existentes é apresentado em [Stinson 2005] e [Stallings 2007].

O segundo modelo é também conhecido como de chave pública. Nesse modelo, os usuários possuem um par de chaves definidos por uma chave pública e por uma privada. Sua principal desvantagem é o desempenho, já que os algoritmos de criptografia de chave pública demandam um maior custo computacional. Um estudo dos fundamentos desse tipo de sistema criptográfico é apresentado em [Stinson 2005], [Mao 2003] e [Stallings 2007].

No contexto desta dissertação, são considerados somente criptosistemas de chave pública. Nas próximas seções são apresentadas as definições e análises do criptosistema e do esquema de assinatura El-Gamal. Esses criptosistemas são utilizados como base teórica para o desenvolvimento do protocolo proposto. Em seguida é apresentada a justificativa de tais escolhas.

3.4.1 Criptosistema El-Gamal

El-Gamal é um criptosistema de chave pública criado em 1984 pelo pesquisador Taher El-Gamal. Esse método consiste na solução do **logaritmo discreto** em um grupo multiplicativo $(\mathbb{Z}_p, \cdot)$, tal que [ElGamal 1985]:

- p é um número primo;
- $\mathbb{Z}_p$ é um grupo multiplicativo de ordem p que possui um elemento gerador α ;

A seguir é apresentada a definição formal do criptosistema El-Gamal segundo a notação utilizada em [Stinson 2005] e [ElGamal 1985].

Criptosistema El-Gamal

Seja p um primo de tal forma que o problema do **logaritmo discreto** em $(\mathbb{Z}_p^*, \cdot)$ é intratável, e $P = \mathbb{Z}_p^*$ o conjunto de textos não cifrados, $C = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ o conjunto dos pares de textos cifrados. Seja,

$$\beta \equiv \alpha^k \pmod{p}$$

e α um elemento gerador de $\mathbb{Z}_p^*$ e b um número aleatório tal que $b \in \mathbb{Z}_{p-1}$.

Considere α, β e p públicos e k a chave secreta.

Dado um texto não cifrado $x \in P$, a cifragem de x obtém um par de textos cifrados (y_1, y_2) .

A cifragem de x utilizando uma chave k é definida por:

$$E(x, b) = (y_1, y_2)$$

$$y_1 = \alpha^b \pmod{p}$$

$$y_2 = x * \beta^b \pmod{p}$$

Para y_1 e $y_2 \in \mathbb{Z}_p^*$, a decifragem é definida por:

$$D_k(y_1, y_2) = y_2 * (y_1^k)^{-1} \pmod{p}$$

Prova: Dado um texto não cifrado x , o resultado da decifragem de $E(x, k)$ é igual a x .

A prova desse fato é dada a seguir. Substituindo y_1 e y_2 em $D(y_1, y_2)$ tem-se:

$$D_k(y_1, y_2) = x * \beta^b * (y_1^k)^{-1} \pmod{p}$$

$$D_k(y_1, y_2) = x * (\alpha^k)^b * (\alpha^{kb})^{-1} \pmod{p}$$

$$D_k(y_1, y_2) = x * \alpha^{bk} * (\alpha^{bk})^{-1} \pmod{p}$$

$$D_k(y_1, y_2) = x * \alpha^{bk - bk} \pmod{p}$$

$$D_k(y_1, y_2) = x * \alpha^0 \pmod{p}$$

Portanto,

$$D_k(y_1, y_2) = x \pmod{p}$$

Exemplo 3.4. O exemplo a seguir considera a cifragem e decifragem de um texto utilizando o criptosistema El-Gamal.

Considere Bob com:

$$p = 2579, \alpha = 2 \text{ e } k = 300.$$

Então, Bob calcula β :

$$\beta = 2^{300} \pmod{2579} = 112$$

Em seguida, Bob publica para Alice os valores de β, α e p .

Suponha que Alice deseja enviar para Bob o texto $x = 1000$. Inicialmente Alice escolhe um número aleatório $b = 800$. Além disso, Alice calcula o par (y_1, y_2) e envia para Bob:

$$y_1 = 2^{800} \bmod 2579 = 523$$
$$y_2 = 100 * 112^{800} \bmod 2579 = 2083$$

Após receber o par (y_1, y_2) , Bob decifra e obtém o valor de x através do cálculo:

$$x = 2083 * (523^{300})^{-1} \bmod 2579 = 1000$$

Propriedades do criptosistema El-Gamal:

O criptosistema El-Gamal é não determinístico, ou seja, dado um mesmo texto não cifrado x , existem diversos pares cifrados (y_1, y_2) diferentes [ElGamal 1985]. Por exemplo, suponha que Alice deseja enviar x para Bob, o resultado é dado por:

$$E_1(x, k) = (y_1, y_2),$$

Além disso, considere que Alice deseja enviar novamente x para Bob, então o valor de $E_2(x, k)$ é dado por:

$$E_2(x, k) = (y_1, y_2).$$

Logo $E_1 \neq E_2$, isso ocorre porque Alice escolhe a cada cifragem um número aleatório b diferente, portanto, um mesmo texto cifrado possui $p - 1$ valores diferentes, pois $b \in \mathbb{Z}_{p-1}$. Essa característica tem como principal vantagem impossibilitar os ataques de texto repetido, aumentando a segurança do criptosistema [Stinson 2005].

A principal desvantagem do criptosistema El-Gamal é o tamanho da chave, conforme dito anteriormente, segundo [Stinson 2005] a chave deve ser no mínimo de 1024 bits para que o **logaritmo discreto** seja intratável.

Além disso, dado um texto x não cifrado, o par cifrado correspondente $E(x, k) = (y_1, y_2)$ tem o dobro do tamanho de x , ou seja, para um texto não cifrado de 1024 bits, o tamanho do texto cifrado correspondente é de 2048 bits. Portanto, esse criptosistema exige um maior custo computacional.

3.4.2 Assinatura digital

Segundo [Stinson 2005] assinatura digital é um método para assinar mensagens armazenadas na forma eletrônica, ou seja, são técnicas de criptografia aplicadas a documentos eletrônicos que equivalem às assinaturas físicas em papéis [Silberschatz et al. 2006]. Para garantir que uma assinatura digital seja válida, são necessárias as seguintes propriedades [Tanenbaum 2003]:

- **Autenticidade:** o receptor deve possuir mecanismos fáceis de verificação da assinatura, para constatar que a assinatura tenha sido realizada pelo verdadeiro emissor.
- **Integridade:** garantir que o conteúdo da mensagem não seja modificado após a assinatura.

- **Não repúdio:** garantir que o emissor da mensagem não possa negar a autenticidade da sua assinatura.

Além disso, um esquema de assinatura é dividido em duas etapas [Stinson 2005]:

- Algoritmo de assinatura;
- Algoritmo de verificação.

Considere inicialmente que Alice deseja assinar uma mensagem x usando sua chave privada k e um algoritmo de assinatura $SIG_K(x)$, o resultado obtido é dado por:

$$SIG_K(x) = y$$

Alice então envia para Bob a mensagem x e sua assinatura, ou seja, o par (x, y) . Após receber o par (x, y) , Bob verifica a assinatura da Alice através do algoritmo de verificação $VER_k(x, y)$ definido a seguir:

$$VER_k(x, y) = \begin{cases} verdadeiro & \text{se } y = SIG_k(x). \\ falso & \text{se } y \neq SIG_k(x). \end{cases}$$

No algoritmo de verificação existem dois resultados possíveis: verdadeiro ou falso. Quando o resultado da verificação é igual à verdadeiro, significa que a assinatura da Alice é válida, caso contrário, a assinatura é inválida.

Segundo a notação apresentada em [Stinson 2005], um esquema de assinatura digital é definido formalmente por:

Assinatura Digital

Uma quintupla (P, A, K, S, V) , tal que:

1. P é um conjunto finito de mensagens.
2. A é um conjunto finito de assinaturas.
3. K é um conjunto finito de chaves.
4. Para cada $k \in K$, existe um algoritmo de assinatura $SIG_K \in S$ e sua correspondente verificação $VER_K \in V$.

Dada uma mensagem $x \in P$, a assinatura de x é definida por:

$$SIG_K(x) = y$$

Para $y \in A$ e $x \in P$, a verificação de uma assinatura é definida por:

$$VER_k(x, y) = \begin{cases} verdadeiro & \text{se } y = SIG_k(x). \\ falso & \text{se } y \neq SIG_k(x). \end{cases}$$

Um esquema de assinatura é seguro se, e somente se, é resistente a qualquer modelo de ataque, ou seja, Oscar não pode forjar uma assinatura válida e se passar por Alice, por exemplo [Stallings 2007]. A seguir são apresentados alguns modelos de ataques [Stinson 2005]:

- **Ataque a chave pública:** Oscar de posse da chave pública de Alice, tenta descobrir uma assinatura válida utilizando o algoritmo de verificação VER_k .
- **Ataque do texto conhecido:** Oscar possui uma lista de mensagens previamente assinadas por Alice e suas correspondentes assinaturas:

$$(x_1, y_1), (x_2, y_2), \dots$$

De posse desta lista, Oscar tenta descobrir a chave privada k de Alice utilizando o algoritmo SIG_k ou construir uma assinatura válida através de VER_k .

- **Força bruta:** Dada uma mensagem x , Oscar testa todas as possibilidades de assinaturas até encontrar um valor $y \in A$, tal que: $VER_k(x, y) = verdadeiro$. Essa forma de ataque é segura se, e somente se, o algoritmo não é quebrado em tempo polinomial.

Esquema de assinatura El-Gamal

É um esquema de assinatura digital de chave pública criado em 1985 e tem como fundamento o criptosistema El-Gamal. Esse esquema de assinatura é não determinístico, ou seja, dada uma mensagem x , existem diversas assinaturas válidas para a mesma mensagem. Além disso, a verificação deve ser capaz de reconhecer essas assinaturas como válidas [ElGamal 1985].

O esquema de assinatura El-Gamal é definido a seguir segundo a notação apresentada em [Stinson 2005] e [ElGamal 1985].

Esquema assinatura El-Gamal (P, A, K, S, V)

Seja p um primo de tal forma que o problema do **logaritmo discreto** em $(\mathbb{Z}_p^*, \cdot)$ é intratável, e $P = \mathbb{Z}_p^*$ o conjunto de mensagens, $A = \mathbb{Z}_p^* \mathbb{Z}_{p-1}$ o conjunto de assinaturas válidas. Seja,

$$\beta \equiv \alpha^k \pmod{p},$$

e α um elemento gerador de $\mathbb{Z}_p^*$ e b um número aleatório tal que $b \in \mathbb{Z}_{p-1}$.

Considere α, β e p públicos e k a chave secreta tal que $k \in K$.

Dada uma mensagem $x \in P$, a assinatura $SIG_k \in S$ de x é definida por:

$$\begin{aligned} SIG_k(x, b) &= (\gamma, \delta) \\ \gamma &= \alpha^b \pmod{p} \\ \delta &= (x - k * \gamma) * b^{-1} \pmod{p-1} \end{aligned}$$

Para $x, \gamma \in \mathbb{Z}^*$ e $\delta \in \mathbb{Z}_{p-1}$ a verificação da assinatura $VER_k \in V$ é definida por:

$$\begin{aligned} VER_k(x, (\gamma, \delta)) &= \text{verdadeiro, se e somente se,} \\ \beta^\gamma \gamma^\delta &\equiv \alpha^x \pmod{p}. \end{aligned}$$

Prova: Dado uma mensagem x , sua assinatura é válida se, e somente se [ElGamal 1985]:

$$\beta^\gamma * \gamma^\delta \equiv \alpha^x \pmod{p}$$

A prova desse fato é dada a seguir. Substituindo β e γ , tem-se:

$$\begin{aligned} \beta^\gamma * \gamma^\delta &\equiv \alpha^{k\gamma} * \alpha^{b\delta} \pmod{p} \\ \alpha^x &\equiv \alpha^{k\gamma} * \alpha^{b\delta} \pmod{p} \end{aligned}$$

Considerando que α é um elemento primitivo módulo p , então a congruência é válida se, e somente se, os expoentes são congruentes módulo $p-1$ [ElGamal 1985], ou seja,

$$x \equiv (k * \gamma + b * \delta) \pmod{p-1}$$

Portanto se a congruência anterior é satisfeita a assinatura de x é válida.

Exemplo 3.5. O exemplo seguinte considera a assinatura de uma mensagem utilizando o esquema de assinatura El-Gamal.

Suponha que Alice tenha: $p = 467, \alpha = 2$ e $k = 300$.

Então Alice calcula β_{Alice} :

$$\beta_{Alice} = 2^{300} \pmod{467} = 317.$$

Em seguida, Alice publica os valores de $\beta_{Alice}, \alpha_{Alice}$ e p_{Alice} .

Considere que Alice deseja assinar a mensagem $x = 100$, primeiramente ela escolhe um número aleatório $b = 139$ e $b^{-1} = 139^{-1} \pmod{466} = 57$. O cálculo de γ e δ é dado por:

$$\gamma = 2^{139} \bmod 467 = 353.$$

$$\delta = (100 - 300 * 353) * 57 \bmod 466 = 372$$

Alice então envia para Bob $(x, (\gamma, \delta))$.

Após receber Bob verifica a assinatura de Alice da seguinte forma:

$$317^{353} * 353^{372} = 2^{100} \bmod 467$$

Portanto se a congruência acima é satisfeita, Bob conclui que a assinatura da Alice é válida.

Propriedades esquema de assinatura El-Gamal

O esquema de assinatura El-Gamal tem as mesmas propriedades do criptosistema apresentado na seção 3.3, além disso, [Tanenbaum 2003] afirma que o algoritmo de verificação desse esquema de assinatura é aproximadamente 10 a 40 vezes mais lento que o RSA.

Até o momento, foram apresentados esquemas de assinaturas sobre toda a mensagem, porém, esses mecanismos tornam o processo de assinatura lento para mensagens grandes. Uma maneira de tornar o processo de assinatura mais eficiente é realizar a assinatura no resumo da mensagem que é bem menor. Esse resumo é obtido através de funções de *hash* [Tanenbaum, 2003].

Um estudo detalhado de assinaturas digitais que utilizam algoritmos de *hash* é apresentado em [Stinson 2005].

No próximo capítulo é definido o protocolo nesta dissertação. Esse protocolo é uma modificação do criptosistema e do esquema de assinatura El-Gamal apresentados neste capítulo. Conforme definido no capítulo 1, o principal problema do contexto apresentado consiste na manipulação dos dados cifrados sem a necessidade de decifrá-los. Diante desse cenário, verificou-se a necessidade da utilização de um criptosistema de chave pública devido a sua estrutura matemática. Dentre os algoritmos existentes, o esquema de assinatura El-Gamal foi o que apresentou as propriedades matemáticas mais adequadas para o contexto desta dissertação. A principal razão para tal escolha, se deve ao elemento gerador definido nesse criptosistema. Através desse elemento é possível obter todos os outros de um grupo multiplicativo. Logo, isso possibilitou criar um mecanismo que verificasse os dados sem a necessidade de conhecê-los.

A seguir é demonstrada a idéia principal e as propriedades matemáticas que possibilitaram essas escolhas.

Inicialmente, considere a definição 3.2.

Seja α um elemento gerador e $\alpha \in \mathbb{Z}_p^*$, logo α gera todos os elementos de $\mathbb{Z}_p^*$, ou seja,

$$\{\alpha^1 \bmod p, \dots, \alpha^{p-1} \bmod p\}$$

Além disso, considere a seguinte propriedade da exponenciação modular [Koblitz 1994] e [Shoup 2008]:

$$\alpha^x * (\alpha^y)^{-1} \bmod p = \alpha^{x-y} \bmod p$$

A partir da fórmula anterior há três casos a considerar:

1. Se $x = y$ então $\alpha^{x-y} \pmod{p} = \alpha^0 \pmod{p} = 1$
2. Se $x > y$ então $\alpha^{x-y} \pmod{p} \in \{\alpha^1 \pmod{p}, \dots, \alpha^{p-1} \pmod{p}\}$
3. Se $x < y$ então $\alpha^{x-y} \pmod{p} \in \{\alpha^{-1} \pmod{p}, \dots, \alpha^{-(p-1)} \pmod{p}\}$

Logo é possível concluir quem é o maior número entre x e y sem a necessidade de conhecê-los.

Dada essa idéia, o algoritmo que apresentou as propriedades mais próximas foi o esquema de assinatura El-Gamal. Ou seja, esse esquema possui o elemento gerador e além disso, utiliza na sua verificação de assinatura exponenciações modulares. Diferentemente do RSA que não possui o elemento gerador. O que impossibilitou a sua escolha.

Portanto o esquema de verificação El-Gamal é o mais adequado dentro do contexto desta dissertação.

A partir da definição do mecanismo de verificação do protocolo proposto, desenvolveu-se uma extensão dos algoritmos de cifragem e decifragem do criptosistema El-Gamal.

A seguir é apresentado um exemplo da idéia inicial mostrada anteriormente.

Exemplo 3.6. Seja $p = 11, \alpha = 2$.

Suponha os números $x = 4$ e $y = 2$, logo,

$$\begin{aligned} 2^4 \pmod{11} &= 5 \\ &\text{e} \\ 2^2 \pmod{11} &= 4 \end{aligned}$$

Considere x e y secretos e 5, 4 e 11 conhecidos. Então é possível verificar se $x > y$ sem conhecer seus valores da seguinte forma:

Inicialmente, tem-se:

$$5 * 4^{-1} \pmod{11} = 5 * 3 \pmod{11} = 4$$

Seja $4 \equiv 2^2 \pmod{p}$, logo $4 \in \{2^1 \pmod{11}, 2^2 \pmod{11}, \dots, 2^{10} \pmod{11}\}$

Portanto, $x > y$, isso ocorre, porque a condição $4 \in \{2^1 \pmod{11}, 2^2 \pmod{11}, \dots, 2^{10} \pmod{11}\}$ é verdadeira.

Os cálculos anteriores apresentam a idéia inicial para a solução do problema, ao longo do capítulo 4, essa idéia é estendida, aprimorada e redefinida.

3.5 Considerações Finais

Neste capítulo foram apresentados o estado da arte e os fundamentos teóricos de computação nas nuvens e criptografia utilizados para a elaboração do novo protocolo. Além disso, foi demonstrado que através das propriedades do elemento gerador e manipulações algébricas é possível comparar dois dados sem conhecê-los. A partir dessa idéia no próximo capítulo é definido o novo protocolo e em seguida é apresentado um protótipo e uma linguagem de consulta em base de dados cifradas.

Capítulo 4

Contribuições, Experimentos e Resultados

Neste capítulo é apresentado um protocolo para consulta a base de dados cifrada. O protocolo proposto manipula dados cifrados em base de dados armazenada remotamente.

O capítulo está estruturado da seguinte forma: Na seção 4.1 é apresentado o novo protocolo proposto e a prova de sua correção. Em seguida, na seção 4.2 é definido o modelo relacional para dados cifrados. Além disso, são apresentados os modelos formais e os algoritmos dos operadores de seleção, inserção, atualização e remoção para dados cifrados.

4.1 Novo Protocolo Proposto

O algoritmo é definido segundo a notação apresentada em [Stinson 2005].

Criptosistema Proposto

Sejam: p um primo grande de tal forma que o problema do **logaritmo discreto** em $(\mathbb{Z}_p, \cdot)$ seja intratável, e $P = \mathbb{Z}_p^*$ o conjunto de textos não cifrados formados com palavras em $\{0, 1\}^*$, $C = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ o conjunto de textos cifrados.

Seja α um elemento gerador de $\mathbb{Z}_p^*$, Nesse caso, $\alpha \in \mathbb{Z}_p^*$, e

$$\mathbb{Z}_p^* = \langle \alpha \rangle = \{\alpha^i : 0 \leq i \leq p-1\}.$$

Portanto, (α) é um grupo cíclico de ordem p .

Seja M um número natural maior que o maior número pertencente à base de dados. O número M deve ser escolhido de tal forma que M é muito menor que $\frac{p}{2}$.

Seja também w um número escolhido de tal forma que $1 < w < M$. A razão dessas escolhas deve-se a questões de complexidade do algoritmo e segurança, que são discutidas adiante.

Seja k a chave privada tal que $k \in K$. Considere, também, que p é público e α, w e k secretos,

Dado um texto não cifrado $x \in P$, a cifragem de x , utilizando a chave secreta k , é definida como:

$$E(x, k) = (\delta, \gamma)$$

na qual:

$$\delta = \alpha^{k+x} \pmod{p}$$

$$\gamma = x * \delta^k \pmod{p}$$

Para δ e $\gamma \in \mathbb{Z}_p^*$, a decifragem, utilizando a chave secreta k , é definida por:

$$D(\delta, \gamma) = \gamma * (\delta^k)^{-1} \pmod{p}$$

Prova: Dado um texto não cifrado x , o resultado da decifragem dado por $E(x, k)$ é igual a x . A prova desse fato é análoga a prova do criptosistema El-Gamal. Substituindo γ e δ em $D(\delta, \gamma)$, tem-se:

$$D(\delta, \gamma) = x * \delta^k * (\delta^k)^{-1} \pmod{p}$$

$$D(\delta, \gamma) = x * \delta^{k-k} \pmod{p}$$

$$D(\delta, \gamma) = x * \delta^0 \pmod{p}$$

Portanto,

$$D(\delta, \gamma) = x \pmod{p}$$

4.2 Modelo Relacional com Dados Cifrados

No protocolo proposto, os dados são armazenados em uma base de dados relacional. As consultas à essa base de dados são efetuadas considerando as consultas da álgebra relacional.

Dado esse contexto de armazenamento e consulta de dados, apresenta-se a seguir os elementos fundamentais do modelo relacional. Em seguida, tal modelo é estendido, tendo como objetivo o armazenamento e consulta a dados cifrados. O resultado de tal extensão é um *modelo relacional com dados cifrados*.

No modelo relacional convencional [Silberschatz et al. 2006] e [Elmasri e Navathe 2000], uma tabela é chamada de relação e cada relação possui um conjunto de linhas chamadas de tuplas.

Uma relação é denotada por $R(A_1, A_2, \dots, A_n)$, onde R é a relação e $A_1, A_2, \dots, A_n$ são os atributos que representam os dados.

Cada atributo pode assumir um conjunto de valores definido como domínio (D).

No *modelo relacional com dados cifrados*, uma relação é denotada por R^C , tal que:

$$R^C(E(A_1, k), E(A_2, k), \dots, E(A_n, k))$$

onde:

$$E(A_1, k), E(A_2, k), \dots, E(A_n, k)$$

representam os atributos cifrados pela função $E(x, k)$ tal que $E(x, k) = (\delta, \gamma)$, e k é uma chave secreta. Além disso, o domínio dos atributos cifrados é o conjunto $\mathbb{Z}_p$.

Um conjunto finito de relações cifradas $R^C = \{R_1^C, \dots, R_n^C\}$ é definido como sendo uma base de dados cifradas.

A figura 4.1 apresenta um exemplo de uma relação não cifrada *funcionario* e seus atributos. Essa relação é utilizada ao longo deste trabalho para exemplificar os operadores da **álgebra relacional para dados cifrados** definida na próxima seção.

<i>Codigo</i>	<i>Nome</i>	<i>Salario</i>	<i>Sexo</i>
1	Beto	900	M
2	Mara	800	F
3	Joao	930	M
4	Silva	830	F
5	Alam	850	M
6	Bob	540	M
7	Adam	950	M
8	Smit	1000	M
9	John	550	M
10	Alex	870	M

Figura 4.1: Relação *funcionario*.

Aplicando na relação anterior o algoritmo de cifragem do protocolo proposto na seção 4.1, suponha, para fins didáticos os valores: $p = 63799$, $k = 3$, $\alpha = 2$, $w = 3$ e $M = 10000$, obtém-se a relação cifrada $funcionario^C$ apresentada na figura 4.4.

Para cifrar a relação *funcionario*, inicialmente, as cadeias de caracteres devem ser transformadas em números. Suponha, por exemplo, que se tenha a correspondência, dada na figura 4.2, entre os nomes de funcionários e sexo e os seus respectivos números. Observe que os números da segunda coluna da tabela a seguir podem ser obtidos considerando qualquer código de conversão de caracteres em números. Como, por exemplo, o código ASCII.

String	Valor
Beto	4245
Mara	4750
Joao	4652
Silva	9088
Alam	4201
Bob	4302
Adam	4205
Smit	9050
John	5256
Alex	4232
M	77
F	70

Figura 4.2: representação numérica das colunas alfanuméricas utilizadas no cálculo da cifragem da relação *funcionario*.

Nesse caso, a cifragem da palavra “Beto”, por exemplo, corresponde a cifragem do número 4245, ou seja, $E(4245, 3)$.

Exemplo 4.1. O processo de cifragem é realizado em cada coluna da relação. Nesse exemplo, a cifragem é feita nas colunas *Codigo*, *Nome*, *Salario* e *Sexo*.

Inicialmente, o cliente cifra o nome da relação *funcionario* e obtém:

$$E(\text{“funcionario”}, k) = \text{funcionario}^C$$

Em seguida, são cifradas as tuplas da relação *funcionario*. Em cada cifragem são utilizadas as funções:

$$\delta = \alpha^{k+x} \pmod{p}$$

e

$$\gamma = x * \delta^k \pmod{p}$$

Então, a cifragem da palavra “*funcionario*”, por exemplo, é dada por:

$$E(\text{“funcionario”}, k)$$

Considerando que a palavra “*funcionario*” corresponda ao número 99, isto é, “*funcionario*” = 99, então,

$$E(\text{“funcionario”}, k) = E(99, 3) = (\delta, \gamma)$$

Onde,

$$\begin{aligned}\delta &= 2^{3+99} \pmod{63799} = 41415 \\ \gamma &= 99 * 41415^3 \pmod{63799} = 44586\end{aligned}$$

Portanto,

$$E(\text{“funcionario”}, k) = (41415, 44586)$$

A cifragem dos atributos da relação *funcionario* são cifrados de forma análoga. A figura 4.3 apresenta os resultados da cifragem dos atributos da relação *funcionario*. Na tabela a seguir, são indicadas as operações para cifragem de cada atributo da tabela *funcionario*. Além disso, ela contém os códigos numéricos correspondentes a cada uma das palavras que nomeiam os atributos. Assim, considera-se, por exemplo, que a palavra “*Nome*” corresponde ao número 425.

Coluna	$\delta = \alpha^{k+x} \pmod{p}$	$\gamma = x * \delta^k \pmod{p}$
“ <i>Codigo</i> ” = 1111	$\delta = 2^{3+1111} \pmod{63799} = 39739$	$\gamma = 1111 * 39739^3 \pmod{63799} = 49717$
“ <i>Nome</i> ” = 425	$\delta = 2^{425+3} \pmod{63799} = 17786$	$\gamma = 425 * 17786^3 \pmod{63799} = 37390$
“ <i>Salario</i> ” = 9000	$\delta = 2^{9000+3} \pmod{63799} = 63248$	$\gamma = 9000 * 63248^3 \pmod{63799} = 15255$
“ <i>Sexo</i> ” = 777	$\delta = 2^{777+3} \pmod{63799} = 5923$	$\gamma = 777 * 5923^3 \pmod{63799} = 7178$

Figura 4.3: Resultado da cifragem das colunas na relação *funcionario*.

A cifragem das tuplas da relação *funcionario* é obtida de forma análoga. A relação cifrada *funcionario*^C = (41415, 44586) é, então, dada pela figura 4.4.

4.2.1 Operadores de Consulta

Considera-se a seguir uma linguagem para consulta à base dados com dados cifrados. Tal linguagem é denominada por **Álgebra relacional para dados cifrados**.

Definição 4.1. (**Álgebra relacional para dados cifrados**) Consiste em um conjunto de operações que usam uma ou duas relações cifradas como entradas e produzem uma nova relação cifrada como resultado.

(39739, 49717)	(17786, 37390)	(63248, 15255)	(5923, 7178)
(16, 4096)	(35468, 31842)	(47636, 41126)	(39236, 6305)
(32, 1737)	(13592, 37108)	(20969, 14370)	(18250, 46155)
(64, 20844)	(51522, 33821)	(60218, 39123)	(39236, 6305)
(128, 30939)	(26235, 7953)	(41055, 63288)	(18250, 46155)
(256, 54194)	(48521, 9983)	(19244, 43657)	(39236, 6305)
(512, 35390)	(10056, 130)	(21049, 11352)	(39236, 6305)
(1024, 32578)	(10748, 49284)	(3288, 58380)	(39236, 6305)
(2048, 42660)	(60552, 38274)	(44965, 10606)	(39236, 6305)
(4096, 1146)	(32492, 25510)	(53913, 35114)	(39236, 6305)
(8192, 31453)	(347, 470)	(2231, 45239)	(39236, 6305)

Figura 4.4: Relação $funcionario^C = (41415, 44586)$ no modelo relacional cifrado.

As operações são as mesmas da álgebra relacional convencional [Guimarães 2005]: seleção, projeção, renomeação, produto cartesiano, união, diferença e junção natural. Além dos operadores de consulta, para extrair os dados, são apresentados os operadores de modificação: inserção, remoção e atualização.

Operador de Projeção(π)

O operador de projeção (π) é um operador unário que extrai alguns atributos de uma relação $R^C = \{E(A_1, k), \dots, E(A_n, k)\}$.

A projeção das colunas $E(A_1, k)$ e $E(A_2, k)$, por exemplo, é denotada por:

$$\pi_{E(A_1, k), \{E(A_2, k)\}} R^C$$

Nesse caso, o resultado obtido pela aplicação do operador é uma relação $R^C' = \{E(A_1, k), E(A_2, k)\}$, tal que o conjunto de atributos $E(A_1, k), E(A_2, k) \in R^C$.

Definida dessa forma, a projeção em uma base de dados cifrada é idêntica a projeção convencional da álgebra relacional.

A seguir, é apresentado um exemplo de utilização do operador de projeção na relação cifrada $funcionario^C$.

Considere que Alice deseja listar todos os códigos e nome dos funcionários cadastrados. Essa consulta é dada por:

$$\pi_{E(Codigo, 3), E(Nome, 3)} (funcionario^C),$$

que corresponde a

$$\pi_{(39739, 49717), (17786, 37390)} (41415, 44586),$$

A relação cifrada resultante dessa consulta é apresentada na figura 4.5:

Da mesma forma, outros operadores da **álgebra relacional para dados cifrados** também são análogos aos operadores utilizados em base de dados convencionais.

A única exceção a essa regra é o operador seleção (σ), que é analisado a seguir.

(39739, 49717)	(17786, 37390)
(16, 4096)	(35468, 31842)
(32, 1737)	(13592, 37108)
(64, 20844)	(51522, 33821)
(128, 30939)	(26235, 7953)
(256, 54194)	(48521, 9983)
(512, 35390)	(10056, 130)
(1024, 32578)	(10748, 49284)
(2048, 42660)	(60552, 38274)
(4096, 1146)	(32492, 25510)
(8192, 31453)	(347, 470)

Figura 4.5: Resultado da projeção na relação *funcionario*^C.

Operador de seleção (σ)

O operador de seleção (σ) é um operador unário que seleciona sequências de tuplas de uma relação R^C que satisfaçam uma determinada condição v . Essa condição é representada por fórmulas booleanas que utilizam os conectivos *e*, *ou* e *não*. O operador de seleção é denotado por:

$$\sigma_v(R^C)$$

No operador de seleção, a condição v é, em geral, uma sentença do tipo:

$$(E(A_j, k) \text{ op } E(q, k))$$

Nessa sentença, **op** é um dos símbolos do conjunto $\{>, <\}$, A_j é um atributo de R^C , k é a chave privada do sistema e q uma consulta realizada pelo usuário, tal que $q \in \mathbb{Z}_p^*$.

No caso de base de dados com dados cifrados, as operações de comparação $\{>, <\}$, não podem ser executadas como no caso convencional. Isso ocorre porque é possível a existência de valores x e y tais que $x < y$, mas $E(x, k) > E(y, k)$. Em outras palavras, as funções de cifragem não são, necessariamente, monotônicas.

A comparação entre os dados cifrados, em uma seleção a uma base de dados com dados cifrados, pode ser feita decifrando os dados e, posteriormente, comparando-os. Infelizmente, tal procedimento mostra quais são os dados armazenados na base de dados, e o objetivo principal desta proposta é apresentar um algoritmo para consulta em base de dados com dados cifrados, sem que seja necessária a decifração de qualquer dado armazenado.

Para simplificar a análise, são consideradas, neste trabalho, apenas seleções $\sigma_v(R^C)$ tais que a condição v é do tipo:

$$E(A_j, k) \text{ op } E(q, k)$$

Tal que *op* é um dos símbolos do conjunto $\{>, <\}$. Além disso, são considerados, também, que os valores dos atributos são apenas números positivos.

A análise do caso mais geral, em que v é uma condição qualquer, como $=, \neq, \leq$ e $\geq$, é análoga e pode ser obtida a partir da análise dos casos em que **op** é $>$ ou $<$. Da mesma forma, o tratamento de números negativos, também é feita analogamente.

Suponha, então, que o cliente deseja realizar uma seleção na base de dados e que o resultado seja formado pelas tuplas que satisfazem a condição:

$$A_j \text{ op } q$$

Nesse caso, se **op** é, por exemplo, o operador $>$, então o resultado da consulta é o conjunto das tuplas, nas quais o atributo A_j é maior que q .

Para efetuar tal consulta, sem decifrar os dados cifrados na base de dados, inicialmente, o cliente cifra q e A_j , utilizando uma chave privada k . O resultado é dado por:

$$E(q, k) \text{ e } E(A_j, k).$$

Mas, como

$$E(q, k) = (\delta_q, \gamma_q) \text{ e } E(A_j, k) = (\delta_{A_j}, \gamma_{A_j}),$$

O cliente calcula dois pares:

$$(\delta_q, \gamma_q) \text{ e } (\delta_{A_j}, \gamma_{A_j})$$

Além disso, o cliente calcula μ_q que é dado por:

$$\mu_q = w * (\delta_q)^{-1} \pmod{p}$$

Em seguida o cliente envia para o servidor os seguintes dados:

$$[\mu_q, \text{op}, (\delta_{A_j}, \gamma_{A_j}), R^C]$$

Após receber a consulta, o servidor identifica a coluna cujo atributo é igual à $(\delta_{A_j}, \gamma_{A_j})$. Para cada tupla i nessa coluna, o servidor extrai o valor armazenado na célula correspondente. Conforme a notação considerada na definição 4.1, tal valor é igual a

$$(\delta_{ij}, \gamma_{ij})$$

Dado o par $(\delta_{ij}, \gamma_{ij})$, obtido da entrada i, j da coluna do atributo $E(A_j, k)$, tal que $E(A_j, k) = (\delta_{A_j}, \gamma_{A_j})$, o servidor utiliza o valor δ_{ij} . Desse valor, o servidor calcula λ_{ij} tal que,

$$\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$$

Observe que:

μ_q corresponde ao valor de referência q , a partir do qual é considerada a comparação efetuada pelo operador **op** e

δ_{ij} corresponde ao j -ésimo atributo da relação R^C na i -ésima tupla.

Dado que o servidor tem armazenado os conjuntos C_1 e C_2 definidos por:

$$\begin{aligned} C_1 &= \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}; \\ &\text{e} \\ C_2 &= \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}; \end{aligned}$$

w deve ser escolhido de tal forma que $C_1 \cap C_2 = \emptyset$ e $1 < w < M$.

Além disso, M é um inteiro muito menor que $\frac{p}{2}$ e maior que o maior valor armazenado na base de dados. As razões para tais condições sobre M e w são determinadas para garantir uma complexidade computacional adequada e uma maior segurança do elemento gerador, como é analisado mais a diante.

Em seguida, o servidor verifica se

$$\begin{aligned} \lambda_{ij} &\in C_1 \\ &\text{ou} \\ \lambda_{ij} &\in C_2 \end{aligned}$$

Tendo calculado λ_{ij} para todo i , o servidor verifica se λ_{ij} satisfaz a condição estabelecida pelo operador op .

Há então dois casos a considerar:

- $op = ">"$
- $op = "<"$

Considere, inicialmente, o caso em que o operador **op** é igual a “menor que”.

- $op = "<"$

Nesse caso, o objetivo do servidor é identificar se o valor armazenado no local ij da tabela é maior que o valor q . Para resolver tal questão, o servidor utiliza a seguinte propriedade.

Propriedade $<$:

Dada uma consulta q cifrada por $E(q, k)$, um atributo A_j cifrado por $E(A_j, k)$ e um par $(\delta_{ij}, \gamma_{ij})$ obtido na tupla i da coluna j , que corresponde ao dado x cifrado então

$$\lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}.$$

se e somente se,

$$q < x$$

Prova:

Inicialmente, observe que:

$$\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$$

$$\lambda_{ij} = \delta_{ij} * w * (\delta_q)^{-1} \pmod{p}$$

$$\lambda_{ij} = w * \alpha^{x+k} * (\alpha^{q+k})^{-1} \pmod{p}$$

$$\lambda_{ij} = w * \alpha^{k-k} * \alpha^{x-q} \pmod{p}$$

$$\lambda_{ij} = w * 1 * \alpha^{x-q} \pmod{p}$$

Logo,

$$\lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}.$$

se, e somente,

$$w * \alpha^{x-q} \pmod{p} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}.$$

Considerando que α é um elemento primitivo modulo p , então, [Stinson 2005],

$$w * \alpha^{x-q} \pmod{p} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}.$$

se, e somente se,

$$x - q \equiv s \pmod{p - 1},$$

para algum s , tal que $s \in [1, M]$. Como $q, x \in [1, M]$, então,

$$x - q = s,$$

Logo,

$$x - q \in [1, M]$$

ou seja,

$$x - q > 0$$

Portanto,

$$q < x$$

- $op = ">"$

Nesse caso, o objetivo do servidor é identificar se o valor armazenado no local ij da relação é menor que o valor q . Para resolver tal questão, o servidor utiliza a seguinte propriedade:

Propriedade >:

Dada uma consulta q cifrada por $E(q, k)$,

um atributo A_j cifrado por $E(A_j, k)$ e um par $(\delta_{ij}, \gamma_{ij})$ obtido na tupla i da coluna j , que corresponde ao dado x cifrado então

$$\lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}.$$

se e somente se,

$$q > x.$$

Prova:

Inicialmente, observe que:

$$\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$$

$$\lambda_{ij} = \delta_{ij} * w * (\delta_q)^{-1} \pmod{p}$$

$$\lambda_{ij} = w * \alpha^{x+k} * (\alpha^{q+k})^{-1} \pmod{p}$$

$$\lambda_{ij} = w * \alpha^{k-k} * \alpha^{x-q} \pmod{p}$$

$$\lambda_{ij} = w * 1 * \alpha^{x-q} \pmod{p}$$

Logo,

$$\lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}.$$

se, e somente se,

$$w * \alpha^{x-q} \pmod{p} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}.$$

Considerando que α é um elemento primitivo modulo p , então, [Stinson 2005],

$$w * \alpha^{x-q} \pmod{p} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}.$$

se, e somente se,

$$x - q \equiv s \pmod{p - 1},$$

para algum s , tal que $s \in [-M, -1]$. Como $q, x \in [1, M]$, então,

$$x - q = s,$$

Logo,

$$x - q \in [-M, -1]$$

ou seja,

$$x - q < 0$$

Portanto,

$$q > x$$

Após a verificação anterior, o servidor retorna ao cliente o conjunto de tuplas que satisfazem **op**. Supondo que cada tupla contém n atributos, os resultados retornados são listas do tipo a seguir, que correspondem às tuplas da tabela R^C que satisfazem à condição da seleção.

$$[(\delta_{i1}, \gamma_{i1}), (\delta_{i2}, \gamma_{i2}), \dots, (\delta_{in}, \gamma_{in}),]$$

Ao receber essa lista, o cliente decifra cada par, utilizando sua chave privada k , da seguinte forma:

$$A_{ij} = D(\delta_{ij}, \gamma_{ij}) = \gamma_{ij} * (\delta_{ij}^k)^{-1} \pmod{p}$$

Observe que A_{ij} é o valor armazenado na j -ésima coluna de R na i -ésima tupla. Dessa forma, o resultado da consulta é dado pela lista:

$$[A_{i1}, A_{i2}, \dots, A_{in}]$$

O protocolo do operador de seleção é apresentado na figura 4.6. Esse protocolo, descreve a operação de seleção com uma condição de comparação do tipo $A_j > q$, como descrita anteriormente.

A análise do operador $<$ no protocolo da figura 4.6 é análoga. A única diferença é no passo 4 (lado servidor). A verificação para esse operador é realizada da seguinte forma:

$$\lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}.$$

O exemplo a seguir considera os passos do protocolo proposto na relação *funcionario^C*.

Cliente	Servidor
Entrada: Chave privada k , Elemento gerador α , w , número primo p , relação $R = \{A_1, A_2, \dots, A_n\}$ condição de seleção $A_j > q$	Entrada: $C = \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}$, número primo p
1. Calcula $E(R, k) = R^C$ $E(q, k) = (\delta_q, \gamma_q)$ $E(A_j, k) = (\delta_{A_j}, \gamma_{A_j})$ $\mu_q = w * (\delta_q)^{-1} \pmod{p}$ 2. Envia para servidor: $[\mu_q, >, (\delta_{A_j}, \gamma_{A_j}), R^C]$ 6. Cliente decifra os pares: $A_{ij} = D(\delta_{ij}, \gamma_{ij})$	3. Calcula para cada tupla i de R^C na coluna $(\delta_{A_j}, \gamma_{A_j})$: $\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$ 4. Se $\lambda_{ij} \in C$ então Retorna $[(\delta_{i1}, \gamma_{i1}), \dots, \delta_{in}, \gamma_{in}]$ 5. Envia: $[(\delta_{i1}, \gamma_{i1}), \dots, \delta_{in}, \gamma_{in}]$

Figura 4.6: Protocolo de seleção com condição $A_j > q$.

Exemplo 4.2. Suponha que Alice deseja selecionar os funcionários com o salário maior que R\$ 800,00 da relação *funcionario*.

Inicialmente, Alice cifra através do algoritmo do protocolo 4.1, o nome da relação e a condição composta pelo nome da coluna e o valor. O resultado de cada cifragem é dado por:

$$E(\text{"funcionario"}, 3) = \text{funcionario}^C$$

$$E(\text{"funcionario"}, 3) = (41415, 44586)$$

$$E(\text{"Salario"}, 3) = (\delta_{sal}, \gamma_{sal})$$

$$E(\text{"Salario"}, 3) = (63248, 15255)$$

Além disso, como o objetivo é selecionar funcionário com salário superior a R\$ 800,00, então o próximo passo é cifrar o número 800.

$$E(800, 3) = \delta_q = 2^{3+800} \pmod{63799} = 20969$$

Além disso, Alice calcula $(\delta_q)^{-1}$ que é dado por:

$$(\delta_q)^{-1} = 50138$$

Finalmente, Alice calcula μ_q que é dado por:

$$\begin{aligned}\mu_q &= w * (\delta_q)^{-1} \pmod{p} \\ \mu_q &= 3 * 50138 \pmod{63799} \\ \mu_q &= 22816\end{aligned}$$

Tendo os valores calculados anteriormente, Alice envia para o servidor a seguinte lista de dados:

$$[\mu_q, >, (\delta_{sal}, \gamma_{sal}), funcionario^C]$$

que é dada por:

$$[22816, >, (63248, 15255), (41415, 44586)]$$

Ao receber essa lista de dados, que corresponde à consulta, o servidor realiza o cálculo de λ_{i3} para cada tupla da relação (41415, 44586), na coluna 3, que corresponde a $(\delta_{sal}, \gamma_{sal}) = (63248, 15255)$, que é dado por:

$$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$$

Tupla	μ_q	δ_{i3}	$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$	Teste se $\lambda_{i3} \in \{3 * 2^1, \dots, 3 * 2^{10000}\}$
1	22816	47636	$\lambda_{i3} = 47011 = 3 * 2^{100}$	verdadeiro
2	22816	20969	$\lambda_{23} = 3 = 3 * 2^0$	falso
3	22816	60218	$\lambda_{33} = 22423 = 3 * 2^{130}$	verdadeiro
4	22816	41055	$\lambda_{43} = 13962 = 3 * 2^{30}$	verdadeiro
5	22816	19244	$\lambda_{53} = 6386 = 3 * 2^{50}$	verdadeiro
6	22816	21049	$\lambda_{63} = 38911 = 3 * 2^{-260}$	Falso
7	22816	3288	$\lambda_{73} = 55183 = 3 * 2^{150}$	verdadeiro
8	22816	44965	$\lambda_{83} = 33520 = 3 * 2^{200}$	verdadeiro
9	22816	53913	$\lambda_{93} = 34288 = 3 * 2^{-250}$	falso
10	22816	2231	$\lambda_{103} = 54693 = 3 * 2^{70}$	verdadeiro

Figura 4.7: Cálculo de λ_{i3} para condição salário > 800,00

A partir do resultado obtido no cálculo anterior, o servidor retorna para Alice as tuplas da relação (41415, 44586), que corresponde a $funcionario^C$, cujo resultado λ_{i3} é tal que:

$$\lambda_{i3} \in \{3 * 2^1, \dots, 3 * 2^{10000}\}.$$

Tais valores são designados pela palavra verdadeiro na última coluna.

Ao receber a lista de resultados do servidor, Alice decifra utilizando sua chave privada k toda a lista recebida da seguinte forma:

<i>Codigo</i>	<i>Nome</i>	<i>Salario</i>	<i>Sexo</i>
1	Beto	900	M
3	Joao	930	M
4	Silva	830	F
5	Alam	850	M
7	Adam	950	M
8	Smit	1000	M
10	Alex	870	M

Figura 4.8: Resultado final consulta $salario > 800,00$ na relação *funcionario*.

$$A_{ij} = D(\delta_{ij}, \gamma_{ij}) = \gamma_{ij} * (\delta_{ij}^k)^{-1} (mod\ p)$$

A relação final não cifrada obtida é representada na figura 4.8:

Operações de Modificação:

O operador de inserção insere uma nova sequência de tuplas em uma relação R^C . A operação de inserção ocorre de forma análoga ao que é considerado na álgebra relacional convencional. A única diferença é que os dados estão cifrados.

A inserção em uma relação cifrada R^C é definida por:

$$R^C \leftarrow R^C \cup [E(D_1, k), ..., E(D_n, k)]$$

Tal que:

- $\leftarrow$ corresponde ao operador de atribuição;
- $\cup$ corresponde ao operador de União;
- $(E(D_1, k), ..., E(D_n, k))$ corresponde a uma tupla cifrada a ser inserida na relação R^C .

Suponha, por exemplo, que o cliente deseja realizar uma inserção da tupla de dados

$$[D_1, D_2, ..D_n]$$

na relação

$$R = \{A_1, A_2, ...A_n\}.$$

Para efetuar tal inserção, a cifragem dos dados em cada coluna da tupla de entrada é executada na máquina do cliente, antes do seu envio ao servidor. Isso, evidentemente, evita que os dados apareçam em sua forma decifrada no servidor. Inicialmente, o cliente cifra o nome da relação R utilizando uma chave privada k , o resultado obtido é dado por:

$$E(R, k) = R^C$$

Em seguida, são cifrados os dados da tupla

$$[D_1, D_2, \dots, D_n]$$

O resultado é a tupla de dados cifrados:

$$[E(D_1, k), E(D_2, k), \dots, E(D_n, k)]$$

Mas, como

$$E(D_n, k) = (\delta_{D_n}, \gamma_{D_n}),$$

O cliente calcula dois pares para cada coluna da tupla cifrada e os envia para o servidor na forma:

$$[R^C, [(\delta_{D_1}, \gamma_{D_1}), (\delta_{D_2}, \gamma_{D_2}), \dots, (\delta_{D_n}, \gamma_{D_n})]],$$

Após receber a lista anterior, o servidor insere a tupla cifrada na relação R^C . Ou seja, a relação cifrada é atualizada na seguinte forma:

$$R^C \leftarrow R^C \cup (E(D_1, k), \dots, E(D_n, k))$$

O protocolo da figura 4.9 descreve a operação de inserção.

Cliente	Servidor
Entrada: Dados: $[D_1, D_2, \dots, D_n]$, Relação $R = \{A_1, A_2, \dots, A_n\}$, Chave privada k , Elemento gerador α , Número primo p ,	
1. Calcula: $E(R, k) = R^C$ $I = [E(D_1, k), \dots, E(D_n, k)]$ 2. Envia: $[R^C, I]$	3. Executa: $R^C \leftarrow R^C \cup I$

Figura 4.9: Protocolo de Inserção em R^C

Exemplo 4.3. Suponha que Alice deseja inserir na relação *funcionario* os seguintes dados:

Codigo = 11, *Nome* = “Joe”, *Salario* = R\$1.500,00 e *Sexo* = “M”. Considere também que o nome “Joe” é representado pelo número 4300.

Inicialmente, Alice cifra o nome da relação *funcionario* utilizando a chave privada $k = 3$, o resultado obtido é dado por:

Coluna	$\delta = \alpha^{k+x} \pmod{p}$	$\gamma = x * \delta^k \pmod{p}$
“11” = 11	$\delta = 2^{3+11} \pmod{63799} = 16384$	$\gamma = 11 * 16384^3 \pmod{63799} = 47110$
“Joe” = 4300	$\delta = 2^{4300+3} \pmod{63799} = 2514$	$\gamma = 4300 * 2514^3 \pmod{63799} = 3463$
“1500” = 1500	$\delta = 2^{1500+3} \pmod{63799} = 19990$	$\gamma = 1500 * 19990^3 \pmod{63799} = 586$
“M” = 77	$\delta = 2^{77+3} \pmod{63799} = 39236$	$\gamma = 77 * 39236^3 \pmod{63799} = 6305$

Figura 4.10: Resultado da cifragem das colunas na relação *funcionario*.

$$E(\text{“funcionario”, } k) = \text{funcionario}^C,$$

$$E(\text{“funcionario”, } 3) = (41415, 44586)$$

Em seguida, Alice cifra o valor de cada coluna da tupla de entrada, conforme a figura 4.10 : Logo após a cifragem das colunas, Alice envia para o servidor a lista:

$$[\text{funcionario}^C, [(\delta_{D_1}, \gamma_{D_1}), (\delta_{D_2}, \gamma_{D_2}), (\gamma_{D_3}, \gamma_{D_3}), (\delta_{D_4}, \gamma_{D_4})]],$$

que é dada por,

$$[(41415, 44586), [(16384, 47110), (2514, 3463), (19990, 586), (39236, 6305)]]$$

Ao receber a lista anterior o servidor insere a tupla cifrada na relação *funcionario*^C.

$$\text{funcionario}^C \leftarrow \text{funcionario}^C \cup$$

$$[(16384, 47110), (2514, 3463), (19990, 586), (39236, 6305)]$$

A relação cifrada resultante é dada pela figura 4.11 :

(39739, 49717)	(17786, 37390)	(63248, 15255)	(5923, 7178)
(16, 4096)	(35468, 31842)	(47636, 41126)	(39236, 6305)
(32, 1737)	(13592, 37108)	(20969, 14370)	(18250, 46155)
(64, 20844)	(51522, 33821)	(60218, 39123)	(39236, 6305)
(128, 30939)	(26235, 7953)	(41055, 63288)	(18250, 46155)
(256, 54194)	(48521, 9983)	(19244, 43657)	(39236, 6305)
(512, 35390)	(10056, 130)	(21049, 11352)	(39236, 6305)
(1024, 32578)	(10748, 49284)	(3288, 58380)	(39236, 6305)
(2048, 42660)	(60552, 38274)	(44965, 10606)	(39236, 6305)
(4096, 1146)	(32492, 25510)	(53913, 35114)	(39236, 6305)
(8192, 31453)	(347, 470)	(2231, 45239)	(39236, 6305)
(16384, 47110)	(2514, 3463)	(19990, 586)	(39236, 6305)

Figura 4.11: Relação *funcionario*^C após a inserção do novo funcionário

Operação de Remoção

O operador de remoção exclui uma ou mais tuplas de uma relação *R*^C. A remoção é realizada de forma quase idêntica a uma seleção. Ou seja, removem-se as tuplas selecionadas pelo cliente através de uma consulta *Q*. A remoção é definida por:

$$R^C \leftarrow R^C - Q$$

Tal que:

Q é uma consulta da **álgebra relacional para dados cifrados** que satisfaz uma condição:

$$A_j \text{ op } q$$

Tal que:

A_j é a j -ésima coluna da relação R ,

op é um dos símbolos do conjunto $\{>, <\}$,

q é o valor da condição.

Suponha, por exemplo, que o cliente deseja remover as tuplas da relação R^C que satisfaz a seguinte condição:

$$A_j > q$$

Inicialmente, o cliente cifra o nome da relação R , da condição composta pelo nome da coluna e o valor utilizando uma chave privada k , o resultado de cada cifragem obtido é dado por:

$$\begin{aligned} E(R, k) &= R^C \\ E(A_j, k) &= (\delta_{A_j}, \gamma_{A_j}) \\ E(q, k) &= (\delta_q, \gamma_q) \end{aligned}$$

Além disso, o cliente calcula:

$$\mu_q = w * (\delta_q)^{-1} \pmod{p}$$

Em seguida, o cliente envia para o servidor a lista:

$$[\mu_q, >, (\delta_{A_j}, \gamma_{A_j}), R^C]$$

Ao receber a lista anterior, o servidor realiza o cálculo de λ_{ij} para cada linha i da relação R^C na coluna cujo atributo é igual a $(\delta_{A_j}, \gamma_{A_j})$. Supõe-se que seja a coluna j .

$$\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$$

A partir do resultado obtido no cálculo anterior, o servidor verifica a condição na linha i :

$$\text{Se } \lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}$$

Então a i -ésima linha de R^C é removida, ou seja,

$$R^C \leftarrow R^C - R_i^C$$

Cliente	Servidor
Entrada: Chave privada k , Elemento gerador α , w , número primo p , relação $R = \{A_1, A_2, \dots, A_n\}$ condição de seleção $A_j > q$	Entrada: $C = \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}$, número primo p
1. Calcula $E(R, k) = R^C$ $E(q, k) = (\delta_q, \gamma_q)$ $E(A_j, k) = (\delta_{A_j}, \gamma_{A_j})$ $\mu_q = w * (\delta_q)^{-1} \pmod{p}$ 2. Envia para servidor: $[\mu_q, >, (\delta_{A_j}, \gamma_{A_j}), R^C]$	3. Calcula para cada tupla i de R^C na coluna $(\delta_{A_j}, \gamma_{A_j})$: $\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$ 4. Se $\lambda_{ij} \in C$ então $R^C \leftarrow R^C - R_i^C$

Figura 4.12: Protocolo de remoção com condição $A_j > q$.

A figura 4.12 descreve o protocolo do operador de remoção.

A análise do operador $<$ é análoga. A única diferença é no passo 4 (lado servidor). A verificação para esse operador é realizada da seguinte forma:

$$\text{Se } \lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}$$

Exemplo 4.4. Suponha que Alice deseja remover os funcionários com salários menores que R\$600,00.

Inicialmente, Alice cifra através do algoritmo do protocolo 4.1, o nome da relação e a condição composta pelo nome da coluna e o valor. O resultado de cada cifragem é dado por:

$$\begin{aligned} E(\text{"funcionario"}, k) &= \text{funcionario}^C, \\ E(\text{"funcionario"}, 3) &= (41415, 44586), \\ E(\text{"Salario"}, k) &= (\delta_{sal}, \gamma_{sal}), \\ E(\text{"Salario"}, 3) &= (63248, 15255), \end{aligned}$$

Além disso, como o objetivo é selecionar funcionário com salário inferior a R\$600,00, então o próximo passo é cifrar o número 600.

$$E(600, 3) = 2^{600+3} \bmod 63799 = 52204.$$

Além disso, Alice calcula $(\delta_q)^{-1}$ que é dado por:

$$(\delta_q)^{-1} = 10834.$$

Finalmente, Alice calcula μ_q que é dado por:

$$\begin{aligned}\mu_q &= w * (\delta_q)^{-1} \bmod p \\ \mu_q &= 3 * 10834 \bmod 63799 \\ \mu_q &= 32502\end{aligned}$$

Em seguida, Alice envia para o servidor a lista:

$$[\mu_q, <, (\delta_{sal}, \gamma_{sal}), funcionario^C]$$

que é igual a

$$[32502, <, (63248, 15255), (41415, 44586)]$$

Ao receber a lista anterior, o servidor realiza o cálculo de λ_{i3} para cada linha da relação (41415, 44586), que corresponde à relação $funcionario^C$, na coluna correspondente a (63248, 15255). Nesse caso, tem-se a coluna 3 da relação (41415, 44586), mostrada na figura 4.4.

O cálculo de λ_{i3} é mostrado pela figura 4.13:

Tupla	μ_q	δ_{i3}	$\lambda_{i3} = \delta_{i3} * \mu_q \bmod p$	Teste se $\lambda_{i3} \in \{3 * 2^{-1}, \dots, 3 * 2^{-10000}\}$
1	32502	47636	$\lambda_{13} = 54939 = 3 * 2^{300}$	falso
2	32502	20969	$\lambda_{23} = 33520 = 3 * 2^{200}$	falso
3	32502	60218	$\lambda_{33} = 43513 = 3 * 2^{330}$	falso
4	32502	41055	$\lambda_{43} = 13525 = 3 * 2^{230}$	falso
5	32502	19244	$\lambda_{53} = 46891 = 3 * 2^{250}$	falso
6	32502	21049	$\lambda_{63} = 17921 = 3 * 2^{-60}$	verdadeiro
7	32502	3288	$\lambda_{73} = 3251 = 3 * 2^{350}$	falso
8	32502	44965	$\lambda_{83} = 8737 = 3 * 2^{400}$	falso
9	32502	53913	$\lambda_{93} = 40791 = 3 * 2^{-50}$	verdadeiro
10	32502	2231	$\lambda_{103} = 36298 = 3 * 2^{270}$	falso

Figura 4.13: Cálculo de λ_{i3} para condição salário < 600,00

A partir do resultado obtido no cálculo anterior, o servidor remove as tuplas da relação (41415, 44586), cujo resultado de λ_{i3} é tal que

$$\lambda_{i3} \in \{3 * 2^{-1}, \dots, 3 * 2^{-10000}\}.$$

Tais valores são designados pela palavra verdade na última coluna. Nesse exemplo, as tuplas removidas são 6 e 9.

A relação resultante (41415, 44586), que corresponde a $funcionario^C$, é apresentada na figura 4.14 :

(39739, 49717)	(17786, 37390)	(63248, 15255)	(5923, 7178)
(16, 4096)	(35468, 31842)	(47636, 41126)	(39236, 6305)
(32, 1737)	(13592, 37108)	(20969, 14370)	(18250, 46155)
(64, 20844)	(51522, 33821)	(60218, 39123)	(39236, 6305)
(128, 30939)	(26235, 7953)	(41055, 63288)	(18250, 46155)
(256, 54194)	(48521, 9983)	(19244, 43657)	(39236, 6305)
(1024, 32578)	(10748, 49284)	(3288, 58380)	(39236, 6305)
(2048, 42660)	(60552, 38274)	(44965, 10606)	(39236, 6305)
(8192, 31453)	(347, 470)	(2231, 45239)	(39236, 6305)

Figura 4.14: Relação (41415, 44586) resultante após a operação de remoção.

Operação de atualização de um único atributo

O operador de atualização modifica um ou mais dados em uma relação cifrada R^C . A atualização é feita de forma análoga a da álgebra relacional convencional.

Suponha, por exemplo, que o cliente deseja atualizar na relação

$$R = \{A_1, A_2, \dots, A_j, \dots, A_n\}$$

o valor do atributo A_j . Nesse caso, se $A_j < q$, então o valor de A_j é atualizado para, por exemplo, por $A_j = A_j + y$. Assim, a atualização é efetuada conforme a uma condição v do tipo:

$$A_j < q$$

Uma atualização como essa é efetuada por uma sequência de operações de seleção, remoção e inserção.

Inicialmente, o cliente cifra o nome da relação R , da condição composta pelo nome da coluna e o valor utilizando sua chave privada k , o resultado de cada cifragem obtido é dado por:

$$\begin{aligned} E(R, k) &= R^C \\ E(A_j, k) &= (\delta_{A_j}, \gamma_{A_j}) \\ E(q, k) &= (\delta_q, \gamma_q) \end{aligned}$$

Além disso, o cliente calcula $(\delta_q)^{-1}$ e μ_q

$$\mu_q = w * (\delta_q)^{-1} \pmod{p}$$

Em seguida, Alice então envia para o servidor a lista.

$$[\mu_q, <, (\delta_{A_j}, \gamma_{A_j}), R^C]$$

Ao receber a lista anterior, o servidor realiza o cálculo de λ_{ij} para cada linha i da relação R^C na coluna j .

$$\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$$

A partir do resultado obtido no cálculo anterior, o servidor verifica a condição na linha i e coluna j :

$$\text{Se } \lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}$$

Então a i -ésima linha de R^C é inserida em uma relação temporária $R^{C'}$, ou seja,

$$R^{C'} \leftarrow R^C \cup R_i^C$$

Em seguida, o servidor envia para o cliente a relação $R^{C'}$. Essa relação contém todos os atributos da relação R , ou seja,

$$[(\delta_{A_1}, \gamma_{A_1}), (\delta_{A_2}, \gamma_{A_2}), \dots, (\delta_{A_j}, \gamma_{A_j}), \dots, (\delta_{A_n}, \gamma_{A_n})]$$

Ao receber a lista anterior, o cliente seleciona a coluna $(\delta_{A_j}, \gamma_{A_j})$ para atualizar. O objetivo do cliente é acrescentar um número y ao valor de A_{ij} que corresponde ao valor armazenado na i -ésima linha e j -ésima coluna de R^C , ou seja,

$$A_{ij} = A_{ij} + y.$$

A cifragem de $A_{ij} + y$ para cada linha i de $R^{C'}$ na j -ésima coluna é dado por $(\delta'_{ij}, \gamma'_{ij})$:

$$\begin{aligned} \delta'_{ij} &= \alpha^y * \delta_{ij} \pmod{p} \\ &\text{e} \\ \gamma'_{ij} &= y * (\delta'_{ij})^k + \alpha^{yk} * \gamma_{ij} \pmod{p} \end{aligned}$$

Tal que:

y é o número acrescentado ao valor da coluna j ,

$(\delta_{ij}, \gamma_{ij})$ é o valor cifrado armazenado na i -ésima linha e j -ésima coluna da relação R ,

α é o elemento gerador do protocolo,

k é a chave privada do cliente.

A prova de que $(\delta'_{ij}, \gamma'_{ij})$ corresponde a cifragem $E(A_{ij} + y, k) = (\delta_{A_{ij}+y}, \gamma_{A_{ij}+y})$ é apresentada a seguir:

Prova: Pela definição tem-se:

$$\delta'_{ij} = \alpha^y * \delta_{ij} \pmod{p}$$

que corresponde à

$$\begin{aligned} \delta'_{ij} &= \alpha^y * \alpha^{A_{ij}+k} \pmod{p} \\ \delta'_{ij} &= \alpha^{y+A_{ij}+k} \pmod{p} \\ \delta'_{ij} &= \delta_{A_{ij}+y} \end{aligned}$$

Já o valor de γ'_{ij} é definido por:

$$\gamma'_{ij} = y * (\delta'_{ij})^k + \alpha^{yk} * \gamma_{ij} \pmod{p}$$

que corresponde à

$$\begin{aligned}\gamma'_{ij} &= y * (\delta'_{ij})^k + \alpha^{yk} * A_{ij} * \delta_{ij}^k \pmod{p} \\ \gamma'_{ij} &= y * (\delta'_{ij})^k + \alpha^{yk} * A_{ij} * (\alpha^{A_{ij}+k})^k \pmod{p} \\ \gamma'_{ij} &= y * (\delta'_{ij})^k + A_{ij} * (\alpha^y * \alpha^{A_{ij}+k})^k \pmod{p} \\ \gamma'_{ij} &= y * (\delta'_{ij})^k + A_{ij} * (\alpha^{y+A_{ij}+k})^k \pmod{p} \\ \gamma'_{ij} &= y * (\delta'_{ij})^k + A_{ij} * (\delta'_{ij})^k \pmod{p}\end{aligned}$$

que é dado por:

$$\gamma'_{ij} = (y + A_{ij}) * (\delta'_{ij})^k \pmod{p} = \gamma_{A_{ij}+y}$$

Observe que o calculo de $(\delta'_{ij}, \gamma'_{ij})$ é realizado sem a necessidade de decifrar os valores armazenados na coluna j , ou seja, a operação de soma é realizada com os dados na sua forma cifrada.

Após o cálculo anterior, o cliente armazena o resultado em uma nova relação cifrada definida por $R^{C''}$. Em seguida, o cliente envia essa relação para o servidor. A relação $R^{C''}$ contém todos os atributos da relação R , entretanto a coluna j está atualizada com os valores $(\delta'_{Aj}, \gamma'_{Aj})$ calculados pelo cliente, conforme a lista a seguir.

$$[(\delta_{A_1}, \gamma_{A_1}), (\delta_{A_2}, \gamma_{A_2}), ..., (\delta'_{A_j}, \gamma'_{A_j}), ..., (\delta_{A_n}, \gamma_{A_n})]$$

Ao receber a lista anterior, o servidor inicialmente remove as tuplas alteradas pelo cliente, ou seja, as tuplas pertencentes à relação R^C que satisfizeram a condição q da seguinte forma:

$$R^C \leftarrow R^C - R^C,$$

Em seguida, o servidor insere na relação R^C resultante a lista $R^{C''}$ enviada pelo cliente. A relação $R^{C''}$ contém todas as tuplas atualizadas pelo cliente. Essa inserção é dada por:

$$R^C \leftarrow R^C \cup R^{C''}$$

A figura 4.15 descreve o protocolo da operação de atualização.

A análise do operador $>$ é análoga. A única diferença é no passo 4 (lado servidor). A verificação para esse operador é realizada da seguinte forma:

$$\text{Se } \lambda_{ij} \in \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [1, M]\}$$

Exemplo 4.5. Suponha que Alice deseja dar um aumento de R\$100,00 para quem ganha um salário inferior à R\$700,00.

Inicialmente, Alice cifra usando-se do algoritmo do protocolo 4.1, o nome da relação e a condição composta pelo nome da coluna e o valor. O resultado de cada cifragem é dado por:

Cliente	Servidor
Entrada: Chave privada k , Elemento gerador α , w , número primo p , relação $R = \{A_1, A_2, \dots, A_n\}$ condição de atualização $A_j < q$ Novo valor de $A_j = A_j + y$	Entrada: $C = \{w * \alpha^s \pmod{p}, \text{ tal que } s \in [-M, -1]\}$, número primo p
1. Calcula $E(R, k) = R^C$ $E(q, k) = (\delta_q, \gamma_q)$ $E(A_j, k) = (\delta_{A_j}, \gamma_{A_j})$ $\mu_q = w * (\delta_q)^{-1} \pmod{p}$ 2. Envia para servidor: $[\mu_q, >, (\delta_{A_j}, \gamma_{A_j}), R^C]$ 6. Atualiza a i -ésima tupla de R^C na coluna $(\delta_{A_j}, \gamma_{A_j})$: $\delta'_{ij} = \alpha^y * \delta_{ij} \pmod{p}$ $\gamma'_{ij} = y * (\delta'_{ij})^k + \alpha^{yk} * \gamma_{ij} \pmod{p}$ 7. Envia tuplas atualizadas $R^{C'}$: $[(\delta_{A_1}, \gamma_{A_1}), \dots, (\delta_{A_j}, \gamma_{A_j}), \dots, (\delta_{A_n}, \gamma_{A_n})]$	3. Calcula para cada tupla i de R^C na coluna $(\delta_{A_j}, \gamma_{A_j})$: $\lambda_{ij} = \delta_{ij} * \mu_q \pmod{p}$ 4. Se $\lambda_{ij} \in C$ então $R^{C'} \leftarrow R^C \cup R_i^C$ 5. Envia $R^{C'}$: $[(\delta_{A_1}, \gamma_{A_1}), \dots, (\delta_{A_j}, \gamma_{A_j}), \dots, (\delta_{A_n}, \gamma_{A_n})]$ 8. Remove tuplas modificadas $R^C \leftarrow R^C - R^{C'}$ 9. Insere tuplas alteradas $R^C \leftarrow R^C \cup R^{C'}$

Figura 4.15: Protocolo operação de atualização para uma condição $A_j < q$.

$$\begin{aligned}
E(\text{"funcionario"}, k) &= \text{funcionario}^C, \\
E(\text{"funcionario"}, 3) &= (41415, 44586), \\
E(\text{"Salario"}, k) &= (\delta_{sal}, \gamma_{sal}),
\end{aligned}$$

$$E("Salario", k) = (63248, 15255) .$$

Além disso, como o objetivo é selecionar funcionário com salário inferior a R\$700, 00, então o próximo passo é cifrar o número 700.

$$E(q, k) = E(700, 3) , \delta_q = 2^{700+3} \bmod 63799 = 2037.$$

Além disso, Alice calcula $(\delta_q)^{-1} = 15754$ e μ_q

$$\begin{aligned} \mu_q &= w * (\delta_q)^{-1} \pmod{p} \\ \mu_q &= 3 * 15754 \pmod{63799} = 47262 \end{aligned}$$

Em seguida, Alice envia para o servidor a lista:

$$[\mu_q, <, (\delta_{sal}, \gamma_{sal}), funcionario^C]$$

que é igual a,

$$[47262, <, (63248, 15255), (41415, 44586)]$$

Ao receber a lista anterior, o servidor realiza o calculo de λ_{i3} para cada linha da relação (41415, 44586), que corresponde a relação $funcionario^C$, na coluna correspondente a (63248, 15255). Nesse caso, tem-se a coluna 3 da relação (41415, 44586), mostrada na 4.4.

O cálculo de λ_{i3} é mostrado na figura 4.16:

Tupla	μ_q	δ_{i3}	$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$	Teste se $\lambda_{i3} \in \{3 * 2^{-1}, ..., 3 * 2^{-10000}\}$
1	47262	47636	$\lambda_{13} = 33520 = 3 * 2^{200}$	falso
2	47262	20969	$\lambda_{23} = 47011 = 3 * 2^{100}$	falso
3	47262	60218	$\lambda_{33} = 13525 = 3 * 2^{230}$	falso
4	47262	41055	$\lambda_{43} = 22423 = 3 * 2^{130}$	falso
5	47262	19244	$\lambda_{53} = 55183 = 3 * 2^{150}$	falso
6	47262	21049	$\lambda_{63} = 31 = 3 * 2^{-160}$	verdadeiro
7	47262	3288	$\lambda_{73} = 46891 = 3 * 2^{250}$	falso
8	47262	44965	$\lambda_{83} = 54939 = 3 * 2^{300}$	falso
9	47262	53913	$\lambda_{93} = 31744 = 3 * 2^{-150}$	verdadeiro
10	47262	2231	$\lambda_{103} = 45574 = 3 * 2^{170}$	falso

Figura 4.16: Cálculo de λ_{i3} para condição salário < 700

A partir do resultado obtido na figura 4.16, o servidor armazena em uma relação temporária $funcionario^C$ as tuplas da relação (41415, 44586), que corresponde a $funcionario^C$, cujo resultado λ_{i3} é tal que:

$$\lambda_{i3} \in \{3 * 2^{-1}, ..., 3 * 2^{-10000}\} .$$

Tupla	$\delta'_{sal} = \alpha^y * \delta_{sal} \pmod{p}$	$\gamma'_{ij} = y * (\delta'_{sal})^k + \alpha^{yk} * \gamma_{sal} \pmod{p}$
6	$\delta'_{sal} = 2^{100} * 21049 \pmod{63799} = 46549$	$\gamma'_{ij} = 100 * 46549^3 + 2^{100*3} * 11352 \pmod{63799} = 45055$
9	$\delta'_{sal} = 2^{100} * 53913 \pmod{63799} = 8323$	$\gamma'_{ij} = 100 * 8323 + 2100 * 3 * 35114 \pmod{63799} = 3245$

Figura 4.17: Resultado atualização dos salários nas tuplas 6 e 9.

Tais valores são designados pela palavra verdadeiro na última coluna. Nesse exemplo, as tuplas inseridas em $funcionario^C$ são 6 e 9.

Em seguida, o servidor envia para Alice as tuplas de $funcionario^C$.

Ao receber a relação $funcionario^C$, Alice calcula o novo salário $(\delta'_{sal}, \gamma'_{sal})$ conforme mostrado na figura 4.17:

Em seguida, Alice constrói a relação $funcionario^{C''}$ com os salários atualizados conforme a figura 4.18. Em seguida, Alice envia $funcionario^{C''}$ para o servidor.

(512, 35390)	(10056, 130)	(46549, 45055)	(39236, 6305)
(4096, 1146)	(32492, 25510)	(8323, 3245)	(39236, 6305)

Figura 4.18: Relação $funcionario^{C''}$ contendo novos salários.

Observe na figura 4.18 que foi atualizado somente o valor da coluna 3, que corresponde à coluna de salário dos funcionários.

Ao receber a lista anterior, inicialmente o servidor remove as tuplas 6 e 9 na relação (41415, 44586).

Em seguida, o servidor insere na relação (41415, 44586), as tuplas de $funcionario^{C''}$, ou seja,

$$funcionario^C \leftarrow funcionario^C \cup [(512, 35390), (10056, 130), (46549, 45055), (39236, 6305)]$$

e

$$funcionario^C \leftarrow funcionario^C \cup [(4096, 1146), (32492, 25510), (8323, 3245), (39236, 6305)]$$

A relação (41415, 44586) resultante é apresentada na figura 4.19 :

(39739, 49717)	(17786, 37390)	(63248, 15255)	(5923, 7178)
(16, 4096)	(35468, 31842)	(47636, 41126)	(39236, 6305)
(32, 1737)	(13592, 37108)	(20969, 14370)	(18250, 46155)
(64, 20844)	(51522, 33821)	(60218, 39123)	(39236, 6305)
(128, 30939)	(26235, 7953)	(41055, 63288)	(18250, 46155)
(256, 54194)	(48521, 9983)	(19244, 43657)	(39236, 6305)
(512, 35390)	(10056, 130)	(46549, 45055)	(39236, 6305)
(1024, 32578)	(10748, 49284)	(3288, 58380)	(39236, 6305)
(2048, 42660)	(60552, 38274)	(44965, 10606)	(39236, 6305)
(4096, 1146)	(32492, 25510)	(8323, 3245)	(39236, 6305)
(8192, 31453)	(347, 470)	(2231, 45239)	(39236, 6305)

Figura 4.19: Relação (41415, 44586) resultante após operação de atualização.

4.3 Considerações Finais

Neste capítulo foi definido o novo protocolo que consulta e modifica base de dados cifrada, cujo principal objetivo é realizar seleções nessas bases sem a necessidade de decifrar os dados no servidor. Além disso, foi definida uma linguagem de consulta à essas bases denominada por **álgebra relacional para dados cifrados**. No caso do operador de seleção foram consideradas somente consultas com operadores $\{<, >\}$. Os demais operadores são obtidos a partir destes. Também foi proposto um protocolo para os operadores remoção, inserção e atualização.

O próximo capítulo apresenta uma análise da segurança desse protocolo e os experimentos realizados para verificar a correção e o desempenho do protocolo.

Capítulo 5

Resultados e Análises

O capítulo está estruturado da seguinte forma: Na seção 5.1 é analisada a segurança do protocolo e os possíveis modelos de ataques. Já na seção 5.2, são descritos os passos utilizados para a construção do protótipo e os parâmetros utilizados nos experimentos. Por fim, apresentam-se os resultados dos experimentos e uma análise desses resultados.

5.1 Segurança do Protocolo

Nessa seção é apresentada uma análise da segurança do protocolo proposto.

Como no criptosistema El-Gamal a segurança de cada passo equivale à dificuldade de resolução do problema do **logaritmo discreto**.

A seguir, são analisados alguns tipos de ataques ao protocolo proposto. Para analisar cada caso, é utilizada a seguinte notação:

- Oscar é o espião que deseja quebrar o protocolo e roubar os dados da base de dados;
- Alice é a cliente que possui e utiliza uma base de dados cifrada nas nuvens.

Pela definição 4.1 do protocolo proposto, tem-se que os valores de p e M são públicos e k , w e α são privados. Nesse contexto, o principal objetivo é assegurar que Oscar não descubra os valores da chave privada e do elemento gerador utilizado pela Alice.

A seguir, são apresentados os dois principais problemas envolvidos na segurança do protocolo proposto.

Problema do Logaritmo Discreto (PLD)

Seja $\mathbb{Z}_p^*$ um grupo multiplicativo de ordem p , α o elemento gerador de $\mathbb{Z}_p^*$ e $\beta \in \mathbb{Z}_p^*$ [Mao 2003].

Encontre um inteiro k , $0 \leq k \leq p - 1$, tal que:

$$\beta = \alpha^k \pmod{p}$$

ou seja,

$$k = \log_{\alpha} \beta \pmod{p}$$

Se o **PLD** tem solução em tempo polinomial, então o protocolo proposto pode ser quebrado. Nesse caso, suponha, então, que o **PLD** tem solução.

No protocolo, se Oscar conhece os elementos α, β e p , então ele determina a chave privada k da seguinte forma:

$$k = \log_{\alpha} \beta \pmod{p}$$

Portanto, nesse caso, o protocolo, como o criptosistema El-Gamal, seria quebrado.

Entretanto, no protocolo, Oscar não conhece α , o que dificulta a descoberta da chave privada, mesmo tendo a solução do **PLD**.

Além do **PLD**, a segurança do protocolo é baseada no problema da **Determinação de um elemento gerador em $\mathbb{Z}_p^*$** definido a seguir.

Determinação de um elemento gerador (DEG) em $\mathbb{Z}_p^*$

Seja p um número primo público, e $\mathbb{Z}_p^*$ um grupo multiplicativo de ordem p que possui n elementos geradores $\{\alpha_1, \alpha_2, \dots, \alpha_n\} \in \mathbb{Z}_p^*$.

Suponha que seja escolhido, aleatoriamente, um elemento gerador α_i no conjunto de elementos geradores de $\mathbb{Z}_p^*$.

Construa então os conjuntos C_1 e C_2 , tal que:

$$C_1 = \{w * \alpha_i^1 \pmod{p}, w * \alpha_i^2 \pmod{p}, \dots, w * \alpha_i^M \pmod{p}\}$$

$$C_2 = \{w * \alpha_i^{-1} \pmod{p}, w * \alpha_i^{-2} \pmod{p}, \dots, w * \alpha_i^{-M} \pmod{p}\}$$

w deve ser escolhido de tal forma que $C_1 \cap C_2 = \emptyset$ e $1 < w < M$.

O problema da determinação de um elemento gerador em $\mathbb{Z}_p^*$, **DEG** corresponde a:

Dados C_1, C_2, p e M , determinar α .

Se **DEG** tem solução em tempo polinomial, então o protocolo proposto pode ser quebrado.

Além disso, o valor de w é utilizado nos conjuntos C_1 e C_2 para aumentar a segurança da **DEG**.

Considere que os conjuntos C_1 e C_2 não utilizem o valor de w , nesse caso, os conjuntos C_1 e C_2 são definidos por:

$$C_1 = \{\alpha_i^1 \pmod{p}, \alpha_i^2 \pmod{p}, \dots, \alpha_i^M \pmod{p}\}$$

e

$$C_2 = \{\alpha_i^{-1} \pmod{p}, \alpha_i^{-2} \pmod{p}, \dots, \alpha_i^{-M} \pmod{p}\}$$

Observe que no conjunto C_1 o elemento α^1 corresponde ao elemento gerador α utilizado por Alice, nesse caso, dado que Oscar sabe que o elemento gerador $\alpha \in C_1$. Então ele descobre por força bruta o elemento gerador. Para aumentar a segurança do **DEG**, Alice multiplica cada elemento dos conjuntos C_1 e C_2 por um número w secreto, ou seja,

$$C_1 = \{w * \alpha_i^1 \pmod{p}, w * \alpha_i^2 \pmod{p}, \dots, w * \alpha_i^M \pmod{p}\}$$

e

$$C_2 = \{w * \alpha_i^{-1} \pmod{p}, w * \alpha_i^{-2} \pmod{p}, \dots, w * \alpha_i^{-M} \pmod{p}\}$$

Logo, Oscar não consegue descobrir diretamente o valor de α . Nesse caso, suponha, então, que a **DEG** tem solução.

No protocolo, se Oscar conhece os elementos α, w, C_1, C_2, p e M . Então ele pode decifrar os dados da relação cifrada por força bruta, utilizando informações cifradas fornecidas por Alice.

Considere, por exemplo, que Alice submeta na relação *funcionario* da figura 4.4 uma consulta de seleção do tipo:

“Selecione os funcionários com o salário inferior à R\$ 1.000,00.”

Inicialmente, Alice cifra através do algoritmo do protocolo 4.1, o nome da relação e a condição composta pelo nome da coluna e o valor. O resultado de cada cifragem é dado por:

$$E(\text{“funcionario”}, 3) = \text{funcionario}^C$$

$$E(\text{“funcionario”}, 3) = (41415, 44586)$$

$$E(\text{“Salario”}, 3) = (\delta_{sal}, \gamma_{sal})$$

$$E(\text{“Salario”}, 3) = (63248, 15255)$$

Além disso, como o objetivo é selecionar funcionário com salário inferior à R\$1.000,00, então o próximo passo é cifrar o número 1000.

$$E(1000, 3) = \delta_q = 2^{3+1000} \pmod{63799} = 44965$$

Além disso, Alice calcula $(\delta_q)^{-1}$ que é dado por:

$$(\delta_q)^{-1} = 52258$$

Finalmente, Alice calcula μ_q que é dado por:

$$\begin{aligned}\mu_q &= w * (\delta_q)^{-1} \pmod{p} \\ \mu_q &= 3 * 52258 \pmod{63799} \\ \mu_q &= 29176\end{aligned}$$

Tendo os valores calculados anteriormente, Alice envia para o servidor a seguinte lista de dados:

$$[\mu_q, <, (\delta_{sal}, \gamma_{sal}), funcionario^C]$$

que é dada por:

$$[29176, <, (63248, 15255), (41415, 44586)]$$

Ao receber essa lista de dados, que corresponde à consulta, o servidor realiza o cálculo de λ_{i3} para cada tupla da relação (41415, 44586), na coluna (63248, 15255), que é dado por:

$$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$$

O cálculo de λ_{i3} é mostrado na figura 5.1 :

Tupla	μ_q	δ_{i3}	$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$	Teste se $\lambda_{i3} \in \{3 * 2^{-1}, ..., 3 * 2^{-10000}\}$
1	29176	47636	$\lambda_{13} = 30520 = 3 * 2^{-100}$	verdadeiro
2	29176	20969	$\lambda_{23} = 22933 = 3 * 2^{-200}$	verdadeiro
3	29176	60218	$\lambda_{33} = 23506 = 3 * 2^{-70}$	verdadeiro
4	29176	41055	$\lambda_{43} = 58254 = 3 * 2^{-170}$	verdadeiro
5	29176	19244	$\lambda_{53} = 31744 = 3 * 2^{-150}$	verdadeiro
6	29176	21049	$\lambda_{63} = 60249 = 3 * 2^{-560}$	verdadeiro
7	29176	3288	$\lambda_{73} = 40791 = 3 * 2^{-50}$	verdadeiro
8	29176	44965	$\lambda_{83} = 3 = 3 * 2^0$	falso
9	29176	53913	$\lambda_{93} = 1343 = 3 * 2^{-450}$	verdadeiro
10	29176	2231	$\lambda_{103} = 16676 = 3 * 2^{-130}$	verdadeiro

Figura 5.1: Cálculo de λ_{i3} para condição salário < 1000.

A partir do cálculo de λ_{i3} da figura 5.1, Oscar decifra por força bruta o valor armazenado da coluna (63248, 15255) na relação cifrada como se segue.

Na linha 1, por exemplo, Oscar sabe que $\lambda_{i3} \in \{3 * 2^{-1}, ..., 3 * 2^{-10000}\}$.

A partir disso, Oscar descobre por força bruta que:

$$\lambda_{i3} = 30520,$$

Corresponde ao elemento do conjunto

$$3 * 2^{-100} \pmod{63799}$$

Além disso, Oscar sabe que $q = 1000$ e que λ_{i3} é definido por:

$$\lambda_{i3} = \delta_{i3} * \mu_q \pmod{p}$$

Em seguida, ele substitui os valores na equação anterior obtendo,

$$30520 = 2^{k+x} * w * (2^{k+1000})^{-1} \pmod{p}$$

que é igual à

$$\begin{aligned} 3 * 2^{-100} &= 3 * 2^{x-1000} * 2^{k-k} \pmod{p} \\ 3 * 2^{-100} &= 3 * 2^{x-1000} \pmod{p} \end{aligned}$$

Oscar então decifra o valor x armazenado na relação cifrada, ou seja, x é igual à:

$$x = 900$$

Portanto, nesse caso, o protocolo proposto é quebrado. Além disso, observe que Oscar decifrou o valor da relação sem descobrir o valor da chave privada da Alice.

A decifragem das outras tuplas e colunas são análogas ao processo anterior.

Porém, Oscar só conseguiu quebrar o protocolo de posse do elemento gerador α e w , por isso, esses elementos devem ser secretos para evitar que Oscar decifre a base de dados conforme explicado anteriormente.

Segundo [Shoup 2008] não existe algoritmo em tempo polinomial que determine todos os elementos geradores de $\mathbb{Z}_p^*$. Além disso, Oscar também precisa descobrir dentre todos os geradores de $\mathbb{Z}_p^*$ qual é o utilizado por Alice.

Logo, o protocolo é seguro, se e somente se, o elemento gerador α é secreto, ou seja, se **DEG** não tem solução.

Ataque do texto conhecido:

No ataque do texto conhecido, Oscar conhece um conjunto X_i de textos cifrados por Alice e seus valores cifrados correspondentes, ou seja, (δ_i, γ_i) , a partir disso, Oscar monta a seguinte lista:

$$[(x_1, (\delta_1, \gamma_1)), (x_2, (\delta_2, \gamma_2)), \dots, (x_n, (\delta_n, \gamma_n))]$$

A partir desta lista, Oscar tenta deduzir a chave privada k da Alice.

Pela definição 4.1 tem-se:

$$\delta_i = \alpha_i^{x_i+k} \pmod{p}$$

Então a chave privada k é calculada por:

$$k = (\log_{\alpha} \delta_i) - x_i \pmod{p}$$

Dado que α é secreto, para Oscar determinar o valor de k é necessario que ele resolva o **PLD**. Nesse sentido, a segurança do protocolo equivale à segurança do sistema criptografico El-Gamal. E como no caso do sistema El-Gamal, o protocolo é seguro para primos maiores que 1024 bits [Stinson 2005].

Além dos problemas apresentados anteriormente, a segurança e o custo computacional do protocolo está relacionado ao número M definido no protocolo.

A seguir é mostrado porque esse número foi definido de tal forma que M é muito menor que $\frac{p}{2}$.

Porque M deve ser muito menor que $\frac{p}{2}$?

Dado um número p , então,

$$\lambda \in \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

se, e somente se, $\exists s$ tal que

$$\lambda = w * \alpha^s \pmod{p}$$

Encontrar o valor de s , que satisfaz a igualdade anterior, equivale a resolver o **PLD** em $\mathbb{Z}_p$.

Seja M o maior número armazenado na base de dados. No protocolo, considera-se, então, o conjunto

$$\{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

no lugar do conjunto

$$\{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^P \pmod{p}\},$$

E como, em geral, M não é um número grande, sendo muito menor que p , determinar se

$$\lambda \in \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

não equivale a resolver o **PLD** em $\mathbb{Z}_p$. Isso, porque se M é um número pequeno, tal cálculo pode ser feito por força bruta, comparando λ com todos os elementos do conjunto:

$$\{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

Entretanto, tal comparação não seria possível se fosse considerado o conjunto:

$$\{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^P \pmod{p}\},$$

que é um conjunto muito maior que o anterior.

Observe que tal fato considera que os números armazenados na base de dados devem ser muito menores que p , para que o teste acima seja possível.

Portanto, dado que M é muito menor que p , responder se

$$\lambda \in \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

ou

$$\lambda \notin \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

não equivale ao **PLD**.

Por outro lado, se Oscar sabe que

$$\lambda \in \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\},$$

e, além disso, ele conhece w e α , então ele pode determinar, por força bruta, o valor de s tal que

$$\lambda = w * \alpha^s \pmod{p}$$

Para isso, basta Oscar testar a igualdade

$$\lambda = w * \alpha^i \pmod{p}$$

para $0 \leq i \leq M$.

Entretanto no protocolo, w e α são secretos. Nesse caso, Oscar não consegue descobrir o valor de s , portanto o protocolo é seguro.

A seguir é apresentado um exemplo e a análise de segurança do conjunto valor de M e dos conjuntos.

Exemplo 5.1. Seja $p = 11$, $\alpha = 2$, $w = 3$ e $M = 4$,

Nesse caso o conjunto C_1 é dado por:

$$C_1 = \{3 * 2^1 \pmod{11}, \dots, 3 * 2^4 \pmod{11}\}$$

que corresponde a

$$C_1 = \{6, 1, 2, 4\}$$

Suponha que Oscar conheça $\lambda = 6$, ele tenta descobrir o valor de s tal que:

$$6 = w * \alpha^s \pmod{11}$$

Entretanto Oscar só conseguiria encontrar o valor de s se e somente se ele conseguir resolver o **PLD**. Nesse caso, conforme apresentado anteriormente o **PLD** não tem solução em tempo polinomial para números grandes.

Além disso, como α e w são secretos, logo o cálculo de s se torna ainda mais difícil.

Então se p é grande e w e α secretos, Oscar não consegue encontrar o valor de s . Portanto, o protocolo é seguro.

5.2 Descrição do Experimento

5.2.1 Arquitetura do protótipo

A figura 5.2 mostra a arquitetura do protótipo proposto que é considerada como base para os experimentos. Existem dois aplicativos nesse contexto: cliente e servidor.

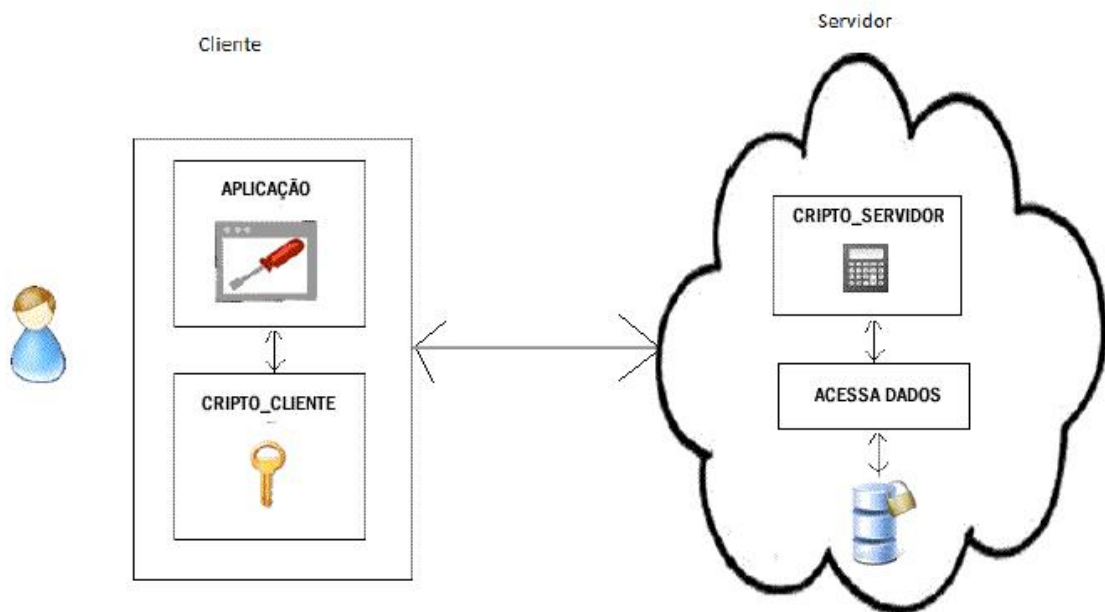


Figura 5.2: Arquitetura protótipo.

No cliente o aplicativo é composto de duas camadas: **Aplicação** e **Cripto_Cliente**.

- **Camada de Aplicação:** Essa camada contém as interfaces que o cliente utiliza para interagir com a base de dados. Essa interação é realizada através de operações SQL informadas pelo cliente. Além disso, essa camada comunica diretamente com a **Cripto_Cliente** durante as operações de cifragem e decifragem. Na primeira, essa camada envia para a **Cripto_Cliente** os comandos e os dados informados para serem cifrados com a chave privada do cliente. No processo de decifragem, essa camada recebe os dados decifrados da **Cripto_Cliente** e os apresenta para o cliente.
- **Cripto_Cliente:** Sua principal função é realizar a cifragem e decifragem dos dados da aplicação. Além disso, ela é responsável pelo armazenamento da chave privada e do elemento gerador secreto do cliente. Inicialmente, os dados são passados pela **camada de Aplicação** e em seguida são cifrados através do algoritmo de cifragem proposto no capítulo 4. No término da operação, os dados cifrados são retornados para a **camada de Aplicação** que os envia para o servidor. O processo de decifragem funciona de forma análoga. Porém nesse caso, o algoritmo utilizado é o de decifragem. No final desse processo os dados decifrados são retornados para a **camada de Aplicação** que exibe os resultados para o cliente.

Por outro lado, o servidor é composto também por duas camadas: **Cripto_Servidor** e **Acessa Dados**.

- **Cripto_Servidor:** Esta Camada é responsável pela execução dos algoritmos do protocolo. Além disso, ela interage diretamente com a camada **Acessa Dados** da seguinte forma: suponha, por exemplo, que o servidor receba uma consulta cifrada do cliente que contenha uma condição com os operadores “<” ou “>”. Inicialmente, a camada **Acessa Dados** seleciona todas as tuplas da relação cifrada armazenadas no SGBD. Nesse momento, o resultado é passado para a camada **Cripto_Servidor**, em seguida, essa camada executa o algoritmo de seleção do protocolo. No término desse processo, a camada **Cripto_Servidor** retorna o resultado final cifrado para o cliente.
- **Acessa Dados:** Essa camada é responsável pela comunicação entre a aplicação cliente e o SGBD hospedado no servidor. Além disso, essa camada pode interagir ou não com a camada **Cripto_Servidor**. Suponha, por exemplo, que o cliente envie uma consulta para o servidor com uma condição, tal que o operador seja um símbolo do conjunto {>, <}. Nesse caso, a camada **Acessa Dados** seleciona os dados da relação e em seguida envia-os para a camada **Cripto_Servidor**.

Por outro lado, quando a condição de uma consulta enviada pelo cliente não tem operadores do tipo {>, <}, então a consulta é realizada diretamente no SGBD, ou seja, nesses casos, não é necessário interagir com a camada **Cripto_Servidor**.

No contexto dessa dissertação os mecanismos de distribuição de chave e do elemento gerador não são discutidos. Nesse caso, considera-se que a máquina do cliente é segura e a entidade que distribui esses elementos é confiável. Além disso, considere também que os conjuntos

$$C_1 = \{w * \alpha_i^1 (mod p), w * \alpha_i^2 (mod p), ..., w * \alpha_i^M (mod p)\}$$

e

$$C_2 = \{w * \alpha_i^{-1} (mod p), w * \alpha_i^{-2} (mod p), ..., w * \alpha_i^{-M} (mod p)\}$$

estão armazenados no servidor.

Na próxima seção é apresentado um exemplo de aplicação do protocolo.

5.2.2 Implementação do protótipo

Inicialmente são definidos os parâmetros do protocolo para a realização dos experimentos, a tabela 5.1 lista os parâmetros utilizados nos exemplos seguintes.

A seguir são apresentadas as principais telas do sistema:

Criação de uma relação cifrada

Na proposta desta dissertação, o nome da relação e das colunas é cifrado pelo protocolo da seção 4.1. Diante desse contexto, ao criar uma relação na base de dados, é necessário inicialmente cifrar o nome da relação e de suas colunas. Porém, nos SGBDs comerciais existe uma

Parâmetro = Valor
$p = 293448575788863462715091072085551541344427287085466935973301370036268383$ $79628743262776820272485451816135560327117589384391159544517959836640159845$ $33500127425773072729145776827651498457638498325758889254764525318620469617$ $99694156511567208687830719493846339784321282027482427818911013609402438747$ 6042795908339 (1025 bits)
$k = 151291675077828606419116784338775515196040717797474920387406079163944318$ $01976948186208016249699384480087961836682955432830665824306502557859493815$ $82051582819835874198972442229376583077653761030865182022901070969217980246$ $95363597773232365460126932288679797325976898002403053880072065199283806313$ 4482824621219 (1024 bits)
$\alpha = 2$
$M = 823267680598805892446185398465219829389943907573001095252543163269870948$ $29090551635400720713621685476436523013616289870387058722548660581701771980$ 7483317 (512 bits)
$w = 3$

Tabela 5.1: Parâmetros para testes do protocolo

limitação da quantidade de caracteres para a definição dos nomes das relações e colunas. O *SQL Server 2008*, por exemplo, utiliza como tamanho máximo 128 caracteres.

Nesse caso, por exemplo, se o cliente utilizar uma chave de 1024 bits, não será possível cifrar o nome da relação e de suas colunas, pois cada número cifrado corresponde a um par (δ, γ) cujo tamanho é de 2048 bits que equivale a 256 caracteres no código ASCII, por exemplo.

Diante desse contexto, foi necessário desenvolver no protótipo um mecanismo para contornar essa limitação. Esse mecanismo consiste em armazenar separadamente os nomes das relações e colunas cifradas. Na próxima seção esses mecanismos são detalhados. Além disso, foi necessário criar um novo tipo de dado para armazenar os pares (δ, γ) na base de dados cifrada. Esse tipo de dado é denominado por **Cripto**. A figura 5.3 mostra a criação do novo tipo de dado no *SQL Server 2008*.

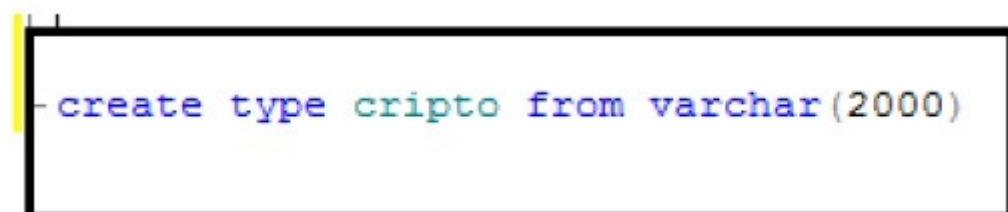


Figura 5.3: Exemplo de criação do tipo de dados **Cripto** no *SQL Server 2008*.

O tipo de dados **Cripto** tem como base o tipo de dados *varchar*, tal escolha se deve pela facilidade na manipulação de *strings* e na integração com o tipo de dados *BigInteger* utilizado na aplicação. Seu tamanho varia de acordo com o tamanho da chave.

Além disso, foi utilizada a técnica *Base64* para representar os dados cifrados. A razão para tal escolha se deve pela eficiência dessa representação. Segundo [Knudsen 1998] a técnica *Base64* é mais rápida que a representação hexadecimal. O apêndice B apresenta uma breve definição dessa técnica.

A figura 5.4 apresenta um exemplo de uma relação cifrada armazenada utilizando a representação *Base64*.

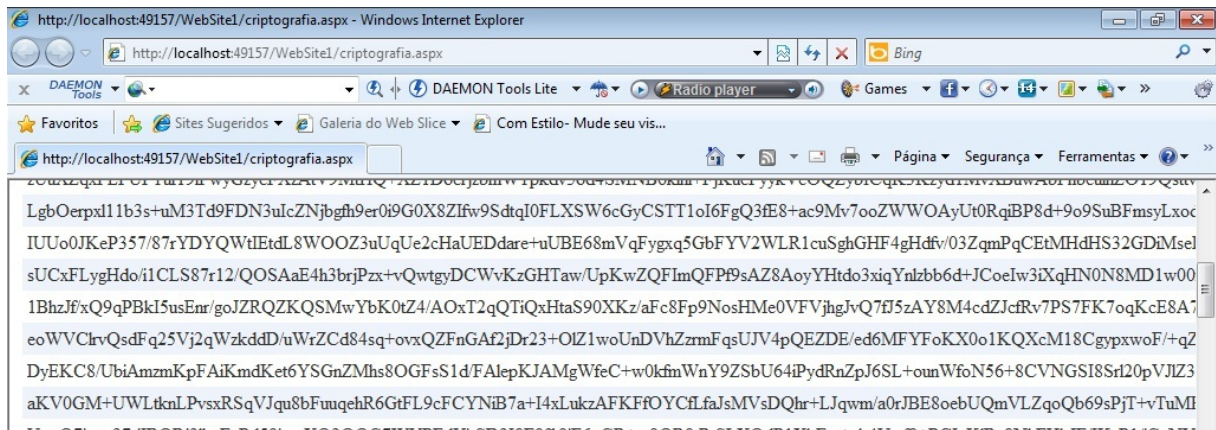


Figura 5.4: Exemplo de relação cifrada utilizando representação *Base64*.

A figura 5.5 mostra a tela do protótipo onde o cliente cria uma nova relação na base de dados.

MÓDULO CRIAÇÃO RELAÇÃO CIFRADA

Nome Relação:	funcionarios
Coluna 1:	codigo
Coluna 2:	nome
Coluna 3:	salario
Coluna 4:	estado_civil

Figura 5.5: Tela do protótipo onde o cliente cria uma nova relação.

Na figura 5.5, o cliente informa o nome da relação e os seus respectivos atributos. Em seguida, o cliente clica no botão “Criar Relação”. Nesse momento, a aplicação do cliente cifra o nome da relação e de seus atributos. Nesse exemplo, cada cifragem é dada por:

$$E(\text{"funcionarios"}, k) = \text{funcionarios}^C,$$

$$\begin{aligned} E(\text{"codigo"}, k) &= (\delta_{cod}, \gamma_{cod}), \\ E(\text{"nome"}, k) &= (\delta_{nome}, \gamma_{nome}), \\ E(\text{"salario"}, k) &= (\delta_{sal}, \gamma_{sal}), \\ E(\text{"estado_civil"}, k) &= (\delta_{est}, \gamma_{est}) \end{aligned}$$

Em seguida, o aplicativo cliente envia para o servidor a lista seguinte:

$$[funcionarios^C, (\delta_{cod}, \gamma_{cod}), (\delta_{nome}, \gamma_{nome}), (\delta_{sal}, \gamma_{sal}), (\delta_{est}, \gamma_{est})]$$

Ao receber a lista anterior o servidor insere cada par (δ, γ) na relação **TbProtocolo**. Essa relação é responsável pelo armazenamento dos nomes das relações e colunas cifradas.

Essa relação é definida por:

$$\mathbf{TbProtocolo} = \{\mathbf{idTabela}, \mathbf{nomeTabela}, \mathbf{idColuna}, \mathbf{nomeColuna}\}$$

Nesse exemplo, a relação **TbProtocolo** resultante após a inserção da relação $funcionarios^C$ e seus atributos é mostrada na figura 5.6.

idTabela	nomeTabela	idColuna	nomeColuna
t_1	$funcionarios^C$	c_1	$(\delta_{cod}, \gamma_{cod})$
t_1	$funcionarios^C$	c_2	$(\delta_{nome}, \gamma_{nome})$
t_1	$funcionarios^C$	c_3	$(\delta_{sal}, \gamma_{sal})$
t_1	$funcionarios^C$	c_4	$(\delta_{est}, \gamma_{est})$

Figura 5.6: Inserção da relação $funcionarios^C$ e seus atributos na relação **TbProtocolo**.

Após a inserção anterior, o servidor cria uma nova relação na base de dados da seguinte forma:

$$\textit{Create table } t_1 (c_1 \textit{ cripto}, c_2 \textit{ cripto}, c_3 \textit{ cripto}, c_4 \textit{ cripto})$$

Observe que o nome da nova relação t_1 é obtida a partir do atributo **idTabela**. Além disso, os nomes das colunas c_1 , c_2 , c_3 e c_4 são obtidos da mesma forma, ou seja, a partir do atributo **idColuna** da relação **TbProtocolo**.

Interface de comandos SQL

O cliente interage com a base de dados armazenada na nuvem através do “Módulo Consulta Base de Dados Cifrada”, a figura 5.7 mostra a interface utilizada pelo cliente.

Observe na figura 5.7 que o cliente utiliza a linguagem SQL padrão para selecionar os registros na relação $funcionarios$. A consulta utilizada na figura 5.7 é análoga a uma consulta convencional para dados não cifrados. Ao clicar no botão “Executar”, a **camada de Aplicação** envia para a camada **Cripto_Cliente** os seguintes dados: $funcionarios$, $salario$ e 3000. Em seguida, a camada **Cripto_Cliente** cifra esses dados utilizando a chave privada k do cliente da seguinte forma:

MÓDULO CONSULTA BASE DE DADOS CIFRADA

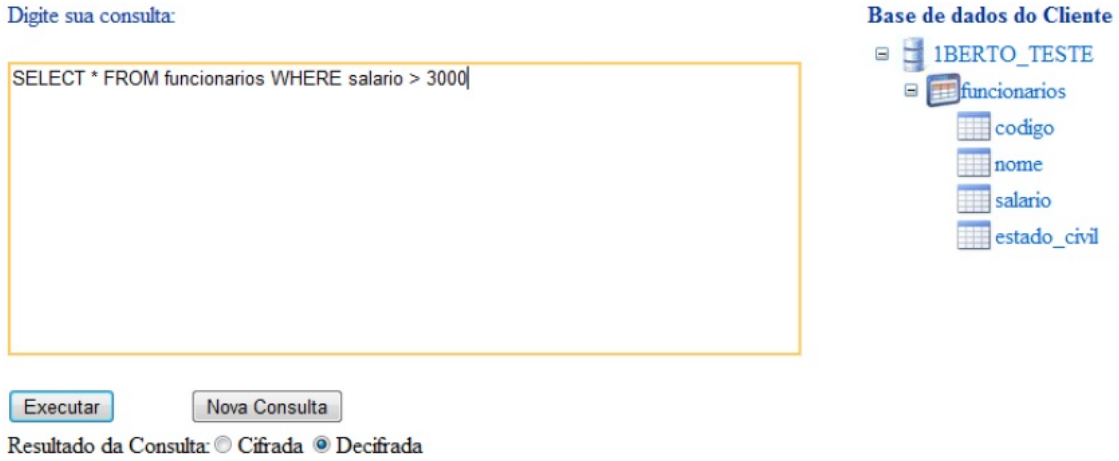


Figura 5.7: Interface da aplicação cliente para consultas SQL.

$$\begin{aligned} E(\text{"funcionarios"}, k) &= \text{funcionarios}^C, \\ E(\text{"salario"}, k) &= (\delta_{sal}, \gamma_{sal}), \\ E(3000, k) &= (\delta_q, \gamma_q), \end{aligned}$$

Além disso, a camada **Cripto_Cliente** calcula $(\delta_q)^{-1}$ e μ_q que é dado por:

$$\mu_q = w * (\delta_q)^{-1} \pmod{p}$$

No fim da cifragem, os dados criptografados são retornados para a **camada de Aplicação** que envia para o servidor a lista seguinte:

$$[\mu_q, >, (\delta_{sal}, \gamma_{sal}), \text{funcionarios}^C]$$

Ao receber a lista de dados anterior, o servidor faz o mapeamento da relação funcionarios^C e da coluna $(\delta_{sal}, \gamma_{sal})$. O resultado obtido do mapeamento é a relação t_1 e a coluna c_3 respectivamente, conforme apresentado na figura 5.6 da seção anterior. Esse processo é realizado pela camada **Acessa Dados**.

Em seguida, o servidor busca todos os dados na relação t_1 e retorna o resultado para a camada **Cripto_Servidor**. A partir desses dados, a camada **Cripto_Servidor** realiza o cálculo de λ para cada tupla da relação t_1 na coluna c_3 , conforme o cálculo seguinte:

$$\lambda_{ic_3} = \delta_{ic_3} * \mu_q \pmod{p}$$

Em seguida, o servidor verifica a condição:

$$\lambda_{ic_3} \in \{w * \alpha^1 \pmod{p}, w * \alpha^2 \pmod{p}, \dots, w * \alpha^M \pmod{p}\}$$

Após a verificação anterior, o servidor retorna ao cliente o conjunto de tuplas que satisfazem a condição da seleção,

$$[(\delta_{cod}, \gamma_{cod}), (\delta_{nome}, \gamma_{nome}), (\delta_{sal}, \gamma_{sal}), (\delta_{est}, \gamma_{est})]$$

Ao receber essa lista, o cliente decifra cada par na camada **Cripto_Cliente**, utilizando sua chave privada k , da seguinte forma:

$$A_{ij} = D((\delta_{ij}, \gamma_{ij})) = \gamma_{ij} * (\delta_{ij}^k)^{-1} \pmod{p}$$

Observe que A_{ij} é o valor armazenado na j -ésima coluna de funcionarios na i -ésima tupla. No fim do processo, o resultado da consulta é dado pela lista

$$[codigo, nome, salario, estado_civil]$$

A lista anterior contém o resultado da consulta do cliente decifrado. Em seguida, essa lista é enviada para **camada de Aplicação** que apresenta o resultado decifrado para o cliente.

A figura 5.8 mostra uma parte do resultado final obtido após o processamento da consulta.

Resultado da Consulta: ☐ Cifrada ☒ Decifrada

Resultado Consulta = 177 registro(s).

codigo	nome	salario	estado_civil
364	Abra	4606	Divorciado
103	Ivor	3216	solteiro
160	Jackson	4279	Casado
394	Abra	3739	Divorciado
163	Acton	3581	Viuvo
178	Acton	4921	Viuvo
186	Aidan	4648	solteiro
57	Aidan	3462	Viuvo
339	Alika	3420	solteiro

Figura 5.8: Resultado da consulta na interface do “módulo consulta no cliente”.

5.2.3 Esquema de Validação

Com o objetivo de verificar o desempenho do protocolo proposto foram utilizadas duas métricas, tempo de execução e tamanho da base de dados. O primeiro experimento consiste em uma consulta cifrada na relação *funcionarios* representada na figura 5.9.

<i>Codigo</i>
<i>Nome</i>
<i>Salario</i>
<i>Estado_Civil</i>

Figura 5.9: Relação *funcionarios*.

Para avaliar os resultados foram utilizados cinco cenários distintos. No primeiro a relação *funcionarios* possui 400 registros. Já nos outros cenários são utilizados: 6.400, 10.050, 80.400 e 643.200 registros respectivamente. Os dados utilizados nos experimentos foram obtidos usando-se softwares geradores de dados para testes que podem ser obtidos em [Benjaminkeen 2001].

A consulta a seguir é considerada nos experimentos para avaliação da métrica de tempo.

“Select * from *funcionarios* where *salario* > 3000”

O primeiro experimento compara o tempo de uma consulta SQL padrão para dados não cifrados com o tempo de execução de uma consulta com os dados cifrados. No primeiro caso, foi executada a consulta convencional em uma relação não cifrada e em seguida foi verificado o tempo gasto para a execução da mesma.

Já no segundo caso, o tempo total da consulta depende do protocolo implementado na camada **Cripto_Servidor**. Inicialmente, é necessário calcular o tempo utilizado para verificar a condição *salario* > 3000 em cada tupla. Logo, o tempo gasto para checar a condição da coluna *salario* na *i*-ésima tupla de *funcionarios* é dado por:

$$\text{Tempo total na tupla } i = T(i) = T_i(\lambda_i) + T_i(\lambda_i \in C)$$

Tal que:

$T_i(\lambda_i)$ é o tempo gasto na *i*-ésima tupla para realizar o cálculo de λ_i .

$T_i(\lambda_i \in C)$ é o tempo gasto para verificar se λ_i existe no conjunto *C*.

Dado que a relação *funcionarios* tem, por exemplo, 400 tuplas, o tempo total necessário para realizar a consulta nessa relação é dado pela soma de todos os tempos, ou seja,

$$\text{Tempo Total Consulta Cifrada} = T(1) + T(2) + \dots + T(400)$$

A figura 5.10 apresenta os tempos totais obtidos nas duas consultas para os cinco cenários considerados. Observe que a consulta cifrada gasta mais tempo que a consulta convencional para dados não cifrados, isto acontece devido aos cálculos realizados pela camada **Cripto_Servidor**.

A segunda métrica compara o tamanho do espaço necessário para armazenar a base de dados não cifrada e cifrada. Os resultados obtidos são mostrados na figura 5.11. Observe que o tamanho da base de dados cifrada é muito maior. Isto acontece porque os dados foram cifrados

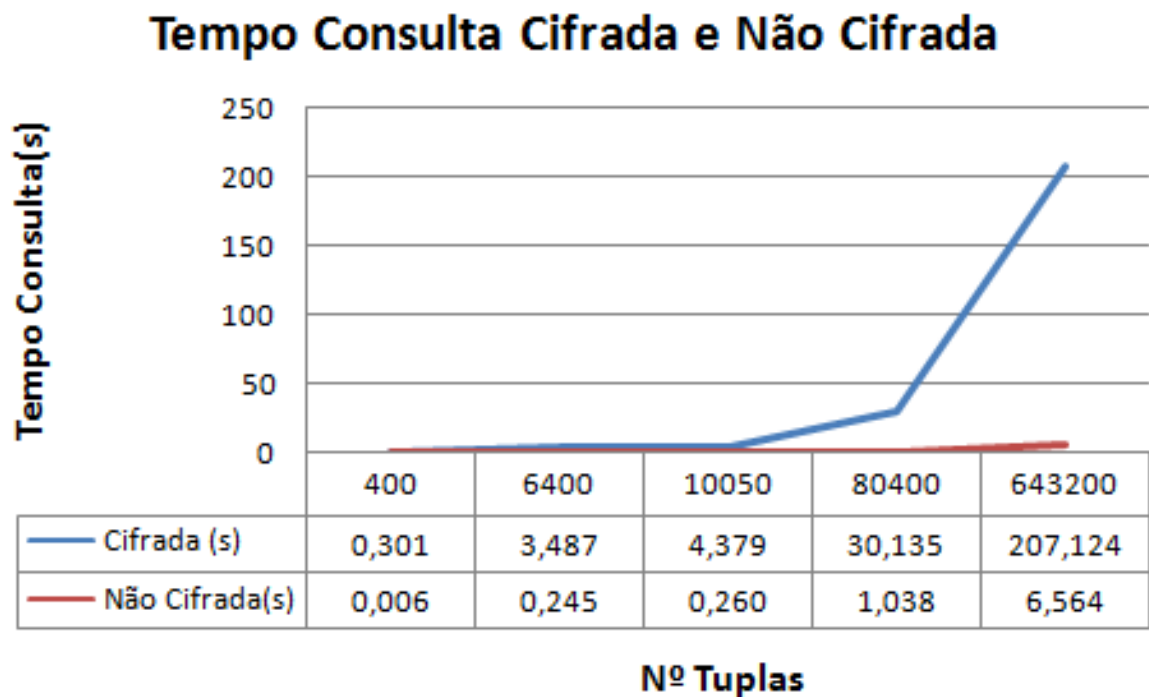


Figura 5.10: Comparação entre tempo (ms) das consultas cifradas x não cifradas.

utilizando chaves de 1024 bits e, além disso, cada texto cifrado possui um par (δ, γ) de 1024 bits cada.

Diante dos resultados apresentados, verificou-se que no pior cenário, por exemplo, a base cifrada é aproximadamente 45 vezes maior que a base não cifrada. Esse tamanho está relacionado diretamente ao tamanho da chave, logo, quanto maior a chave maior é o tamanho da base cifrada. Além disso, uma base de dados cifrada muito grande, implica em um maior tráfego na rede e um maior espaço em disco no servidor.

Métrica de segurança

Nos experimentos anteriores, o tamanho da chave utilizada foi de 1024 bits. Nesse caso, para avaliar o quanto essa chave representa em relação à segurança de um ambiente, utilizou-se como métrica a tabela 5.2 que mostra o tamanho de chaves e o nível de segurança correspondente [Ecrypt 2010]. A partir da tabela 5.2, verificou-se que o nível de segurança obtido por um criptosistema de chave pública de 1024 bits é igual a 3, ou seja, uma proteção média. Além disso, cada nível apresenta um valor representado pela coluna peso. Essa coluna varia em uma escala entre 0 e 1. No caso dos experimentos para a uma chave de 1024 bits o peso corresponde à 0,375.

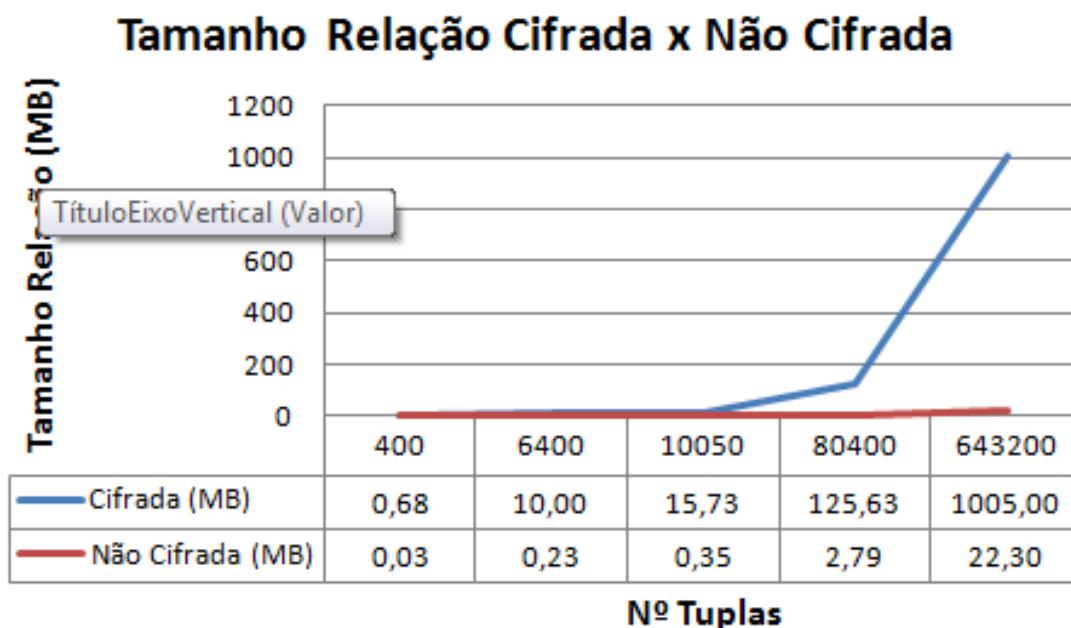


Figura 5.11: Comparação entre os tamanhos em (MB) das relações cifradas e não cifradas.

Nível	Proteção	Tamanho da chave em bits	Peso
1	Ataques em tempo real	-	0,125
2	Proteção curta, não deve ser usada para manter confidencialidade de dados. Utilizado por pequenas empresas	816	0,250
3	Proteção média para pequenas empresas	1008	0,375
4	Curta proteção para dados de empresas de médio porte (2008-2011)	1248	0,5
5	(2009 até 2020)	1776	0,625
6	Médio prazo de proteção (2009 até 2030)	2432	0,75
7	Proteção a longo prazo (2009 até 2040)	3248	0,875
8	Boa proteção contra os computadores quânticos, com exceção do algoritmo Shor's	15424	1

Tabela 5.2: Relação nível de segurança com o tamanho da chave, tabela adaptada de [Ecrypt 2010].

5.3 Considerações Finais

Neste capítulo foram apresentados os principais problemas de segurança que o protocolo está exposto. Nesse caso, verificou-se que se o **problema do logaritmo discreto (PLD)** ou de **determinação do elemento gerador (DEG)** tem solução então o protocolo seria quebrado. Além disso, foi apresentada a arquitetura utilizada como base dos experimentos, as principais telas do sistema e os resultados obtidos. Verificou-se também que o tamanho da chave tem um papel importante na segurança do protocolo, ou seja, para números menores que 1024 bits, o **problema do logaritmo discreto (PLD)** e o de **determinação do elemento gerador** tem solução em tempo polinomial, logo, o criptosistema é quebrado. Nesse caso, o tamanho da

chave deve ser sempre levado em consideração segundo as recomendações existentes, como a apresentada em [Ecrypt 2010], por exemplo.

No próximo capítulo é apresentada uma análise crítica desses resultados e as principais contribuições obtidas desta dissertação.

Capítulo 6

Conclusão e Trabalhos Futuros

Neste capítulo são apresentadas as análises críticas do protocolo proposto, sugestões de trabalhos futuros e em seguida as considerações finais.

6.1 Análise crítica dos resultados

A partir dos resultados obtidos nos experimentos apresentados no capítulo 5, foi possível constatar as hipóteses levantadas no capítulo 1, ou seja, verificou-se que através da extensão do algoritmo de chave pública El-Gamal é possível realizar consultas em base de dados cifrada armazenada em um servidor sem a necessidade de decifrar os dados nele armazenados. Portanto as consultas realizadas pelos clientes são mais seguras. Através da arquitetura proposta, o cliente realizou todo o processo de cifragem antes de enviar sua consulta ao servidor. Nesse caso é importante considerar que no contexto desta dissertação a máquina do cliente deve ser segura. Conforme mostrado no capítulo de segurança, é necessário que os elementos α e w não sejam descobertos por Oscar, uma vez que de posse desses elementos ele pode decifrar os dados da base sem a necessidade de conhecer a chave privada.

Além disso, constatou-se também a outra hipótese de que o protocolo proposto tem um alto custo computacional se comparado a uma consulta convencional. No entanto, essa queda do desempenho aconteceu principalmente em razão do tamanho da chave de 1024 bits. Entretanto, os tempos obtidos podem ser considerados adequados, pois em sua maioria foram inferiores a 4 minutos. No pior caso, por exemplo, a consulta demorou 207,124s(3 minutos e 45s) para uma tabela de 643.200 registros. Além disso, deve-se levar em consideração que esses tempos podem ser melhorados se o protocolo for executado em um ambiente de computação nas nuvens.

Outro fator importante na segurança do protocolo é o tamanho da chave. Esse tamanho depende do nível de segurança requisitado por um cliente ou empresa. A chave de 1024 bits utilizada nos experimentos por exemplo, apresentou segundo a métrica um nível de segurança nível 3 e um peso de 0,375 em uma classificação de 1 a 8. Logo, dependendo da aplicação, uma chave de 1024 bits pode ser não segura. Em [Lenstra e Verheul 1999] é apresentado um estudo

detalhado sobre os tamanhos das chaves para os próximos anos. Este estudo teve como objetivo determinar os tamanhos das chaves utilizados para permitir que os criptosistemas sejam mais seguros. Segundo essa estimativa, o tamanho da chave recomendado para algoritmos de chave pública no ano de 2011 deveria ser de 1416 bits.

Um aumento no tamanho da chave também implica em uma bases de dados maior. Nos experimentos verificou-se que o espaço necessário para armazenar uma base de dados cifrada é bem maior, no pior cenário, por exemplo, o tamanho da base cifrada é aproximadamente 45 vezes maior que a base convencional. Portanto, dependendo da quantidade de registros, pode ser necessário uma grande quantidade de espaço em disco para armazenamento dos dados cifrados. Além disso, se o resultado da consulta possuir uma grande quantidade de dados, implica em um maior tráfego na rede.

6.2 Trabalhos futuros

A seguir são apresentadas as sugestões para trabalhos futuros que podem ser realizados a partir desta proposta.

1. Implementação em hardware usando-se de FPGAs dos algoritmos de cifragem e decifragem executados no cliente;
2. Implementação em hardware usando-se de FPGAs de um processador criptográfico para os cálculos do operador de seleção executado no servidor;
3. Implementação dos operadores de Atualização e Remoção do protocolo proposto;
4. Executar o protocolo proposto em servidores de computação nas nuvens e verificar o desempenho nesses ambientes;
5. Desenvolver um protocolo proposto equivalente utilizando curvas elípticas.

6.3 Considerações finais

Nesta dissertação foi proposto um protocolo de segurança para bases de dados armazenadas em computação nas nuvens, o principal objetivo foi garantir uma maior segurança nesses ambientes. Para atingir esse objetivo, foi desenvolvido um protocolo proposto que tem como fundamento o criptosistema e o esquema de assinatura do El-Gamal. A justificativa para tal escolha é o aparato matemático que esse criptosistema apresenta para manipulação dos dados na sua forma cifrada. A partir dessa propriedade, é possível decidir dentre dois dados cifrados qual deles é o maior, por exemplo. No contexto apresentado, isso é muito importante, pois possibilita a execução de uma consulta no servidor sem a necessidade de decifrá-la no servidor e transmiti-la na sua forma original. Em seguida, foi definida a álgebra relacional para a manipulação das relações cifradas. A partir dessa definição, verificou-se que as operações desta

álgebra são similares à álgebra relacional convencional, com exceção do operador de seleção. Diante disso, foi proposto um protocolo para esse operador.

Em relação aos ataques que o protocolo está exposto, constatou-se que a segurança do protocolo depende da dificuldade em ter a solução de dois problemas matemáticos: **Problema do Logaritmo discreto (PLD)** e **determinar o elemento gerador (DEG)** de um grupo $\mathbb{Z}_p^*$. Verificou-se também através de pesquisas que estudos recomendam que o tamanho da chave em sistemas de chave pública sejam de 1416 bits, por exemplo. Portanto no momento de utilização do protocolo é importante verificar o nível de segurança exigido pela aplicação do cliente e utilizar um tamanho de chave adequado. Portanto, se essa condição é satisfeita então o protocolo é seguro.

Além disso, observou-se que o protocolo proposto apresentou como desvantagens o desempenho e o maior espaço em disco para armazenamento das informações cifradas. Essas observações foram constatadas nos testes realizados no protótipo desenvolvido. Esse protótipo é uma aplicação cliente-servidor desenvolvida em um ambiente local para validar a correção do protocolo e o desempenho das consultas cifradas. Além disso, utilizou-se o SBGD sqlServer 2008 nos experimentos, esse SGBD apresentou uma limitação de 128 caracteres na definição de nomes de colunas e relações. Portanto para tamanho de chaves grandes(maior que 1024 bits) não é possível definir diretamente no SBGD os nomes das relações e colunas cifradas. Nesse caso, para contornar essa limitação foi construído um mecanismo que mapeia o nome das relações e colunas com os seus valores cifrados.

Diante das desvantagens apresentadas, em um ambiente empresarial, por exemplo, podem existir centenas ou até milhares de tabelas em uma mesma base de dados, nesse caso, os algoritmos de criptografia propostos devem ser aplicados em tabelas cuja segurança é crítica, como por exemplo, tabelas que armazenam: salários de funcionários, senhas de clientes, número do cartão de crédito, entre outras.

Referências Bibliográficas

- [Anciaux et al. 2006] Anciaux, N., Bouganim, L., e Pucheral, P. (2006). Data confidentiality: to which extent cryptography and secured hardware can help. *Annales des Telecommunications - Annals of Telecommunications*.
- [Benjaminkeen 2001] Benjaminkeen (2001). *Software Data Generator*. Acessado em 21/04/2011 Disponível em: <http://www.benjaminkeen.com/software/>.
- [Breitman e Viterbo 2010] Breitman, K. e Viterbo, J. (2010). Computação na Nuvem , Uma Visão Geral. *III Congresso Internacional Software Livre e Governo Eletrônico*, pp. 17–46.
- [Brodkin 2008] Brodtkin, J. (2008). Seven cloud-computing security risks. Acessado em 01/06/2011 Disponível em <http://www.networkworld.com/news/2008/070208-cloud.html?page=1>.
- [Cambiucci 2009] Cambiucci, W. (2009). SQL Azure Database Um banco de dados Relacional na nuvem. Acessado em 06/05/2011 Disponível em: <http://www.profissionaisiti.com.br/2009/12/sql-azure-database-um-banco-de-dados-relacional-na-nuvem/>.
- [Ecrypt 2010] Ecrypt (2010). *cryptographic key length Recommendation*. Acessado em 01/07/2011 Disponível em: <http://www.keylength.com/en/3/>.
- [ElGamal 1985] ElGamal, T. (1985). A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*.
- [Elmasri e Navathe 2000] Elmasri, R. e Navathe, S. B. (2000). *Fundamentals of Database Systems*. Addison Wesley, 3 edition.
- [Evdokimov e Guenther 2007] Evdokimov, S. e Guenther, O. (2007). Encryption Techniques for Secure Database Outsourcing. Cryptology ePrint Archive, Report 2007/335. <http://eprint.iacr.org/>.
- [Guimarães 2005] Guimarães, C. (2005). *Fundamentos de banco de dados: Modelagem, projeto e linguagem SQL*. Unicamp.
- [Hacigumus et al. 2002] Hacigumus, H., Iyer, B., Li, C., e Mehrotra, S. (2002). Executing over Encrypted Data in Database-Service-Provider Model. *Technical Report TRDB-0202, Database research Group at University of California*.
- [Knudsen 1998] Knudsen, J. (1998). *Java Cryptography*. O'REILLY, 1 edition.
- [Koblitz 1994] Koblitz, N. (1994). *A Course in Number Theory and Cryptography*. Springer Verlag.

- [Laur et al. 2011] Laur, S., Willemson, J., e Zhang, B. (2011). Round-efficient Oblivious Database Manipulation. Cryptology ePrint Archive, Report 2011/429. <http://eprint.iacr.org/>.
- [Lenstra e Verheul 1999] Lenstra, A. K. e Verheul, E. R. (1999). Selecting Cryptographic Key Sizes in Commercial Applications. *PricewaterhouseCoopers Cryptographic Centre of Excellence (CCE) Quarterly Journal*.
- [Li et al. 2009] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., e Lou, W. (2009). Enabling Efficient Fuzzy Keyword Search over Encrypted Data in Cloud Computing. Cryptology ePrint Archive, Report 2009/593. <http://eprint.iacr.org/>.
- [Mao 2003] Mao, W. (2003). *Modern Cryptography: Theory and Practice*.
- [Martins 2010] Martins, A. (2010). Fundamentos de Computação Nuvem para Governos. *III Congresso Internacional Software Livre e Governo Eletrônico*, pp. 47–64.
- [Microsoft 2011] Microsoft (2011). *SQL Server (2011) Technical Reference (Database Engine)*. Acessado em 04/05/2011, Disponível em <http://msdn.microsoft.com/en-us/library/bb500275.aspx>.
- [Mysql 2011] Mysql (2011). *Mysql Technical Reference*. Acessado em 04/05/2011, Disponível em <http://www.mysql.com/>.
- [Novais et al. 2010] Novais, G. M., Kosloski, I. B. D., Leocadio, J. M., e Gomes, L. R. G. (2010). Segurança de Computação em Nuvem. *III Congresso Internacional Software Livre e Governo Eletrônico*, pp. 131–158.
- [Oracle 2011] Oracle (2011). *Oracle Technical Reference*. Oracle. Acessado em 04/05/2011, Disponível em <http://psoug.org/reference/library.html>.
- [Pollard 1978] Pollard, J. (1978). Monte Carlo methods for index computation mod p. 32.
- [Reinhold 2000] Reinhold, A. G. (2000). *Big Number Calculator Applet*. Acessado em 15/05/2011, Disponível em <http://world.std.com/reinhold/BigNumCalc.html>.
- [Scheinerman 2003] Scheinerman, E. (2003). *Matemática Discreta: Uma introdução*. Thomson.
- [Shoup 2008] Shoup, V. (2008). *A computational Introduction to Number Theory and Algebra*.
- [Silberschatz et al. 2006] Silberschatz, A., Korth, H., e Sudarshan, S. (2006). *Sistemas de Banco de Dados*. Elsevier, 5 edition.
- [Sousa et al. 2009] Sousa, F. R. C., Moreira, L. O., e Machado, C. (2009). Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios. *ERCEMAPI*.
- [Stallings 2007] Stallings, W. (2007). *Criptografia e Segurança de Redes: Princípios e práticas*. Prentice Hall, 4ª edição edition.
- [Stinson 2005] Stinson, D. R. (2005). *Cryptography: Theory and Practice, Third Edition (Discrete Mathematics and Its Applications)*. Chapman & Hall/CRC, 3 edition.
- [Tanenbaum 2003] Tanenbaum, A. (2003). *Redes de Computadores*. Campus, 4 edition.

- [Tujal 2010] Tujal, L. C. P. (2010). Modelo de Referência de Cloud. *III Congresso Internacional Software Livre e Governo Eletrônico*, pp. 67–110.
- [Wang et al. 2009] Wang, Q., Wang, C., Jin, L., Ren, K., e Lou, W. (2009). Enabling Verifiability and Data Dynamics for Storage Security in Cloud Computing. *Cryptology ePrint Archive*.
- [Wang et al. 2004] Wang, Z.-f., Dai, J., Wang, W., e Shi, B.-l. (2004). Fast Query Over Encrypted Character data in Database. *International Press*, 4:289–300.

Apêndice A

Código Fonte

A.1 Classe Cripto_Servidor

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Numerics;
public
class cripto_servidor
private BigInteger mP; // valor de P utilizado no calculo da multiplicao modular
private BigInteger mLambda;
private BigInteger M = 5000;
private BigInteger w = 3;
private BigInteger[] conjunto1 = new BigInteger[5000];
public cripto_servidor(BigInteger P) {
    mP = P;
}
public BigInteger getmP() {
    return this.mP;
}
```

```
public void SetmP(BigInteger P)
    this.mP = P;
}
public BigInteger calculaLambda(Byte[] d1 ,Byte[] d2 ) {
    BigInteger deltaQ = new BigInteger(d1);
    BigInteger deltaTb = new BigInteger(d2);
    mLambda = BigInteger.Multiply(deltaQ, deltaTb);
    mLambda = BigInteger.Remainder(mLambda, mP);
    return mLambda; }
public bool existeLamba(string operador)
if (operador == ">") {
//verifica se lamba existe em conjunto 1
    int i;
    for (i = 0; i < M; i++) {
        if (BigInteger.Compare(conjunto1[i], mLambda) == 0) {
            return true;
        }
    }
    return false;
}
else {
//verifica se lamba existe em conjunto 1
    int i;
    for (i = 0; i < M; i++) {
        if (BigInteger.Compare(conjunto1[i], mLambda) == 0) {
            return true;
        }
    }
    return false;
}
}
public void geraConjunto()
int i;
BigInteger auxM = BigInteger.One;
for (i = 0; i < M; i++) {
    conjunto1[i] = BigInteger.ModPow(2, auxM , mP);
    conjunto1[i] = BigInteger.Multiply(w, conjunto1[i]);
    auxM = BigInteger.Add(auxM, BigInteger.One); }
}
```

```
// Converte uma string em um byte[]
private static byte[] StrToByteArray(string str)
{
    System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
    return encoding.GetBytes(str);
}

public BigInteger imprimeCon(int i) {
    return conjunto1[i];
}

public DataTable select(DataTable dt, String operador, String nomeColuna, String vlrCon-
dicao) {
    int indiceColuna;
    DataTable dtRes;
    geraConjunto();
    dtRes = dt.Clone();
    indiceColuna = dt.Columns.IndexOf(nomeColuna);
    bool teste;
    if (operador == ">") {
        teste = true;
    } else {
        teste = false;
    }
    int i;
    for (i = 0; i < dt.Rows.Count; i++)
    {
        String[] auxDeltaGama = dt.Rows[i][indiceColuna].ToString().Split('$');
//busca registro e armazena valor coluna em um vetor
//de duas colunas delta e gama
        calculaLambda(Convert.FromBase64String(vlrCondicao),
            Convert.FromBase64String(auxDeltaGama[0])).ToString();
        if (BigInteger.Compare(BigInteger.One, mLambda) != 0) {
            if (existeLamba(operador) == teste) {
                dtRes.ImportRow(dt.Rows[i]);
            }
        }
    }
    return dtRes;
}
}
```

A.2 Classe Cripto_Cliente

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Numerics;
using System.Data;
public class cripto_cliente
private BigInteger mP;
private BigInteger mX;
private BigInteger mK;
private BigInteger mAlfa=2;
private BigInteger mDelta;
private BigInteger mGama;
private String auxk = "";
private String auxP = "";
public cripto_cliente(){
    P = BigInteger.Parse(auxP);
    mK = BigInteger.Parse(auxk);
}
public BigInteger getmP(){
    return this.mP;
} public BigInteger getDelta() {
    return this.mDelta;
public BigInteger getGama() {
    return this.mGama;
}
public void cifra(string msg) {
    Int64 auxInt;
    if (Int64.TryParse(msg, out auxInt) == true)
    {
        mX = new BigInteger(Convert.ToInt64(msg));
    }
    else {
        mX = new BigInteger(StrToByteArray(msg));
    }
    mDelta = BigInteger.ModPow(mAlfa, BigInteger.Add(mK, mX), mP);
    mGama = BigInteger.Multiply(mX, BigInteger.ModPow(mDelta, mK, mP));
```

```

        mGama = BigInteger.Remainder(mGama, mP);
    }
    public string decifra(byte[] b1, byte[] b2, int tipo) {
        BigInteger auxGama = new BigInteger(b1);
        BigInteger auxDelta = new BigInteger(b2);
        BigInteger auxDec = new BigInteger();
        auxDec = BigInteger.ModPow(auxGama, mK, mP);
        auxDec = inverso(mP, auxDec);
        auxDec = BigInteger.Multiply(auxDelta, auxDec);
        auxDec = BigInteger.Remainder(auxDec, mP);
        if (tipo == 0 || tipo == 2) {
            return auxDec.ToString();
        }
        return ByteArrayToStr(auxDec.ToByteArray());
    }
    public BigInteger inverso(BigInteger a, BigInteger b)
    BigInteger a0;
    BigInteger b0;
    BigInteger t;
    BigInteger t0;
    BigInteger q;
    BigInteger r;
    BigInteger temp;
    a0 = a;
    b0 = b;
    t0 = BigInteger.Zero;
    t = BigInteger.One;
    q = BigInteger.DivRem(a0, b0, out r);
    while(BigInteger.Compare(r, BigInteger.Zero) == 1) {
        temp = BigInteger.Subtract(t0, BigInteger.Multiply(q, t));
        temp = BigInteger.Remainder(temp, a);
        t0 = t;
        t = temp;
        a0 = b0;
        b0 = r;
        q = BigInteger.DivRem(a0, b0, out r);
    }
    if (BigInteger.Compare(b0, BigInteger.One) == 0) {
        if (BigInteger.Compare(t, BigInteger.Zero) == -1) {

```

```
        return BigInteger.Add(a, t);
    }
    else {
        return t;
    }
}
return t;//return false nao tem inverso
}
public DataTable decifraTabela(DataTable dt) {
    int i, j;
    for (i = 0; i < dt.Rows.Count; i++) {
        for (j = 0; j < dt.Columns.Count; j++) {
            string[] auxDeltaGama = dt.Rows[i][j].ToString().Split('$');
            dt.Rows[i][j] = decifra(Convert.FromBase64String(auxDeltaGama[0]), Convert.FromBase64String(auxDeltaGama[1]),j).ToString();
        }
    }
    return dt;
}
private static byte[] StrToByteArray(string str) {
    System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
    return encoding.GetBytes(str);
}
private static string ByteArrayToStr(byte[] b) {
    System.Text.ASCIIEncoding encoding = new System.Text.ASCIIEncoding();
    return encoding.GetString(b);
}
}
```

Apêndice B

Representação Base64

Base64 é um sistema utilizado para representação de *bytes* de dados. Essa representação tem como entrada um conjunto de *bytes* e a partir destes utiliza-se seis bits do total de oito para representar um dígito. Cada dígito é representado por um caractere ASCII [Knudsen 1998].

A figura B.1 mostra o processo de conversão de *byte* para *Base64*.

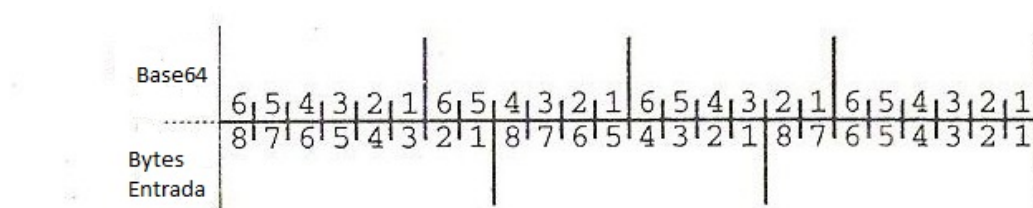


Figura B.1: Conversão de *byte* para *Base64* [Knudsen 1998]

Dada a representação da figura B.1 há três casos a considerar [Knudsen 1998]:

- Se a entrada de dados é múltipla de 24, logo não é necessário nem um dígito complementar.
- Se a entrada tem um *byte* a mais, então é necessário completar com dois dígitos extras.
- Se a entrada tem dois *bytes* a mais, então é necessário completar com um dígito extra.

Segundo a representação *Base64* é mais eficiente que a representação hexadecimal.

Um estudo mais detalhado dessa representação é realizado em [Knudsen 1998].