UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS OLIVEIRA LEITE DE SÁ

A IMPLEMENTAÇÃO DE UM PROTOCOLO CRIPTOGRÁFICO PARA GERAÇÃO DISTRIBUÍDA DE CREDENCIAIS NO SISTEMA CIVIS

Belém
2018

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MATHEUS OLIVEIRA LEITE DE SÁ

A IMPLEMENTAÇÃO DE UM PROTOCOLO CRIPTOGRÁFICO PARA GERAÇÃO DISTRIBUÍDA DE CREDENCIAIS NO SISTEMA CIVIS

Trabalho de Conclusão de Curso apresentado como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Roberto Samarone dos Santos Araújo

Belém
2018

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S111i Sá, Matheus Oliveira Leite de.
A IMPLEMENTAÇÃO DE UM PROTOCOLO CRIPTOGRÁFICO PARA GERAÇÃO DISTRIBUÍDA DE CREDENCIAIS NO SISTEMA CIVIS / Matheus Oliveira Leite de Sá. — 2018.
61 f. : il. color.

Orientador(a): Prof. Dr. Roberto Samarone dos Santos Araújo
Trabalho de Conclusão de Curso (Graduação) - Faculdade de Computação, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2018.

1. Segurança da Informação. 2. Criptografia Limiar. 3. Votação Eletrônica. I. Título.

CDD 004

Este trabalho é dedicado ao sonho de criança de
viajar ao espaço, ver um dinossauro e se tornar um cientista.

AGRADECIMENTOS

Agradeço aos meus pais, Francisco Nilton e Maria, e à minha irmã, Beatriz, pelo apoio e incentivo nessa jornada. Se este é o fim desta primeira jornada, vocês foram os responsáveis por que eu desse o primeiro passo.

Agradeço ao Prof. Samarone Araújo pela orientação e apoio desde os primeiros semestres. Todos os momentos de aprendizado com o senhor foram fundamentais para que eu chegasse até aqui, e desde já agradeço também a sua paciência e me desculpo pelos momentos em que fui cabeça dura e não segui seus conselhos.

Agradeço também aos membros do LabSC, principalmente ao André, que me ajudou com qualquer problema que aparecesse, desde que ingressei ao laboratório até hoje. Você foi meu irmão mais velho da vida acadêmica, sou muito grato por toda a ajuda e parceria.

Agradeço aos professores e professoras do curso, que transmitiram ensinamentos fundamentais para o desenvolvimento deste trabalho e saciaram todas as minhas muitas dúvidas. Em especial agradeço ao professor Josivaldo, por não rasgar a minha beca; ao professor Sandro, por responder pacientemente as minhas infinitas perguntas; e ao professor Bitar, por ser um grande avô para todos da minha turma. Agradeço em especial também à professora Marcelle, por me ensinar a não cair mais em pegadinhas nas provas (mesmo eu tendo caído em todas as suas); à professora Kawasaki, por ser a pessoa mais divina da faculdade, nos dando aulas do começo ao final do curso e chocolates na Páscoa; e à professora Fabíola, por manter a mim e aos meus amigos calmos nos mostrando que tudo apenas tende a piorar. Minha experiência acadêmica foi sem igual principalmente pela grande proximidade que tive com a maioria de vocês.

Agradeço aos amigos que tive durante a graduação, em especial ao Leo, Italo, Fernando, Jayme, Adiel, Renan, Gabriel e Josy, pelos pelos vários momentos inesquecíveis que tivemos em todos esses anos. Mesmo que daqui para frente não sigamos os mesmos caminhos, essa etapa será eternamente marcada, para mim, por todos vocês.

Agradeço também ao meu amigo David, que mesmo não muito presente no meu ambiente acadêmico, sempre se manteve bastante próximo de mim e preocupado em ajudar com o que fosse possível.

Agradeço a todos que direta ou indiretamente contribuíram para a realização deste trabalho, desde as encarregadas pela limpeza do instituto até os bolsistas encarregados em manter os laboratórios funcionais.

E por fim, sou grato também pelos momentos em que jogava pelos corredores e filas da universidade, com meus amigos. Esses foram um alívio para as situações de cansaço e estresse que surgiam pelo caminho, e vale o agradecimento, mesmo que tenham sido momentos simples, pois os compartilhei com muitas pessoas de quem gosto.

Muito obrigado.

"Quando acordei hoje de manhã, eu sabia quem eu era,
mas acho que já mudei muitas vezes desde então."
(Alice no País das Maravilhas - Lewis Caroll)

RESUMO

O sistema para eleições CIVIS é baseado em um protocolo criptográfico que possibilita resistência à ataques coercivos. A fim de resistir a tais ataques, esse protocolo utiliza a ideia de credenciais. Uma credencial é formada por um conjunto de bits que deve ser gerado por um conjunto de autoridades eleitorais e entregue a cada votante em sigilo. Do contrário, o protocolo não garante resistência a ataques coercivos. Dessa forma, a geração de credenciais torna-se um ponto crítico no protocolo. Na versão atual do sistema CIVIS, entretanto, cada credencial é gerada por única autoridade eleitoral. Como consequência, o sistema depende que essa autoridade seja confiável para garantir a geração segura da credencial. Nesse contexto, este trabalho apresenta uma implementação de um protocolo para a geração distribuída de credencias a fim de tornar a geração de credenciais mais segura no sistema CIVIS.

Palavras-chave: Segurança da Informação. Criptografia Limiar. votação via *Internet*.

ABSTRACT

The electronic election system CIVIS is based on a cryptographic protocol that enables resistance to coercive attacks. In order to resist such attacks, this protocol uses the concept of credentials. A credential consists of a set of bits that must be generated by a set of electoral authorities and delivered to each voter in secrecy. Otherwise, the protocol does not guarantee resistance to coercive attacks. In this way, the generation of credentials becomes a critical point in the protocol. In the current version of the CIVIS system, however, each credential is generated by single electoral authority. As a consequence, the system relies on this authority to be trusted to ensure secure credential generation. In this context, this work presents an implementation of a protocol for the distributed generation of credentials in order to make credential generation more secure in the CIVIS system.

Keywords: Information Security. Threshold Cryptosystem. Electronic Voting.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

BPMN	*Business Process Model and Notation*
DKG	*Distributed Key Generation*
HTML	*HyperText Markup Protocol*
JSBN	*JavaScript BigInt*
LabSC	Laboratório de Segurança e Criptografia Aplicada
SJCL	*Stanford JavaScript Crypto Library*
UFPA	Universidade Federal do Pará
WZF	*Wang, Zhang & Feng*

LISTA DE SÍMBOLOS

a, b, c	Termos auxiliares para geração de credenciais
f_i, d_i	Polinômios de *Lagrange*
E	Variável de verificação de compartilhamento
F	Polinômio de *Lagrange* Público
g	Gerador do grupo multiplicativo
h	Chave de compartilhamento
i, j, k	Identificadores de participantes
m	Mensagem a ser assinada
n	Número de participantes
p, q	Números primos
r	Número aleatório
R	Número aleatório secreto
Ri, Si	Frações de segredo
S	Segredo compartilhado
t	limiar
u, v	Polinômios públicos auxiliares para geração de credenciais
x, y	Chaves privadas
$\mathbb{Z}_p^*$	Grupo multiplicativo de p sem elementos nulos
Λ	Grupo de participantes válidos; Lambda
σ	Elemento de uma credencial; sigma
$\sum$	Somatório
$\prod$	Produtório

SUMÁRIO

1 INTRODUÇÃO

A tecnologia da informação tem se tornado cada vez mais presente no cotidiano. Uma quantidade significativa da população em todo o mundo tem acesso a dispositivos computacionais, como computadores e celulares inteligentes, com acesso à *Internet*. É natural que tais tecnologias passem a ser aplicadas em diversos processos.

Um processo de muita relevância é o de eleição. A possibilidade de introdução deste processo ao ambiente da *Internet* proporciona uma série de benefícios quando comparada ao modelo tradicional de eleições presenciais, como a comodidade aos eleitores e uma maior abrangência de localidades. Tal processo ainda pode ter como benefícios a diminuição de custos e maior facilidade de organização.

Entretanto, problemas também são destacados quando sistemas de votação via *Internet* são uma opção. Ameaças à integridade e ao sigilo dos dados enviados pelos votantes ao serem transmitidos via *Internet*, um ambiente onde aplicações ficam mais expostas a atacantes, são problemas relacionados à tecnologia utilizada. Outros problemas podem ser destacados, como o risco de coerção de votantes, onde um eleitor é obrigado por um agente malicioso a votar numa determinada opção.

Para realização de um processo de votações seguro, é necessário que medidas de segurança sejam tomadas. O sistema de votação utilizado é de muita relevância para cada problema destacado, por isso também é necessário que o mesmo tenha sido desenvolvido tendo formas de minimizar cada um desses. Um exemplo, é o sistema de votação via *Internet* CIVIS, que se destaca por utilizar um protocolo que garante resistência à coerção, proposto por (ARAÚJO et al., 2010).

O sistema de votação via *Internet* CIVIS (ARAÚJO; NETO; TRAORÉ, 2018) utiliza um esquema de credenciais, geradas a partir de primitivas criptográficas, que são enviadas com o voto de cada eleitor. As credenciais são únicas para cada participante, e se alguma for alterada e enviada junto de um voto, este voto será anulado na fase de apuração. Assim, o votante tem a capacidade de enganar um atacante em caso de coerção.

Para o funcionamento de uma eleição criada no sistema, se faz necessária a definição de um conjunto de autoridades, que serão responsáveis por organizar as etapas dessa eleição. As autoridades tem participação no uso de primitivas criptográficas do sistema, por meio do uso de chaves secretas para cada autoridade.

No CIVIS, as credenciais são geradas por uma única autoridade, a autoridade de registro. A autoridade possui uma chave secreta, utilizada no processo de geração de credenciais. Entretanto, é necessário que essa autoridade seja confiável para que seja garantida a segurança do processo. Isso resulta numa potencial vulnerabilidade para todo o sistema de votação.

Para solucionar tal problema é proposta a utilização de um esquema de geração de chaves

limiar, onde seria definido um número maior de autoridades de registro, formando um conjunto. Este conjunto deve colaborar para a criação de credenciais de forma distribuída, a partir do protocolo proposto por (WANG; ZHANG; FENG, 2005) para criação de assinaturas digitais, de forma distribuída.

Este trabalho apresenta uma implementação do protocolo limiar de (WANG; ZHANG; FENG, 2005), aplicado para a criação de credenciais de forma distribuída, e propõe sua posterior integração ao sistema de votação via *Internet* CIVIS (ARAÚJO; NETO; TRAORÉ, 2018).

1.1 Motivação

A informatização do processo eleitoral proporciona maior praticidade e eficiência se comparado ao seu modelo tradicional, mas também proporciona uma variedade de riscos à segurança, que devem ser considerados. Uma forma de minimizar tais riscos é a utilização de criptossistemas para proporcionar maior garantia de segurança em seu funcionamento. O sistema de votação via *Internet* utilizado como base para este trabalho foi o CIVIS ((ARAÚJO; NETO; TRAORÉ, 2018), que é baseado em um protocolo criptográfico para votação via *Internet* seguro.

Entretanto, mesmo o CIVIS sendo um sistema que atenda bem aos requisitos de segurança de uma votação via *Internet*, o mesmo ainda pode ser adaptado para aumentar ainda mais a segurança proporcionada pelo seu uso. Uma possibilidade a ser explorada é o uso de técnicas de descentralização para a geração de credenciais.

A proposta de geração de credenciais de maneira distribuída consiste em ampliar a quantidade de autoridades de registro presente no sistema, e utilizar técnicas de criptografia limiar para que este conjunto de autoridades passe a criar um segredo em conjunto, distribuindo frações desse segredo para cada membro do conjunto de autoridades de registro.

Para a utilização do segredo é necessária a colaboração de ao menos uma quantidade de autoridades maior que o limiar estabelecido, evitando que a segurança seja comprometida caso algum membro mal intencionado tente utilizar o segredo sozinho ou com uma pequena assosciação de membros maliciosos.

Esta proposta é baseada no esquema de (WANG; ZHANG; FENG, 2005), e utiliza protocolos estabelecidos por Pedersen (PEDERSEN, 1991a) (PEDERSEN, 1991b) e Gennaro et al. (GENNARO et al., 1999).

1.2 Objetivos

Os objetivos deste trabalho são:

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é implementar um protocolo criptográfico para geração de credenciais de forma distribuída, a fim de integrá-lo ao sistema de votação CIVIS.

1.2.2 Objetivos Específicos

- Estudar a arquitetura do sistema de votação CIVIS;
- Estudar os mecanismos criptográficos associados à geração de credenciais do CIVIS;
- Analisar os protocolos criptográficos relacionados à geração de credenciais de forma distribuída;
- Coletar os requisitos necessários para o planejamento da implementação;
- Propor modificações na arquitetura do sistema CIVIS a fim de integrá-lo ao esquema de geração de credenciais implementado.

1.3 Trabalhos Relacionados

O protocolo de votação implementado para sistema CIVIS requer o uso de credenciais com uma estrutura matemática específica, portanto não é possível utilizar todo tipo de geração de credencial. Como o CIVIS é a primeira implementação desse protocolo, não foram encontrados na literatura trabalhos relacionados a implementações desse tipo.

Outros trabalhos utilizam o protocolo de geração de assinaturas digitais de (WANG; ZHANG; FENG, 2005) como fundamentação para suas implementações. Um trabalho que utiliza este protocolo também aplicado a eleições via *Internet*, é o de (SOUHEIB; STEPHANE; RIADH, 2012), mas como forma de minimizar a complexidade do sistema implementado, que é quadrática. É proposto no trabalho um esquema de marca d'água para as credenciais, que indicarão se são verdadeiras ou falsas, e estas são geradas de forma distribuída.

Já o trabalho de (HERRANZ; RUIZ; SÁEZ, 2014) é proposto um modelo de assinatura e criptografia em conjunto, de forma a tornar o procedimento mais eficiente do que quando realizados separadamente. A forma de decriptação apontada no trabalho conta com o uso de um conjunto limiar, propondo dois protocolos para este procedimento. Os protocolos garantem anonimidade aos participantes, e ao mesmo tempo mantém a segurança do sistema. A proposta do sistema é ser implementado em um sistema de leilão eletrônico.

Melhorias ao sistema CIVIS também foram o foco de outros trabalhos. O trabalho de (NETO, 2016) propõe a implementação de todo o criptossistema distribuído, o El Gamal Limiar, utilizado pelos CIVIS. O trabalho de (NETO et al., 2018) propõe como melhoria ao CIVIS uma

análise da usabilidade do sistema, e a adaptação do procedimento de envio de voto por cada participante.

1.4 Estrutura do Trabalho

Este trabalho está organizado da seguinte forma:

- Capítulo 2 – Referenciais Teóricos

  Neste capítulo são apresentadas as bases teóricas para o desenvolvimento e compreensão deste trabalho. Como principais informações destacam-se os conceitos e esquemas de criptografia limiar, amplamente utilizados no decorrer dos capítulos. O sistema CIVIS também é apresentado, com mais informações de seu funcionamento, sendo detalhado principalmente o seu processo de geração de credenciais atual.

- Capítulo 3 – A Implementação do Protocolo Criptográfico para Geração Distribuída de Credenciais

  Neste capítulo a implementação é definida, mostrando primeiramente o esquema de funcionamento do protocolo para geração de credenciais implementado, e logo após são apresentadas suas informações de planejamento, como requisitos e diagramas, e as funções disponibilizadas e como se comportam. Ao fim do capítulo é proposta a integração do protocolo implementado ao sistema CIVIS, onde é detalhado onde devem ocorrer modificações no sistema de votação para que o mesmo possa receber a implementação.

- Capítulo 4 – Conclusões

  O capítulo apresenta as considerações finais sobre o trabalho e os trabalhos futuros propostos.

2 REFERENCIAIS TEÓRICOS

Para a compreensão deste trabalho, é preciso estabelecer uma base de conceitos dentro da área de segurança. São utilizados diversos protocolos, com funcionamentos e utilizações diferentes. O sistema de votação utilizado também deve ser abordado, permitindo que todos possam compreender seu funcionamento e onde o trabalho atua sobre ele.

O capítulo está dividido da seguinte forma: a Seção 2.1 descreve o conceito de segurança da informação, onde é contextualizado o funcionamento de algoritmos criptográficos e dada uma visão geral sobre suas aplicações; a Seção 2.2 detalha o conceito de criptografia limiar, destacando os esquemas fundamentais para seu funcionamento e os protocolos utilizados como base para o protocolo implementado; na Seção 2.3 o sistema de votação CIVIS é apresentado, onde será detalhado seu funcionamento e, principalmente, seu esquema de geração de credenciais; finalmente, a Seção 2.4 conclui o capítulo.

2.1 A segurança da informação

Com a utilização de serviços computacionais para as mais diversas atividades, a necessidade de manter tais serviços funcionando de forma correta e sem oferecer riscos aos seus usuários se faz ainda mais presente. Esse é o foco da segurança da informação e suas tecnologias.

A segurança da informação pode ser definida como a proteção oferecida a fim de preservar a integridade, disponibilidade e confidencialidade em qualquer sistema computacional, de acordo com (STALLINGS, 2008). A integridade é a garantia de que uma determinada informação sempre será consistente e precisa; Confidencialidade é a capacidade de impedir que uma informação seja acessada ou disponibilizada a participantes não autorizados; e a Disponibilidade é a capacidade de garantir que um serviço ou recurso esteja disponível para os usuários autorizados a acessá-lo. Estes três objetivos são os principais pilares da segurança da informação.

A criptografia é um conjunto de mecanismos utilizados para atender aos requisitos de segurança citados. De acordo com (GATHEN, 2015), sua principal tarefa é a transmissão segura de alguma informação, tornando-a ininteligível para qualquer receptor que não seja o correto. Adicionalmente, outros objetivos estão incluídos na criptografia, como a utilização de assinaturas digitais ou formas de estabelecimento de identidade de alguma das partes envolvidas.

Este trabalho trata mais especificamente dos mecanismos de codificação criptográficos, que possuem como objetivo o uso de algoritmos matemáticos para transformar dados inteligígeis para um formato em que esteja impossível compreendê-los. No caso dos protocolos utilizados neste trabalho, todas as cifragens realizadas são reversíveis, ou seja, é possível obter a mensagem original de volta.

Mecanismos de codificação necessitam de uma chave criptográfica, utilizada como um segredo necessário para alterar uma informação inteligível, a fim de torná-la ininteligível, impossível de ser compreendida normalmente. Cada algoritmo utiliza uma chave criptográfica para realizar o processo de cifragem, e o tipo de chave determina o tipo de criptografia realizada: se for utilizada duas chaves de mesmo valor, é denominada a esse esquema criptografia simétrica; caso ambas as chaves tenham valores diferentes, o esquema é denominado criptografia assimétrica.

A criptografia de chave simétrica não é utilizada neste trabalho, mas seu funcionamento ajuda a elucidar requisitos fundamentais para ambos os tipos de criptografia com uso de chaves. Um algoritmo criptográfico de cifragem deve possuir termos específicos para o seu funcionamento. Estes termos são, de acordo com (STALLINGS, 2008) e (GATHEN, 2015):

1. um texto claro, que é a informação original, em seu estado inteligível;
2. um texto cifrado, que é a informação embaralhada, impossível de ser compreendida;
3. um algoritmo de criptografia, responsável por realizar procedimentos de substituições e transposições no texto recebido, para cifrá-lo ou decifrá-lo;
4. uma chave criptográfica, que é segredo utilizado nos processos de substituições e transposições do algoritmo de criptografia.

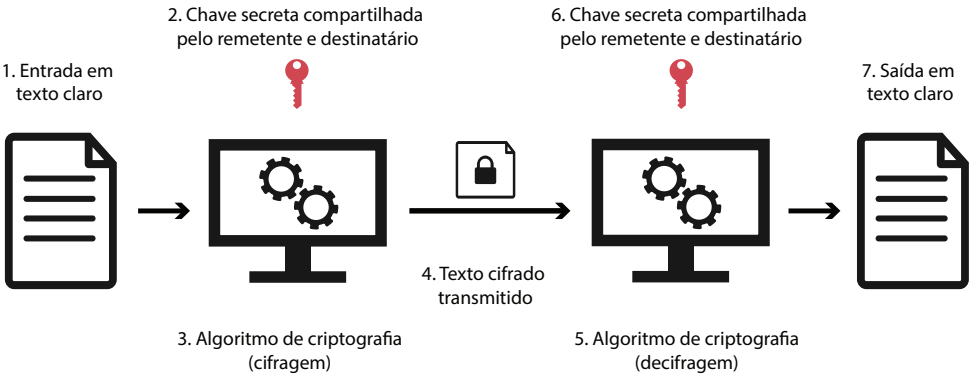
A chave criptográfica é responsável por garantir que o algoritmo de cifragem ou decifragem processe de forma correta a mensagem recebida, a fim de embaralhar seu conteúdo (caso seja um texto claro) ou decifrá-lo (caso seja um texto cifrado). Cada participante do esquema deverá possuir uma chave criptográfica, e nesse caso ambas devem ser iguais, sendo simétricas.

Entretanto, a criptografia de chave simétrica deve seguir os seguintes requisitos para seu uso seguro, ainda de acordo com (STALLINGS, 2008) e (GATHEN, 2015):

1. Um oponente que conheça o algoritmo criptográfico utilizado não deve ser capaz de decifrar qualquer texto cifrado ou descobrir a chave utilizada, mesmo que possua diversos textos cifrados e seus textos claros;
2. As chaves utilizadas devem ser mantidas de forma segura por quem as possuir. Uma vez que algum oponente tiver acesso a alguma chave, toda comunicação que a utilizou poderá ser lida;

Assim, a chave criptográfica utilizada deve permanecer em segredo de qualquer, um exceto os participantes do processo que devam ter acesso à mensagem em texto claro. Por esse motivo, o esquema de criptografia de chave simétrica é também conhecido como criptografia de chave secreta, como apresentado na Figura 1.

Figura 1 – Modelo simplificado de criptografia de chave simétrica



Fonte: (STALLINGS, 2008), adaptado

Porém, a necessidade de manter a chave criptográfica sempre em segredo torna o compartilhamento de chaves entre membros uma grande dificuldade a ser enfrentada. É muito difícil manter a chave em segredo se existe a necessidade de que ela seja compartilhada entre vários membros de um conjunto, porquanto a criptografia de chaves assimétricas torna-se mais interessante quanto o compartilhamento.

Nos protocolos critográficos descritos a seguir considera-se um grupo cíclico $\mathbb{G}$ de ordem prima p onde o problema de Decisão de Diffie-Hellman (BONEH, 1998) é difícil. Como na versão atual do sistema CIVIS, aqui considera-se o grupo multiplicativo de inteiros Z_p^* onde $p = 2q + 1$, onde p e q são dois números primos.

2.1.1 Criptografia de chaves assimétricas

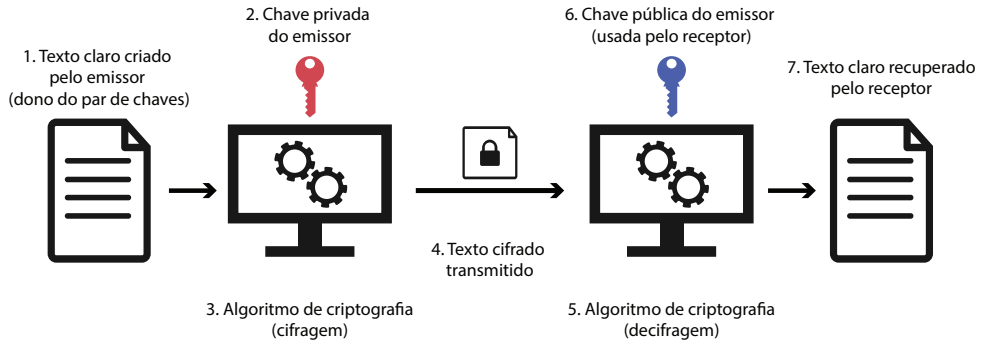
Diferentemente do modelo de criptografia simétrica, onde ambas as chaves utilizadas (para cifrar e decifrar uma informação) possuíam os mesmos valores, o esquema de criptografia de chaves assimétricas as utiliza com valores diferentes uma da outra. Estas chaves não são mais denominadas apenas como chave secreta, e sim como chaves pública e privada. (GATHEN, 2015)

Ambas as chaves são valores numéricos relacionados matematicamente, onde a chave privada é responsável por gerar o valor da chave pública. Assim, cada uma possui a capacidade de desfazer qualquer ação da outra chave. Por exemplo, uma informação cifrada com uma chave pública será decifrada apenas pela chave privada correspondente a ela.

Cada integrante inserido em um conjunto de participantes que aplique a criptografia assimétrica deve gerar seu par de chaves, e compartilhar sua chave pública aos demais membros do conjunto. Qualquer participante que possuir esta chave pública poderá compartilhar informações de forma cifrada com o integrante que possui a chave privada correspondente à chave pública recebida, onde apenas este poderá decifrar a mensagem recebida, como apresentado na Figura 2.

A criptografia de chaves assimétricas também possui requisitos que devem ser atendidos,

Figura 2 – Modelo simplificado de criptografia assimétrica



Fonte: Criação do autor

ainda de acordo com (STALLINGS, 2008) e (GATHEN, 2015). São eles:

1. Um oponente que conheça a chave pública de determinado participante não deve ser capaz de determinar o valor da chave privada correspondente a esta;

2. A chave privada utilizada por cada participante deve ser mantida de forma segura por quem a possuir. Uma vez que algum oponente tiver acesso à chave, toda a segurança da comunicação é comprometida;

Desta forma, é possível manter o esquema de criptografia de chaves assimétricas em um conjunto com diversos participantes, já que o compartilhamento de chaves não é mais um problema, desde que apenas chaves públicas sejam compartilhadas. Por utilizar uma chave que pode ser compartilhada desta forma, o esquema também é chamado de criptografia de chave pública.

Este trabalho utiliza a criptografia de chave pública na implementação do protocolo de geração de credenciais de forma distribuída. Entretanto, é necessário detalhar ainda uma outra vertente da criptografia, fundamental para a realização de procedimentos descentralizados dentro do protocolo. Esta é a criptografia limiar, que será abordada na Seção 2.2.

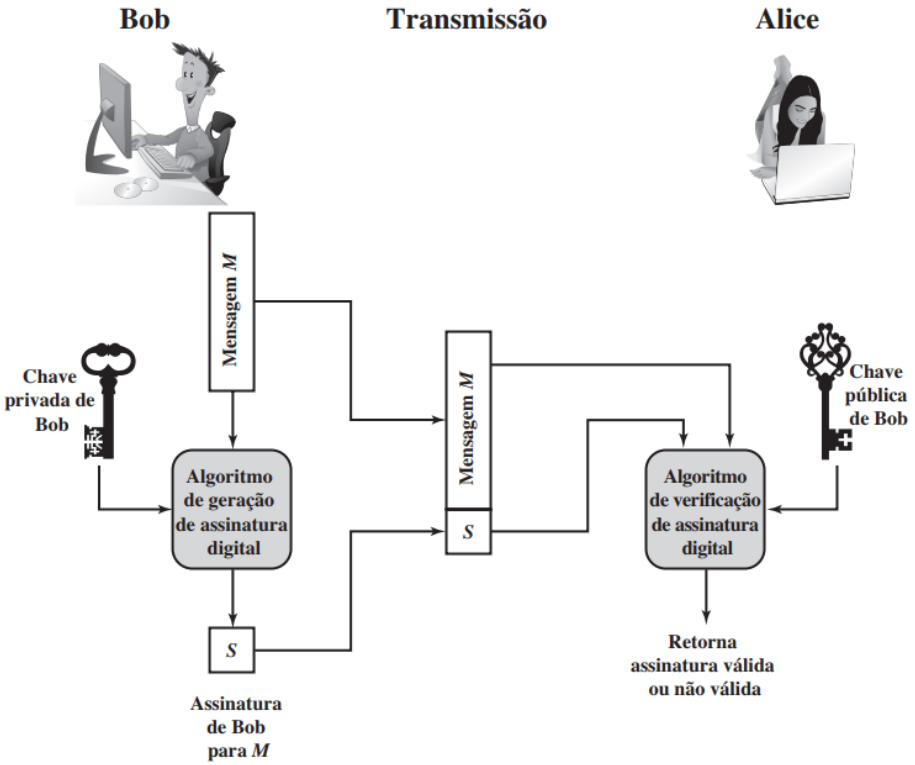
2.1.2 Assinatura digital

Outro mecanismo criptográfico são as assinaturas digitais. Estas oferecem um conjunto de capacidades de segurança que seria difícil de implementar de qualquer outra maneira. Neste trabalho, focaremos nas capacidades de autenticação e integridade. Diferente da codificação, que tem como objetivo primordialmente a confidencialidade de uma informação, uma assinatura digitai tem como objetivo primordial assegurar a integridade de uma informação, garantindo ainda uma forma de autenticação. (STALLINGS, 2008)

A Figura 3 representa um modelo genérico de criação e uso de assinaturas digitais. Um dos participantes pode gerar uma assinatura, utilizando sua chave privada e a mensagem enviada, e inclui essa assinatura na mensagem a ser enviada. Qualquer outro usuário que receber

a mensagem pode verificar a assinatura, e se esta condiz com a mensagem enviada ou com o remetente correto, utilizando a própria mensagem recebida, com a assinatura, e a chave pública do remetente.

Figura 3 – Modelo simplificado de criptografia assimétrica



Fonte: (STALLINGS, 2008)

Este processo garante que a mensagem foi enviada e assinada pelo remetente, que no caso da Figura 3 é representado por Bob. Isso porque cada assinatura é relacionada diretamente a apenas uma chave privada e a uma mensagem. Assim, como apenas Bob possui acesso a sua chave privada, e nenhum outra mensagem pode ser inserida no lugar da mensagem dele e se manter compatível com a assinatura, nenhum outro participante deste processo pode alterar a mensagem enviada por Bob à Alice. Portanto, as principais características de uma assinatura digital podem ser listadas como:

- verificar o autor da assinatura;
- autenticar o conteúdo no momento da assinatura;
- ser verificável por terceiros.

O processo detalhado nessa subseção trata da visão geral sobre assinaturas, onde foi exemplificado seu uso de forma individual. Entretanto, a mesma pode ser usada em conjunto com outras técnicas de criptografia, como a codificação e a criptografia limiar. Estas demais tecnicas potencializam as capacidades de segurança do mecanismo, e por isso são utilizadas no

protocolo de geração de assinaturas digitais de forma distribuída de (WANG; ZHANG; FENG, 2005).

2.2 Criptografia limiar

A criptografia é um mecanismo que auxilia a proporcionar segurança em um sistema. Entretanto, para que a segurança seja mantida é necessário que os participantes envolvidos em seu processo sejam confiáveis. Se algum destes agir de forma inapropriada, como compartilhando indevidamente o valor de alguma chave a qual tem acesso, toda a segurança é debilitada.

Para minimizar tal vulnerabilidade utiliza-se a criptografia limiar, que consiste de um conjunto de procedimentos criptográficos distribuído entre membros de um conjunto, como proposto por (CRAMER; GENNARO; SCHOENMAKERS, 1997) em sua proposta limiar do criptossistema *El Gamal*.

A criptografia limiar é utilizada quando se trabalha com a idea de um conjunto, onde nenhum de seus membros deve possuir informações privilegiadas em relação aos demais, de acordo com (DESMEDT, 1987). Assim, não deve existir sistemas de hierarquia verticais entre os participantes do conjunto, já que nenhum participante pode ser considerado melhor ou mais importante que outro.

Desta forma, é esperado que o acesso a qualquer informação compartilhada entre membros de um conjunto não seja limitada por algum membro em específico. Ou seja, se uma informação deve ser decifrada, por exemplo, não deve ser possível que apenas um membro possa impedir o procedimento. Para tal, é utilizado a ideia de um esquema de compartilhamento limiar de informações, como o proposto por (SHAMIR, 1979).

Em um conjunto de n participantes, é possível compartilhar uma informação secreta a todos os membros, em forma de frações de segredo. Cada fração de segredo será entregue a um participante, até que todos tenham recebido uma. A informação secreta compartilhada só poderá ser decifrada caso uma quantidade mínima de t participantes colaborarem entre si. Neste esquema temos n como o número total de membros do conjunto e t como o limiar, ou limite mínimo, de participantes para recuperação de uma informação. Estes parâmetros são denominados como definições de um esquema (t,n)-limiar.

Esse esquema assume que, dentro do conjunto definido, podem existir membros corruptíveis, que tentem infringir a organização do conjunto, como tentando impedir que um segredo seja recuperado ou decifrar a mensagem sem o consentimento dos demais. No entanto, enquanto estes sejam menos que t participantes, a segurança do sistema não será impactada. O tamanho do limiar t utilizado varia para cada algoritmo utilizado.

Cada algoritmo é baseado em um problema de compartilhamento seguro de um segredo. Este problema é o que deve ser resolvido pelo algoritmo ao decifrar a informação compartilhada

ao conjunto. Neste trabalho, o esquema utilizado é baseado no proposto por (SHAMIR, 1979), que utiliza a interpolação polinomial como problema de compartilhamento.

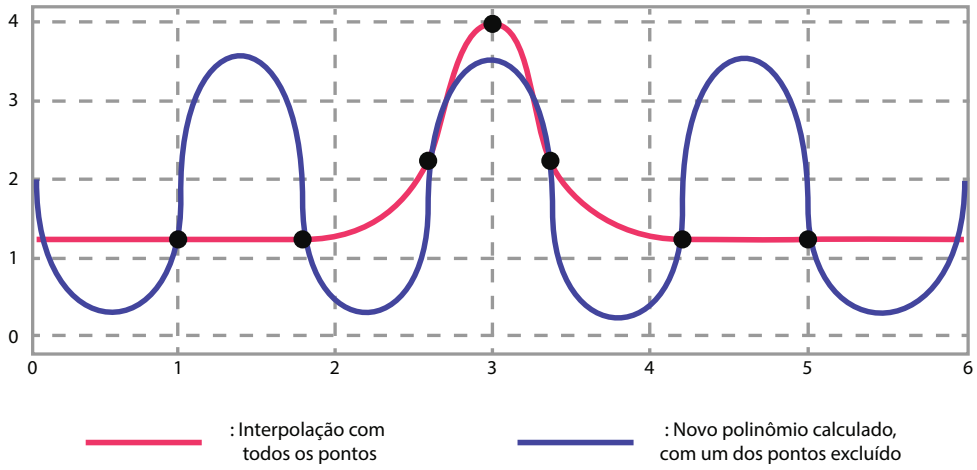
2.2.1 Interpolação polinomial de *Lagrange*

Um polinômio é uma expressão matemática que representa uma soma de coeficientes multiplicados por potências de variáveis não determinadas. Um polinômio qualquer pode ser definido como $a_{t-1}x^{t-1} + \ldots + a_2x^2 + a_1x + a_0$, dada uma quantidade t de coordenadas.

Todo polinômio pode ser representado em um plano cartesiano, em forma gráfica, onde cada coordenada presente nele será representada por um ponto no plano, e o polinômio é representado como uma curva que toca em todos os pontos do plano. Portanto, um determinado polinômio, representado como um $f(x)$, deve ser capaz de atingir o valor de qualquer uma de suas coordenadas, dado um valor de x. O processo de determinar o polinômio $f(x)$ por meio de coordenadas é chamado interpolação polinomial.

Caso uma ou mais coordenadas de um polinômio sejam invalidadas, é possível recalcular o polinômio gerador das coordenadas válidas, resultando em um novo polinômio $f(x)$, que será representado por outra curva no plano cartesiano, onde apenas as coordenadas válidas serão contidas nesta curva.

Figura 4 – Interpolação polinomial de polinômio original e reconstruído



Fonte: Criação do autor

O método de interpolação polinomial de *Lagrange* utiliza uma quantidade t de coordenadas matemáticas dentro do formato (x, y), onde $y = f(x)$. Cada coordenada t deve ser única. Caso haja uma colisão de coordenadas, ou seja, uma coordenada t onde os valores de y são iguais para valores de x diferentes, uma das coordenadas deve ser recalculada. O objetivo é que, utilizando cada coordenada t, seja possível obter o polinômio $f(x)$ gerador de todas as coordenadas. O

polinômio de Lagrange será definido em (2.1):

$$f(x) = \sum_{s=0}^{t} y_s \prod_{j=1, j\neq s}^{t} \frac{(x - x_j)}{(x_s - x_j)} \tag{2.1}$$

O polinômio de *Lagrange* $f(x)$ possui grau equivalente a $t-1$, onde t é a quantidade total de coordenadas utilizadas. Todos os polinômios $f(x)$ gerados por esse algoritmo seguirão o padrão $f(x) = a_{t-1}x^{t-1} + \ldots + a_2x^2 + a_1x + a_0 \mod p$.

O algoritmo de interpolação polinomial de *Lagrange* é considerado determinístico, pois dado um grupamento de coordenadas, o polinômio gerado pela fórmula será sempre o mesmo para o grupamento informado, e nunca grupamentos de coordenadas diferentes possuirão o mesmo polinômio. Este esquema possui um vasto volume de especificações, que não são de interesse para a continuidade deste trabalho, mas podem ser encontrados com mais detalhes em (JR, 2002).

A partir do método de interpolação polinomial de *Lagrange*, um modelo de compartilhamento de segredo limiar pode ser especificado. O primeiro modelo proposto foi o de *Shamir*, detalhado a seguir.

2.2.2 Compartilhamento de segredo de *Shamir*

A proposta do esquema de compartilhamento de segredo de *Shamir* (SHAMIR, 1979), é utilizar um segredo dividido em n frações, e ao menos t dessas frações serão necessárias para recuperar o segredo. A base do funcionamento deste esquema é a utilização de interpolações polinomiais para recuperar o segredo dentro do limiar t.

O segredo compartilhado nesse esquema é inserido em um polinômio de *Lagrange*, gerado aleatoriamente. Serão geradas frações de segredo no formato $y_i = f(x_i)$, onde x_i será um identificador para cada membro; e y_i o resultado do polinômio aleatório, em que o segredo foi inserido, para cada um dos membros.

Os procedimentos para o compartilhamento de segredo são dados da seguinte forma:

1. Devem ser definidos o segredo a ser compartilhado, chamado S, e os parâmetros do esquema (t,n)-limiar, onde n é o número de participantes do conjunto e t é o limiar para recuperar o segredo. Um número primo p deve ser definido também, para que todas as operações realizadas sejam mantidas dentro do grupo multiplicativo Z_p;

2. É selecionado, aleatoriamente, um polinômio de Lagrange $f(x)$ com grau $t-1$. Em seu termo constante a_0 deve ser inserido o segredo S;

3. Cada membro será identificado por um x_i, e calcula-se $y_i = f(x_i)$, onde y_i será a fração de segredo recebida por cada participante, individualmente.

Para a recuperação do segredo, é necessário que ao menos t participantes realizem uma interpolação polinomial de *Lagrange*, inserindo as frações de segredo recebidas. Se menos de t participantes realizarem tal processo, ou alguma das frações enviadas estiver fraudada ou for inválida, o resultado da interpolação será diferente do esperado, resultando em um segredo diferente do inserido inicialmente. Isso não altera o segredo, já que o mesmo continua podendo ser recuperado caso ao menos t participantes insiram frações válidas.

O procedimento para a recuperação do segredo é baseado em recuperar o valor do polinômio $f(0)$, pois este retornará apenas o valor constante de $f(x)$. O valor de S poderá ser recuperado por (2.2):

$$f(0) = \sum_{s=0}^{t} y_s \prod_{j=1, j\neq s}^{t} \frac{(0 - x_j)}{(x_s - x_j)} \mod p \tag{2.2}$$

Este esquema de compartilhamento de segredo se mostra resistente contra a presença de usuários maliciosos inseridos no conjunto, impedindo que um membro não confiável obtenha o valor de S compartilhado aos participantes.

2.2.3 Compartilhamento de segredo verificável de *Pedersen*

O esquema proposto por *Shamir* funciona bem para o compartilhamento de um segredo de forma segura, mas possui como desvantagem a falta de uma verificação sobre cada uma das frações. Caso um dos membros do conjunto seja corrupto, ele não terá acesso ao segredo, mas os outros participantes não terão como saber quem entre eles enviou uma parte de segredo falsa. Para isso, existe o esquema de compartilhamento de segredo verificável, proposto por (PEDERSEN, 1991a).

Para compartilhar um segredo S, o emissor do segredo deve gerar aleatoriamente dois polinômios de *Lagrang*e de grau t, chamados $f(x)$ e $d(x)$, de forma que $f(0) = S$, ou seja, o segredo ficará contido no termo constante do polinômio $f(x)$. Desta vez, deverá ser gerado um segredo aleatório, que denominaremos R, e esse será o termo contido em $d(0)$, o termo constante do polinômio $d(x)$.

Neste esquema, devem ser usados um elemento gerador do subgrupo multiplicativo $g \in \mathbb{Z}_p^*$, pertencente ao grupo definido $\mathbb{Z}_p$, e o elemento h, também presente em $\mathbb{Z}_p$, definido como a chave de compartilhamento do conjunto de participantes. Para ambos os segredos utilizados todos os participantes receberão uma fração P_i, que contém uma coordenada de cada polinômio. Nesse caso, cada participante receberá um $S_i = f(i) \mod p$ e um $R_i = d(i) \mod p$.

Um termo de verificação, para todos os participantes, também deverá ser gerado pelo emissor do segredo. Este termo, que será denominado E, e é definido como $E_k = g^{f_k} h^{d_k}$, onde k são todos os identificadores de membros do conjunto, de 0 até t. Ou seja, $k = 0, \ldots, t$, todos

utilizados como coeficientes dos polinômios $f(x)$ e $d(x)$ em E. Logo, E_k é um conjunto de valores de E, assumindo um valor para cada par de coeficientes dos polinômios utilizados.

O termo E_k é definido como um *Pedersen Commitment* (CHAUM; PEDERSEN, 1992). Este procedimento criptográfico permite que uma determinada informação seja entregue a um participante, sem que esse tenha a capacidade de revelar imediatamente a informação, mas que em algum momento depois isso possa ocorrer. Isso garante que nenhum participante poderá alterar as variáveis de verificação utilizadas neste protocolo, por exemplo, garantindo que todos possam verificar os demais compartilhamentos de frações de segredo.

A verificação de frações recebidas são feitas por cada participante, que testa a igualdade entre suas frações de segredo e as variáveis de verificação publicadas, na forma (2.3):

$$g^{S_i} h^{R_i} = \prod_{k=0}^{t} E_k^{i^k} \tag{2.3}$$

Se a verificação não for positiva, o participante P_i que verificou tal falha envia um comunicado a todos os participantes sobre a verificação que fez. Se o compartilhamento do emissor for determinado como falho em mais de t participantes, esse membro é desqualificado e retirado do conjunto. Caso contrário, o emissor deve compartilhar as frações de segredo corretas aos membros que detectaram falhas, ou também será desclassificado.

É definido um conjunto chamado Λ_G, que contém todos os membros não desclassificados. Caso este possua menos membros que o conjunto original, o segredo S pode ser reconstruído a partir de (2.4):

$$S = \sum_{i \in \Lambda_G} \lambda_{i,\Lambda_G} S_i \mod p \tag{2.4}$$

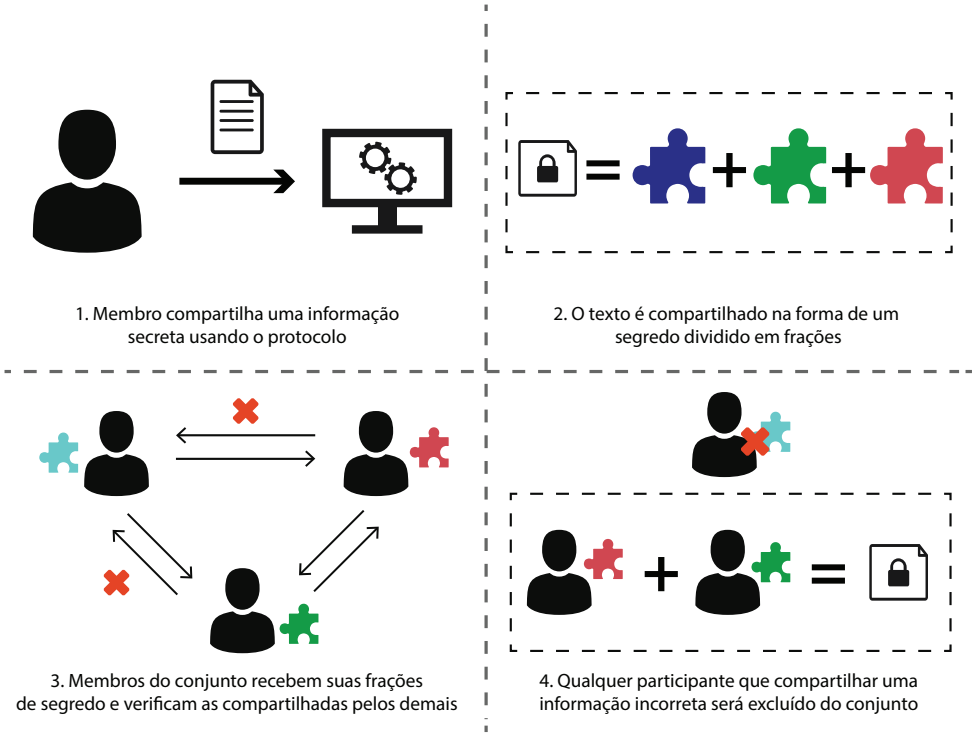
onde λ_{i,Λ_G} é o coeficiente de interpolação de Lagrange (JR, 2002). Assim o polinômio que guarda o segredo S será reconstruído, e não sofrerá com a falta dos membros desclassificados na etapa de verificação.

A organização dos procedimentos expostos para o compartilhamento de segredo verificáveis de *Pedersen* podem ser detalhados, e forma mais resumida, como:

1. Devem ser definidos o segredo a ser compartilhado, chamado S, e os parâmetros do esquema (t,n)-limiar. Um número primo p deve ser definido, e devem ser selecionados um gerador do subgrupo multiplicativo $g \in \mathbb{Z}_p^*$ e uma chave de compartilhamento h, onde $g, h \in \mathbb{Z}_p$;

2. São selecionados, aleatoriamente, dois polinômios de Lagrange, $f(x)$ e $d(x)$, com grau t. Devem ser inseridos o segredo S e o valor aleatório R, em $f(0)$ e $d(0)$, respectivamente;

Figura 5 – Exemplo de distribuição de frações de segredo verificáveis



Fonte: Criação do autor

3. Cada membro P_i deverá receber as frações $f(i) = S_i$ e $d(i) = R_i$ individualmente, e $E_k = g^{f_k} h^{d_k}$ deve ser publicado a todos os participantes;

4. Individualmente, todos os participantes deverão verificar as frações recebidas, comparando se está de acordo com (2.3). Caso $t+1$ membros identificarem falha no compartilhamento, o emissor deve ser retratado;

5. Se algum membro for retirado do conjunto, o polinômio deverá ser recalculado por a partir de (formula reconstrução S).

O esquema de Pedersen adiciona mais segurança ao compartilhamento de segredo em conjunto, mas seu funcionamento é limitado a um emissor compartilhando seu segredo aos demais participantes. A partir do seu esquema de segredo verificável, foram realizadas expansões para esta proposta.

2.2.4 Compartilhamento de segredo aleatório distribuído

O protocolo de *Pedersen* é centralizado em um dos membros do conjunto, que age como um emissor e tem poder para criar um segredo e compartilhá-lo com os demais. Entretanto, o ideal é que a etapa de escolha do valor a ser compartilhado não seja determinado por apenas uma pessoa, pois isso recai na possibilidade dele ser um membro malicioso, mesmo que isso possa ser verificado. Para tanto, uma expansão para o modelo de *Pedersen* é a criação de um segredo pelo próprio conjunto. (PEDERSEN, 1991b)

Esta expansão propõe a utilização do protocolo de *Pedersen* para cada participante, compartilhando um segredo $S_0^{(i)}$ e um número aleatório $R_0^{(i)}$, ambos conhecidos apenas por seus respectivos criadores, e compartilhados a todos os membros como $S_j^{(i)}$ e $R_j^{(i)}$, onde o sobrescrito (i) indica o participante que receberá cada termo. As frações $S_j^{(i)}$ e $R_j^{(i)}$ atuarão como frações intermediárias de segredo, uma vez que serão utilizadas para gerar as verdadeiras frações de segredo utilizadas por cada participante.

Após todas as frações de segredo $S_j^{(i)}$ e $R_j^{(i)}$, geradas por cada participante, terem sido distribuídas e verificadas corretamente, cada participante não desqualificado na etapa de verificação do protocolo de Pedersen deverá computar suas frações finais do segredo. Estas frações serão utilizadas para descobrir o segredo gerado em conjunto por todos. Para computar suas frações, cada participante deve calcular o somatório de todas as frações de segredo recebidas, nos formatos (2.5) e (2.6):

$$S_i = \sum_{j \in \Lambda_G} S_i^{(j)} \mod p \tag{2.5}$$

$$R_i = \sum_{j \in \Lambda_G} R_i^{(j)} \mod p \tag{2.6}$$

As variáveis de verificação de cada compartilhamento de segredo também devem utilizadas para computar as novas variáveis de verificação do segredo compartilhado, no formato (2.7):

$$E_k = \prod_{j \in \Lambda_Q} E_k^{(j)}, k = 0, \ldots, t \tag{2.7}$$

A organização dos procedimentos expostos para o compartilhamento de segredo aleatório distribuído podem ser detalhados, e forma mais resumida, como:

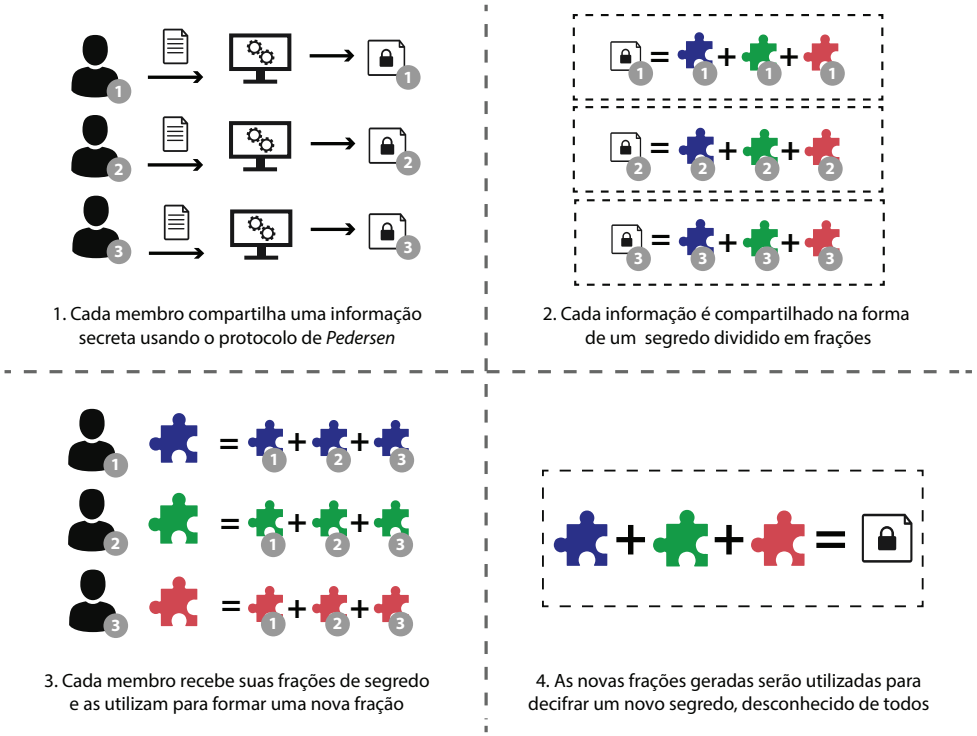
1. Cada participante deve utilizar o protocolo de compartilhamento de segredo verificável de *Pedersen*, criando um segredo $S_0^{(i)}$ e utilizando um número aleatório $R_0^{(i)}$, conhecido apenas por este participante;

2. Cada membro P_i deverá receber as frações $f(i) = S_j^{(i)}$ e $d(i) = R_j^{(i)}$ individualmente, criando um conjunto de frações de segredo recebidas. Cada parte recebida deverá ser utilizada para computar as frações do segredo do conjunto, no formato $S_i = \sum_{j \in \Lambda_G} S_i^{(j)}$ mod p e $R_i = \sum_{j \in \Lambda_G} R_i^{(j)}$ mod p;

3. Todas as variáveis de verificação $E_k = g^{f_k} h^{d_k}$ devem ser publicadas a todos os participantes. Elas serão utilizadas para calcular as variáveis de verificação de segredo do conjunto, no formato $E_k = \prod_{j \in \Lambda_Q} E_k^{(j)}, k = 0, \ldots, t$.

Este esquema proporciona a criação de um segredo em conjunto, por todos os participantes, de forma descentralizada. O segredo S criado pelo conjunto é definido como $S = \sum_{i \in \Lambda_Q} S_0^{(i)} \mod p$. A Figura 6 apresenta um modelo simplificado do funcionamento deste esquema.

Figura 6 – Criação de frações de segredo compartilhado



Fonte: Criação do autor

2.2.5 Geração distribuída de chaves

Outra expansão para o protocolo de *Pedersen* é o esquema de geração de chave distribuída, proposto por (GENNARO et al., 1999). Este usa como base a própria expansão de compartilhamento de segredo aleatório, gerando de forma descentralizada um par de chaves assimétricas.

Para tal, o protocolo utiliza um polinômio de *Lagrange* público para gerar uma chave pública a partir de um segredo criado no mesmo formato do protocolo de *Pedersen* expandido, detalhado na subseção anterior. O segredo criado de forma distribuída será a própria chave privada, que pertence a todo o conjunto de participantes.

É necessário que todo o protocolo de *Pedersen* expandido seja executado normalmente, mas a partir do compartilhamento de segredo verificável de *Pedersen* será possível determinar um polinômio público para o segredo, definido como $F_k^{(i)} = g^{f_k^{(i)}}$, onde $f_k^{(i)}$ são os coeficientes do polinômio $f(x)$ utilizado. Este procedimento deve ocorrer após as verificações de compartilhamento de cada participante, onde os não desclassificados devem compartilhar seus valores de $F_k^{(i)}$, que deve ser compreendido como o polinômio público utilizado pelo participante durante o esquema de *Pedersen*.

Para realizar a verificação das frações recebidas neste protocolo, cada participante deverá comparar se o gerador do subgrupo multiplicativo $g \in \mathbb{Z}_p^*$ do grupo $\mathbb{Z}_p$, quando elevado à fração de segredo recebida, equivale ao produtório dos polinômios públicos do participante que emitiu tal fração de segredo, elevado ao identificador do participante verificador. Isso pode ser descrito em (2.8):

$$g^{S_j^{(i)}} = \prod_{k=0}^{t} (F_k^{(i)})^{j^k}. \tag{2.8}$$

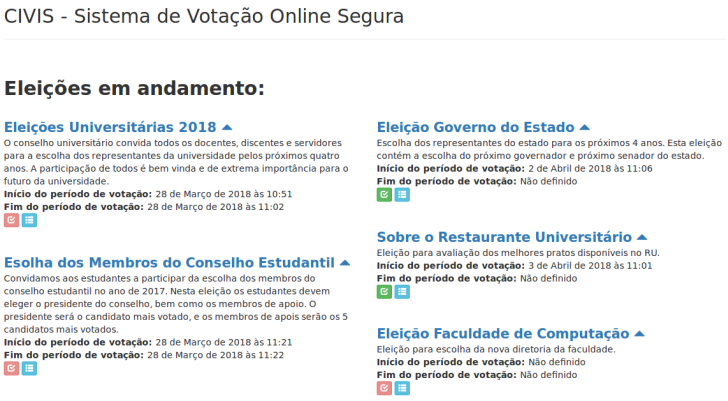
Todos os participantes computam a verificação final de polinômio público, onde $F_k = \prod_{j \in \Lambda_G} F_k^{(j)}$, onde $k = 0, \ldots, t$ e o valor de $F_0 = g^S$, que será definido como a chave pública do conjunto.

2.3 O Sistema de Votação CIVIS

O sistema de votação via *Internet* CIVIS (ARAÚJO; NETO; TRAORÉ, 2018) foi desenvolvido no Laboratório de Segurança e Criptografia Aplicada (LabSC - UFPA), sendo o primeiro sistema a implementar o protocolo criptográfico para votação digital proposto por (ARAÚJO et al., 2010). Este protocolo estabelece uma votação via *Internet* segura e resistente à coerção, utilizando credenciais geradas com base em primitivas criptográficas.

Desenvolvido na linguagem de programação *Python* (PYTHON, 2018), utilizando o framework de desenvolvimento *web Django* (DJANGO, 2018), o sistema e possui seus mecanismos criptográficos implementados em *JavaScript* (JAVASCRIPT, 2018). O algoritmo criptográfico de chave assimétrica El Gamal (GAMAL, 1984)) é utilizado para cifragem e decifragem de suas informações. A Figura 7 apresenta a página inicial do sistema CIVIS.

Figura 7 – Página inicial do sistema CIVIS



Fonte: Captura de tela do sistema

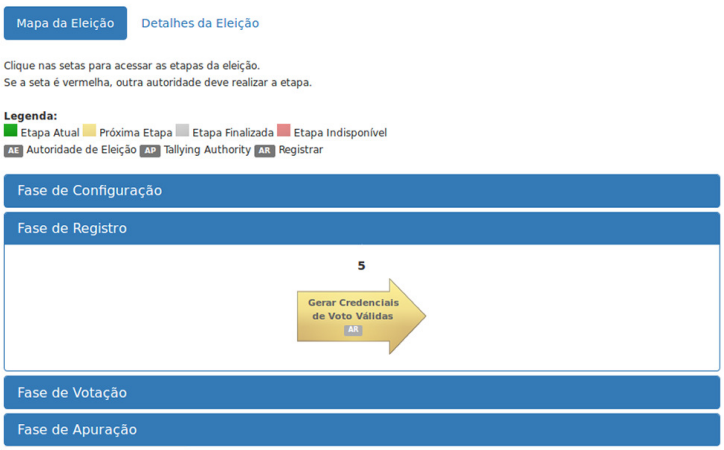
O sistema é organizado em eleições, onde cada eleição possui um conjunto de fases, e autoridades são responsáveis por estas. Três autoridades atuam numa eleição; a autoridade de eleição, responsável por configurar a eleição num geral; a autoridade de registro, responsável por

gerar credenciais para os participantes; e a autoridade de apuração, responsável pelos votos de uma eleição.

Para que uma eleição seja realizada usando o CIVIS, é necessário que todas as fases sejam concluídas. Para isso, deverão ser definidas as autoridades de configuração, registro e apuração.

A primeira fase, a de configuração, é onde as autoridades irão gerar suas chaves criptográficas. Nela também serão definidas as disputas presentes na eleição. Na fase de registro, representada na Figura 8, são definidos quais votantes poderão participar da eleição, e a eles são entregues credenciais únicas e intransferíveis. A fase de votação é o momento em que os eleitores poderão enviar seus votos, que será criptografado pelo sistema até o momento da apuração. Por fim, a fase de apuração é onde os votos serão descriptografados e o resultado contabilizado. Após o resultado ser definido, ele é publicado a todos os participantes.

Figura 8 – Mapa de fases do sistema, em destaque a Fase de Registro



Fonte: Captura de tela do sistema

2.3.1 Credenciais

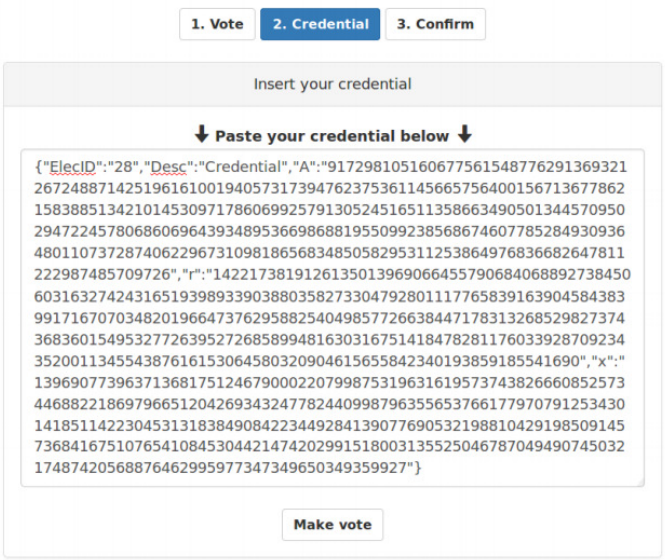
As credenciais são um mecanismo utilizado pelo sistema para possibilitar a resistência à coerção de votantes. Elas são geradas unicamente pela autoridade de registro, e possuem uma estrutura matemática específica. Uma credencial é única para cada votante, impossibilitando que uma mesma credencial possa ser utilizada por dois votantes diferentes. Cada credencial recebida será utilizada para enviar votos para o sistema, que serão contabilizados pelo sistema durante a apuração.

Entretanto, as credenciais do sistema CIVIS são utilizadas também como uma estratégia de resistência ao ataque de coerção de votantes. O sistema possibilita que um votante, caso utilize uma credencial que não foi recebida por ele, ainda possa enviar um voto para o sistema, mas esse voto não será considerado durante a apuração. Esse processo acontece sem qualquer indicação no sistema, impedindo que qualquer usuário do sistema além do votante possa diferenciar o

envio de voto que utilize uma credencial recebida pelo votante, ou uma forjada. Desta forma, é possível que o votante possa enviar votos que não serão contabilizados, utilizando uma credencial que será chamada credencial inválida. Esta se contrapõe à credencial recebida unicamente pelo votante, que é uma credencial válida. Assim, qualquer credencial será considerada inválida se possuir valores diferentes da gerada para aquele votante específico, pelo sistema.

Essa estratégia permite que um usuário possa enganar um impositor que o esteja coagindo a votar numa determinada opção de uma disputa. O votante poderá enviar um voto utilizando uma credencial falsa, e este não será computado pelo sistema, mas o sistema não informa o voto inválido. Assim, apenas o votante saberá que seu voto é inválido, até a fase de apuração.

Figura 9 – Votante inserindo sua credencial no sistema



Fonte: Araújo *et al.* (2018)

As credenciais são geradas pela autoridade de registro. Elas possuem uma estrutura matemática composta de três valores (2.9):

$$(A = (g_1 g_3^x)^{\frac{1}{y+r}}, r, x), \tag{2.9}$$

onde $g_1, g_3 \in \mathbb{Z}_p^*$ são dois geradores aleatórios, $r, x \in \mathbb{Z}_p^*$ são dois números aleatórios e y é a chave privada da autoridade de registro (ARAÚJO; NETO; TRAORÉ, 2018). O sistema possui apenas uma autoridade de registro por eleição, logo a geração de credenciais é centralizada por eleição, a apenas uma autoridade de registro.

Conforme o protocolo em que o CIVIS se baseia, credenciais válidas autorizam os votos que devem ser contados na apuração. Cada credencial válida deve ser gerada por um conjunto de autoridades e entregue ao votante em sigilo. Na versão atual do CIVIS uma única autoridade gera uma credencial válida para cada votante. Dessa forma, o sistema requer que essa autoridade seja confiável. No entanto, tal confiança é uma suposição forte tendo em vista que, se essa autoridade

for maliciosa, ela pode arruinar a segurança do sistema. Por exemplo, revelando credenciais válidas para opressores. A proposta deste trabalho é a implementação de um protocolo de geração de credenciais realizada de forma distribuída, com mais de uma autoridade de registro.

2.4 Considerações finais do capítulo

Este capítulo apresentou o material necessário para a compreensão deste trabalho. Destaca-se, na Seção 2.4, o funcionamento da criptografia limiar, com conceitos fundamentais para o protocolo implementado neste trabalho; e na Seção 2.5.2 o esquema utilizado atualmente no sistema CIVIS para geração de credenciais.

Nos próximos capítulos os conceitos definidos neste capítulo serão utilizados para a implementação do protocolo de geração de credenciais distribuídas, assim como sua proposta de integração com o sistema de votação CIVIS.

3 IMPLEMENTAÇÃO DO PROTOCOLO CRIPTOGRÁFICO PARA GERAÇÃO DISTRIBUÍDA DE CREDENCIAIS

Como descrito na Seção 1.1, a principal moticação dese trabalho é tornar mais seguro o processo de geração de credenciais de votação no sistema CIVIS. De forma a atender tal objeetivo esse trabalho apresenta uma implementação de um esquema que possibilita a geração de credenciais de forma distribuída, que possa ser aplicado ao sistema CIVIS, auxiliando a garantir uma segurança ainda maior durante o registro de participantes numa eleição virtual. Para tal, é necessário utilizarmos os conceitos apresentados na Seção 2.4, onde foram apresentadas as bases teóricas da criptografia limiar.

A organização do capítulo está de acordo com a seguinte sequência: a Seção 3.1 apresenta a visão geral do protocolo a ser implementado, apontando seu esquema de funcionamento e as adequações realizadas para seu funcionamento dentro do sistema de votação; na Seção 3.2 são descritas as nformações necessárias para a implementação, contando com descrições dos requisitos, tecnologias utilizadas e arquitetura geral da implementação; a Seção 3.3.1 propõe uma integração do protocolo implementado ao sistema de votação CIVIS; finalmente, a Seção 3.3 conclui este capítulo.

3.1 O protocolo de Wang, Zhang & Feng (WZF)

Esta seção apresenta o esquema de funcionamento do protocolo de geração de assinaturas digitais distribuídas de WZF (WANG; ZHANG; FENG, 2005), apontando seus passos e como pode ser aplicado para a geração de credenciais distribuídas ao sistema CIVIS.

Vale destacar que este protocolo não se dispõe a gerar credenciais para sistemas de votação, e sim uma assinatura digital gerada por um conjunto limiar. Entretanto, o modelo criptográfico aplicado é equivalente ao utilizado para geração de credenciais no CIVIS. As adequações do protocolo para geração de credenciais são detalhadas na Seção 3.1.2.

3.1.1 O protocolo de WZF

O esquema proposto por WZF é uma versão limiar do protocolo de assinaturas digitais digital curtas de *Boneh & Boyen* (BONEH; BOYEN, 2004). Tal protocolo utiliza originalmente grupos bilineares, mas também funciona em grupos genéricos. Como na versão atual do sistema CIVIS, a descrição a seguir considera o grupo multiplicativo de inteiros Z_p^* onde $p = 2q + 1$, p e q são dois números primos.

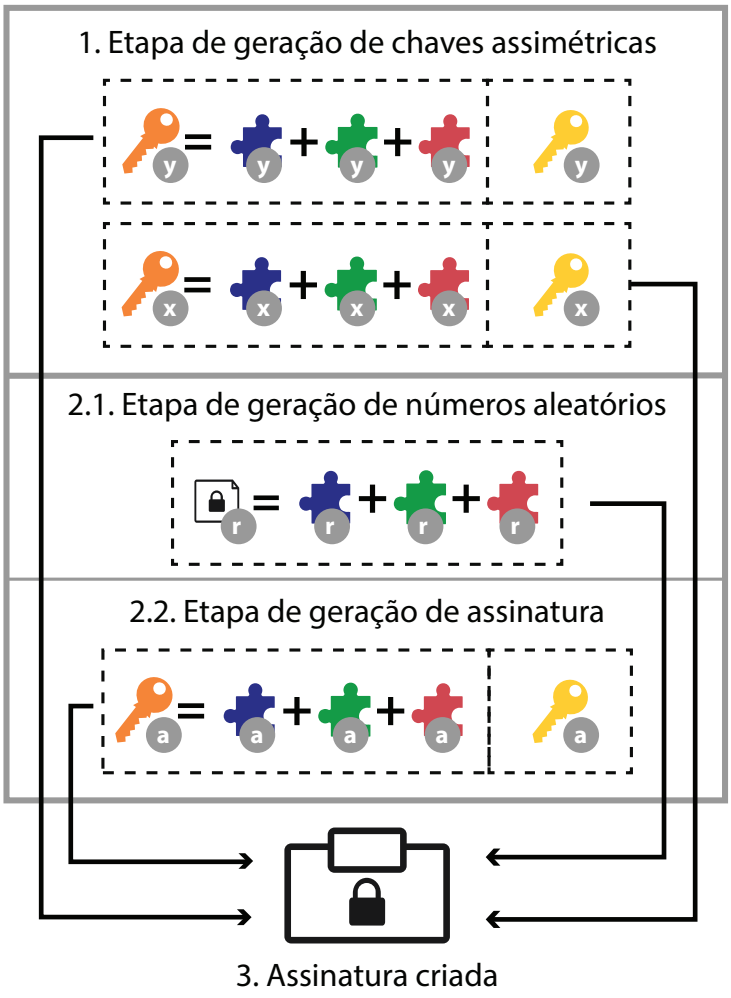
O protocolo de WZF utiliza também os protocolos de *Lagrange*, *Pedersen* e suas expansões, apresentadas anteriormente. Este protocolo gera uma assinatura digital na estrutura

apresentada em (3.1):

$$(\sigma = (g)^{\frac{1}{x+m+ry}}, r). \tag{3.1}$$

Para melhor compreensão, o esquema pode ser dividido em fases, onde cada protocolo visto anteriormente será utilizado devidamente. As fases serão divididas como *setup* e geração de assinatura digital. Cada fase poderá ser dividida em etapas. A fase de *setup* é dividida em geração de chave de compartilhamento; e geração de chaves assimétricas do conjunto. A fase de geração de assinaturas pode ser dividida em geração de números aleatórios; e geração da estrutura matemática necessária para a assinatura.

Figura 10 – Estrutura do esquema proposto por Wang, Zhang & Feng



Fonte: Criação do autor

A Figura 10 apresenta um diagrama que organiza a estrutura do protocolo de WZF. Destaca-se nele as etapas de funcionamento do protocolo e os termos relacionados à geração de uma assinatura digital de forma distribuída.

O protocolo inicia na fase de setup, onde devem ser definidos os valores de chaves utilizadas durante todo o processo de assinatura. Entretanto, é necessário que antes a essa etapa sejam definidos os parâmetros iniciais, sendo estes $\mathbb{Z}_p^*$, como grupo cíclico multiplicativo; e $g \in \mathbb{Z}_p^*$, como gerador de subgrupo.

Tendo estes parâmetros iniciais definidos, é necessário que seja gerada uma chave de compartilhamento h. Esta chave é gerada a partir do gerador do subgrupo multiplicativo $g \in \mathbb{Z}_p^*$, e é utilizada como parâmetro para os algoritmos de cada protocolo de compartilhamento, citados nas seções 2.2.3 a 2.2.5. Esta é a etapa de geração de chave de compartilhamento, dentro da fase de *setup*.

Após a etapa detalhada anteriormente, a etapa de geração de chaves assimétricas é iniciada. Todos os participantes deverão, em conjunto, gerar a dupla de segredos $x, y \in \mathbb{Z}_q$, utilizando o protocolo de geração de chave distribuída (ver Seção 2.2.5). Ambos os termos serão tratados como as chaves privadas do conjunto.

Cada chave privada gerada também possuirá um conjunto de polinômios públicos, denominados $(F_0, \ldots, F_t)$. Os polinômios $F_0^{(x)}, F_0^{(y)}$ são necessários para compor a chave pública do conjunto, sendo nomeados u, v, respectivamente, onde $u = F_0^{(x)} = g^{(x)}$ e $v = F_0^{(y)} = g^{(y)}$. Desta forma, u, v serão as chaves públicas do conjunto.

Com as chaves x, y geradas, a fase de *setup* está concluída. Temos definidos como parâmetros públicos os termos (p, q, g, h, u, v), e como parâmetros privadas os valores (x, y).

A próxima etapa é a de geração de assinatura digital. Esta etapa é iniciada pela geração de um número aleatório $r \in \mathbb{Z}_q$, de forma distribuída e secreta, utilizado como parte da assinatura. Tal valor deve ser gerado utilizando o protocolo de compartilhamento de segredo aleatório distribuído (ver Seção 2.2.4), dado um número n de participantes.

As assinaturas digitais são criptografadas utilizado o expoente parcial $(x+m+ry)^{-1}$ mod p, referente ao elemento σ da assinatura, onde m é o valor a ser assinado digitalmente; e x, y são as chaves privadas criadas anteriormente, no processo de geração de chaves assimétricas distribuídas.

Para realizar o processo de assinatura digital, serão utilizadas variáveis auxiliares a, b. Ambas devem ser recuperáveis, ou seja, podem ter seus valores obtidos mesmo sem o participante ter acesso aos termos que os constroem. Nesse caso, as variáveis serão reconstruídas por qualquer participante, mesmo que estes não possuam as chaves de cada um dos outros membros do conjunto.

O valor de a deve ser gerado utilizando o protocolo de geração de chave distribuída. Assim, cada participante receberá uma fração da chave privada do conjunto, e será disponibilizado um polinômio público $F_0^{(a)}, \ldots, F_t^{(a)}$, onde $F_0^{(a)}$ é a chave pública relacionada à a; e variáveis de verificação $E_0^{(a)}, \ldots, E_t^{(a)}$. Sua recuperação pode ser realizada pela forma $E_0^{(a_i)} = \prod_{k=0}^{t} (E_k^{(a)})^{i^k}$, onde $E_0^{(ai)}$ é a variável de verificação inicial de a_i caso executado o protocolo de compartilha-

mento de segredo verificável de *Pedersen*. Desta forma, tem-se em (3.2):

$$E_0^{(a_i)} = \prod_{k=0}^{t} (E_k^{(a)})^{i^k} = g^{S_0^{(a_i)}} h^{R_0^{(a_i)}}. \tag{3.2}$$

Já o valor de b é referente ao próprio expoente parcial referente ao elemento σ da assinatura, utilizado para a assinatura. Assim, cada participante poderá calcular $b_i = x_i + m + ry_i \mod p$. O valor de cada b_i poderá ser recuperado pela equação (3.3):

$$E_0^{(b_i)} = g^{b_i} h^{R_i^{(b)}} = g^m h^m \prod_{k=0}^{t} (E_k^x (E_k^y)^r)^{i^k}, \tag{3.3}$$

onde $E_0^{(b_i)}$ é a variável de verificação inicial de b_i caso executado o protocolo de compartilhamento de segredo verificável de Pedersen, e $R_i^{(b)}$ é a variável aleatória auxiliar utilizada no mesmo protocolo, que deve possuir valor $R_i^{(b)} = R_i^{(x)} + m + rR_i^{(y)}$.

Todos os participantes que conseguirem recuperar os valores de a, b corretamente serão adicionados ao subconjunto $\Lambda_(a,b)$. Cada participante $P_i \in \Lambda_(a,b)$ deve então compartilhar o valor de $c_i = a_i b_i \mod p$ pelo protocolo de *Pedersen*, onde c será um valor recuperável por meio da variável de verificação $E_0^{(va_i)} = (E_0^{(b_i)})^{a_i} h^{R_i^{(a_i)}}$, equivalente à variável de verificação inicial do protocolo de *Pedersen* executado.

O valor do segredo c será a combinação linear de todos os elementos $c = c_1, \ldots, c_{2t+1}$. O conjunto de participantes que recuperou corretamente o valor de c será dado como $\Lambda_{(c)}$, composto de $2t+1$ participantes. Assim, para qualquer combinação mínima de $t+1$ participantes de $\Lambda_{(c)}$, cada participante poderá computar localmente (3.4):

$$\sigma = (g^a)^{c^{-1}} = (F_0^{(a)})^{c^{-1}} = g^{\frac{1}{x+m+ry}}, \tag{3.4}$$

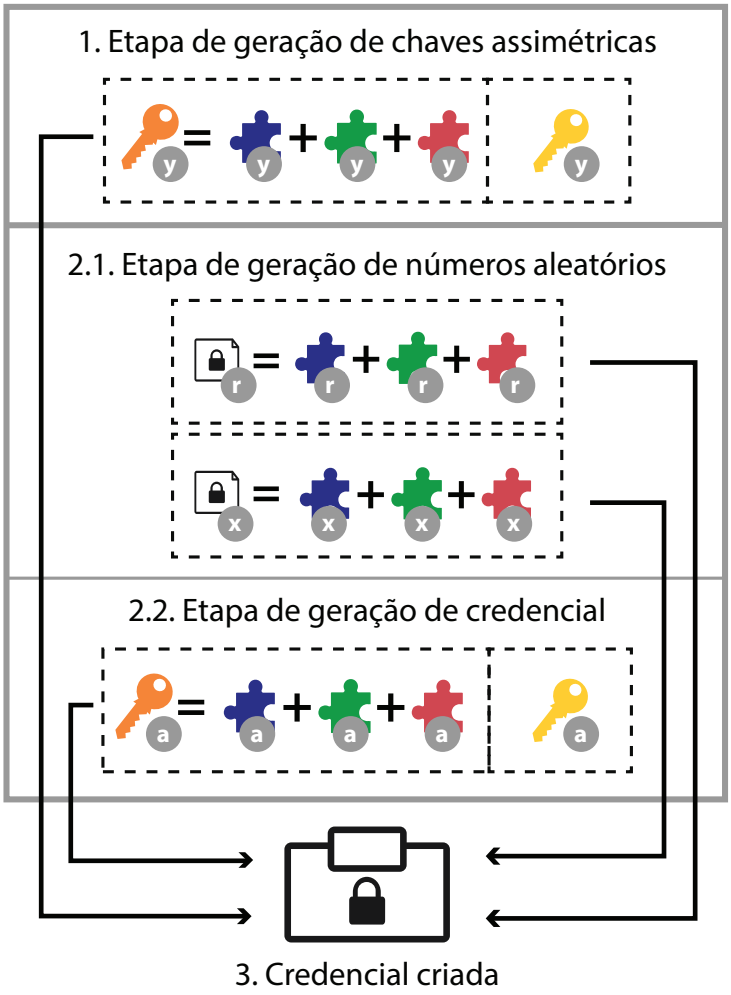
compondo a assinatura final (σ, r).

3.1.2 Adaptação do protocolo de WZF ao sistema CIVIS

O protocolo de WZF foi concebido para geração de assinaturas digitais. O sistema de votação digital CIVIS utiliza um sistema de credenciais, como requerido pelo protocolo de Araujo (ARAÚJO et al., 2010) e este deve seguir o formato matemático (2.9). Entretanto, a estrutura matemática das assinaturas digitais do protocolo de WZF é bastante similar à estrutura de credenciais utilizada no CIVIS. Dessa forma, o protocolo de WZF pode ser utilizado para a geração de credenciais de forma distribuída, mas é necessário realizar uma pequena adaptação para que o mesmo funcione de acordo com o utilizado no sistema de votação via *Internet*.

A Figura 11 apresenta as adaptações ao diagrama que organiza a estrutura do protocolo de WZF. Destaca-se nele as etapas de funcionamento do protocolo e os termos relacionados à

Figura 11 – Estrutura do esquema proposto por Wang, Zhang & Feng, adaptado para geração de credenciais



Fonte: Criação do autor

geração de uma credencial de forma distribuída. Todas as etapas deste esquema serão melhor detalhadas na Seção 3.2, que trata da implementação do protocolo.

Como apresentado na Seção 2.3, as credenciais utilizadas no CIVIS possuem uma estrutura matemática igual a (2.9), onde $g_1, g_3 \in \mathbb{Z}_p^*$ são dois geradores aleatórios, $r, x \in \mathbb{Z}_q$ são dois números aleatórios e y é a chave secreta da autoridade de registro. Esta estrutura difere da gerada como assinatura pelo protocolo de WZF, pois esta retorna os valores de (3.1).

Para que o protocolo de WZF possa ser utilizado no CIVIS, ele deve finalizar retornando uma credencial na estrutura definida por (ARAÚJO et al., 2010). Para adaptá-lo, devem ser considerados os termos em comum dentre o resultado do protocolo de WZF com o de uma credencial gerada no CIVIS. Destaca-se a necessidade de existência de uma chave privada; a geração de dois números aleatórios distribuídos; e o expoente parcial referente ao elemento A da credencial deve atender a estrutura $\frac{1}{y+r}$.

Levando em conta a o protocolo de WZF, a fase de *setup* deve ser alterada apenas na

etapa de geração de chaves assimétricas. A estrutura de credencial de (ARAÚJO et al., 2010) exige apenas um par de chaves assimétricas, portanto apenas será gerado o par de chaves y nesta adaptação. Os parâmetros públicos desta etapa serão p, q, g, h, v e y será uma informação privada do conjunto de autoridades de registro. O par de chaves assimétricas x, u se torna desnecessário, não sendo geradas, pois é necessário o uso de apenas um par, sendo este y, v.

A fase de geração de assinatura digital agora se tornará a fase de geração de credencial, e sofrerá alterações nas etapas de geração de números aleatórios distribuídos e na estrutura matemática de compartilhamento da credencial.

Para a geração de números aleatórios, que no protocolo de WZF é necessária a geração distribuída apenas da variável r, agora terá de ser complementada com a geração de um novo termo aleatório, que será chamado x, e também gerado utilizando o protocolo de *Pedersen* expandido. Note que a variável de valor aleatório x, criada nesta etapa, não possui relação com a chave x utilizada no protocolo de WZF. A nomenclatura de ambas permanece igual para que agora esta esteja adequada ao exigido pelo protocolo de (ARAÚJO et al., 2010).

Ainda na fase de geração de credencial, a etapa relacionada à estrutura matemática de compartilhamento sofrerá alterações referentes ao expoente parcial referente ao elemento σ da assinatura. No CIVIS, o formato deste expoente é $\sigma = \frac{1}{y+r}$, enquanto no protocolo de WZF é utilizado no formato $\frac{1}{x+m+ry}$. Portanto, o termo m se torna desnecessário, tendo em vista que o protocolo adaptado não deve gerar uma assinatura, e sim uma credencial, logo nenhum texto plano precisa ser assinado; e a chave privada x não é mais utilizada, logo também se torna desnecessária e deve ser retirada.

Finalmente, seguindo o proposto para adaptações do protocolo de WZF, o novo resultado gerado é apresentado em (3.5):

$$(\sigma = (g_1 g_3^x)^{\frac{1}{y+r}}, r, x), \tag{3.5}$$

completamente adequado à estrutura matemática de credenciais utilizada no sistema de votação CIVIS.

3.2 Implementação do protocolo de WZF

A seção anterior apresentou o protocolo de WZF, bem como sua adaptação necessária ao CIVIS. A seguir, será apresentada a implementação desse protocolo, onde serão elicitados os requisitos necessários para o desenvolvimento, assim como uma descrição das tecnologias empregadas durante o processo e detalhes de sua arquitetura geral. Serão apresentadas, também, toda a sequência de execução da implementação, e as funções que compõem o protocolo.

3.2.1 Requisitos da implementação

Antes de iniciar a implementação, é fundamental elucidar os requisitos necessários para o desenvolvimento da aplicação. Estes foram divididos em requisitos funcionais e não funcionais.

Os requisitos funcionais descrevem as funcionalidades necessárias para a execução da implementação proposta neste trabalho. O Quadro 1 apresenta os requisitos funcionais utilizados para esta implementação.

Quadro 1 – Requisitos funcionais da implementação

Código	Requisito	Descrição
RF01	Gerar credencial	Os participantes do conjunto devem gerar uma credencial na estrutura requerido pelo CIVIS.
RF02	Gerar polinômio aleatório	O sistema deve gerar aleatoriamente um polinômio.
RF03	Interpolação polinomial de *Lagrange*	O sistema deverá calcular um polinômio utilizando o método de *Lagrange*.
RF04	Compartilhar segredo verificável	Os participantes do conjunto devem compartilhar segredos utilizando o esquema de *Pedersen*.
RF05	Verificar fração de segredo	Os participantes do conjunto devem verificar uma fração de segredo.
RF06	Calcular fração de segredo aleatório	Os participantes do conjunto devem calcular sua fração de segredo a partir de suas demais frações recebidas.
RF07	Gerar polinômio de *Lagrange* público	O sistema deve gerar um polinômio público com as informações compartilhadas pelos usuários.
RF08	Verificar polinômio de *Lagrange* público	Os participantes do conjunto devem verificar um polinômio público.

Fonte: Criação do autor

Já os requisitos não funcionais descrevem características gerais do sistema, e que não estão diretamente ligadas às funcionalidades disponibilizadas pelo protocolo implementado. O Quadro 2 apresenta os requisitos não funcionais utilizados para esta implementação.

3.2.2 Tecnologias empregadas

O esquema implementado possui etapas onde devem ser geradas chaves assimétricas para o conjunto. O próprio processo de criação de credenciais também deve ser considerado como uma etapa onde existe o uso de informações secretas. Portanto, é necessário o processamento local de dados para a execução deste esquema.

A linguagem de programação *JavaScript* (JAVASCRIPT, 2018) foi utilizada para a implementação por realizar computação diretamene no computador no cliente ao invés do servidor.

Quadro 2 – Requisitos funcionais da implementação

Código	Tipo de Requisito	Descrição
RNF01	Desenvolvimento	O protocolo deve funcionar em um ambiente *web*, inseridos em páginas *HTML*
RNF02	Desenvolvimento	O protocolo deve utilizar a metodologia de orientação à objetos em sua arquitetura
RNF03	Desenvolvimento	O protocolo deve ser dividido nos seguintes módulos: Lagrange, Pedersen, Pedersen-extendido, DKG e WZF
RNF04	Desenvolvimento	O protocolo deve utilizar as bibliotecas *JSBN* e *SJCL* para realizar cálculos criptográficos
RNF05	Segurança	O protocolo deve ser executado localmente, no computador cliente, via navegador *web*

Fonte: Criação do autor

Como a linguagem atua no computador cliente, ela permite um processamento completamente local, evitando que as informações secretas que estão sendo utilizadas sejam divulgadas a outros participantes, ou enviadas a algum servidor desconhecido.

Para funcionar em um ambiente *web*, é necessário que os códigos *JavaScript* sejam inseridos em páginas *HTML (HyperText Markup Language)*, para que possam ser executados em um navegador *web*.

Bibliotecas específicas foram utilizadas para a realização de cálculos criptográficos, sendo elas a SJCL (*Stanford JavaScript Crypto Library*), desenvolvida pelo Laboratória de Segurança da Computação na universidade de Stanford (STARK; HAMBURG; BONEH, 2018), a qual foi usada para gerar números aleatórios; e a JSBN (*JavaScript Big Number*), que possibilita a utilização de números do tipo *BigInt* (também chamado como *LongInt*, fundamental para o uso de grandes números inteiros, como os utilizados para criptografia.

Por fim, foram utilizados módulos presentes no próprio sistema de votação CIVIS para a implementação, tendo em vista que o mesmo já implementa a versão limiar do criptosistema El Gamal e portando implementa os algorimos de *Lagrange* e *Pedersen* (veja Seção 2.2).

A utilização do módulo de *Lagrange* e do módulo de *Pedersen* presentes no sistema estão de acordo com o necessário para esta implementação. Entretanto, o módulo de Pedersen disponibilizado no sistema corresponde apenas a parte do definido na Seção 2.2.3, portanto foi necessário ainda implementar partes do protocolo, como definido nas seções anteriores.

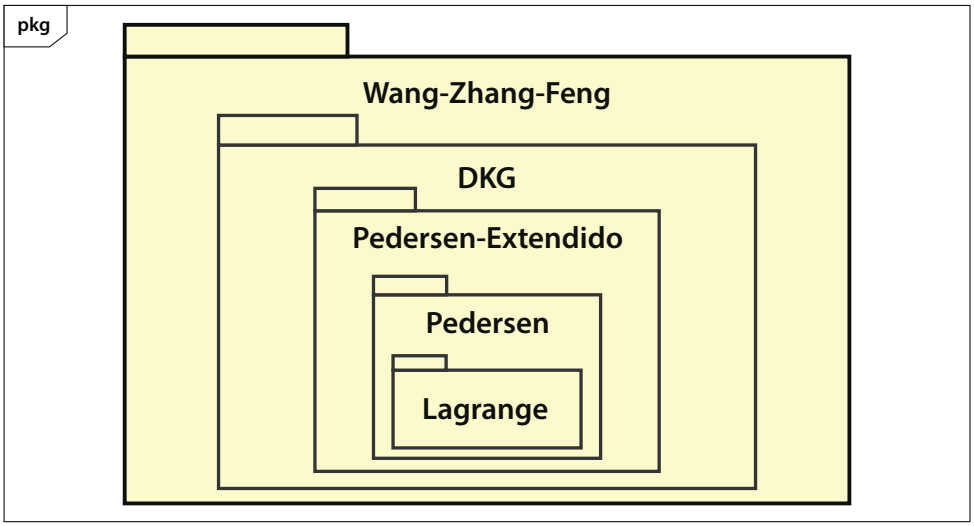
3.2.3 Arquitetura geral

Como descrito na Seção 3.1.1, o esquema de WZF requer um conjunto de protocolos que interagem entre sí. A fim de organizar essa interação, a seguir é definida a arquitetura

da implementação. A arquitetura apresentada nesta seção descreve a organização interna do protocolo de geração distribuída de credenciais implementado. Isso permite uma compreensão ampliada sobre o sistema, garantindo uma melhor visualização de seus componentes.

A metodologia utilizada para o desenvolvimento do protocolo de geração de credenciais distribuídas é orientada a objetos, portanto toda a arquitetura aborda seus elementos considerando as classes utilizadas para a implementação.

Figura 12 – Visão geral do sistema representada em um diagrama de classes



Fonte: Criação do autor

A Figura12 apresenta uma visão geral do protocolo implementado, dividido entre módulos que interagem entre si. Cada módulo representa um protocolo, e possui suas próprias classes e funções para a utilização do sistema. Cada módulo interno funciona de forma independente aos módulos aos quais está mantido.

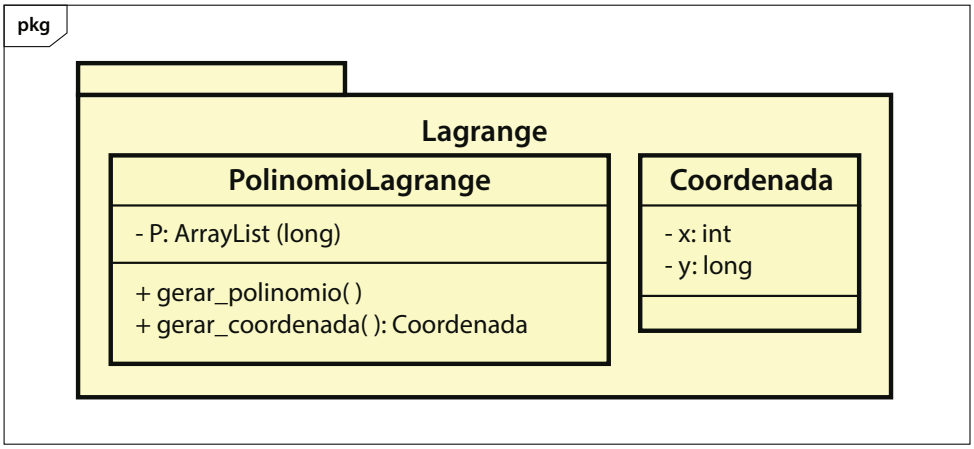
Esta forma de organização foi selecionada por conta de alguns módulos já estarem presentes no CIVIS, como o módulo de *Lagrange*. Por conta disso, sua implementação não foi necessária, sendo apenas reutilizado o código existente no próprio sistema. Entretanto, é necessário detalhar sua estrutura, pois os demais protocolos dependem direta ou indiretamente dele.

O protocolo de *Lagrange* utiliza diversas coordenadas para calcular um polinômio único, onde seus pontos podem ser recuperados por meio de uma interpolação, desde que uma quantidade limiar de pontos seja informado. (NETO, 2016)

A Figura13 apresenta as classes utilizadas no módulo de *Lagrange*. A classe *Polinomio-Lagrange* apresenta um polinômio calculado. Este é composto por um *array* de números inteiros longos, onde cada um destes é um dos termos do polinômio, variando de 0 até t.

Esta classe possui dois métodos: *gerar_polinomio()*, que seleciona um polinômio com fator aleatório; e *gerar_coordenada()*, que calcula e retorna um objeto da classe *Coordenada*.

Figura 13 – Diagrama de classes do módulo *Lagrange*

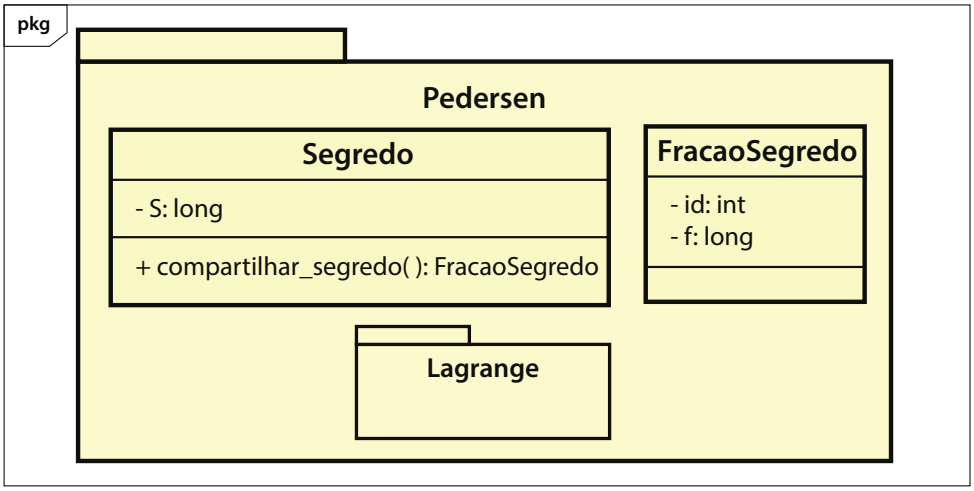


Fonte: (NETO, 2016), adaptado

A classe *Coordenada* abrange as coordenadas de um polinômio de *Lagrange*. A mesma é composta por dois atributos, ambos números inteiros, representados por x, y. Note que y é um valor inteiro longo.

O módulo imediatamente superior ao módulo de *Lagrange* é o de *Pedersen*, ver Figura 12. Este protocolo é responsável pelo compartilhamento de segredo verificável no sistema, utilizando a interpolação polinomial do módulo anterior.

Figura 14 – Diagrama de classes do módulo *Pedersen*



Fonte: Criação do autor

A Figura 14 apresenta seu diagrama de classes. Este módulo possui uma classe *Segredo*, que representa o segredo a ser compartilhado. Seu valor está contido no atributo S, representado como um inteiro longo, e possui como método $compartilhar\_segredo()$, que resulta em um objeto do tipo *FracaoSegredo*.

A classe *FracaoSegredo* abrange todas as frações geradas a partir de um compartilhamento. A mesma é composta por dois atributos, ambos números inteiros: f, representando o

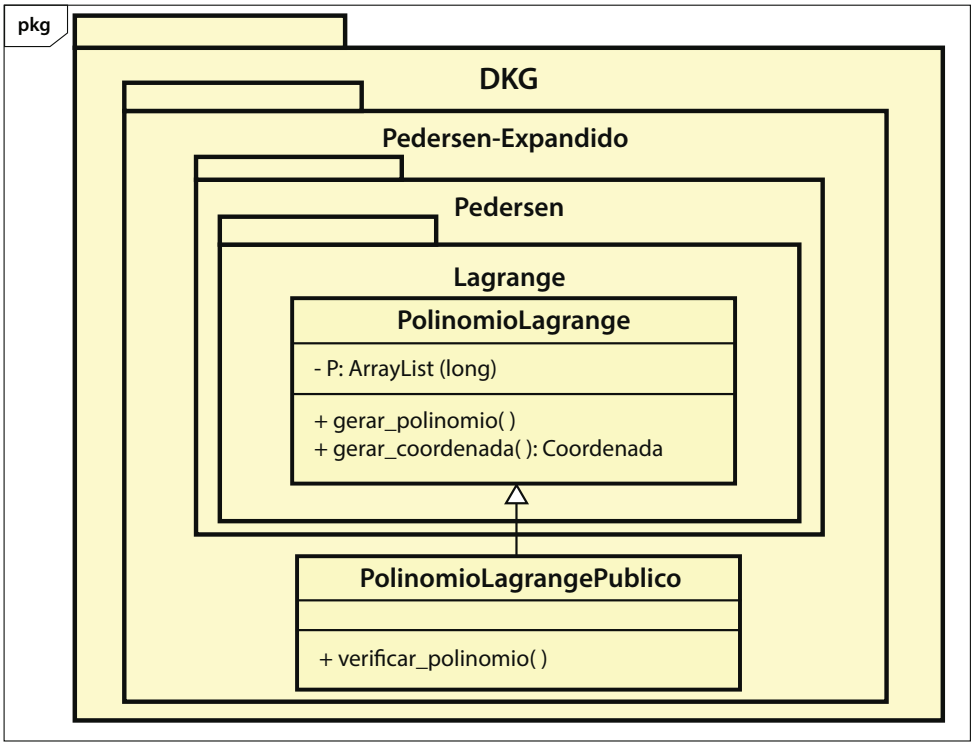
valor da fração, e por isso é um inteiro longo; e *id*, representando o identificador do usuário que receberá tal fração.

O protocolo de *Pedersen* se relaciona ao de *Lagrange* dada a necessidade de utilizar um polinômio aleatório para gerar as frações de segredo. Portanto, o protocolo deste módulo não pode ser executado sem o uso do protocolo de *Lagrange*. Note que o protocolo de *Shamir*, implementado por (NETO, 2016), é bastante similar ao protocolo de *Pedersen* aqui implementado, entretanto mesmo que suas classes sejam idênticas, o funcionamento de ambos é diferente, como detalhado nas seções 2.2.2 e 2.2.3.

O módulo seguinte é o de *Pedersen*-Extendido, mas este protocolo não apresenta classes novas para a implementação. Seu funcionamento se diferencia do de *Pedersen* apenas pelas funções utilizadas e a ordem de sequência, que serão discutidas nas seções seguintes.

Como módulo externo seguinte, chamado de DKG (*Distributed Key Generation*), é o protocolo de geração de chaves distribuído. Ele é responsável por criar um compartilhamento de segredo emitindo também um polinômio público, que é utilizado como a forma de guardar o segredo criado aleatoriamente pelo conjunto.

Figura 15 – Diagrama de classes do módulo *DKG*



Fonte: Criação do autor

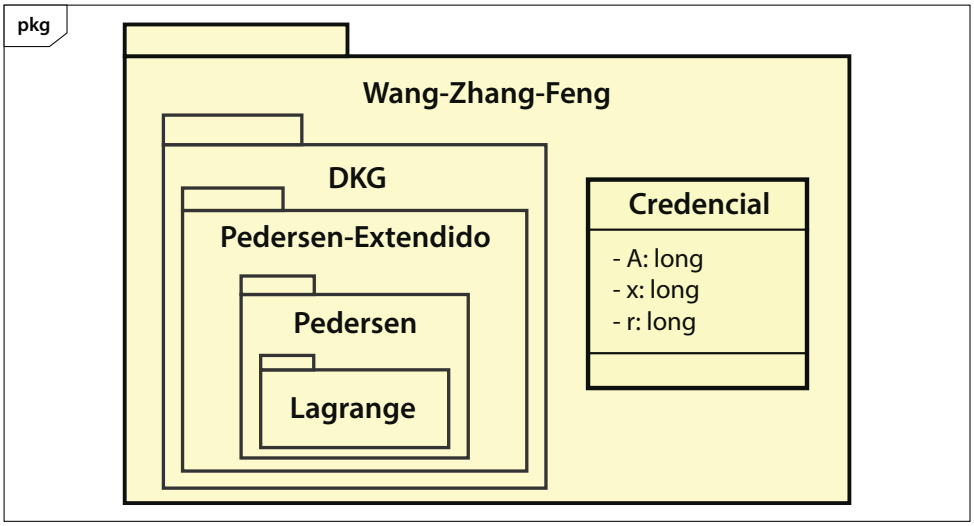
A Figura 15 apresenta a relação deste módulo com os anteriores. A nova classe criada, *PolinomioLagrangePublico* se relaciona diretamente como uma herança da classe *PolinomioPublico*, detalhada anteriormente no módulo de *Lagrange*.

Este módulo é similar ao módulo de *Pedersen* implemetado por (NETO, 2016). O mesmo

se diferencia do *DKG* por trabalhar com seleção e distribuição de chaves criptográficas de conjunto, enquanto o *DKG* trata apenas do compartilhamento utilizado nos métodos anteriores e à criação do polinômio público.

O módulo mais externo do diagrama apresentado na Figura 12 é o *Wang-Zhang-Feng*, que representa o protocolo de assinaturas digitais distribuído de (WANG; ZHANG; FENG, 2005), com as alterações propostas. Este será responsável pela geração de credenciais de forma distribuída.

Figura 16 – Diagrama de classes do módulo *Wang-Zhang-Feng*



Fonte: Criação do autor

A Figura 16 apresenta a arquitetura deste módulo. A classe *Credencial* se faz presente, e possui três atributos: A, que tem como valor um inteiro longo, e representa o valor expresso por $(g_1 g_3^x)^{\frac{1}{y+r}}$; x e r, ambos que têm como valor um inteiro, onde x e r são valores aleatórios gerados de forma distribuída durante o processo de criação da credencial.

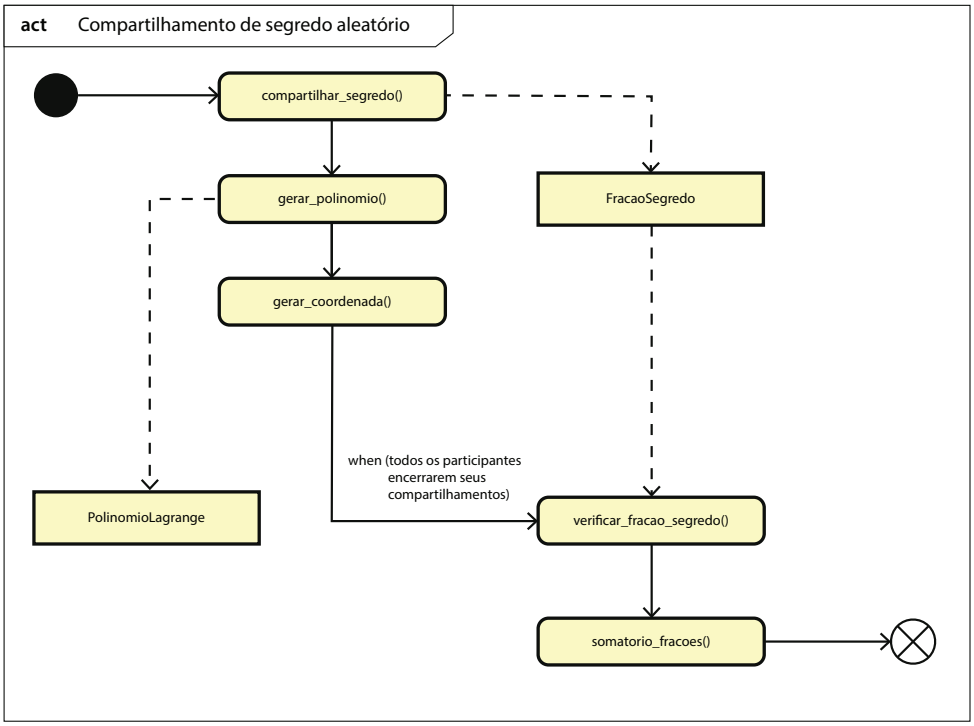
3.2.4 Sequência de execução da implementação

A seção anterior apresentou as funções implementadas para o protocolo de geração distribuída de credenciais. Ainda é necessário detalhar a sequência a qual as funções deverão ser executadas, baseado no funcionamento do protocolo de (WANG; ZHANG; FENG, 2005).

Para detalhar a sequência de funções executadas no protocolo de geração de credenciais de forma distribuida, primeiramente é necessário detalhar os protocolos de compartilhamento de segredo, utilizados em algumas de suas etapas.

Os diagramas de atividades descritos nas Figuras 17 e 18 não apontam a criação dos parâmetros de grupo, que são um pré requisito para o funcionamento da implementação, pois é implícito que ambos já estarão definidos ao início do processo. A criação da chave de compartilhamento do conjunto, utilizada para os compartilhamentos iniciais, também não é apontada no diagrama, pelos mesmos motivos anteriores.

Figura 17 – Diagrama de atividades indicando a sequência necessária para geração de segredo aleatório de forma distribuída



Fonte: Criação do autor

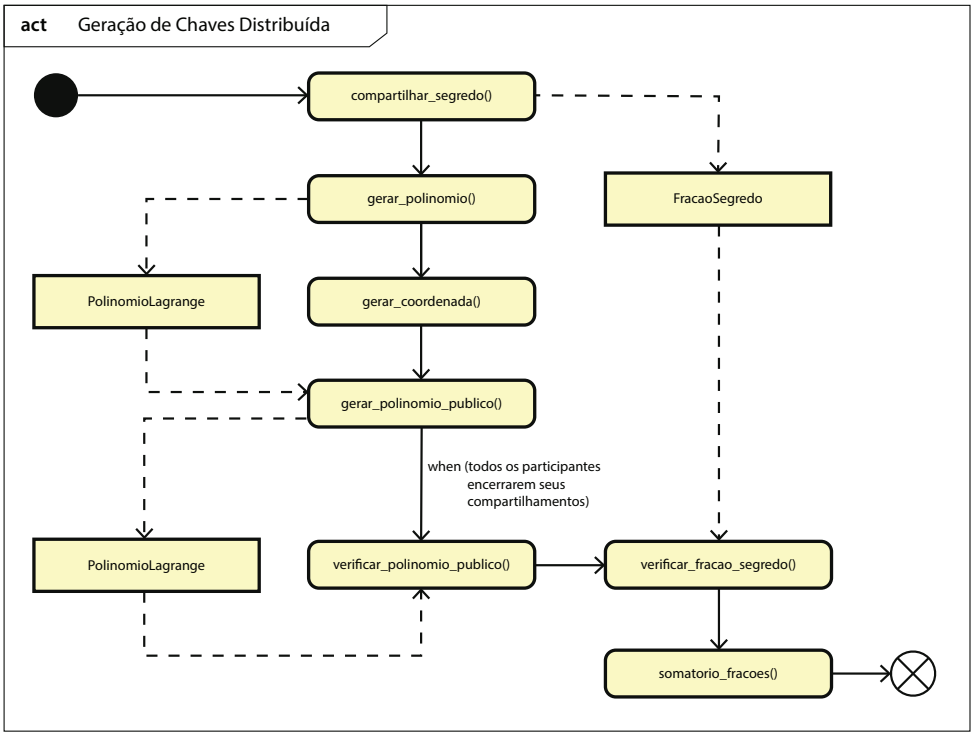
Na Figura 17 a sequência de atividades do protocolo de compartilhamento de segredo aleatório é detalhada. Todos os participantes devem utilizar o método ***compartilhar_segredo()*** a partir de um objeto *Segredo*, individual de cada um. Este resulta em objetos da classe *FracaoSegredo*, e para tal o método ***compartilhar_segredo()*** necessita chamar a função ***gerar_polinomio()***, que por sua vez resulta em um objeto da classe *PolinomioLagrange*.

Quando todos os participantes encerrarem a etapa de compartilhamento de segredo, a etapa de verificação é iniciada. Cada participante executa a função ***verificar_fracao_segredo()***, utilizando os objetos *FracaoSegredo* gerados. Caso o participante não seja desclassificado, suas frações válidas serão utilizadas para calcular sua fração do segredo gerado em conjunto pelo conjunto, utilizando a função ***somatorio_fracao_segredo()***, encerrando este procedimento.

Na Figura 18 a sequência de atividades do protocolo de geração de chave distribuído, ou *DKG*, é detalhada. Todos os participantes devem utilizar o método ***compartilhar_segredo()*** a partir de um objeto *Segredo*, individual de cada um. Este resulta em objetos da classe *FracaoSegredo*, e para tal o método ***compartilhar_segredo()*** necessita chamar a função ***gerar_polinomio()***, que por sua vez resulta em um objeto da classe *PolinomioLagrange*. Este objeto será utilizado pela função ***gerar_polinomio_publico()*** para gerar um objeto *PolinomioLagrangePublico*.

Quando todos os participantes encerrarem a etapa de compartilhamento de segredo, as etapas de verificação são iniciadas. Cada um executa as funções ***verificar_polinomio_publico()***, utilizando o objeto *PolinomioLagrangePublico* criado; e ***verificar_fracao_segredo()***, utilizando

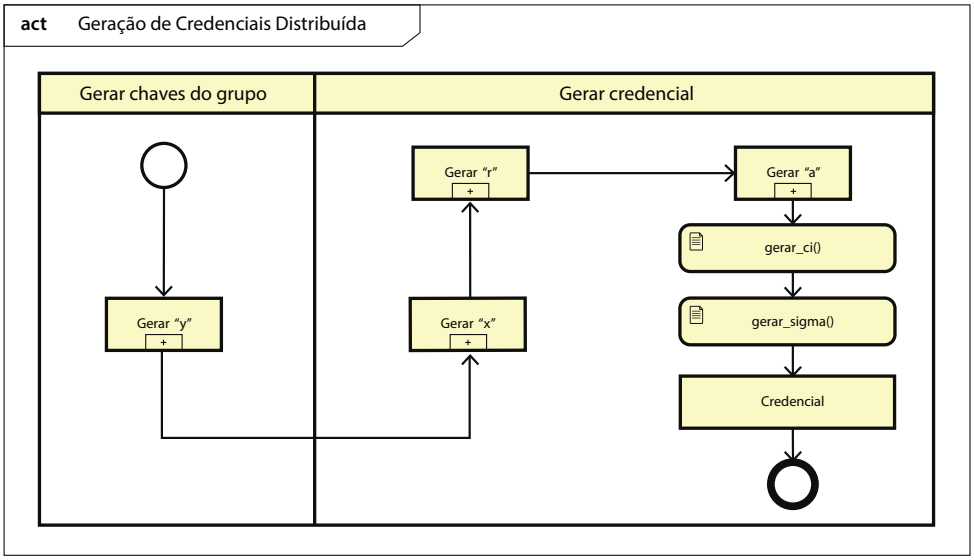
Figura 18 – Diagrama de atividades indicando a sequência necessária para geração de chaves de forma distribuída



Fonte: Criação do autor

os objetos *FracaoSegredo* gerados. Caso o participante não seja desclassificado, suas frações válidas serão utilizadas para calcular sua fração do segredo gerado em conjunto pelo conjunto, utilizando a função ***somatorio_fracao_segredo()***, encerrando este procedimento.

Figura 19 – Diagrama de atividades indicando a sequência necessária para geração de credenciais de forma distribuída



Fonte: Criação do autor

A Figura 19 apresenta um diagrama *BPMN* (BPMN, 2019) que descreve a sequência de

atividades executadas no protocolo de geração de credenciais. Destaca-se que, todas as atividades declaradas como subprocessos, executarão as atividades descritas nas figuras 17 e 18.

Cada participante deverá colaborar, em paralelo, para a geração de variáveis x, y, utilizando o módulo *DKG*. Após esta etapa, referida como geração de chaves de conjunto, é iniciada a segunda etapa, de geração de credencial. Nela é necessário que cada participante colabore com a geração do termo aleatório r, utilizando o método de compartilhamento de segredo aleatório; e após isso gere a, utilizando o *DKG*.

Após geradas todas as variáveis, cada participante deverá calcular um valor de c_i utilizando a função ***gerar_ci()***. Por fim, deve ser gerada a credencial pela função ***gerar_sigma()***, que retornará o último dos termos necessários para formar uma credencial.

3.2.5 Funções implementadas

A partir da arquitetura definida para o sistema, apresentada na seção anterior, foi possível implementar o protocolo de Wang-Zhang-Feng. Nesta seção são apresentadas cada uma das funções *JavaScript* implementados para a execução deste protocolo.

Para as funções serem utilizadas, é necessário que alguns parâmetros já estejam definidos inicialmente. Estes são o grupo multiplicativo $\mathbb{Z}_p^*$, definido pelos primos p, q e o gerador do subgrupo multiplicativo $g \in \mathbb{Z}_p^*$; o limiar utilizado, representado como t; e o número de membros do conjunto, representado como n. Cada membro também deverá possuir um identificador, representado como k.

Como a linguagem *JavaScript* não possui o tipo de dados *BigInt*, é necessária a utilização de bibliotecas externas para que os números utilizados para os cálculos criptográficos do protocolo funcionem corretamente. A biblioteca *JSBN* permite a conversão dos parâmetros iniciais, recebidos pelo sistema, ao tipo de dados *BigInt*. Desta forma, os cálculos matemáticos dos protocolos utilizados podem ser realizados. Nos diagramas, este sempre será chamado como *long*.

A segunda biblioteca utilizada, a *SJCL*, é necessária para o funcionamento de grande parte das funções implementadas, pois a própria biblioteca possui diversos mecanismos criptográficos fundamentais para a implementação do protocolo num geral.

Para melhor organização das funções do protocolo implementado, foram criados quatro módulos separados, sendo cada um referente aos protocolos utilizados como base teórica e ao próprio protocolo a ser implementado. Assim, os módulos existentes são referentes aos protocolos de compartilhamento de segredo verificado (PEDERSEN, 1991a); compartilhamento de segredo aleatório (PEDERSEN, 1991b); compartilhamento de chave distribuída (GENNARO et al., 1999); e o próprio protocolo de assinaturas digitais distribuídas (WANG; ZHANG; FENG, 2005).

O primeiro módulo implementado foi o de compartilhamento de segredo verificável (PEDERSEN, 1991a), que será chamado como protocolo de *Pedersen*. Ele é responsável por permitir que cada participante crie um segredo e o compartilhe, permitindo que o segredo possa ser descoberto pelo conjunto de forma distribuída.

Esse módulo possui uma função, e é utilizado um método da classe Segredo. Este método, representado pelo Algoritmo 1, é utilizado para gerar as frações de um segredo determinado. Ele está presente em todos os objetos da classe *Segredo*, onde este objeto deve conter o valor do segredo a ser compartilhado.

Algoritmo 1: Segredo.compartilhar_segredo()

Entrada: p, q, g, h, t, C

Saída: *array*

Como entrada este método recebe os parâmetros de grupo p, q, g; o parâmetro h, que é a chave de compartilhamento do conjunto; o parâmetro limiar t e um *array* C contendo os identificadores de cada um dos membros do conjunto.

Inicialmente, o método deverá gerar dois polinômios de *Lagrange*, utilizando os parâmetros q, t. Um dos polinômios utilizará ainda o parâmetro S, recebido do objeto *Segredo*, e criará o *polinômio f*. O outro polinômio deverá receber um valor aleatório r, e será responsável pela criação do *polinômio d*. Para ambos os casos, o polinômio criado será um *array P* contendo t valores.

Após gerados os polinômios, é necessário gerar coordenadas para cada participante. Estas coordenadas serão utilizadas na criação das frações de segredo de cada membro do conjunto. Este método é executado n vezes, para cada n valores contidos em C Cada execução recebe como entrada o valor p e um dos valores de C_i.

São produzidos n objetos da classe *Coordenada*. Para cada um destes é criado um objeto da classe *FracaoSegredo*, que herda os atributos de *Coordenada*. Todos os objetos *FracaoSegredo* são armazenados em posições de um *array*. Serão gerados dois *array*, um para o *polinômio f* e outro para o *polinômio d*.

A variável de verificação de compartilhamento será criada no formato de um *array*, onde cada item deste array definido em (3.6, e será executada t vezes.

$$E_i = g^{f_i} h^{d_i} \tag{3.6}$$

Finalmente, o método retorna um *array* com os valores de S, R, E, que são parâmetros públicos; e o *polinômio f*, que é uma informação privada e será utilizada apenas em outro módulo.

Algoritmo 2: verificar_fracao_segredo()

Entrada: p, q, g, h, si, ri, E

Saída: boleano

Esta função, representada pelo Algoritmo 2, é responsável por realizar uma comparação entre dois termos independentes, calculados por cada membro. Os parâmetros recebidos pela função são p, q, g, h como parâmetros iniciais do grupo $\mathbb{Z}_p^*$; as frações de segredo si, ri; e a variável de verificação E.

Serão calculados os valores de $verifica1 = g^{s_i} h^{r_i} \mod p$ e $verifica2 = \sum E_k^{id^k} \mod p$. Os valores de si, ri se referem ao atributo f de cada parte de segredo, enquanto os valores de id se referem aos identificadores de cada parte de segredo, de cada usuário. O termo E_k se refere a cada valor dentro do *array* E.

Ao final da função, será verificad se $verifica1 = verifica2$. Caso verdadeiro, a função retornará o valor *verdadeiro*. Caso contrário, o valor retornado será *falso*.

O segundo módulo implementado foi o de compartilhamento de segredo aleatório (PEDERSEN, 1991b). Este protocolo consiste do compartilhamento de um segredo completamente distribuído, sem precisar de um membro específico para a criação deste segredo.

Cada membro deverá executar o método de compartilhamento de segredo *compartilhar_segredo(p, q, g, h, t, C)* do protocolo de *Pedersen*, e compartilhar as frações de segredo com os demais membros. Ao término desta etapa, cada participante deverá utilizar todas as suas fraações de segredo para gerar uma única parte de chave nova, que será utilizada para a recuperação do segredo, de forma distribuída. As variáveis de verificação devem ser utilizadas para gerar uma nova variável de verificação que funcione para o novo segredo gerado em conjunto.

Este protocolo é denominado *Pedersen* expandido, e possui duas funções:

Algoritmo 3: somatorio_fracao_segredo()

Entrada: p, Si, Ri

Saída: *array*

Esta função, representada pelo Algoritmo 3, é responsável por realizar o somatório de todas as frações de segredo pertencentes a cada participante. Os parâmetros utilizados são o número primo p, relativo ao grupo; e as frações de segredo Si, Ri, que são *arrays* que contém todas as frações relativas aos segredos S, R compartilhados pelos participantes.

Cada um dos n termos de Si será somado, assim como os termos de Ri, resultando numa única parte de chave para Si e outra para Ri. A função deve retornar um *array* com ambas as novas chaves Si, Ri calculadas.

Algoritmo 4: somatorio_E()

Entrada: p, Ek

Saída: *array*

Esta função, representada pelo Algoritmo 4, é responsável por realizar o somatório de todas as variáveis de verificação de segredo pertencentes a cada compartilhamento. Os parâmetros utilizados são o número primo p, relativo ao grupo; e o parâmetro Ek, que é um *array* que contém

todas as variáveis de verificação pertecentes aos compartilhados de *Pedersen* realizados pelos participantes.

Cada um dos *n* termos de *Ek* será somado, resultando em um único *array* de variáveis de verificação. A função deve retornar este *array*.

O terceiro módulo implementado é o protocolo de criação de chave de forma distribuída (GENNARO et al., 1999), que trabalha utilizando o protocolo de *Pedersen* expandido, permitindo que, para um determinado segredo compartilhado de forma aleatória, seja possível compartilhar seu polinômio público. Este polinômio pode ser utilizado como forma de recuperação deste segredo.

Cada membro deverá executar o método de compartilhamento de segredo *compartilhar_segredo(p, q, g, h, t, C)* do protocolo de *Pedersen*, e compartilhar as frações de segredo com os demais membros. Ao término desta etapa, cada participante deverá utilizar a função *somatorio_fracao_segredo(p, Si, Ri)* do protocolo de *Pedersen* expandido para gerar suas novas chaves. A função *somatorio_E(p, Ek)* deve ser utilizada para gerar as novas variáveis de verificação do segredo compartilhado. Por fim, os participantes devem gerar um polinômio público para o compartilhamento, e o mesmo será um novo fator de verificação para o compartilhamento de chaves do protocolo.

Este protocolo é denominado *DKG (Distributed Key Generation*), e possui duas funções:

Algoritmo 5: gerar_polinomio_publico()

Entrada: $p, g, polinomio$

Saída: PolinomioLagrangePublico

Esta função, representada pelo Algoritmo 5, é utilizada para calcular a versão pública de um polinômio de *Lagrange*, e também utilizada para verificar as frações obtidas a partir do polinômio de *Lagrange* original.

Os parâmetros recebidos por esta função são p, g para noção de grupo, e $polinomio$, que é um objeto recebido diretamente do método *compartilhar_segredo(p, q, g, h, t, C)* do protocolo de *Pedersen*.

Para cada ponto f_i de $polinomio$ será calculado o valor $F_i = g^{f_i} \mod p$. Cada valor F_i será inserido na posição i correspondente em um *array A* resultante. A função retorna um objeto *PolinomioLagrangePublico* com *array A* como atributo.

Algoritmo 6: verificar_polinomio()

Entrada: $p, q, g, fracao, polinomio$

Saída: boleano

Esta função, representada pelo Algoritmo 6 é responsável por realizar uma comparação entre dois termos independentes, calculados por cada membro. Os parâmetros recebidos pela função são p, q, g como parâmetros iniciais do grupo $\mathbb{Z}_p^*$; uma fração de segredo $fracao$; e o polinômio público $polinomio$.

Serão calculados os valores de $verifica1 = g^{f} \mod p$ e $verifica2 = \prod_{k=0}^{t} F_k^{id^k} \mod p$. O valor de f se referem ao atributo f de cada fração de segredo $fracao$, enquanto os valores de id se referem aos identificadores de cada fração de segredo $fracao$, de cada usuário. O termo F_k se refere a cada valor dentro do *array* de polinômios públicos $polinomio$.

Ao final da função, será verificado se $verifica1 = verifica2$. Caso verdadeiro, a função retornará o valor *verdadeiro*. Caso contrário, o valor retornado será *falso*.

O último módulo implementado, que condiz com o protocolo de geração de credenciais distribuído, é baseado em (WANG; ZHANG; FENG, 2005) e possui adequações detalhadas anteriormente na seção 3.1.2. Ele trabalha utilizando o protocolo *DKG* e o protocolo de *Pedersen* expandido.

Após os membros colaborarem para a geração de variáveis utilizando ambos os módulos anteriores, este módulo será responsável pelo cálculo das variáveis b, c, utilizadas para gerar a credencial. Após estas variáveis serem definidas, é possível iniciar o processo de geração de credencial.

Este protocolo é denominado Wang-Zhang-Feng, e possui duas funções:

Algoritmo 7: gerar_ci()

Entrada: p, r, yi, ai

Saída: *BigInt*

Esta função, representada pelo Algoritmo 7 é responsável por gerar a variável *ci*, utilizada como parâmetro para a função de geração de credencial. Cada membro deve calcular seu valor de c. Os parâmetros recebidos pela função são p como parâmetro do grupo $\mathbb{Z}_p^*$; o número aleatório r, gerado de forma distribuída entre os participantes; a fração de segredo *yi* pertencente ao membro; e a fração de segredo *ai*, também única do participante.

Será calculado o valor da variável *bi* internamente, no formato $bi = r * yi \mod p$, e após determinado seu valor será calculado $ci = ai * bi \mod p$. O valor de b corresponde ao formato utilizado para criptografar as credenciais, e será gerada a variável *ci* para manter b em segredo. Ao final da função, será retornado o valor obtido de *ci*.

Algoritmo 8: gerar_sigma()

Entrada: $p, q, polinomioPublico\_a, c$

Saída: *BigInt*

Esta função, representada pelo Algoritmo 8, é responsável por gerar a variável σ, utilizada como o valor A da estrutura matemática de credencial utilizada no CIVIS. Os parâmetros recebidos pela função são p, q como parâmetros do grupo $\mathbb{Z}_p^*$; o polinomio $polinomioPublico\_a$, gerado no momento de criação distribuída da variável a, durante a execução desse módulo; e o termo c, criado como resultado da soma de todos os *ci* gerados por cada participante.

Será calculado o valor da variável σ seguindo a estrutura definida em (3.7). Ao final da

função, será retornado o valor obtido de σ, utilizado como atributo para a criação de alguma credencial.

$$\sigma = polinomioPublico\_a^{c^{-1}} = (g^a)^{c^{-1}} = g^{\frac{1}{(ry)}}. \tag{3.7}$$

3.2.6 Testes realizados

Para a verificação do funcionamento de todas as funções implementadas, foram realizados testes que simulavam o uso de todo o protocolo funcionando em sequência.

Tais testes ocorreram de forma local e individual, com apenas um participante, que fornecia as informações necessárias de todos os membros do grupo. Todas as etapas que necessitavam da participação de todo o conjunto de participante foram simuladas dentro de um for, onde todas as informações de cada participante eram determinadas no ambiente de testes, sem ser executado realmente de forma distribuída. Isso foi necessário para poder testar as funções, já que um teste de forma realmente distribuída exigiria mais pessoas e seria inviável para o prazo.

Individualmente, foram testados todos os módulos, com exceção do módulo de *Lagrange*, pois este já se encontra implementado no sistema CIVIS. Como seus códigos foram reutilizados, não houve necessidade de retestá-lo de forma individual. Para cada módulo, foram realizados dois testes: o primeiro utilizava números inteiros pequenos, possibilitanto que todos os cálculos pudessem ser feitos também manualmente, permitindo a realização de testes de mesa como auxilio para depuração de erros; e um teste com números inteiros grandes, para verificar se a biblioteca *JSBN* estava sendo usada corretamente. Em todos os testes, foram definidos apenas três participantes para o conjunto.

O primeiro módulo testado foi o de *Pedersen*. Para este, foi necessário definir um segredo para apenas um dos membros, e esse executava o método ***Segredo.compartilhar_segredo()***, resultando em frações, que eram exibidas para que seus valores pudessem ser conferidos com os cálculos manuais. Após geradas as frações, foi realizada a etapa de verificação de frações de segredo, dentro de um laço *Para* que rodava para todos os participantes. Cada participante verificava as frações resultantes do compartilhamento de segredo inicial, e postava o resultado de sua verificação. O teste apenas era bem sucedido se todas as verificações resultassem no valor boleano verdadeiro.

O segundo módulo testado foi o de compartilhamento de segredo de forma aleatória e distribuída, ou *Pedersen*-Extendido. O teste consiste de execuções repetidas do módulo anterior, e caso todos os compartilhamentos ocorram corretamente, todas as frações de segredo recebidas por cada participante devem ser somadas, formando as novas frações de cada um. Para este teste, foi definido um segredo para cada participante, e dentro de um laço *Para*, que rodava para cada participante, foi executado o módulo de *Pedersen*. O teste apenas era bem sucedido se todas as verificações dos compartilhamentos resultassem no valor boleano verdadeiro. Após todos os

compartilhamentos serem válidos, era utilizada as funções ***somatorioSR()*** e ***somatorioE()***.

O terceiro módulo testado foi o de compartilhamento de chaves de forma distribuída, ou *DKG*. O teste consiste da execução do módulo anterior, adicionado à geração de um polinômio público. Para gerar tal polinômio, foi necessário receber o valor de polinômio de *Lagrange* utilizado por cada participante, gerando o polinômio público de cada compartilhamento dentro de um laço *Para*, que rodava para cada participante. O teste apenas era bem sucedido se, além de todas as verificações de compartilhamento serem válidas, também as verificações do polinômio público resultassem no valor boleano verdadeiro.

Por fim, o último módulo testado foi o protocolo adaptado de WZF. O teste consiste da execução dos módulos anteriores de acordo com o descrito na Figura 16. As utilizações de cada módulo tinham resultado bem sucedido dadas as mesmas condições de sucesso dos testes de seus módulos. Para a verificar se os termos gerados nesse módulo eram válidos, foi criadas funções exclusivas para testes, que verificavam o formato dos termos *ci* e σ. O teste apenas era bem sucedido se, além de todas as verificações de módulos anteriores serem válidas, as verificações de elementos para geração de credenciais também resultassem no valor boleano verdadeiro.

Mesmo que os testes possuam complexidade computacional elevada, o mesmo não ocorre no sistema de votação, pois cada participante executará suas etapas de forma paralela aos outros. Em apenas um computador, onde os testes foram realizados, isso não seria possível, e portanto os testes tiveram de ser executados de forma sequencial.

3.3 Considerações finais do capítulo

Este capítulo apresentou a implementação de um protocolo de geração de credenciais de forma distribuída, baseado no protocolo proposto por (WANG; ZHANG; FENG, 2005). O esquema foi alterado para que pudesse atender aos requisitos de geração de credenciais do sistema CIVIS, já que o protocolo gera assinaturas digitais de forma distribuída em uma estrutura matemática bastante similar à exigida pelo sistema de votação via *Internet*.

É necessário ainda propor uma forma de integração, do protocolo implementado, ao sistema CIVIS, que será abordado na Seção 3.3.1. Dificuldades encontradas durante o desenvolvimento do trabalho também serão detalhadas, na Seção 3.3.2.

3.3.1 Proposta de integração do protocolo de *WZF* ao sistema CIVIS

A Seção 3.2 detalhou a implementação do protocolo de WZF para a geração de credenciais. De forma a integrá-lo ao sistema CIVIS, no entanto, são necessárias modificações no sistema. Essa subseção apresenta uma proposta de integração desse protocolo ao sistema.

Atualmente, as credenciais são geradas de forma centralizada no CIVIS. Apenas uma autoridade de registro é definida, e esta é responsável por utilizar sua chave secreta para gerar

as credenciais. Entretanto, o sistema trabalha com a proposta de ser resistente a membros corruptíveis,portanto permitir que apenas um membro seja definido como autoridade de registro é uma potencial falha a ser explorada no sistema.

O protocolo implementado gera credenciais de forma distribuída, sem necessitar de uma autoridade centralizadora que define valores para variáveis ou coordena o funcionamento da etapa de registro de credenciais. A introdução desse protocolo no CIVIS resultaria em um aumento na segurança do sistema, por atuar como uma alternativa à geração centralizada.

Para que a integração seja possível, é fundamental que o sistema possa cadastrar mais de uma autoridade de registro por eleição. As autoridades de registro são definidas na fase de configuração, pela autoridade de eleição, portanto a ela deve ser permitido adicionar mais autoridades de registro.

Os módulos implementados necessariamente precisariam ser adicionados ao CIVIS, para o funcionamento do protocolo de forma correta. O sistema já conta com alguns módulos que atuam de forma similar, mas apenas o módulo de *Lagrange* trabalha de forma alinhada ao proposto ao trabalho, logo é o único que pode ser reutilizado. Os demais módulos necessários para o funcionamento do protocolo devem ser adicionados ao sistema.

Por fim, destaca-se que a implementação foi completamente baseada nas funções que trabalham no lado cliente da aplicação. Funções de configurações de membros dos conjuntos após verificações, caso detectado algum membro a ser excluído, devem ser implementadas no lado servidor do sistema, funcionando na linguagem de programação *Python*, e não *JavaScript*, como utilizado neste trabalho.

3.3.2 Desafios da implementação

A implementação, baseada no protocolo proposto por WZF, possuiu como principal desafio a compreensão teórica do protocolo. Como as assinaturas e credenciais possuem uma estrutura matemática similar, a compreensão de todo o processo de geração foi fundamental.

A grande extensão de variáveis utilizadas, e a limitação de representação de determinados termos, foram um fator de grande desafio para a continuidade do trabalho. Entretanto, após toda a base matemática ser compreendida, a implementação teve continuidade sem a ocorrência de demais desafios.

4 CONCLUSÃO E TRABALHOS FUTUROS

Produzir o presente trabalho de pesquisa garantiu uma vasta ampliação nos conhecimentos do autor sobre um tema tão importante para a área de estudo, permitindo a compreenção mais detalhada de diversos protocolos criptográficos e o estudo da criptografia limiar, além de uma maior aproximação com a área de pesquisa relacionada à votações eletrônicas. Implementar um protocolo para geração de credenciais de forma distribuída é uma colaboração para o sistema de votação via *Internet* CIVIS, que auxiliará o sistema a se manter ainda mais seguro.

A implementação veio como forma de resolver um dos desafios relacionados aos sistemas de votação CIVIS, que mesmo que esse proporcione segurança em diversas etapas de seu funcionamento, a forma como as suas credenciais eram geradas, exclusivamente de forma centralizada a uma autoridade de registro, tornava a segurança deste processo dependente da probidade da autoridade que desempenhasse essa função.

A utilização de um protocolo para geração distribuída de credenciais permite que o sistema consiga evitar que uma única autoridade de registro manipule informações de forma indevida. Ao invés de utilizar apenas uma autoridade de registro, seria formado um conjunto de autoridades para realizar a mesma tarefa, evitando que uma autoridade maliciosa tenha controle sobre a eleição.

Por ser um conjunto dentro de um sistema que utiliza criptografia limiar, ainda existe a vantagem de que mesmo que alguma das autoridades do conjunto seja maliciosa, ela não terá capacidade de comprometer a eleição, pois é necessário que ao menos $t + 1$ participantes colaborem para tal.

Tendo sido finalizada a implementação do protocolo, originalmente proposto por (WANG; ZHANG; FENG, 2005), mas aqui foram realizadas alterações para se adequar à geração de credenciais exigidas conforme (ARAÚJO et al., 2010), e utilizando os esquemas propostos por (GENNARO et al., 1999) e (PEDERSEN, 1991b), conclui-se que o protocolo para geração de credenciais de forma distribuída é eficaz, gerando credenciais de forma descentralizada e dentro da estrutura necessária.

4.1 Trabalhos futuros

A partir do conteúdo desenvolvido neste trabalho, demais aspectos podem ser trabalhados futuramente. É necessário realizar a integração do protocolo implemetado com o CIVIS, modificando algumas funcionalidades do sistema para que isso seja possível. Deve ser possível compor um conjunto de autoridades de registro, assim como deve ser implementada uma forma de remover membros que não obtiveram compartilhamentos válidos, por meio das verificações realizadas pelo conjunto de autoridades. Além disso, é necessário avaliar o desempenho do protocolo integrado ao sistema de votação, verificando quanto tempo leva para gerar uma credencial

de forma distribuída e também verificar mais detalhadamente a segurança do código fonte.

REFERÊNCIAS

ARAÚJO, R.; NETO, A.; TRAORÉ, J. Civis-a coercion-resistant election system. 2018.

ARAÚJO, R. et al. Towards practical and secure coercion-resistant electronic elections. In: SPRINGER. **Cryptology and Network Security - 9th International Conference, CANS 2010**. [S.l.], 2010. v. 6467, p. 278–297.

BONEH, D. The decision diffie-hellman problem. In: SPRINGER. **International Algorithmic Number Theory Symposium**. [S.l.], 1998. p. 48–63.

BONEH, D.; BOYEN, X. Short signatures without random oracles. In: SPRINGER. **International Conference on the Theory and Applications of Cryptographic Techniques**. [S.l.], 2004. p. 56–73.

BPMN. **Object Management Group Business Process Model and Notation**. 2019. Disponível em: <http://www.bpmn.org/>.

CHAUM, D.; PEDERSEN, T. P. Wallet databases with observers. In: SPRINGER. **Annual International Cryptology Conference**. [S.l.], 1992. p. 89–105.

CRAMER, R.; GENNARO, R.; SCHOENMAKERS, B. A secure and optimally efficient multi-authority election scheme. **European transactions on Telecommunications**, Wiley Online Library, v. 8, n. 5, p. 481–490, 1997.

DESMEDT, Y. Society and group oriented cryptography: A new concept. In: SPRINGER. **Conference on the Theory and Application of Cryptographic Techniques**. [S.l.], 1987. p. 120–127.

DJANGO. **Django: The web framework for perfectionists with deadlines.** 2018. Disponível em: <https://www.djangoproject.com/>. Acesso em: 02, dez, 2018.

GAMAL, T. E. A public key cryptosystem and a signature scheme based on discrete logarithms. workshop on the theory and application of cryptographic techniques. p. 10–18, 1984.

GATHEN, J. Von zur. **CryptoSchool**. [S.l.]: Springer, 2015.

GENNARO, R. et al. Secure distributed key generation for discrete-log based cryptosystems. In: SPRINGER. **International Conference on the Theory and Applications of Cryptographic Techniques**. [S.l.], 1999. p. 295–310.

HERRANZ, J.; RUIZ, A.; SÁEZ, G. Signcryption schemes with threshold unsigncryption, and applications. **Designs, codes and cryptography**, Springer, v. 70, n. 3, p. 323–345, 2014.

JAVASCRIPT. **JavaScript Community**. 2018. Disponível em: <https://www.javascript.com/>.

JR, S. S. W. **Cryptanalysis of number theoretic ciphers**. [S.l.]: CRC Press, 2002.

NETO, A. A implementação de um criptossistema el gamal limiar e sua integração a um sistema de votação eletrônica. In: UFPA. [S.l.], 2016.

NETO, A. S. et al. Usability considerations for coercion-resistant election systems. In: ACM. **Proceedings of the 17th Brazilian Symposium on Human Factors in Computing Systems**. [S.l.], 2018. p. 40.

PEDERSEN, T. P. Non-interactive and information-theoretic secure verifiable secret sharing. In: SPRINGER. **Annual International Cryptology Conference**. [S.l.], 1991. p. 129–140.

PEDERSEN, T. P. A threshold cryptosystem without a trusted party. In: SPRINGER. **Workshop on the Theory and Application of of Cryptographic Techniques**. [S.l.], 1991. p. 522–526.

PYTHON. **Python Software Foundation**. 2018. Disponível em: <https://www.python.org/>. Acesso em: 02 dez. 2018.

SHAMIR, A. How to share a secret. **Communications of the ACM**, ACm, v. 22, n. 11, p. 612–613, 1979.

SOUHEIB, Y.; STEPHANE, D.; RIADH, R. Watermarking in e-voting for large scale election. In: IEEE. **Multimedia Computing and Systems (ICMCS), 2012 International Conference on**. [S.l.], 2012. p. 130–133.

STALLINGS, W. Cryptography and network security: principles and practice. **Practice (6th Edition)**, v. 9, 2008.

STARK, E.; HAMBURG, M.; BONEH, D. **Stanford Javascript Crypto Library**. 2018. Disponível em: <http://bitwiseshiftleft.github.io/sjcl/>. Acesso em: 06 dez. 2018.

WANG, H.; ZHANG, Y.; FENG, D. Short threshold signature schemes without random oracles. In: SPRINGER. **International Conference on Cryptology in India**. [S.l.], 2005. p. 297–310.