ANDRÉ DE CARVALHO RODRIGUES
FÁBIO FERREIRA VIDILLE
HÉLIO VIEIRA DE JESUS

CONSTRUÇÃO DE UM SISTEMA DE *E-VOTING* SEGURO BASEADO NO PROTOCOLO VOTEREMOTE

São Paulo
2010

ANDRÉ DE CARVALHO RODRIGUES
FÁBIO FERREIRA VIDILLE
HÉLIO VIEIRA DE JESUS

CONSTRUÇÃO DE UM SISTEMA DE *E-VOTING* SEGURO BASEADO NO PROTOCOLO VOTEREMOTE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

São Paulo
2010

ANDRÉ DE CARVALHO RODRIGUES
FÁBIO FERREIRA VIDILLE
HÉLIO VIEIRA DE JESUS

CONSTRUÇÃO DE UM SISTEMA DE *E-VOTING* SEGURO BASEADO NO PROTOCOLO VOTEREMOTE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia.

Área de Concentração:
Engenharia de Computação

Orientador: Prof. Dr. Paulo S. L. M. Barreto

São Paulo
2010

"Ninguém pretende que a democracia seja perfeita ou sem defeito. Tem-se dito que a democracia é a pior forma de governo, salvo todas as demais formas que têm sido experimentadas de tempos em tempos."

Winston Churchill
Estadista britânico
(1874 - 1965)

Resumo

Descreve-se aqui a construção de um sistema completo de *e-voting* baseado no protocolo VoteRemote, desenvolvido na Technische Universität Darmstadt, e que, até o momento, nunca havia sido implementado. Este protocolo visa à simplicidade, permitindo uma implementação eficiente do sistema, ao mesmo tempo em que atende aos requisitos básicos de segurança de um sistema de votação eletrônico. Além disso, a segurança deste protocolo independe do módulo responsável por contabilizar o resultado, chamado de Escrutinador. Inicialmente, estudaram-se as características e a viabilidade do protocolo. Em seguida, especificou-se o sistema, descrevendo-se detalhadamente todos seus componentes, com suas características e funcionamento a serem implementados. O sistema além de realizar o protocolo VoteRemote, deve também atender a certos requisitos adicionais exigidos para sistemas reais, como usabilidade e facilidade de distribuição. De posse desta especificação completa, passou-se para a fase de implementação, que consiste na realização de um protótipo deste sistema proposto. Comparou-se o protótipo com o sistema de e-voting Helios, disponível para uso na internet e já largamente utilizado entre usuários de eleições eletrônicas, destacando-se suas semelhanças e diferenças, principalmente no quesito requisitos de segurança atendidos. As conclusões obtidas demonstram um sistema robusto, seguro, de fácil distribuição, simples de se utilizar por parte do eleitor e que, dependendo das limitações impostas à implementação utilizada, pode ser usado para realizar eleições dos mais variados portes e importância.

Palavras-chave: VoteRemote, Votação Eletrônica, e-Voting, Criptografia

Abstract

In this work, the development of a complete e-voting system is described. The e-voting system is based on the VoteRemote protocol, which was developed in the Technische Universitaet Darmstadt. The highlight of this protocol, which still has not been implemented, is that the Counter does not need to be trustworthy. The protocol aims for simplicity, which allows for an efficient implementation of the voting system, but meets anyhow the main security requirements of an e-voting system. The protocol's feasibility are initially analyzed, alongside with its main characteristics. To analyze the protocol's feasibility, the required technologies were studied to guarantee that these premises were fulfilled. The proposed system was then specified: a detailed description of each component is given, alongside with its main characteristics and algorithms. The proposed system must also fulfill some additional requirements besides those of the protocol, such as usability and to be easily distributable. Having the complete specification at hand, technologies were selected to build the proposed system; the system was described through class and sequence diagrams and a graphical user interface was designed. The system was then implemented and compared with the Helios e-voting system, which is available in the internet and is widely used. The differences between both protocols were highlighted, in particular regarding the security requirements each protocol fulfills. The conclusions drawn from this work point to a robust and easily distributable system, which can be used in any kind of election, independently of its importance and magnitude.

Keywords: VoteRemote, e-Voting, Cryptography

Lista de Figuras

Lista de Abreviaturas

AES	Advanced Encryption Standard
API	Application Programming Interface
BPS	Ballot Preparation System
CBC	Cipher-Block Chaining
DES	Data Encryption Standard
JDBC	Java Database Connectivity
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
MINA	Multi-purpose Infrastructure for Network Applications
NIO	Non-Blocking I/O
NIST	National Institute of Standards and Technology
NSA	National Security Agency
PKCS	Public Key Cryptography Standards
PKI	Public Key Infrastructure
PSS	Probabilistic Signature Scheme
QN	Quadro Negro
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
XML	Extensible Markup Language

Sumário

Capítulo 1

INTRODUÇÃO

1.1 Objetivo

O objetivo deste projeto é implementar um sistema seguro de votação eletrônica pela internet. Além de assegurar fortes requisitos de segurança, a fim de garantir aos seus usuários que nenhuma informação sobre o conteúdo do voto de um eleitor ou resultado parcial da eleição poderão ser acessados, o sistema deve ser fácil de usar, possuir boa escalabilidade e fácil distribuição.

1.2 Motivação

Métodos de votação tradicionais, como aqueles usando cédulas de papel e urnas, não podem ser confiados para suprir os requisitos básicos de seguranca de uma eleição considerada justa e válida. Isso porque a complexidade de contabilização do resultado torna quase inviável a conferência do escrutínio, e há uma grande facilidade de adulteração, remoção e duplicação de votos válidos. Ademais, o processo tradicional, com urnas e cédulas de papel, é demorado e envolve questões de logística que se provam obsoletas frente à tecnologia existente atualmente. Tendo isto em vista, este projeto pretende implementar uma solução segura e eficiente de e-voting, esperando assim criar um sistema que agilize e torne mais seguras as eleições.

O protocolo escolhido para ser implementado se chama VoteRemote e foi desenvolvido na Technische Universität Darmstadt, por Lucie Langer, Axel Schmidt, Melanie Volkamer e Johannes Buchmann. Este protocolo está descrito no paper "Ein PKI-Basiertes Protokoll Für Sichere und Praktikable Onlinewählen"(LANGER et al., 2009), e foi escolhido devido aos fortes requisitos de segurança por ele atendidos, sem no entanto exigir premissas complexas. Assim, a implementação de um sistema robusto e seguro baseado nesse protocolo torna-se realística. Esta característica não é encontrada na maior parte dos protocolos de *e-voting* propostos na literatura especializada, que apresentam em geral premissas extremamente complexas a fm de garantir a sua segurança, de modo que sistemas reais tornam-se pouco viáveis ou de uso restrito.

Após implementado o sistema, uma comparação pode ser realizada com outros sistemas já existentes. Um exemplo é um sistema de *e-voting* baseado no protocolo Helios 1 (ADIDA, 2008), que já é amplamente utilizado por usuários finais.

1.3 Organização

O presente documento está dividido nas seguintes seções:

- Capítulo 2 - Protocolo VoteRemote: descreve o protocolo VoteRemote, no qual é baseado o funcionamento do sistema. Inicialmente, suas premissas são elencadas; em seguida, todos os requisitos de segurança garantidos pelo protocolo são descritos; seu funcionamento é explicado em detalhes, com a definição das funções de cada módulo; e por fim, demonstra-se como cada um dos requisitos é atingido, além de discutir-se alguns pontos sobre cuidados de implementação.
- Capítulo 3 - Sistema Proposto: descreve o sistema proposto neste trabalho. Descreve-se como o sistema implementa o protocolo descrito no capítulo 2, como suas premissas são asseguradas, e seu funcionamento como um sistema de votação completo. Também discutem-se requisitos adicionais a fim de torná-lo um sistema viável de ser utilizado por qualquer pessoa, além de alguns possíveis ataques e medidas de segurança adicionais.
- Capítulo 4 - Segurança da Informação: descreve os principais algoritmos criptográficos usados na implementação da camada de criptografia do sistema.
- Capítulo 5 - Desenvolvimento: descreve toda a arquitetura do sistema implementado, como suas dependências, filtros e classes principais. Diagramas de classe, de estado e de sequência são apresentados para melhor entendimento do sistema.
- Capítulo 6 - Testes: descreve os principais testes executados para a validação das funcionalidades do sistema e para prova de sua segurança.
- Capítulo 7 - Comparação com o Helios: realiza uma comparação entre o sistema aqui proposto e o sistema de *e-voting* Helios, facilmente encontrado na internet para download. Descreve rapidamente o princípio de funcionamento desse sistema e, em seguida, compara os requisitos de segurança atendidos por cada um dos protocolos.
- Capítulo 8 - Considerações finais: avalia os resultados globais do projeto, analisando suas dificuldades e aspectos positivos, além de ressaltar possíveis futuras melhorias.

Capítulo 2

PROTOCOLO VOTEREMOTE

Neste capítulo descreve-se em detalhes o funcionamento do protocolo VoteRemote, no qual é baseado o sistema proposto neste trabalho. Este se trata de um protocolo de votação eletrônica que visa atender os requisitos elencados na seção 2.2. Descrevem-se os objetivos e requisitos almejados pelo protocolo; as funções de cada um de seus módulos componentes; o funcionamento das fases do protocolo e, por fim, comentam-se detalhes de implementação que influenciam nos requisitos alcançados pelo protocolo.

2.1 Objetivo do Protocolo

O protocolo escolhido para ser implementado chama-se VoteRemote e foi desenvolvido na Technische Universität Darmstadt, por Lucie Langer, Axel Schmidt, Melanie Volkamer e Johannes Buchmann, e foi selecionado por sua robustez e simplicidade (LANGER et al., 2009). Ele foi desenvolvido a partir do protocolo descrito em Ohkubo et al. (MAMBO; ZHENG, 1999). A diferença principal entre esses dois protocolos está no fato de que no lugar de dividir o processo de contagem dos votos em várias instâncias, o VoteRemote utiliza apenas um módulo Escrutinador, tornando o processo de apuração dos votos mais simples e eficiente. Este módulo por sua vez, nem confiável precisa ser, já que sua chave privada é publicada em um Quadro Negro ao fim da votação. Dessa forma, qualquer pessoa pode verificar a contagem dos votos. Sua descrição completa encontra-se em (LANGER et al., 2009).

A simplicidade do VoteRemote baseia-se na ausência de premissas complexas, ainda que sua implementação consiga atender a todos os requisitos essenciais para um sistema de e-voting seguro.

Neste capítulo descreve-se seu funcionamento e quais requisitos de segurança são por ele atendidos.

2.2 Requisitos de um sistema de e-voting seguro

Um protocolo de eleição eletrônica deve satisfazer uma gama de requisitos de segurança, elencados a seguir.

- Exatidão:

  - Um voto não pode ser repetido, apagado ou alterado.
  - Todos os votos válidos são contabilizados.

- Democracia:
  - Somente os eleitores cadastrados podem votar.
  - Cada eleitor tem direito a lançar no máximo um voto.

- Confidencialidade:
  - Anonimato: Não é possível associar um voto com o eleitor que o emitiu.
  - Equidade: Todos os votos permanecem em segredo até o final do período de votação.
  - Incomprovabilidade: Nenhum eleitor pode provar que votou de determinada maneira.
  - Incoagibilidade: Nenhum eleitor pode ser forçado a votar de maneira determinada.

- Verificabilidade:
  - Universal: Qualquer um pode verificar que todos os votos válidos foram contados.
  - Individual: Cada eleitor pode verificar que seu voto foi considerado válido.

2.3 Descrição do Protocolo

Nesta seção descreve-se o funcionamento do protocolo VoteRemote, os módulos que o compõem e suas funcionalidades, além das premissas assumidas e os requisitos de segurança alcançados.

2.3.1 Módulos Componentes

Os seguintes atores compõem o protocolo.

- Validador: módulo que, através de sua assinatura, comprova o direito de votar de um eleitor.

- Quadro Negro: quadro público, no qual apenas alguns atores possuem permissão de escrita, mas todos possuem permissão de leitura. Deve ser impossível apagar ou sobrescrever qualquer mensagem publicada no Quadro Negro.
- Rede de Mistura: mistura os votos recebidos, de forma a quebrar qualquer relação entre a ordem de chegada dos votos.
- Escrutinador: contabiliza os votos e apresenta o resultado final da eleição.
- Votador: representa um ator que submete um voto através de uma conexão segura. Na descrição do protocolo aqui apresentada, o Votador representa tanto o eleitor que interage com o sistema para produzir seu voto quanto o módulo Votador, que é a entidade do sistema desenvolvido onde o eleitor realiza seu voto, ou seja, a urna onde o eleitor depositará seu voto. A ambiguidade presente no que se refere ao Votador na descrição do protocolo foi tratada na implementação do sistema, de forma a se obter o funcionamento proposto na descrição do protocolo.

2.3.2 Premissas

Na especificação do protocolo, assumem-se as seguintes premissas:

- Existe uma infraestrutura de chave pública (PKI) confiável disponível. Todas as chaves públicas são válidas. Uma entidade certificadora fornece um certificado PKI, ou seja, todas as encriptações são feitas com a chave pública correta. Todos os atores utilizam a PKI. A criptografia utilizada é robusta e extremamente complexa de ser quebrada.
- A fase de registro de eleitores e candidatos foi executada corretamente.
- O eleitor não é observado durante a votação.
- Durante a votação, o voto do eleitor não é alterado no aparelho de votação. O voto gerado deve corresponder à opção escolhida pelo eleitor.
- O votador não pode gerar um voto inválido (como por exemplo, com uma estrutura ou encriptação incorretas). Pode existir a possibilidade do eleitor não votar em nenhum dos candidatos através, por exemplo, de uma opção adequada na cédula de votação.
- Durante o processo de votação, são gerados números aleatórios, utilizados para realizar a encriptação probabilística do voto e durante o processo de assinaturas

cegas. O eleitor não conhece nem o fator utilizado para mascarar seu voto durante o processo de Assinaturas Cegas, nem o número aleatório utilizado para encriptar seu voto.

- Ao eleitor é fornecido um recibo para que ele possa verificar se seu voto foi contabilizado ao final da eleição.
- O Validador deve ser confiável nos seguintes aspectos:
  - Somente votos de eleitores aptos a votar são assinados.
  - Ele não divulga sua chave privada.
  - Ele separa a assinatura do votador do seu respectivo voto.
  - Ele não conspira com outros atores.
- O Quadro Negro deve ser confiável nos seguintes aspectos:
  - Ele autentica os atores corretamente e autoriza os acessos correspondentes a cada função.
  - Ele não pode rejeitar a publicação de informação para um ator autenticado.
  - Ele não pode alterar ou apagar nenhuma informação.
  - Ele não conspira com nenhum outro ator.
- A Rede de Mistura deve ser confiável nos seguintes aspectos:
  - Ela mistura corretamente.
  - Ela não divulga sua chave privada nem a ordem original dos votos.
  - Ela não altera, apaga ou duplica nenhum voto.
  - Ela não conspira com outros atores.
- Os atores confiáveis são então:
  - Validador
  - Quadro Negro
  - Rede de Mistura
- Os atores não necessariamente confiáveis são:
  - Votador

  - Escrutinador

- Um voto é válido quando:
  - Possui o formato técnico correto.
  - É assinado pelo Validador.
  - É encriptado pelas chaves públicas do Escrutinador e da Rede de Mistura na ordem correta.
  - Está publicado no Quadro Negro.

2.3.3 Funcionamento do Protocolo

Nesta seção descreve-se o funcionamento do protocolo VoteRemote, divido em três fases: Registro, Votação e Apuração. Cada uma dessas fases encontra-se explicada em detalhes a seguir.

Fase 0: Registro

Esta fase ocorre fora do escopo do protocolo. Ao final dessa fase, uma lista de candidatos e uma lista de eleitores aptos a votar e seus respectivos certificados devem estar publicadas no Quadro Negro. Esta lista pode ser verificada por qualquer pessoa, evitando que pessoas mal intencionadas possam omitir candidatos das cédulas, impedindo os eleitores de os selecionarem, ou omitir eleitores, impedindo estes de votarem.

Fase 1: Votação

Uma única vez e apenas no início da fase de votação, o Validador recebe do Quadro Negro a lista de todos os eleitores aptos a votar. Então, os seguintes passos são executados a cada vez que um voto é submetido por um eleitor.

Quando um eleitor for votar, ao entrar com seu ID (por exemplo, número do título de eleitor ou número USP), o módulo de votação envia as informações de autenticação para o Quadro Negro. Caso o número do eleitor seja válido e este ainda não tenha votado, o quadro negro envia a cédula para o módulo votador.

O eleitor pode então efetuar seu voto através do módulo Votador, produzindo um Voto v . Esse voto é então mascarado com um fator aleatório r , produzindo $x = B(v, r)$. O Votador então assina o voto mascarado e envia ao Validador x e sua assinatura, ou seja,

$[x, S_{votador}(x)]$. Isto é feito para que o Validador possa, sem ter acesso ao candidato escolhido pelo eleitor, verificar a assinatura do votador e confirmar que este se trata de um voto válido, ou seja, de um voto oriundo de um eleitor que possui permissão para votar.

Em posse de voto e assinatura, o Validador utiliza a lista de eleitores cadastrados para verificar sua validade. Caso o voto seja válido, o Validador retira a assinatura $S_{votador}(x)$ do Votador e assina com a sua própria chave privada obtendo $S_{validador}(x)$, devolvendo ao módulo Votador $[x, S_{validador}(x)]$, ou seja, o voto mascarado e uma assinatura gerada com sua chave privada, que garante que este voto é válido.

O Votador retira então o fator aleatório *r*, obtendo assim um voto não encriptado *v* assinado pelo Validador. Vale notar que $S_{validador}(v) = S_{validador}(x)$, pois $S_{validador}(x)$ foi obtido através do sistema de assinaturas cegas de Chaum. Este voto assinado $[v, S_{validador}(v)]$ é encriptado em sequência, primeiramente com a chave pública do Escrutinador, obtendo $E_{escrutinador}([v, S_{validador}(v)])$ e, em seguida, com a chave pública da Rede de Mistura, obtendo $E_{rededemistura}(E_{escrutinador}([v, S_{validador}(v)]))$. O Votador autentica-se com o Quadro Negro e submete este voto assinado e duplamente encriptado. Caso o eleitor esteja cadastrado e ainda não tenha votado, o voto $E_{rededemistura}(E_{escrutinador}([v, S_{validador}(v)]))$ é publicado no Quadro Negro.

Fase 2: Apuração

A segunda fase do processo de votação consiste na apuração dos votos e publicação do resultado final. Esta fase inicia-se somente após o término completo da fase de votação, ou seja, quando todos os eleitores já tiverem publicado seus votos, e envolve o funcionamento de três módulos: Quadro Negro, Rede de Mistura e Escrutinador.

Após finalizada a votação, o Quadro Negro envia todos os votos duplamente encriptados $E_{rededemistura}(E_{escrutinador}([v, S_{validador}(v)]))$ para a Rede de Mistura. Esta por sua vez, retira a encriptação correspondente a sua chave e embaralha todos os votos. Uma nova lista com os votos $E_{escrutinador}([v, S_{validador}(v)])$ é então gerada e enviada de volta ao Quadro Negro. Esta etapa tem como função quebrar a relação entre a ordem de publicação dos votos no Quadro Negro e seu conteúdo. Como a chave privada do Escrutinador é publicada no final da Fase de Apuração, sem esta etapa seria possível associar a cada eleitor o seu voto, violando o requisito da Anonimidade de uma votação considerada justa e segura.

Os votos embaralhados e sem a encriptação com a chave da Rede de Mistura $E_{escrutinador}([v, S_{validador}(v)])$ são então publicados no Quadro Negro e enviados para o Escrutinador, componente responsável pela apuração da votação. O Escrutinador decripta os votos, obtendo os pares $[v, S_{validador}(v)]$, e verifica as assinaturas do Validador, confirmando sua

autenticidade. O escrutinador faz a contagem de todos os votos válidos, ou seja, desconsiderando os votos com assinaturas inválidas (votos fabricados) e os votos com assinaturas duplicadas, atribuindo a cada candidato o número de votos recebido e calculando o resultado baseado no edital da eleição. Esta etapa é essencial para garantir a Equidade, ou seja, que o resultado da eleição permaneca secreto até o final desta.

O Quadro Negro recebe os votos não encriptados com suas respectivas assinaturas de validação, $[v, S_{validador}(v)]$, o resultado da eleição e chave privada do Escrutinador, e publica estas informações. Com essas informações em mãos, qualquer usuário pode confirmar o resultado da eleição através do uso da chave privada do Escrutinador, assim como confirmar que seu voto foi contabilizado.

2.4 Segurança do Protocolo

Nesta seção analisam-se quais requisitos de segurança são atendidos pelo protocolo e de que forma isso é alcançado.

2.4.1 Exatidão

O primeiro requisito de segurança de Exatidão garante que um voto produzido não pode ser duplicado, apagado ou alterado. Isto é assegurado pelo protocolo pelo fato de que, desde o momento que o voto é produzido, ele é assinado e encriptado. Desta forma, a fim de alterar um voto, sua assinatura e encriptações devem ser quebradas. Além do mais, o Quadro Negro e a Rede de Mistura são por premissa confiáveis e, por isso, não alteram, apagam ou duplicam nenhum voto. A não-duplicação dos votos é também garantida até a fase de escrutinação pela encriptação probabilística, a qual impede dois votos de possuírem a mesma forma. Por fim, como o Escrutinador publica sua chave privada ao final da votação, não é possivel que ele altere, apague ou duplique nenhum voto sem que isso possa ser detectado. Ao final da apuração, a integridade de todos os votos ainda é garantida pela assinatura do Validador.

Outro requisito da Exatidão é o de que todos os votos válidos devem ser contabilizados no resultado final. Esta garantia vem do fato de que todos os votos são publicados e misturados em um Quadro Negro e Rede de Mistura confiáveis, que não apagam nem invalidam nenhum voto. Sendo o Escrutinador não confiável, este poderia publicar um falso resultado. Porém, por ter sua chave privada publicada ao final do processo, o resultado final e a contabilização de cada voto podem ser verificados por qualquer um ao final da apuração. Assim, garante-se que todos os votos válidos são contados no resultado.

2.4.2 Democracia

Como premissas do protocolo tem-se um Quadro Negro e um Validador confiáveis. Dessa forma, apenas eleitores aptos a votar recebem uma cédula de votação; apenas votos pertencentes a eleitores válidos são assinados pelo Validador; e, caso algum eleitor não cadastrado tente enviar um voto ao Quadro Negro, este voto é recusado. Logo, o primeiro requisito da democracia de que apenas eleitores cadastrados podem votar é completamente atendido.

Outro requisito exige que cada eleitor cadastrado tem direito a lançar no máximo um voto. Esse requisito é cumprido pelo Quadro Negro confiável. Ele envia cédulas apenas para eleitores que ainda não votaram e recusa votos provenientes de eleitores que já possuam um voto publicado.

2.4.3 Confidencialidade

O primeiro requisito da Confidencialidade, o Anonimato, exige que um voto não possa ser associado ao eleitor que o produziu. Este requisito é cumprido por uma série de operações do protocolo. Primeiramente, o Validador não possui acesso ao conteúdo do voto recebido de um determinado eleitor. Além disso, por ser confiável, ele separa completamente o voto da assinatura do eleitor a ele associada. Em seguida, ao enviar o seu voto ao Quadro Negro, o Votador encripta-o com a chave pública da Rede de Mistura e do Escrutinador, ocultando assim seu conteúdo. Esta encriptação mantém-se até o escrutínio. Também, o embaralhamento dos votos realizados pela Rede de Mistura garante que um voto não possa ser associado a um eleitor através do conhecimento da ordem ou tempo de chegada dos votos.

A Equidade exige que todos os votos permaneçam em segredo até o final da fase de votação. Durante todo o processo de votação os votos permanecem armazenados encriptados duplamente, com a chave pública da Rede de Mistura e do Escrutinador. Mesmo que o Escrutinador esteja comprometido e divulgue sua chave privada para um agressor, nenhum voto pode ser decriptado antes de passar pela Rede de Mistura. Como esta última é confiável, o agressor só conseguiria acesso ao conteúdo dos votos após o seu embaralhamento. Nesse momento, todo o processo de votação já estaria finalizado. Assim, não é possível nenhum acesso a um resultado parcial.

A Incomprovabilidade exige que nenhum eleitor possa provar que votou de determinada maneira. Por não conhecer os fatores aleatórios utilizados para mascarar o voto enviado ao Validador e para encriptar o voto, o eleitor não pode associar o conteúdo de seu voto nem

ao voto enviado ao Validador nem ao voto publicado no Quadro Negro. Além disso, nesse requisito possui também um papel fundamental a utilização de assinaturas probabilísticas ou determinísticas pelo sistema, que será discutido adiante.

Por fim, a Incoagibilidade exige que nenhum eleitor possa ser forçado a votar de determinada maneira. Por exigir premissas extremamente complexas, este requisito não é atendido pelo protocolo VoteRemote e pode apenas ser atendido através de medidas tomadas pela organização da eleição, como, por exemplo, a alocação de mesários.

2.4.4 Verificabilidade

A Verificabilidade divide-se em Universal e Individual. A primeira exige que cada eleitor possa verificar que todos os votos foram contabilizados. A segunda, que cada eleitor possa verificar que o seu voto foi contabilizado. Ambas são garantidas pelo protocolo.

Após a fase de registro, publica-se uma lista com todos os eleitores cadastrados no Quadro Negro. Esta lista permite a todos verificar quem são os eleitores aptos a votar e seus respectivos certificados. Também, tanto a lista com os votos duplamente encriptados, quanto a lista com os votos encriptados apenas com a chave do Escrutinador são publicados no Quadro Negro, garantindo a todos os direito de verificar se o total de votos em cada fase está correto. Sendo o Quadro Negro e a Rede de Mistura confiáveis, nenhum voto é apagado, alterado ou duplicado por estes componentes. Mesmo o Escrutinador não sendo confiável, sua chave privada é publicada ao final da eleição, garantindo a todos os direito de decriptar os votos, verificar suas assinaturas e o conferir o resultado final da eleição. Assim, a Verificabilidade Universal é assegurada.

A Verificabilidade Individual é garantida caso seja usada a encriptação probabilística, que faz com que cada voto encriptado possua uma forma única. Assim, cada eleitor pode comprovar que seu voto consta no Quadro Negro (mesmo sem poder provar seu conteúdo). Como visto anteriormente, pode-se provar que todos os votos publicados no Quadro Negro foram contabilizados. Logo, prova-se que o voto de um determinado eleitor foi contabilizado.

2.5 Detalhes de Implementação

A comunicação entre os atores é baseada em uma estrutura PKI. Assim, cada ator possui o seu par de chave privada / certificado público para autenticação. Além disso, a Rede de Mistura e o Escrutinador possuem pares adicionais para a encriptação dos dados.

Toda comunicação entre o Validador, o Votador e o Quadro Negro deve ser mutualmente autenticada e encriptada utilizando uma conexão TLS/SSL.

A encriptação utilizada nesse protocolo deve ser do tipo probabilística, visando fornecer ao eleitor a Verificabilidade Individual de seu voto, além de evitar que um agressor consiga duplicar um voto.

Quanto ao tipo de assinatura, pode-se usar tanto assinaturas probabilísticas quanto assinaturas determinísticas na implementação do protocolo. Dependendo da escolha realizada, diferentes requisitos de segurança serão atendidos pelo sistema.

A assinatura probabilística dificulta a duplicação de votos e garante uma maior Verificabilidade. Por outro lado, facilita a venda do voto por parte do eleitor, já que torna possível a um eleitor comprovar a sua escolha através da assinatura única e do voto legível publicado no Quadro Negro, caso a Rede de Mistura e o Escrutinador sejam implementados como um só módulo. Já assinaturas determinísticas fortalecem o requisito de Incomprovabilidade, por dificultarem a associação de uma assinatura a um voto. Por outro lado, não servem como comprovante para que um eleitor possa posteriormente verificar que seu voto foi contabilizado.

Dessa forma, o tipo de assinatura a ser usada no sistema deve ser escolhido baseado na importância dos requisitos Verificabilidade e Incomprovabilidade para o sistema proposto.

2.6 Sinopse

Neste capítulo, descreveram-se os requisitos alcançados pelo protocolo VoteRemote, que incluem Exatidão, Democracia, Anonimato, Equidade e Verificabilidade. Por outro lado, o requisito Incomprovabilidade é atendido apenas dependendo da implementação do sistema, enquanto a Incoagibilidade não é alcançada. Estes requisitos são garantidos, desde que um conjunto de premissas sejam asseguradas, entre elas a existência de uma infraestrutura de chave pública confiável, e que o Validador, Quadro Negro e a Rede de Mistura sejam confiáveis. Descreveu-se também a diferença entre o uso de assinatura probabilística ou determinística no sistema. Enquanto a primeira fortalece o requisito de Verificabilidade, a segunda atende mais amplamente o requisito de Incomprovabilidade.

Capítulo 3

SISTEMA PROPOSTO

Neste capítulo descreve-se o sistema proposto baseado no protocolo VoteRemote. Esta descrição, que independe das tecnologias a serem utilizadas, visa à especificação de um sistema que atenda a todas as premissas do protocolo e execute corretamente seu funcionamento. Além disso, dois módulos adicionais são descritos a fim de garantir características de sistemas reais. Por fim, descrevem-se ataques e possíveis medidas preventivas fora do escopo do protocolo.

3.1 Descrição do sistema

O sistema proposto neste projeto consiste em um sistema de votação eletrônica, baseado no protocolo VoteRemote descrito no capítulo anterior. Isto significa que o sistema desenvolvido é responsável por garantir todas as premissas elencadas pelo protocolo, implementar seu funcionamento e, ainda, atender a outros requisitos esperados de sistemas reais.

Para melhor entendimento do funcionamento do sistema, sua arquitetura simplificada está representada na figura 3.1. Nesta, observa-se que o sistema é coordenado pelo Quadro Negro, componente que comunica-se com grande parte dos outros módulos.

Neste capítulo explica-se o funcionamento do sistema, depois, quais requisitos adicionais são por ele atingidos, além de medidas adicionais cabíveis para aumentar seu nível de segurança.

3.1.1 Funcionamento do sistema

Tipos de votação

Um sistema de votação eletrônica pode ser utilizado tanto para um processo de votação no qual os componentes utilizados baseiam-se em hardwares eletrônicos, ao invés de cédulas e papéis, quanto para realizar-se uma votação pela internet, no qual eleitores podem escolher seus candidatos sem precisar sair do conforto de seu lar.

No que diz respeito à implementação, o que difere os dois tipos de processo é apenas a implementação do módulo Votador. O sistema aqui proposto pretende permitir a realização de eleições utilizando ambos os tipos de processo.

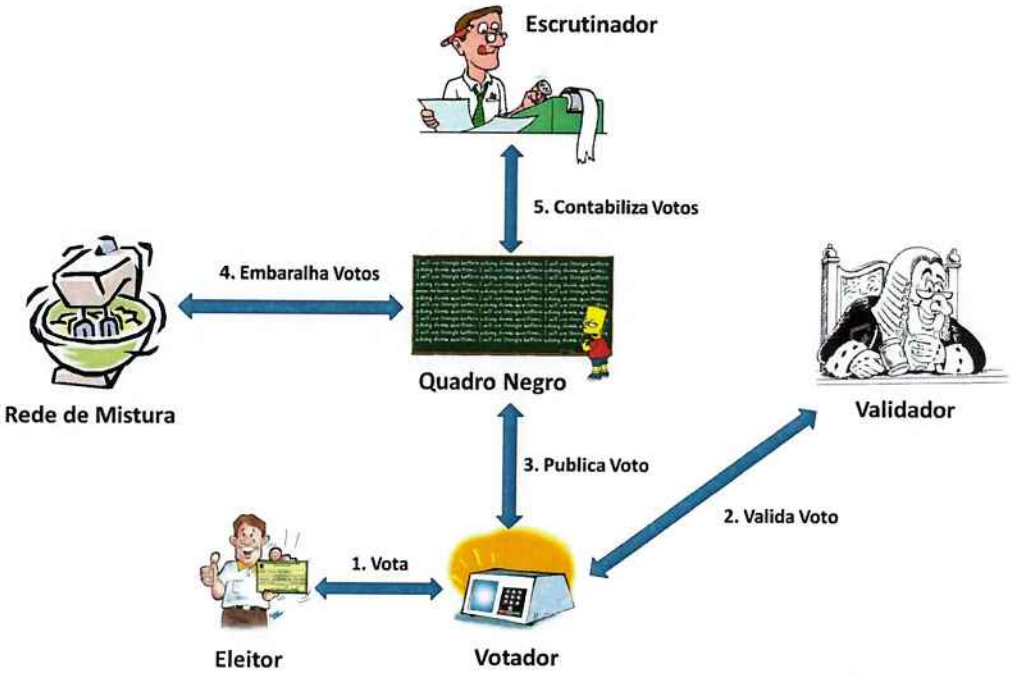

Figura 3.1: *Arquitetura simplificada do sistema.*

Em uma votação eletrônica do tipo presencial, ou seja, que o eleitor necessite ir até uma zona eleitoral para registrar seu voto, basta que exista uma "urna eletrônica"executando uma implementação do módulo Votador. Apesar de ser possível implementar esta urna através de um hardware especifico e dedicado, um dos objetivos desse sistema é o de facilitar a execução de um processo eleitoral seguro exigindo premissas pouco complexas. Assim sendo, a implementação do protocolo foi pensada de forma a evitar a necessidade de hardware específico, permitindo que um computador pessoal executando um sistema operacional facilmente encontrado no mercado seja utilizado atendendo a todos os requisitos do sistema. Dessa forma, os requisitos mínimos para execução do aplicativo de votação não se tornariam obstáculos impedindo usuários de organizar ou participar do processo eleitoral.

Já no caso de uma votação pela internet, pode-se facilmente distribuir o módulo Votador como um componente de software a ser instalado e configurado na máquina do eleitor ou mesmo implementar a interface gráfica do Votador na forma de uma página Web. Este procedimento não fere nenhum requisito de segurança atentido pelo protocolo, mas não garante a Incoagibilidade, a qual pode ser garantida por um mesário caso a votação seja presencial.

Apesar de um sistema de votação pela Internet proporcionar uma maior facilidade de distribuição e de acesso ao processo eleitoral, por outro lado, requisitos de segurança como a Incomprovabilidade e a Incoagibilidade tornariam-se difíceis de serem alcançados. A escolha, então, entre um dos dois métodos de votação deve levar em conta as vantagens de uma votação *online* contra os requisitos de segurança exigidos para o processo eleitoral em questão.

Registro do eleitor

O registro do eleitor deve ser realizado por uma entidade responsável estabelecida pelo edital de votação. Este registro consiste em atribuir para todos os eleitores aptos a votar um número de identificação único e um par de chaves pública e privada.

Seu número de identificação, certificado público e informações adicionais sobre o eleitor (nome, foto, data de nascimento, cartório em que foi registrado, etc.) devem ser então guardados de forma a serem carregados no sistema antes do início do processo de votação. Este procedimento corresponde à fase 0 do protocolo e é implementado pelo sistema através da leitura de um arquivo contendo as informações de todos os eleitores e seu armazenamento no Quadro Negro.

É de responsabilidade da entidade registradora garantir a consistência dos dados carregados no sistema, impedindo eleitores de serem cadastrados mais de uma vez, ou com informações incorretas. Porém, por ser o Quadro Negro público, todos podem verificar o cadastro de cada eleitor, garantindo uma maior segurança ao processo.

Já a chave privada pertence exclusivamente ao eleitor e não deve ser compartilhada por ele com mais ninguém. A forma de distribuição desta chave varia com o nível de segurança apresentado pelo sistema. Para uma maior segurança, recomenda-se gerá-la e distribuí-la armazenada em um hardware criptográfico como, por exemplo, um *smartcard*. Porém, para projetos com maiores restrições orçamentárias, pode-se gerá-la por software e armazená-la em uma mídia digital convencional, como um *pen drive*.

A senha correspondente à chave privada pode ser, por exemplo, um código ASCII conhecido apenas pelo eleitor, ou, para garantir uma maior comodidade aos eleitores e para evitar que eleitores emprestem suas chaves privadas e senha para terceiros votarem por eles, uma cadeia de bytes gerada a partir de uma leitura biométrica, como, por exemplo, da impressão digital ou retina do eleitor.

Registro dos candidatos

Assim como o registro dos eleitores, o registro dos candidatos deve ser realizado por uma entidade responsável. Este registro consiste em atribuir a cada um dos candidatos um número de identificação único. Também, caso aplicável a cada votação em específico, pode constar uma opção na qual o eleitor se abstém de votar.

O número de identificação de cada candidato, junto com possíveis informações adicionais como foto, partido, entre outras, ficam armazenados em um arquivo de configuração até o início da eleição, quando são carregados pelo Quadro Negro. Não há necessidade de se proteger esse arquivo, pois por ser publicada, qualquer pessoa pode verificar a autenticidade desta lista.

Pares de chaves

Para a implementação da camada de criptografia do sistema são necessários pares de chaves públicas e privadas para os módulos Validador, Rede de Mistura, Escrutinador e para cada um dos eleitores. Todos esses pares são gerados antes da votação, sendo carregados na inicialização do sistema.

As chaves públicas são armazenadas no formato de certificados digitais X509, os quais devem ser assinados por uma entidade certificadora confiável para garantir sua autenticidade. No momento da inicialização do sistema, armazenam-se todos os certificados públicos no Quadro Negro, de forma que estejam sempre disponíveis para qualquer módulo que os necessitem. Além do mais, por ser o Quadro Negro público, qualquer pessoa pode verificar se os certificados utilizados realmente correspondem aos certificados públicos dos módulos, aumentando a segurança do sistema.

As chaves privadas dos módulos, por outro lado, devem permanecer secretas, acessíveis somente aos módulos correspondentes. É necessário também analisar alternativas para armazenar as senhas associadas às chaves privadas de cada módulo.

Uma alternativa seria armazenar as senhas de forma *hardcoded* no código dos módulos. Esta solução apresenta, porém, uma falha de segurança. Caso um agressor consiga furtar o hardware correspondente a um dos módulos da eleição, ele será capaz de inicializar e executar o sistema sem maiores problemas, já que a senha encontra-se salva em memória não-volátil. Seria possível também, em alguns casos, obter a senha a partir do bytecode de execução do programa através de técnicas de decompilação.

Um alternativa melhor seria a separação física entre módulo e senha, atribuindo a uma pessoa a responsabilidade de guardar esta última. Assim a senha poderia ser gerada e

salva em um *smartcard* e a responsabilidade delegada a um juiz eleitoral, por exemplo. O módulo lê a senha e, em seguida, armazena-a em sua memória volátil para utilizá-la durante sua execução. Caso houver qualquer interrupção na execução do sistema, este não poderá ser reinicializado sem a autorização do juiz responsável e a senha não poderá ser recuperada a partir do *hardware* do módulo.

Publicação

O Quadro Negro é o elemento central do protocolo VoteRemote. Grande parte da segurança oferecida pelo protocolo, e portanto pelo sistema que o implementa, advém da existência de um componente central público. Toda informação escrita no Quadro Negro pode ser lida por qualquer um, ou seja, o Quadro Negro *publica* tudo que recebe.

No contexto de uma votação eletrônica, pode-se definir a publicação como a distribuição de um arquivo eletrônico contendo as informações a serem publicadas. Um arquivo contendo todas as informações publicadas pelo Quadro Negro pode ser disponibilizado para download no site da votação, ou ainda, a informação pode ser compartilhada diretamente através de uma página html.

Com isso, o conteúdo fica acessível a todos, permitindo que entidades fiscalizadoras, auditorias, organizações não governamentais, ou até os próprios eleitores possam verificar diversos passos do processo eleitoral.

Recibo

Um dos requisitos de segurança mais fortes atendido por este sistema proposto é o da Verificabilidade. Todos os eleitores podem comprovar que o seu voto foi contabilizado no resultado final da eleição, mesmo sem poder comprovar o seu conteúdo.

Para que isto seja possível, ao terminar de votar e ter o seu voto publicado, o eleitor recebe um comprovante de votação, chamado Recibo. Este é constituído por uma cadeia de bytes e pode ser entregue ao eleitor em formato digital ou impresso em uma folha de papel.

Este recibo representa o voto assinado pelo Validador e duplamente encriptado pelas chaves públicas da Rede de Mistura e do Escrutinador. Desta forma, por ser o resultado de dois processos criptográficos probabilísticos, o Recibo não revela nenhuma informação sobre o conteúdo do voto do eleitor, não precisando permanecer secreto.

Por outro lado, esta cadeia de bytes é exatamente igual à que é publicada pelo Quadro Negro após o voto ser registrado. Logo, ao final do processo de votação, o eleitor pode facilmente consultar o Quadro Negro e verificar se o seu voto encontra-se ali publicado.

Como a Rede de Mistura é por premissa confiável, o voto do eleitor também estará entre os votos registrados na segunda publicação do Quadro Negro e, portanto, entre os votos enviados para o Escrutinador. Após o escrutínio, qualquer um pode verificar o funcionamento do módulo Escrutinador, utilizando sua chave privada publicada. Desta forma, garante-se que todos os votos foram contabilizados. Portanto, o Recibo garante ao eleitor o direito de verificar que seu voto realmente foi registrado.

Inicialização do Sistema

Seguindo a explicação dos itens anteriores, a inicialização do sistema começa com o carregamento dos certificados públicos de todos os módulos e da lista de eleitores aptos a votar. Em seguida, é requisitada a inserção dos meios digitais contendo as senhas das chaves privadas de cada um dos módulos, que também são carregadas.

Após esta autorização, o Quadro Negro recebe a carga com informações sobre todos os candidatos que podem ser votados. Neste momento, os módulos votadores são habilitados para que a eleição possa começar.

Fase de Votação

Cada eleitor que desejar votar deve seguir o seguinte procedimento:

No caso de uma votação presencial, o votador deve comparecer até uma urna de votação, enquanto em uma votação pela internet, basta que instale o software no seu computador ou acesse a página Web correspondente.

O sistema pede então sua identificação, e o eleitor deve inserir a mídia contendo sua chave privada e inserir sua senha, sendo ela uma cadeia de caracteres memorizada pelo eleitor ou uma cadeia de caracteres automaticamente gerada a partir de um leitor biométrico. Se essas informações estiverem corretas, o módulo Votador carregará a chave privada do eleitor e tentará se autenticar com o Quadro Negro. Somente se o eleitor estiver na lista de eleitores cadastrados e ainda não tiver votado o Quadro Negro autorizará seu voto enviando uma cédula ao módulo Votador.

O eleitor escolhe então o candidato de sua preferência e confirma seu voto. O voto é assim gerado pelo Votador e encaminhado ao Validador. Este novamente verifica se o eleitor está cadastrado e ainda não votou, verifica a integridade do voto e o assina. O voto é devolvido para o Votador, que o encripta duplamente com as chaves públicas da Rede de Mistura e do Escrutinador antes de ser publicado no Quadro Negro.

Caso o voto tenha sido corretamente publicado, o Recibo é impresso ou salvo em uma mídia digital e entregue ao eleitor, finalizando o processo de votação.

Todo esse processo está descrito no diagrama de sequência da figura 3.2.

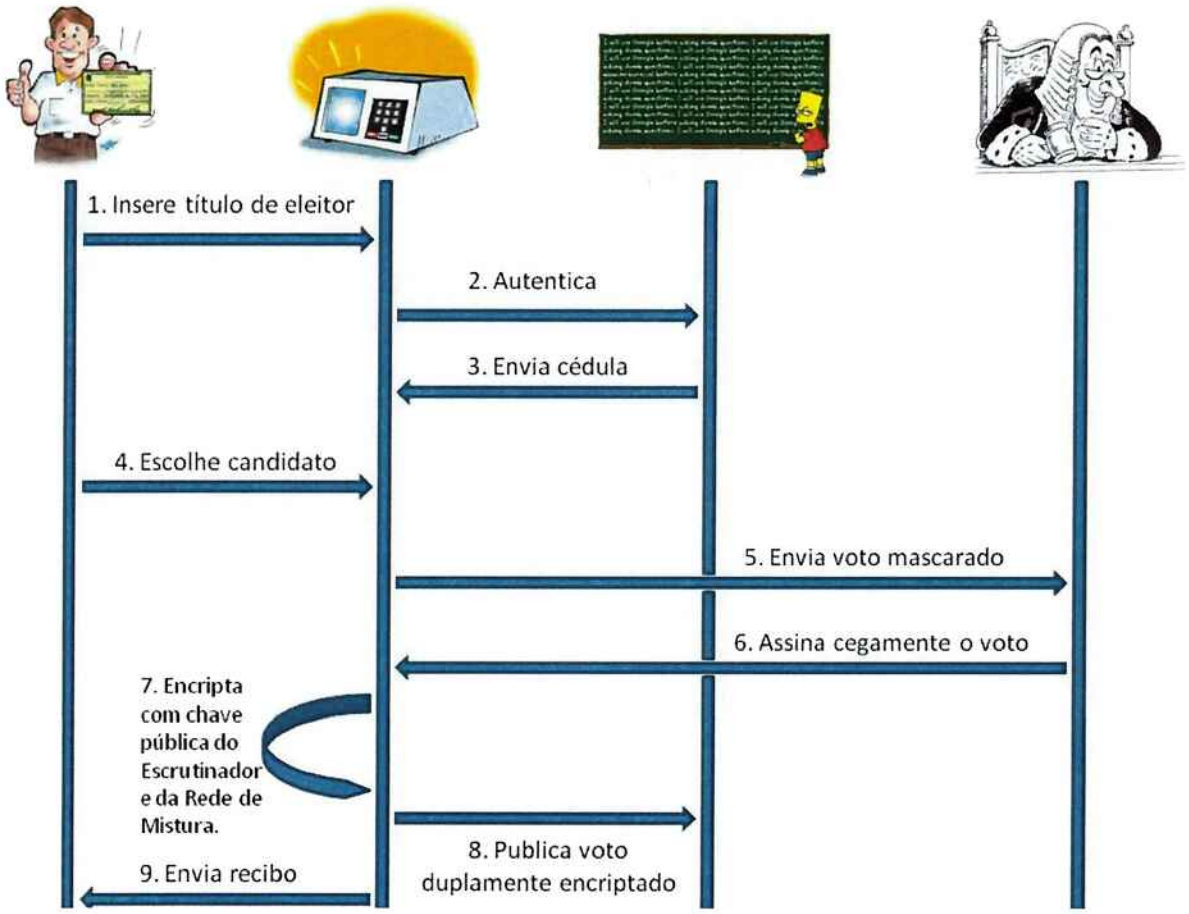

Figura 3.2: *Diagrama de sequência da Fase de Votação.*

Embaralhamento dos votos

Quando for desejado terminar a votação, um juiz eleitoral responsável deve acionar o fim da Fase de Votação no sistema. É iniciado então o processo de apuração, que tem como primeira parte o embaralhamento dos votos.

Todos os votos publicados no Quadro Negro são enviados de uma única vez à Rede de Mistura, que os decripta com sua chave privada e aplica um algoritmo de sorteio aleatório para embaralhá-los. O conjunto resultante, que naturalmente deve ter o mesmo tamanho do conjunto de entrada, é então devolvido ao Quadro Negro.

Vale ressaltar que há uma diferença entre o sistema aqui proposto e uma recomendação do protocolo VoteRemote. Ao utilizar-se assinaturas do tipo probabilística, o protocolo sugere eliminar o elemento Escrutinador, agregando sua função à Rede de Mistura do

sistema. Neste caso, a Rede de Mistura também deixaria de ser confiável por premissa, publicando sua chave privada ao fim do processo de apuração.

Retirar a premissa de confiabilidade de um elemento do sistema e ainda manter a Verificabilidade conseguida pelo protocolo seria uma excelente vantagem, mas o preço a se pagar por essa conquista não é baixo. Ao se publicar a chave privada da Rede de Mistura, qualquer pessoa terá ao final do processo de apuração todas as informações necessárias para decriptar os votos encriptados registrados no Quadro Negro. Dessa forma, seria possível descobrir o conteúdo do voto de um eleitor com base na ordem de chegada dos votos e na ordem de votação, permitindo a condenável prática de venda de votos. Ademais, bastaria decriptar o voto contido no Recibo do eleitor para provar o seu conteúdo.

Por estas razões, decidiu-se neste trabalho não seguir esta recomendação do protocolo. Apesar de se utilizar assinaturas probabilísticas, decidiu-se manter um elemento confiável a mais no sistema, adicionando complexidade às suas premissas, mas, ainda assim, continuando a alcançar os requisitos de Verificabilidade, Anonimato e Incomprovabilidade.

Este processo de embaralhamento dos votos está representado no diagrama de sequência da Fase de Apuração, figura 3.3.

Escrutínio dos votos

Finalmente, a última fase do processo é o escrutínio dos votos. Todos os votos encriptados no Quadro Negro são enviados ao escrutinador, que simplesmente retira a criptografia com sua chave privada, verifica a assinatura de cada voto e, então, contabiliza o resultado da eleição.

Tanto os votos com assinatura, quanto o resultado e a chave privada do próprio Escrutinador são enviados de volta ao Quadro Negro para publicação.

Neste momento, todo o processo eleitoral estará terminado e o sistema encerra seu funcionamento. Restam apenas os arquivos de publicação do Quadro Negro, que podem ser acessados por qualquer pessoa mesmo depois do fim da votação.

Este processo de escrutínio dos votos também está representado no diagrama de sequência da Fase de Apuração, figura 3.3.

Modo de recuperação

O sistema proposto além de implementar o protocolo VoteRemote deve também estendê-lo em alguns aspectos. Um modo que necessita fazer parte de um sistema real, mas que

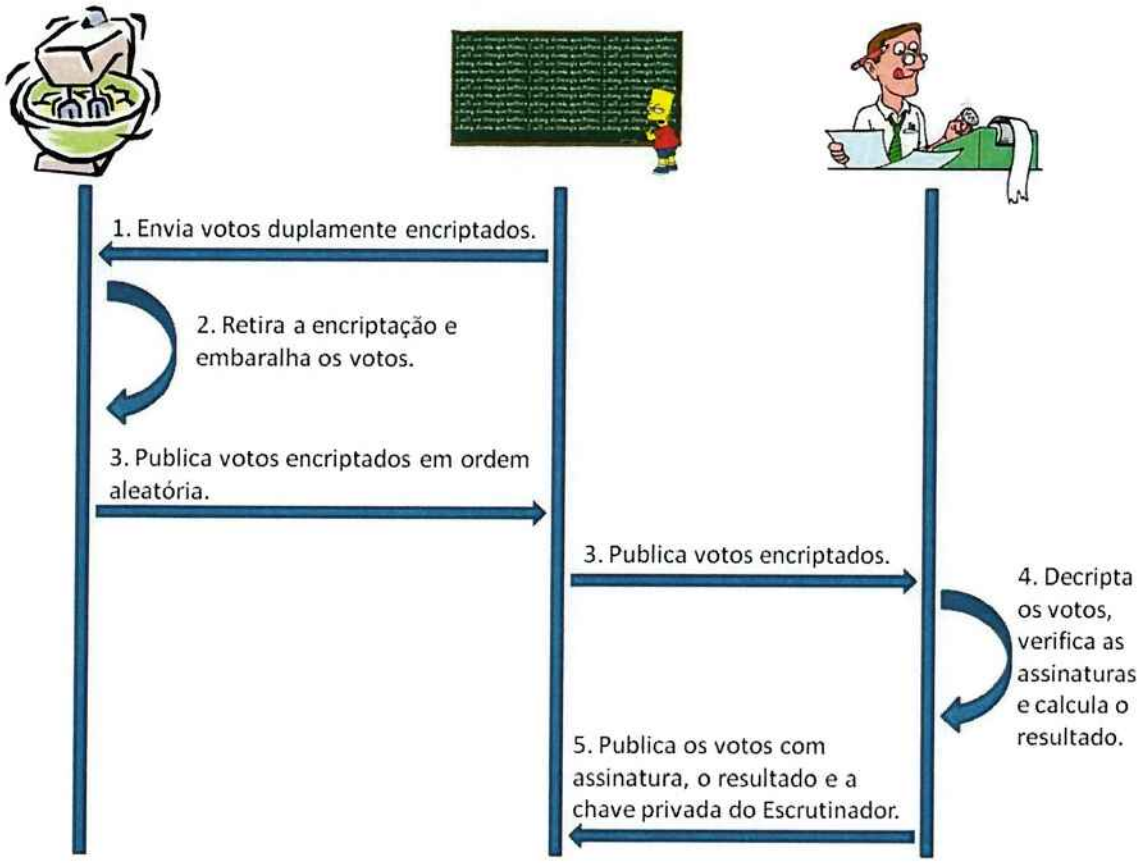

Figura 3.3: *Diagrama de sequência da Fase de Apuração.*

não está descrito no protocolo, é o Modo de Recuperação. O que aconteceria se, por exemplo, o fornecimento de eletricidade do servidor falhasse por alguns minutos após cerca de duas horas de votação e o sistema precisasse ser reiniciado? O que aconteceria com os votos registrados até o momento?

O Modo de Recuperação é acionado se o sistema precisar ser reiniciado após o início da votação. Por sistema, estamos nos referindo ao Quadro Negro, único componente que armazena informações durante o processo de votação.

Esse Modo de Recuperação deve atender a dois requisitos básicos: ser confiável e seguro. Confiável no sentido de recuperar completamente todos os votos já registrados, e seguro para não permitir o acesso indevido às informações nele contidas, nem carregar votos falsos.

Como descrito anteriormente, ao publicar-se um voto no Quadro Negro, este é escrito em um arquivo de saída. Dessa forma, basta que esse arquivo seja consistente após a escrita de cada voto para que ele também sirva como *backup* do sistema. No caso de

reinicialização do Quadro Negro, todos os votos do arquivo devem ser carregados para que a votação possa continuar. Porém, é necessário também que outra informação seja armazenada durante o andamento da eleição: o número de identificação de cada eleitor que já votou. Caso o Modo de Recuperação não recarregue esse dado, um mesmo eleitor poderia produzir um novo voto toda vez que o Quadro Negro fosse reiniciado.

Este procedimento é confiável, pois retoma o sistema exatamente ao ponto em que havia parado, com todos os votos registrados completamente recarregados e todos os eleitores que já votaram impossibilitatos de fazê-lo novamente.

Além disso, todos os votos publicados estão duplamente encriptados com a chave da Rede de Mistura e do Escrutinador, ambas secretas até esse momento. Ou seja, não fornece nenhuma informação sobre o eleitor ou seu voto.

Ademais, estando as premissas do protocolo atendidas, não é possível alterar ou duplicar votos de forma não detectável. Alterar um voto tornaria a assinatura do Validador inválida; e duplicar um voto seria uma ação facilmente detectável pelo fato da assinatura probabilística do Validador ser única.

Para garantir que nenhum voto será apagado no arquivo de *backup* é necessário uma proteção adicional. Criptografar ou assinar o arquivo com os votos afetaria bastante o desempenho do sistema, já que esta operação teria de ser realizada toda vez que um voto novo fosse registrado e com um arquivo de tamanho considerável. Por outro lado, encriptar apenas o arquivo que salva o número de identificação dos eleitores que já votaram não afetaria de forma significativa o desempenho geral do sistema. Comparando-se os dois arquivos, um com o *backup* dos votos e o outro com a identificação dos eleitores, qualquer ausência de voto seria facilmente identificável.

Verificador

Como dito em seções anteriores, um dos pontos fortes do sistema proposto é a Verificabilidade alcançada. Não só o resultado final pode ser verificado, como também diversos passos intermediários do funcionamento do sistema. Mesmo operações consideradas confiáveis por premissa possuem passos que podem ser verificados.

Isto é alcançado pela existência do Quadro Negro público, que divulga informações durante o processo de votação e apuração, sem no entanto comprometer a segurança do sistema. Estas informações podem ser analisadas individualmente e comparadas entre si, a fim de localizar tentativas de falsificação de votos ou fraude no resultado final.

A possibilidade deste processo de verificação retira a necessidade de alguns módulos serem confiáveis, transferindo-a para entidades responsáveis por verificar o processo

eleitoral. Por exemplo, o módulo Escrutinador do sistema não precisa ser confiável, pois publica sua chave privada após o fim do processo de apuração. Assim, todos podem verificar a exatidão de seu resultado publicado.

Pode-se definir um módulo novo complementador para o sistema: o módulo Verificador. Este componente independe do protocolo VoteRemote, e tem como função facilitar o processo de inspeção. O módulo não precisa ser confiável e a idéia é que, baseado em um arquivo de saída padrão do Quadro Negro especificado pelo sistema, qualquer entidade, seja ela uma entidade regulamentadora, uma auditoria, uma organização não-governamental ou mesmo um indivíduo, possa ser capaz de criar o seu próprio módulo Verificador e inspecionar o funcionamento do sistema.

A seguir, encontram-se listadas algumas informações passíveis de serem verificadas:

- Lista de todos os eleitores aptos a votar, junto com seus respectivos certificados públicos.
- Lista de todos os candidatos concorrendo na eleição.
- Certificados públicos de todos os módulos do sistema.
- Quantidade de votos registrados durante a fase de votação.
- Quantidade de votos após o embaralhamento pela Rede de Mistura.
- Quantidade de votos após o escrutínio.
- Resultado da eleição baseado nos votos decriptados após o escrutínio.
- Validade das assinaturas de todos os votos decriptados após o escrutínio.
- Existência de assinaturas duplicadas.
- Presença de um voto específico na lista de votos duplamente encriptados, baseado no recibo de um eleitor.

3.1.2 Requisitos adicionais

O sistema proposto atende a todos os requisitos de segurança abrangidos pelo protocolo VoteRemote. Porém, como um sistema real, para que seja viável, outros requisitos também devem ser alcançados.

Um requisito importante do sistema a ser desenvolvido diz respeito à usabilidade: o sistema não pode ser desenvolvido sob a premissa de que o usuário terá acesso *a priori*

a informações sobre como usá-lo, e por se tratar de um sistema que visa ser utilizado por um grupo grande e heterogêneo de pessoas, deve ser portanto intuitivo para os eleitores. Não adianta a este sistema atingir um grau de segurança elevado, mas que possua uso tão complicado, que apenas especialistas em criptografia conseguiriam utilizá-lo.

Outro requisito é a facilidade de distribuição. Caso a votação seja do tipo não-presencial, ou seja, pela Internet, basta que seja fácil instanciar um servidor para uma nova eleição, já que o acesso do eleitor será bem simples através de páginas Web. No caso de uma votação presencial, o aplicativo cliente do Votador deve apresentar instalação e configuração simples na máquina de votação. Este requisito é essencial para popularização e sucesso do sistema.

3.1.3 Análise de ataques

Nesta seção descrevem-se alguns ataques que vão além do nível de segurança do protocolo, suas consequências e possíveis medidas para evitá-los.

Ataques

- Divulgação da chave privada do Validador: o agressor conseguir acesso à chave privada do Validador seria uma das maiores falhas de segurança que este sistema poderia sofrer. De posse dessa chave, o agressor seria capaz de fabricar votos válidos. Ainda assim, ele precisaria conseguir acesso a chaves privadas de eleitores para conseguir se autenticar, ou ser capaz de escrever diretamente no Quadro Negro.

- Divulgação da chave privada da Rede de Mistura: de posse da chave privada da Rede de Mistura, o agressor conseguiria decriptar, após o fim da fase de apuração, todos os votos duplamente encriptados publicados no Quadro Negro. Obtendo também informações sobre a ordem de votação dos eleitores, seria possivel mapear cada voto ao eleitor correspondente, quebrando o requisito de Anonimato. Porém, mesmo de posse dessa informação, o agressor ainda não seria capaz de alterar, duplicar ou apagar votos registrados. Todas essas ações ficariam evidentes no processo de verificação.

- Divulgação da chave privada do eleitor: uma desvantagem de um sistema de votação eletrônica é a falta de fiscalização sobre o eleitor no momento da votação. Por exemplo, caso o eleitor compartilhe seu *smartcard* com um outro eleitor, este último pode se passar pelo primeiro e computar um voto.

Medidas

- Mantendo a chave privada secreta: para se evitar o vazamento de uma chave privada, a sua senha deve ser armazenada em um local confiável. No protótipo implementado, por exemplo, a senha fica sob a posse somente de um juiz eleitoral. Uma solução para aumentar o nível de segurança, seria atribuir essa responsabilidade a mais de uma pessoa. Um grupo de juízes poderia por exemplo carregar uma chave privada cada um. Dessa forma, em vez de encriptar-se somente uma vez um voto com a chave pública da Rede de Mistura, repete-se esse procedimento cinco vezes, cada um correspondendo a um certificado diferente. Ou ainda, além de utilizar um certificado público do módulo, cada candidato disputando a eleição poderia fornecer o seu própro par de chaves públicas e privadas. Assim, além de descobrir a chave privada do módulo, o agressor precisaria obter a chave privada sob a posse de cada um dos candidatos.

- Fiscalização da cabine de votação: esta medida é aplicável para o caso de uma votação do tipo presencial. Nesse caso, para evitar que um eleitor empreste sua chave secreta para outro eleitor, a cabine pode ser equipada com um monitor externo. Ao inicar o processo de votação, esse monitor pode apresentar algumas informações relacionadas ao número de identificação do eleitor votante, como por exemplo foto e sexo. Assim, qualquer pessoa que estiver observando por fora a cabine pode constatar uma irregularidade caso a foto não corresponda à pessoa dentro da cabine.

3.2 Sinopse

Neste capítulo descreveu-se um sistema de votação eletrônica completo baseado no protocolo VoteRemote. Esta descrição inclui as características que o sistema deve apresentar, de forma que as premissas do protocolo sejam atendidas; requisitos adicionais presentes em sistemas reais; e o comportamento do sistema de forma que o funcionamento do protocolo seja corretamente implementado. Apesar desta especificação independer de tecnologias a serem utilizadas, comentou-se a influência da escolha da tecnologia sobre os requisitos de segurança. Além disso, apresentaram-se módulos, cujas funcionalidades não faziam parte do escopo do procotolo, como o Modo de Recuperação, mas cuja presença é imprescindível em um sistema real, já que influenciam diretamente na robustez do sistema.

Capítulo 4

SEGURANÇA DA INFORMAÇÃO

Neste capítulo descrevem-se os algoritmos que compõem a camada de criptografia do sistema, como suas bases teóricas, mecanismo de funcionamento e parâmetros aceitos.

4.1 Algoritmo RSA

O algoritmo RSA (RIVEST; SHAMIR; ADLEMAN, 1978) é um dos algoritmos criptográficos mais usados atualmente em protocolos eletrônicos. É baseado no modelo de chave pública e pode ser usado tanto para encriptação quanto para assinaturas. É considerado seguro dadas chaves suficientemente longas e implementações atualizadas.

O algoritmo realiza encriptação probabilística e, para isso, utiliza um par de chaves, uma pública e outra privada. A chave pública é usada para encriptar mensagens, e pode ser conhecida por todos que queiram enviar mensagens. É distribuída na forma de certificados públicos. A chave privada é a única que consegue decriptar uma mensagem cifrada pela chave pública, e portanto só deve ser conhecida pelo destinatário da mensagem.

A operação do algoritmo RSA consiste nas seguintes etapas:

- Geração das Chaves:
  - Escolha dois números primos suficientemente grandes, p e q.
  - Calcule $n = p * q$.
  - Calcule $phi = (p-1) * (q-1)$.
  - Escolha um inteiro e, tal que $1 < e < phi$ e e e phi sejam primos entre si.
  - Calcule o expoente secreto d, tal que $1 < d < phi$ e $ed \equiv 1 \pmod{phi}$
  - Os pares (n, e) e (n, d) são a chave pública e privada, respectivamente.
- Encriptação:
  - Obtenha a chave pública (n, e) do destinatário.
  - Calcule $c = m^e \bmod n$, onde m é mensagem a ser transmitida representada por um inteiro positivo.
  - Envie c.

- Decriptação:
  - Use a chave privada *(n, d)* e calcule $m = c^d \ mod \ n$.

Esta sequência de procedimentos é utilizada para se transmitir uma mensagem encriptada do remetente para o destinatário. Além disso, o RSA também pode ser utilizado para assinar uma mensagem, garantindo que o remetente é realmente quem ele diz ser. Para isto, a operação é exatamente a mesma descrita acima, apenas utilizando-se a chave privada para encriptação e a chave pública para decriptação. Dessa forma, por ser a chave privada apenas de conhecimento do remetente, todos que possuírem a chave pública podem garantir a origem da mensagem. Além disso, um esquema de assinatura do RSA, chamado de RSA PSS , pode ser usado para gerar assinaturas probabilísticas.

A segurança do algoritmo RSA está baseada na dificuldade de se resolver o problema da fatoração inteira, que apresenta complexidade sub-exponencial. Dessa forma, para uma chave de 1024 bits, o algoritmo apresenta uma segurança de aproximadamente 80 bits, enquanto que para uma chave de 3072 bits, a segurança é de 128 bits.

4.2 Assinatura Cega

Assinatura cega é um tipo de assinatura introduzida em (CHAUM, 1982), no qual o conteúdo da mensagem é mascarado antes de ser assinado. Ainda, a assinatura realizada sobre a mensagem mascarada também é válida para a mensagem original. É equivalente a assinar um documento contido em um envelope selado, porém permitindo que a assinatura passe através deste, alcançando o conteúdo do documento lacrado.

Pode-se implementar o esquema de assinatura cega com o uso do algoritmo RSA e um fator aleatório. A sequência de procedimentos encontra-se a seguir.

- Escolha um fator aleatório *r* , tal que *gcd(r, N) = 1*, sendo *N* o módulo da chave pública RSA.
- Calcule $m' = m * r^e \ mod \ n$, sendo *m* a mensagem a ser assinada e *e* o expoente da chave pública RSA.
- *m'* representa a mensagem mascarada e não fornece nenhuma informação sobre a mensagem original, podendo ser enviada a outro módulo para ser assinada.
- A assinatura é calculada por $s' = (m')^d \ mod \ N$, sendo *d* o expoente da chave privada RSA.

- O fator *R* pode ser então removido pela primeira entidade através do cálculo $s \equiv s' * r^{-1} \ mod \ N$.
- *s* equivale a assinatura da mensagem original *m*.

Um cuidado que deve ser tomado no uso desse algoritmo é o de nunca usar a mesma chave privada para as funções de assinar cegamente e encriptar dados. Caso isso seja feito, a implementação estará sujeita a um ataque conhecido como *"RSA blinding attack"*, no qual o agressor convence o sistema a decriptar uma mensagem encriptada, passando-a como se fosse uma mensagem mascarada a espera de ser assinada.

4.3 AES

O AES (Advanced Encryption Standard) (SPECIFICATION..., 2001) é um padrão de encriptação de chave simétrica usado pelo governo dos EUA. Originalmente publicado como Rijndael, o padrão é formado por três cifras de bloco: AES-128, AES-192, AES-256, apresentando níveis de segurança de 128, 192 e 256 bits, respectivamente.

A cifra trabalha em blocos de 128 bits e baseia-se na aplicação de repetidos *rounds* de transformações, como descrito em (SPECIFICATION..., 2001). A quantidade de *rounds* varia com o tamanho de chave escolhido, e cada *round* constitue-se de um número de passos, incluindo um que depende da chave. O processo de decriptação, por sua vez, aplica *rounds* reversos usando a mesma chave de encriptação.

O algoritmo segue a seguinte descrição.

- *KeyExpansion*: a partir da chave da cifra são geradas chaves para cada *round*.
- *Initial Round*
  - *AddRoundKey*: o bloco processado é somado bit a bit com a chave do *round*.
- *Rounds*
  - *SubBytes*: aplicação da *S-Box* no bloco.
  - *ShiftRows*: cada linha do bloco é deslocada um certo número de vezes.
  - *MixColumns*: multiplica cada coluna do bloco por um polinômio fixo.
  - *AddRoundKey*
- *Final Round*

- *SubBytes*
- *ShiftRows*
- *AddRoundKey*

4.4 Algoritmo SHA-2

SHA-2 é um conjunto de funções de *hash* desenvolvidas pela NSA e publicadas pelo NIST (NIST, 2010) como padrão para o governo dos EUA. É constituído por quatro funções com diferentes tamanhos de *hash* de saída, que são de 224, 256, 384 ou 512 bits, apresentando níveis de segurança de 112, 128, 192 ou 256 bits, respectivamente.

No momento, um novo padrão de *hash* (SHA-3) está em desenvolvimento. Em 2012, este novo padrão provavelmente já estará definido.

4.5 Sinopse

Neste capítulo descreveu-se a base da camada de criptografia do sistema proposto, composta por algoritmos largamente utilizados por sistemas reais. Apresentou-se seu funcionamento resumido, além de parâmetros aceitos e detalhes sobre seus níveis de segurança.

Capítulo 5

DESENVOLVIMENTO

O sistema proposto foi implementado na linguagem de programação Java, escolhida por sua versatilidade: tendo o sistema sido implementado em Java, ele pode ser executado em qualquer sistema operacional que apresente suporte à Java Virtual Machine. Ademais, Java é a linguagem com a qual os desenvolvedores mais têm intimidade, possibilitando assim um código mais limpo e um menor tempo de desenvolvimento dedicado ao aprendizado de uma nova linguagem.

A arquitetura do sistema aqui descrita diz respeito apenas à implementação da camada de negócios e persistência dos módulos, que são o cerne do trabalho executado. O sistema foi projetado de forma modular, fazendo com que a camada de visualização funcione de forma independente do resto do sistema. Dessa forma, a interface gráfica torna-se facilmente cambiável de modo a se adaptar da melhor forma possível às características de uso de um processo eleitoral específico.

A arquitetura da camada de negócios foi feita baseada fortemente na API de Networking Apache MINA , por ser uma API extremamente completa e com suporte a uma grande variedade de características da camada de comunicação. Ainda que cada módulo tenha sido implementado como um programa independente, a implementação de todos os módulos segue a mesma arquitetura, que foi fortemente baseada naquela sugerida pelo Apache MINA.

Inicialmente é descrita a implementação e a arquitetura comuns a todos os módulos descritos no protocolo VoteRemote, assim como as dependências do sistema, ou seja, as APIs utilizadas na implementação do sistema. As peculiaridades de cada módulo serão discutidas nas seções seguintes.

5.1 Dependências de APIs externas

Nesta seção são descritas as dependências das APIs externas que foram utilizadas na implementação do algoritmo VoteRemote, sua características e vantagens.

5.1.1 Apache MINA

O Apache Mina ('Multi-purpose Infrastructure for Network Applications') é um arcabouço que ajuda os usuários a desenvolverem facilmente aplicações de rede de alto desempenho

e alta escalabilidade (APACHE, 2010a). Ele provê uma API assíncrona, dirigida por eventos, sendo possível o uso com várias camadas de transporte como TCP/IP e UDP/IP através do Java NIO ('non-blocking I/O') (APACHE, 2010a).

Uma das vantagens de se utilizar o Apache MINA é o ganho em manutenibilidade e reusabilidade do código, devido à separação dos códigos de rede (como *sockets*), de protocolo e da lógica de negócio da aplicação. Alguns recursos que o Apache Mina oferece são:

- API unificada para diversos tipos de transporte
- Extensibilidade através de filtros
- API de baixo e alto níveis
- Modelo de *thread* altamente customizável
- Testabilidade unitária usando *mocks*
- Gerência JMX
- Suporte E/S baseado em *stream*
- Integração com containers conhecidos, como Spring

Toda a comunicação entre os módulos deste projeto foram feitos através do Apache MINA.

5.1.2 LOG4J

O log4j é um utilitário fornecido pela Apache Software Foundation para aplicações escritas em Java, que facilita o uso de registro (*log*) de dados (APACHE, 2010c). A utilização de *logs* é importante para conhecer o comportamento passado de um software, sendo assim possível fazer diagnósticos de diversas utilidades, como por exemplo para a solução de problemas. Os *logs* também podem ser utilizados para *debugging*.

Incluir códigos para a execução de *loggins* em uma aplicação pode poluir o código fonte e diminuir a legibilidade. Na linguagem Java, na qual não existe um pré-processador, as declarações no código referentes a *logs* aumentam o tamanho do código e reduzem a velocidade consideravelmente. Com o log4j é possível habilitar *logs* em tempo de execução sem acarretar um custo alto de desempenho, sendo que o comportamento do *log*

pode ser controlado através da edição de um arquivo de configuração, sem influenciar os executáveis da aplicação.

Uma das características diferenciais do log4j é o uso de herança. Assim sendo é possível criar uma hierarquia que pode controlar os registros de *log* de forma granular e fácil. Isto ajuda a reduzir o volume de informações registradas e o custo do registro.

5.1.3 Commons Configuration

O Commons Configuration provê uma interface de configuração genérica, que permite a uma aplicação Java ler dados de configuração de uma variedade de fontes (APACHE, 2008). Utilizando o Commons Configuration é possível ter acesso a parâmetros de configuração simples e multivalorados através de simples chamadas de apenas um método.

Os parâmetros de configuração podem estar nas seguintes fontes:

- Arquivos de propriedades
- Documentos XML
- Arquivos INI do Windows
- Arquivos de lista de propriedade (Property list files - plist)
- JNDI
- Fonte de dados JDBC
- Propriedades do sistema
- Parâmetros de applet
- Parâmetros de servlet

Diferentes fontes de parâmetros de configuração podem ser combinadas. O uso de fontes adicionais pode ser feito usando objetos de configuração customizados, através da herança de algumas classes providas pelo Commons Configuration.

5.1.4 OpenSSL

Como previsto no protocolo VoteRemote, a comunicação segura entre os módulos é garantida por uma estrutura do tipo PKI (Public Key Infrastructure). Esta estrutura baseia-se na existência de certificados emitidos por entidades certificadoras contendo as chaves públicas a serem utilizadas para o estabelecimento da comunicação.

A fim de gerar esses certificados e suas respectivas chaves privadas, utliza-se neste projeto o OpenSSL (APACHE, 2010b). Essa ferramenta é uma implementação de código aberto dos protocolos SSL e TLS, que está disponível para diversas linguagens de programação (através de *wrappers*) e sistemas operacionais.

O projeto OpenSSL vem de um esforço colaborativo para o desenvolvimento de um robusto conjunto de ferramentas que além de implementar os protocolos SSL e TLS também provê uma biblioteca de criptografia de propósito geral. O projeto é gerenciado por uma comunidade de voluntários que planejam e desenvolvem o OpenSSL e sua documentação relacionada. O seu uso é livre tanto para fins comerciais como não comerciais sujeito a apenas algumas condições de licença.

O OpenSSL permite a geração de certificados auto-assinados e chaves privadas nos mais variados formatos. Para isso, basta executar os comandos descritos a seguir.

O primeiro comando gera um arquivo de saída de extensão .PEM, contendo uma chave privada RSA de 2048 bits.

openssl genrsa -out privkey.pem 2048

Já o comando indicado abaixo, gera um certificado completo com extensão .PEM, contendo uma série de informações específicas, além da chave pública a ser usada durante o estabelecimento da conexão.

openssl req -new -x509 -key privkey.pem -out cacert.pem -days 1095

Como descrito no protocolo, o arquivo contendo o certificado é publicado para que todos possam acessá-lo, enquanto o arquivo contendo a chave privada deve estar acessível apenas ao seu módulo proprietário.

5.1.5 Bouncy Castle

BouncyCastle é um conjunto de APIs para as linguagens Java e C-Sharp, usadas em criptografia, que foram desenvolvidas em conjunto por uma comunidade (BouncyCastle Legion) (THE LEGION OF THE BOUNCY CASTLE, 2010). Essas bibliotecas, além de serem livres para uso ilimitado, possuem a vantagem de ter o código aberto. Portanto, a correção de eventuais erros pode ser executada por toda a comunidade de usuários dos algoritmos criptográficos.

A biblioteca do BouncyCastle possui implementação de diversos algoritmos de criptografia simetria, criptografia assimétrica, assinaturas e funções de hash, com uma interface simples de ser usada. Alguns exemplos usados neste projeto são:

- RSA (criptografia assimétrica)
- AES (criptografia simétrica)
- Família de funções SHA (funções de hash)

5.2 Arquitetura do sistema

5.2.1 Arquitetura de um sistema desenvolvido no Apache MINA

A arquitetura proposta pela API Apache MINA envolve, para cada conexão a ser aberta, um *Handler*, que representa a camada de aplicação e trata as mensagens recebidas e enviadas, e de um conjunto de *Filters*, que podem ser utilizados para manipular a mensagem antes que ela chegue ao e logo que é enviada pelo *Handler* (APACHE, 2010a).

Cada conexão é encapsulada por um objeto do tipo *Session*.

A arquitetura de uma aplicação genérica pode ser vista na figura 5.1.

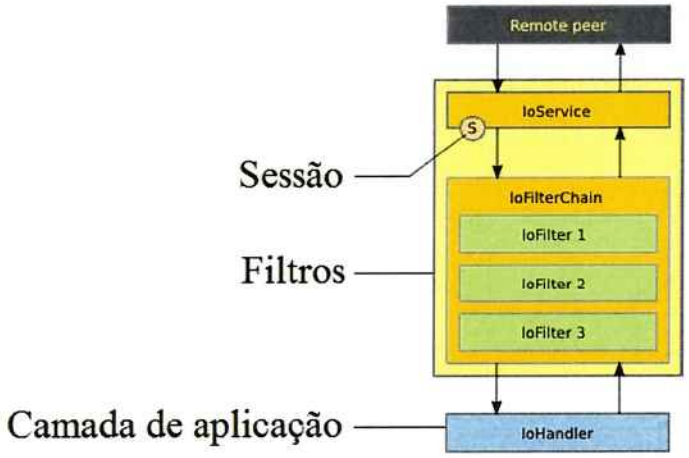

Figura 5.1: *Arquitetura de um servidor.*

Na figura 5.1, é possível identificar os seguintes elementos:

- *IoSession* - Representa um *Acceptor* caso o comportamento do programa desenvolvido for de servidor e um *Connector* caso o comportamento for de cliente.
- *Session* - Representa uma instância de uma conexão. Um *Acceptor*pode tratar diversas conexões simultaneamente, enquanto *Connectors* tratam apenas uma conexão por vez.

- *IoFilters* - Representa uma série de filtros responsáveis por realizar alterações nas mensagens enviadas e recebidas antes delas serem tratadas pela camada de aplicação.
- *IoHandler* - Representa a camada de aplicação.

5.2.2 Arquitetura dos módulos do sistema proposto

Todos os módulos foram desenvolvidos como programas completamente independentes, de forma que é possível executá-los tanto na mesma máquina quanto em máquinas separadas.

Comuns à implementação de todos os módulos são os filtros utilizados e todas as classes utilitárias, como as classes que encapsulam as informações dos eleitores, candidatos e votos e as classes da camada de segurança da informação. Os fatores diferenciais entre os módulos são a camada de aplicação, ou seja, o *Handler* e suas classes de apoio, e a camada de visualização, ou seja, as classes pertencentes à interface gráfica.

5.2.3 Filtros utilizados

Filtro para a criação de Logs

Foi utilizado um filtro para a geração de logs. Através deste filtro, é possivel configurar facilmente o destino onde os logs serão armazenados ou exibidos e o nível de detalhe das mensagens que serão apresentadas. É possível configurar quatro diferentes níveis para a exibição de mensagens:

- *Error* - Apenas mensagens marcadas como mensagens de erro são exibidas.
- *Warn* - Apenas mensagens marcadas como mensagens de erro ou *warn* são exibidas.
- *Info* - Apenas mensagens marcadas como mensagens de erro, *warn* ou *info* são exibidas.
- *Debug* - Todas as mensagens são exibidas.

Este filtro tem como dependência a API log4j, citada na seção Dependências. Este filtro foi implementado na classe *LoggingFilter*.

Filtro para a comunicação segura através de SSL

Este filtro, chamado *SSLFilter* foi utilizado para garantir a comunicação através do protocolo SSL entre os módulos. Ele tem como responsabilidade garantir que todo o protocolo SSL seja seguido de forma a tornar a utilização deste protocolo imperceptível para a camada de aplicação. As mensagens recebidas são decriptadas por ele antes de serem repassadas ao filtro seguinte e são encriptadas por ele antes de serem enviadas aos outros módulos.

Filtro para o envio de objetos complexos

Este filtro, chamado *ProtocolCodecFilter*, existe para tornar transparente o envio de objetos complexos pelo sistema. Inicialmente, só é possivel enviar mensagens encapsuladas por *ByteBuffers*. Com a utilizaçao deste filtro, a camada de aplicação pode enviar qualquer objeto serializável, isto é, que implementa a interface *Serializable* do Java, e o filtro transformará este objeto em um *ByteBuffer* para que ele possa ser enviado.

5.2.4 Classes utilitárias

O sistema proposto conta com diversas classes de apoio, que são armazenadas no pacote *voteremote.util*e seus subpacotes.

Mensagens

Foram criadas classes para encapsular as mensagens trocadas pelos módulos do sistema proposto. Estas classes são armazenadas no pacote *voteremote.util.messages* e podem ser vistas na figura 5.2.

As mensagens são:

- *listaEleitoresRequest* - usada pelo Validador para pedir a Lista de Eleitores ao Quadro Negro.
- *listaEleitores* - lista de eleitores enviada pelo Quadro Negro ao validador.
- *isVotingPhaseRequest* - mensagem enviada ao Quadro Negro para saber se a eleição se encontra em fase de votação.
- *isVotingPhaseNOTOK* - resposta negativa à pergunta acima.
- *isVotingPhaseOK* - resposta afirmativa à pergunta acima.

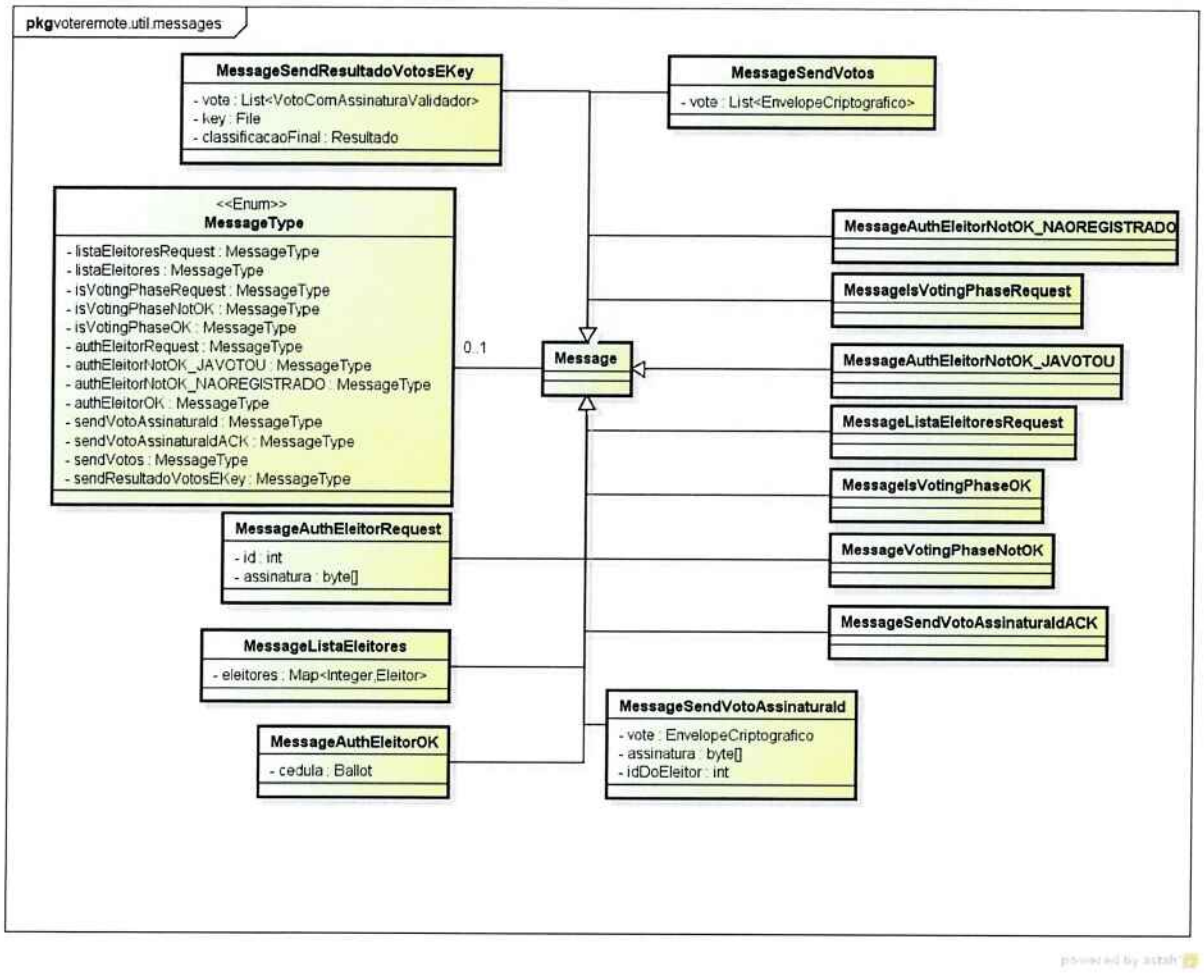

Figura 5.2: *Mensagens trocadas pelos módulos.*

- *authEleitorRequest* - pedido de autenticação de eleitor, enviada pelo modulo Votador ao Quadro Negro.
- *authEleitorNOTOK_JAVOTOU* - resposta ao pedido de autenticação informando que o eleitor não tem direito de votar pois já votou nesta eleicão.
- *authEleitorNOTOK_NAOREGISTRADO* - resposta ao pedido de autenticacão dizendo que o eleitor não tem direito de votar pois ele não está registrado.
- *authEleitorOK* - mensagem indicativa de que o eleitor foi autenticado, envia uma cedula de votação.
- *sendVotoAssinaturaId* - envio de voto com validação de eleitor.
- *sendVotoAssinaturaIdACK* - resposta à mensagem acima dizendo que a mensagem foi recebida com sucesso.

- *sendVotos* - usada pelo Quadro Negro para enviar os votos à Rede de Mistura e ao Escrutinador.
- *sendResultadoVotosEKey* - usada pelo escrutinador para enviar o resultado ao Quadro Negro.

Todas as mensagens são enviadas contendo um número de série. Este número é mantido pelos módulos ao responder as mensagens, de forma que o emissor possa saber, ao receber uma resposta, a que mensagem enviada esta resposta corresponde.

Eleitores, candidatos e votos

O restante das classes do pacote *voteremote.util* podem ser vistas na figura 5.3.

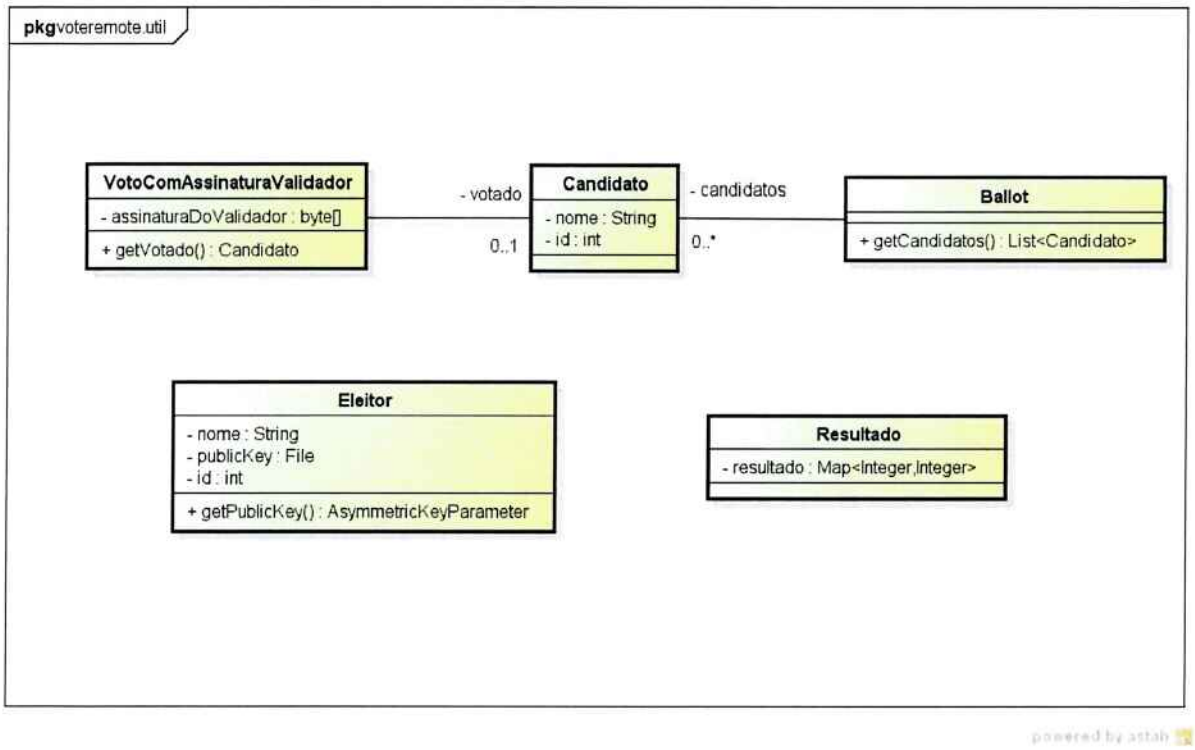

Figura 5.3: *Classes no pacote voteremote.util* .

As classes são:

- *Candidato* - Encapsula um candidato, que é representado por um identificador numerico único e pelo seu nome.
- *VotoComAssinaturaValidador* - Encapsula um candidato votado por um eleitor e a assinatura dada pelo Validador a este voto, tornando-o válido.

- *Ballot* - Representa uma lista de candidatos que estão concorrendo na eleição. É enviado pelo Quadro Negro ao Votador após autenticá-lo.
- *Eleitor* - Encapsula um eleitor, contendo seu nome, um identificador numérico único e sua chave pública.
- *Resultado* - Encapsula o resultado da eleição no formato de um mapa que relaciona o identificador numérico único de cada candidato ao número de votos recebidos.

5.2.5 Segurança da informação

Como descrito em documentos anteriores, a implementação em Java dos algoritmos escolhidos tem como base a biblioteca BouncyCastle.

A seguir, discutem-se as escolhas dos algoritmos e de seus parâmetros, além de apresentarem-se as classes e métodos criados.

Parâmetros escolhidos

A criptografia assimétrica representa o gargalo criptográfico do sistema. Isso acontece devido ao fato de seu desempenho ser muita mais lento que a de uma cifra simétrica para o mesmo padrão de segurança. Dessa forma, sua escolha define o padrão que deve ser seguido no resto do projeto.

Sistemas reais atuais implementam usualmente RSA-1024, que apresenta uma segurança de 80 bits, ou RSA-2048, com uma segurança de aproximadamente 112 bits, ou ainda, em menor número de vezes, RSA com tamanhos de chaves maiores que 2048 bits. Por outro lado, ao duplicar-se o tamanho de uma chave RSA, o desempenho do algoritmo cai aproximadamente 5 vezes.

Algoritmos RSA com chaves abaixo de 1024 bits não possuem uso encorajado, por representarem atualmente uma fraca segurança. Levando-se estas informações em conta, optou-se por utilizar o RSA-2048 com *padding* PKCS1 v1.5 no sistema VoteRemote, que apresenta um bom *trade-off* entre desempenho e segurança para este projeto.

Assim, ao utilizar um algoritmo de 112 bits de segurança, definimos um padrão para escolha dos demais algoritmos. Isso acontece porque qualquer escolha que envolva uma ordem de segurança muito maior que esta representará apenas uma queda de desempenho do sistema, pois o nível de segurança geral continuará em 112 bits.

Para cifra simétrica, possíveis opções seriam o 3DES , com segurança de 112 bits, ou o AES-128, com segurança de 128 bits. Apesar do 3DES apresentar uma conformidade maior com o padrão de 112 bits escolhido, seu desempenho é inferior ao desempenho

do AES-128. Dessa forma, escolheu-se o segundo para ser implementado no sistema proposto. Como mode de operação, escolheu-se o modo CBC e padding PKCS7, por apresentarem boa segurança em conjunto com o algoritmo escolhido.

Finalmente, para a função de *hash* pode-se escolher tanto o SHA-224, com segurança de 112 bits, quanto o SHA-256, com segurança de 128 bits. Novamente, apesar do primeiro corresponder exatamente ao padrão de segurança geral do sistema, o segundo apresenta uma maior conformidade com padrões usados em sistemas reais. Assim, definiu-se o SHA-256 como função de *hash* para esse sistema.

RSAKeys

Essa classe é responsável por receber os arquivos .PEM referentes ao certificado e à chave privada a serem utilizados pelo algoritmo de criptografia assimétrica e encapsulá-los em um formato que possa ser utilizado pela classe CryptoUtil.

A classe estática é composta por dois métodos descritos a seguir.

- getPublicKey(File pubKeyFile): recebe um arquivo contendo um certificado público no formato X509 e retorna o objeto da biblioteca Bouncy Castle que encapsula uma chave pública.
- getPrivateKey(File privKeyFile, char[] password): recebe um arquivo contendo uma chave privada encriptada e a senha necessária para sua decriptação. Retorna então um objeto da biblioteca Bouncy Castle que encapsula uma chave privada.

Envelope Criptografico

O algoritmo RSA de encriptação assimétrica aceita blocos de informação com tamanho máximo de um byte a menos do que o tamanho da chave utilizada. Dessa forma, para o algoritmo RSA-2048 utilizado nesse projeto, o tamanho máximo de bloco seria de 255 bytes. Porém, ao utilizar-se a codificação PKCS 1 v1.5 deve-se descontar também o tamanho do cabeçalho. Dessa forma, por ser o cabeçalho utilizado do tamanho de 10 bytes, somente blocos de até 245 bytes são aceitos pelo algoritmo. O problema encontra-se então em como encriptar dados maiores do que 245 bytes.

A solução utilizada para cifras simétricas consiste na utilização de modos de operação. Porém, para cifras assimétricas, essa solução representa apenas uma curiosidade teórica, e não é utilizada em sistemas reais. A solução mais adequada é a de se utilizar uma combinação de cifras simétricas e assimétricas, mantendo-se assim as vantagens de ambos os métodos de encriptação.

O procedimento consiste nas seguintes etapas. Primeiramente, gera-se uma chave simétrica aleatória, encriptando-a com o algoritmo de cifra assimétrica. Os dados são então encriptados de forma simétrica com essa mesma chave. Em seguida, dados e chave encriptados são encapsulados em um único objeto e enviados para o destinatário. A este, resta apenas a função de decriptar a chave simétrica usando o algoritmo assimétrico e, assim, obter a chave necessária para decriptar o conjunto completo de dados.

Essa combinação aproveita a estrutura de chaves pública e privada da cifra assimétrica e o fato da cifra simétrica aliada a um modo de operação aceitar qualquer tamanho de vetor de entrada e possuir um melhor desempenho.

Portanto, a classe EnvelopeCriptográfico é responsável por encapsular um conjunto de dados de tamanho arbitrário encriptados pela cifra simétrica e uma chave AES de 128 bits junto com seu vetor de inicialiazação de também 128bits, encriptados pelo algoritmo RSA. Através dos métodos getAesKeyEncrypted() e getDataEncrypted(), cada uma dessas informações pode ser recuperada.

CryptoUtil

Classe principal da camada de criptografia do sistema, a classe estática CryptoUtil implementa, utilizando a biblioteca BouncyCastle, todos os algoritmos descritos nos itens anteriores deste documento.

Todos os métodos são estáticos, assim a chamada de métodos de encriptação/decriptação e assinaturas poderá ser facilmente utilizada a fim de atender ao protocolo previamente definido.

- generateBlindingFactor(AsymmetricKeyParameter publicKey): gera o fator R usado para mascarar uma mensagem baseado em um fator aleatório e na chave pública da entidade responsável por assinar cegamente.

- sign(byte[] data, AsymmetricKeyParameter key): assina probabilisticamente uma mensagem usando a chave privada do módulo e o algoritmo RSA. Essa mensagem deve ser sempre o resultado de uma função de hash, com exceção apenas do caso de ser uma assinatura cega. Neste caso, a entrada será o resultado de uma função de hash mascarado por um fator R.

- verify(byte[] data, byte[] signature, AsymmetricKeyParameter key): verifica a assinatura recebida utilizando a chave pública do módulo signatário e o algoritmo RSA.

- blindMessage(byte[] data, RSABlindingParameters blindingFactor): mascara a mensagem usando um fator aleatório *R*. Essa função é usada sempre para mascarar o resultado de uma função de hash.

- unblindMessage(byte[] data, RSABlindingParameters blindingFactor):recebe uma assinatura e retira desta o fator aleatório *R* usado para mascarar a mensagem. Devido a propriedades específicas do RSA, ao desmascarar-se por *R* uma assinatura proveniente de uma mensagem mascarada por *R*, o resultado será equivalente a uma assinatura da mensagem original (sem máscara).

- getHash(byte[] data): extrai o resumo criptográfico de um conjunto de dados de entrada de tamanho arbitrário usando o algoritmo SHA-256.

- encrypt(byte[] data, AsymmetricKeyParameter key): encripta uma mensagem de tamanho arbitrário usando em conjunto criptografia simétrica e assimétrica. Como o RSA não possui modo de operação, o seguinte método é executado. Inicialmente, gera-se uma chave AES de 128 bits aleatória e encripta-se essa informação usando a chave pública do módulo de destino. Em seguida, encripta-se a mensagem de entrada usando o algoritmo simétrico. Por fim, encapsula-se tanto a chave AES encriptada pelo RSA quanto a mensagem encriptada pelo AES em um objeto do tipo EnvelopeCriptografico.

- decrypt(EnvelopeCriptografico dataWithAESKey, AsymmetricKeyParameter key): decripta uma mensagem de tamanho arbitrário usando em conjunto criptografia simetrica e assimétrica. Como o RSA não possui modo de operação, o seguinte método é executado. Decripta-se a chave AES recebida usando-se a chave privada do módulo. Em seguida, usando-se a chave encontrada, decripta-se a informação de entrada pelo algoritmo AES.

- encryptAES(byte[] data, ParametersWithIV aesKey): método privado utilizado como parte do método público encrypt, este método encripta uma mensagem de tamanho arbitrário usando criptografia simétrica (AES-128, com modo de operação CBC e padding PKCS7).

- decryptAES(byte[] data, ParametersWithIV aesKey): método privado utilizado como parte do método público decrypt, este método decripta uma mensagem de tamanho arbitrário usando criptografia simétrica (AES-128, com modo de operação CBC e padding PKCS7).

- generateAESKey(): método privado utilizado como parte do método público encrypt, este método gera uma chave AES aleatória, junto com seu vetor de inicialização.
- convertAESKeyToBytes(ParametersWithIV aesKey): método privado utilizado como parte do método público encrypt, este método converte o objeto da chave AES em um vetor de bytes, no qual os primeiros 128 bits representam a chave AES e os outros 128 bits o vetor de inicialização.
- getAESKeyFromBytes(byte[] input): método privado utilizado como parte do método público decrypt, este método converte um vetor de bytes, no qual os primeiros 128 bits representam a chave AES e os outros 128 bits o vetor de inicialização, em um objeto do tipo ParametersWithIV.

5.2.6 Configuração parametrizada

Todos os módulos possuem um arquivo de configuração, no qual podem ser configurados parâmetros como a porta a ser utilizada para comunicação com outros módulos e o local de armazenamento das chaves públicas e privadas utilizadas. Este arquivo chama-se *voteremote.properties*.

5.3 Arquitetura do Quadro Negro

O módulo Quadro Negro foi escolhido como o pivot de toda a eleição, pois ele é o único módulo que se comunica com todos os outros. É através de um paramêtro de configuração no Quadro Negro que é definido se a eleição se encontra atualmente na Fase de Contagem ou na Fase de Votação. É também no módulo Quadro Negro que são registrados todos os eleitores (com suas respectivas chaves públicas) e candidatos participantes da eleição, assim como quais foram os eleitores que já votaram e seus votos, ainda que de forma encriptada.

A arquitetura do Quadro Negro pode ser vista na figura 5.4.

Como pode ser visto na figura 5.4, o Quadro Negro possui três *Handlers*: um para a Fase de Votação e dois para a Fase de Contagem.

Ao iniciar sua execução, o Quadro Negro verifica em um arquivo de configuração se a votação se encontra em Fase de Contagem ou em Fase de Votação e executa as

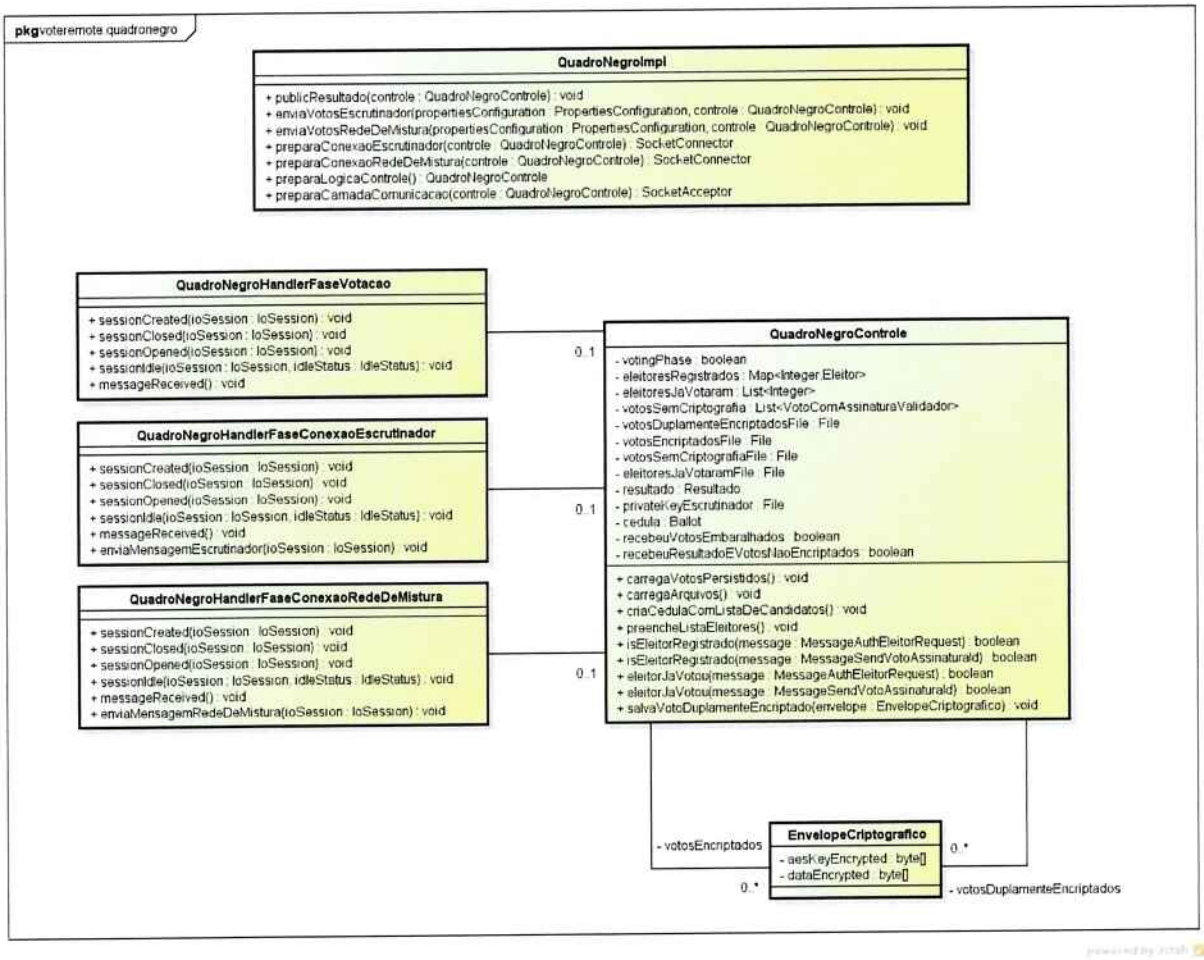

Figura 5.4: *Arquitetura do Quadro Negro.*

ações devidas para cada fase. Este arquivo pode ser alterado em tempo de execução para passar da Fase de Votação para a Fase de Contagem, mudança que será percebida pelo Quadro Negro em tempo real.

Durante a Fase de Votação, o Quadro Negro atua como servidor e recebe solicitações do Validador e do Votador. Ele executa funções de autenticação de eleitor, de envio da lista de eleitores cadastrados e de armazenamento dos votos duplamente encriptados enviados pelo votador, conforme pode ser visto na figura 3.2.

Durante a Fase de Contagem, o Quadro Negro envia os votos duplamente encriptados para a Rede de Mistura e, ao receber dela os votos encriptados embaralhados, envia estes para o Escrutinador. Ao receber o resultado e a chave privada do Escrutinador, ele publica estes para o público geral.

5.3.1 Modo de recuperação

Todos os votos duplamente encriptados recebidos pelo Quadro Negro do votador durante a Fase de Votação assim como todos o registro de todos os eleitores que já votaram são armazenados em arquivos em tempo real. Caso o sistema tenha seu funcionamento interrompido, é possivel retomar a eleição no mesmo estado em que ela se encontrava no momento da interrupção. Nenhuma premissa de segurança é violada pelo sistema em decorrência deste comportamento, dado que tanto os votos quanto os eleitores são armazenados de forma encriptada. Adicionalmente, o arquivo com os votos duplamente encriptados é o arquivo que será publicado pelo Quadro Negro conforme descrito no protocolo.

5.3.2 Publicação dos votos duplamente encriptados, votos encriptados, resultado e chave privada do Escrutinador

A publicação dos votos duplamente encriptados, votos encriptados, resultado e da chave privada do Escrutinador, conforme descrito no protocolo, dá-se através da publicação de arquivos contendo estas informações. O local de armazenamento destes arquivos pode ser definido no arquivo de configuração *voteremote.properties*.

5.4 Arquitetura do Validador

A arquitetura do Validador pode ser vista na figura 5.5.

O módulo Validador tem um funcionamento simples em comparação com o Quadro Negro. O Validador atua como cliente ao ser inicializado, buscando do Quadro Negro as informações dos eleitores registrados e suas chaves públicas. Após ter recebido estas informações, ele atua como servidor, recebendo votos do módulo Votador, verificando se os eleitores que enviaram estes votos estão registrados e assinando os votos recebidos caso os eleitores estiverem registrados.

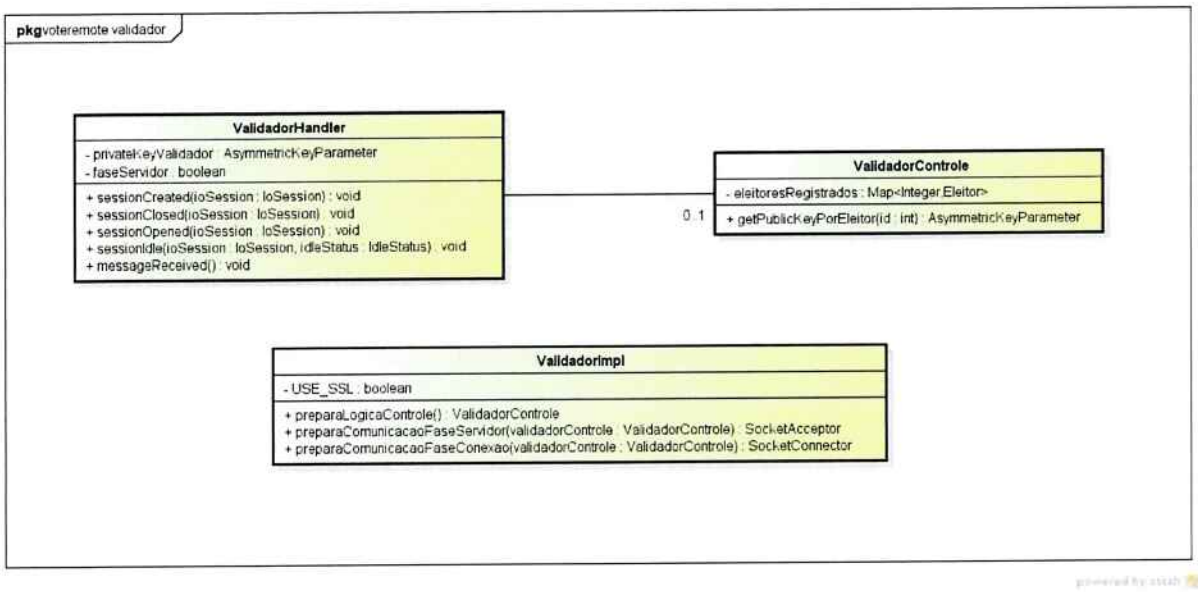

Figura 5.5: *Arquitetura do Validador.*

5.5 Arquitetura da Rede de Mistura

A arquitetura da Rede de Mistura pode ser vista na figura 5.6.

A Rede de Mistura apresenta comportamento de servidor. Ela aguarda por mensagens do tipo *sendVotos* e, ao recebê-las, retira a encriptação com sua chave privada, as embaralha e as envia de volta para o remetente da mensagem, conforme pode ser visto na figura 5.7.

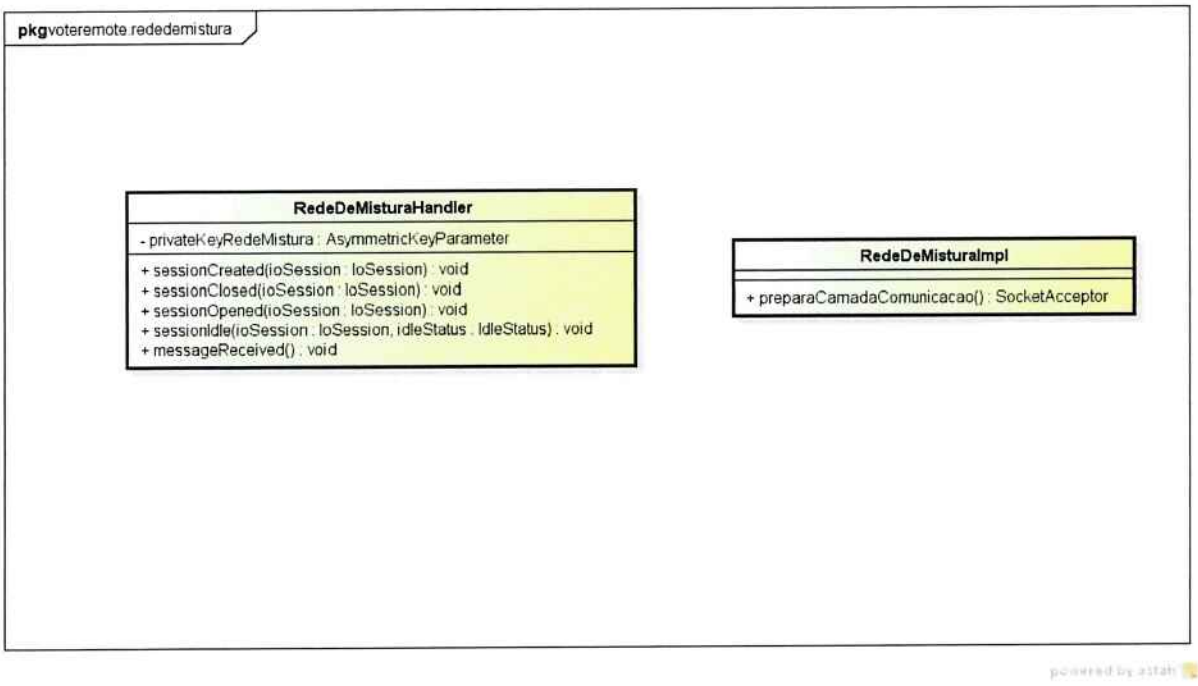

Figura 5.6: *Arquitetura da Rede de Mistura.*

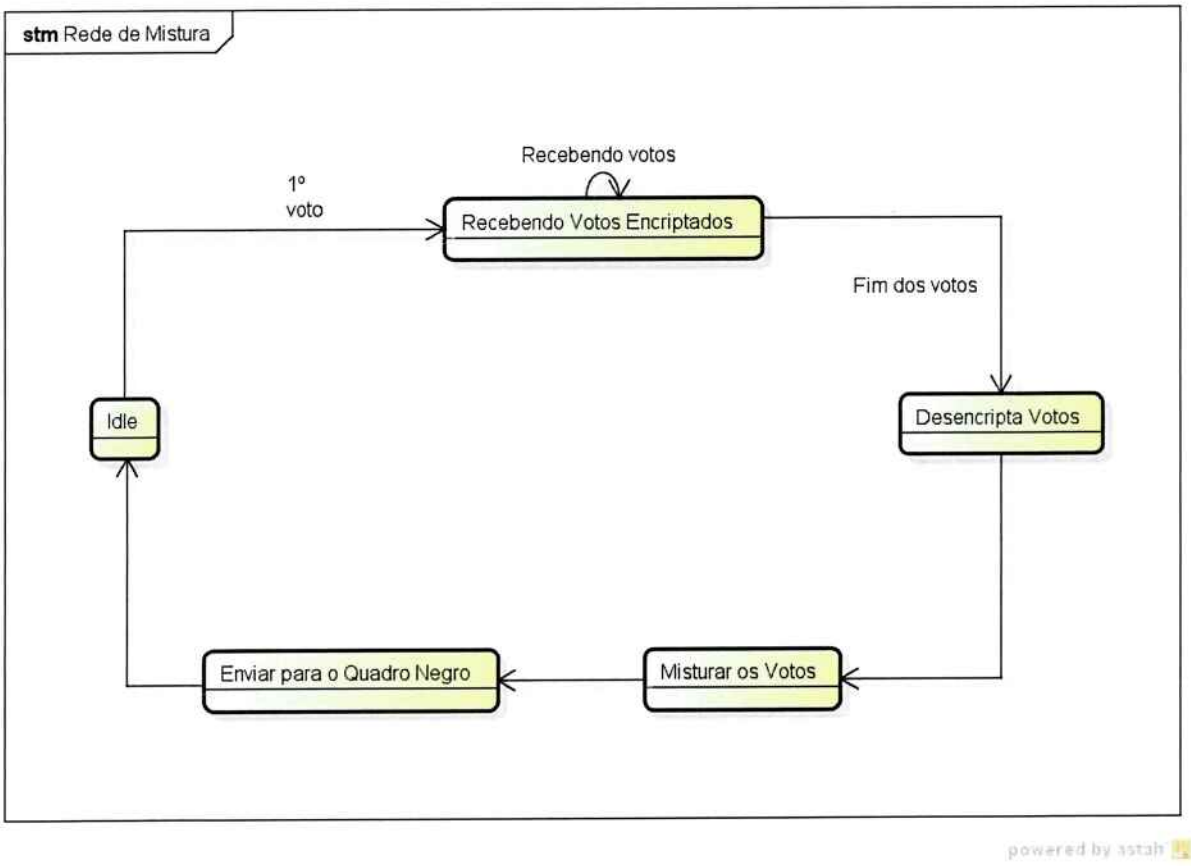

Figura 5.7: *Comportamento da Rede de Mistura.*

5.6 Arquitetura do Escrutinador

A arquitetura do Escrutinador pode ser vista na figura 5.8.

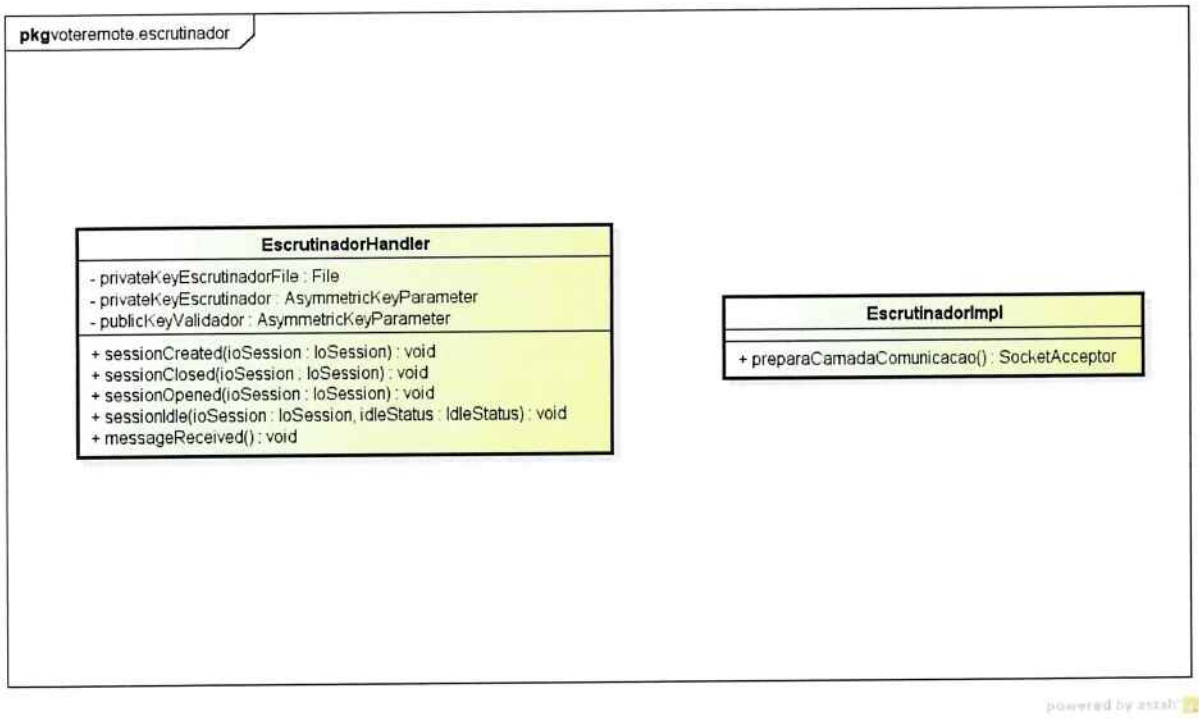

Figura 5.8: *Arquitetura do Escrutinador.*

O escrutinador apresenta comportamento de servidor. Ele aguarda por mensagens do tipo *sendVotos* e, ao recebê-las, retira a encriptação com sua chave privada, verifica a validade de cada voto a partir da assinatura do Validador contida na mensagem e realiza a contagem dos votos. Ao obter o resultado final, o Escrutinador envia o resultado da votação, assim como sua chave privada, para o Quadro Negro para publicação. O comportamento do Escrutinador pode ser visto na figura 5.9.

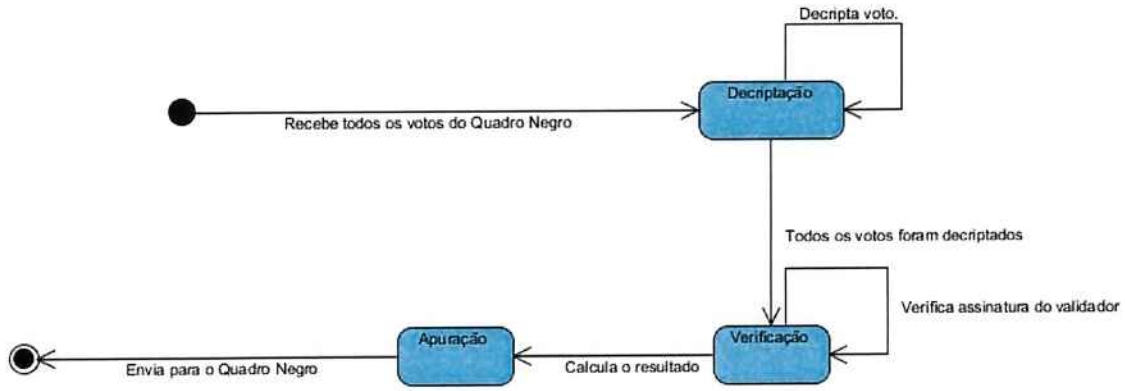

Figura 5.9: *Comportamento do Escrutinador.*

5.7 Arquitetura do Votador

A arquitetura do Escrutinador pode ser vista na figura 5.10.

O módulo Votador apresenta comportamento de cliente. Ele possui duas conexões: uma com o módulo Quadro Negro e outra com o módulo Validador.

5.7.1 Interface com os eleitores

Este módulo é o que fará o papel de "urna", e o módulo com o qual os eleitores terão contato. Assim sendo, é de fundamental importância que seja desenvolvida uma interface amigável para facilitar o processo de votação dos eleitores.

Para manter a implementação aqui proposta o mais versátil possível, foi desenvolvida uma interface, chamada *Votador* e que pode ser visto na figura 5.10, para realizar a comunicação com a interface gráfica. Desta forma, pode-se desenvolver a interface gráfica que representará a urna eleitoral independente do funcionamento do módulo Votador. É possivel, por exemplo, criar uma interface web que funcionaria como uma página na Internet que os eleitores poderiam acessar para realizar seu voto. Outra alternativa seria utilizar uma máquina para cada urna, cada uma rodando uma interface gráfica programada em Java, para realizar a eleição.

As figuras 5.11, 5.12, 5.13, 5.14 e 5.15 são exemplos da interface gráfica do sistema implementado nesse trabalho.

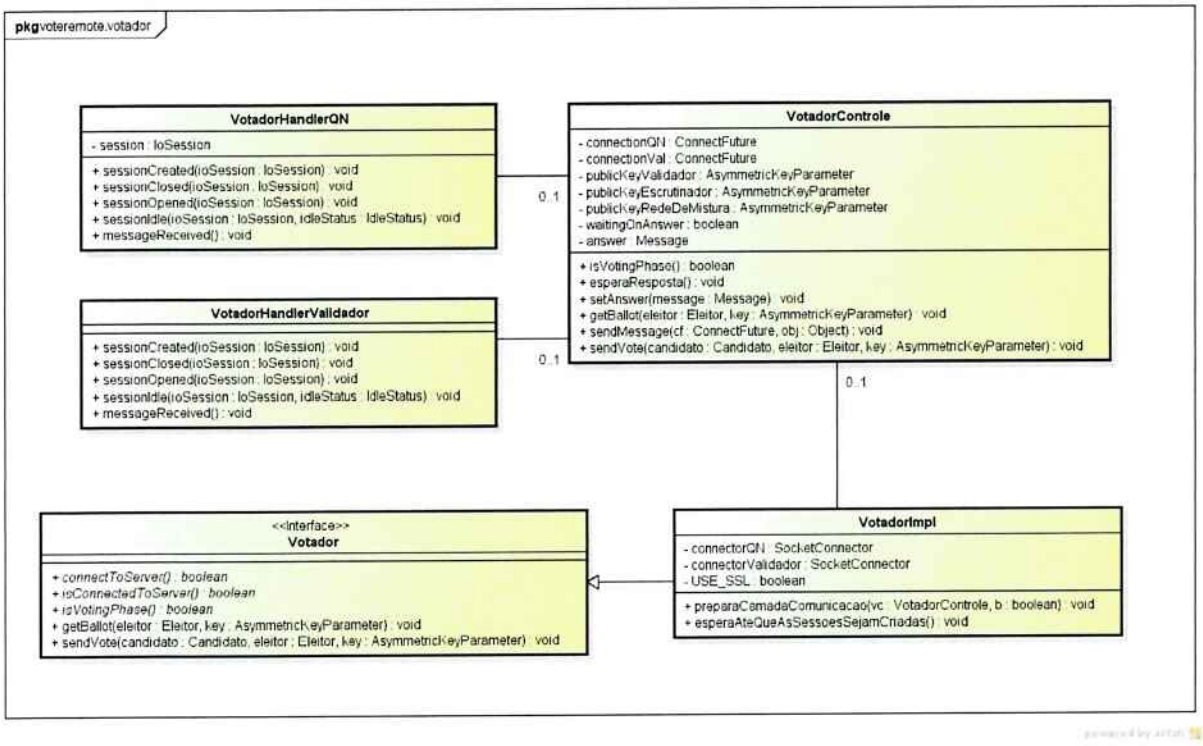

Figura 5.10: *Arquitetura do Votador.*

5.7.2 Armazenamento das chaves privadas e senhas dos eleitores

No protocolo VoteRemote é previsto que cada eleitor possua uma chave privada e uma senha a ela associada. Estas chaves seriam idealmente armazenadas em hardware criptográfico, como smartcards, que seriam os títulos dos eleitores. A senha da chave privada poderia ser tanto uma senha escolhida pelo eleitor, o que exigiria a presença de um fiscal próximo a cada urna para garantir que nenhum terceiro vote em nome de um eleitor tendo posse de sua chave privada e senha, ou uma cadeia de caracteres gerada por um leitor biométrico, fazendo com que apenas os próprios eleitores possam votar utilizando suas chaves privadas.

Para fins demonstrativos, foi desenvolvido neste projeto um sistema que busca a chave privada armazenada em um *pen drive*, no qual cada eleitor tem sua senha memorizada. Entretanto, esta característica pode ser facilmente modificada.

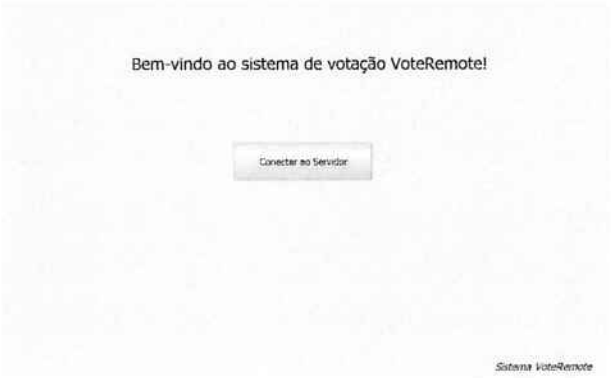

Figura 5.11: *Tela de conexão com o servidor do sistema implementado.*

Figura 5.12: *Tela de identificação do sistema implementado. O eleitor deve, nesse momento, inserir seu título de eleitor eletrônico, que pode estar em um* pen drive *ou* hardware *criptográfico.*

Figura 5.13: *Tela com o nome do eleitor identificado, e na qual o eleitor deve inserir sua senha secreta.*

Figura 5.14: *Tela com a cédula de votação do sistema implementado.*

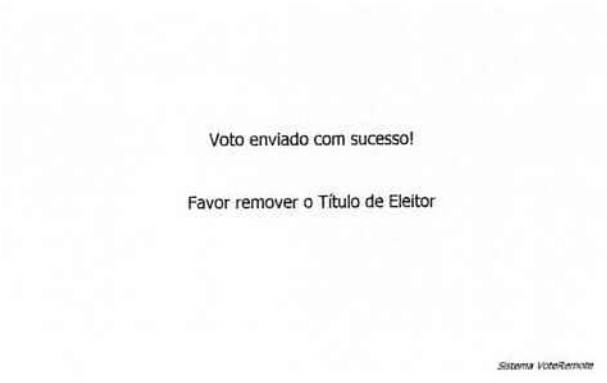

Figura 5.15: *Tela de confirmação do voto registrado.*

5.8 Sinopse

Neste capítulo descreveram-se a arquitetura e os detalhes de implementação de um protótipo do sistema proposto. Assim, inicialmente definiram-se as tecnologias a serem utilizadas pelo protótipo, como a linguagem Java para programação e o uso de *pen drives* para armazenamento da chave privada do eleitor. Definiram-se também os algoritmos e parâmetros a serem utilizados pela camada de criptografia do sistema. Estando esses detalhes de implementação acertados, estruturou-se a arquitetura do sistema, que é basicamente dividida em três camadas: comunicação, criptografia e controle. A camada de comunicação é baseada na arquitetura sugerida pelo Apache MINA, contando com a existência de *Handlers*, *Filters* e *Sessions*. A comunicação ocorre pela troca de variados tipos de Mensagens. A camada de criptografia por sua vez, fornece métodos de encriptação, decriptação e assinatura baseados na biblioteca BouncyCastle. Já a camada de controle é responsável por implementar o funcionamento do protocolo em si. Uma interface gráfica exemplo também foi desenvolvida para o software protótipo, optando-se por uma interface do tipo Aplicação Cliente, em detrimento de uma aplicação Web.

Capítulo 6

TESTES

Neste capítulo, os testes aplicados ao protótipo desenvolvido são descritos e seus resultados apresentados, a fim de provar seu funcionamento.

6.1 Votador

- Teste 1: Eleitor entra com sua chave privada ou número de identificação e senha corretos.

  Resultado: Votador aceita o eleitor e tenta se autenticar com o Quadro Negro.

- Teste 2: Eleitor entra com número de identificação e senha não correspondentes.

  Resultado: Votador recusa eleitor e não fornece uma cédula de votação.

- Teste 3: Eleitor escolhe um candidato dentre os da cédula de votação.

  Resultado: Votador gera um voto com estrutura válida, assina este voto e o envia para o Validador.

- Teste 4: Votador recebe um voto autenticado pelo Validador.

  Resultado: Votador verifica a assinatura do Validador, encripta com as chaves públicas da Rede de Mistura e do Escrutinador, enviando-o para o Quadro Negro.

- Teste 5: Voto do eleitor é recusado pelo Validador.

  Resultado: Votador mostra uma mensagem de erro para o eleitor indicando que ele não está cadastrado ou que já votou.

6.2 Validador

- Teste 1: Eleitor cadastrado envia voto pela primeira vez.

  Resultado: Validador retira a assinatura do eleitor, assina com a sua própria chave privada e envia de volta para o Votador.

- Teste 2: Eleitor cadastrado que já votou, tenta enviar um segundo voto.

  Resultado: Validador recusa-se a assinar o voto e envia uma mensagem de erro.

- Teste 3: Eleitor não cadastrado envia voto.

  Resultado: Validador recusa-se a assinar o voto e envia uma mensagem de erro.

- Teste 4: Teste de carga. Dez mil módulos Votadores enviam votos ao Validador para serem assinados.

  Resultado: Validador realiza corretamente a validação de todos os dez mil votos.

6.3 Quadro Negro

- Teste 1: Eleitor cadastrado e que ainda não votou tenta se autenticar.

  Resultado: Quadro Negro envia uma cédula de votação para o Votador.

- Teste 2: Validador requisita a lista de eleitores cadastrados.

  Resultado: Quadro Negro devolve a lista de eleitores cadastrados.

- Teste 3: Eleitor não cadastrado tenta se autenticar.

  Resultado: Quadro Negro recusa a autenticação e envia uma mensagem de erro.

- Teste 4: Eleitor que já votou tenta se autenticar.

  Resultado: Quadro Negro recusa a autenticação e envia uma mensagem de erro.

- Teste 5: Eleitor cadastrado e que ainda não votou tenta enviar voto encriptado.

  Resultado: Quadro Negro aceita eleitor, publica seu voto e salva seu número de identifcação na lista de eleitores que já votaram.

- Teste 6: Fase de apuração é iniciada.

  Resultado: Quadro Negro envia todos os votos publicados para a Rede de Mistura.

- Teste 7: Rede de Mistura retorna os votos embaralhados.

  Resultado: Quadro Negro publica todos os votos recebidos e os envia para o Escrutinador.

- Teste 8: Escrutinador retorna os votos decriptados, o resultado da eleição e sua chave privada.

  Resultado: Quadro Negro publica todas essas informações.

- Teste 9: Teste de carga. Quadro Negro recebe 5 mil pedidos de autenticação e 5 mil votos para serem publicados.

  Resultado: Quadro Negro resolve corretamente os pedidos de autenticação e, caso venham de eleitores válidos, publica os votos recebidos.

6.4 Rede de Mistura

- Teste 1: Quadro Negro envia todos os votos duplamente encriptados de forma correta.

  Resultado: Rede de Mistura decripta, embaralha e envia todos os votos novamente para o Quadro Negro.

- Teste 2: Quadro Negro envia um voto alterado dentre os votos duplamente encriptados.

  Resultado: Rede de Mistura detecta erro na decriptação, envia uma mensagem de erro, descarta o voto e continua a operação de embaralhamento.

- Teste 3: Teste de carga. Rede de Mistura recebe um conjunto de 100 mil votos para serem embaralhado.

  Resultado: Rede de Mistura embaralha os votos de forma aleatória e os envia de volta para o Quadro Negro.

6.5 Escrutinador

- Teste 1: Quadro Negro envia todos os votos encriptados de forma correta.

  Resultado: Escrutinador decripta os votos, verifica suas assinaturas, contabiliza o resultado e devolve as informações para o Quadro Negro.

- Teste 2: Quadro Negro envia um voto alterado dentre os votos encriptados.

  Resultado: Escrutinador detecta erro na decriptação, envia uma mensagem de erro, descarta o voto e continua a operação de escrutínio.

- Teste 3: Voto com assinatura inválida é encontrado.

Resultado: Escrutinador detecta assinatura inválida, envia uma mensagem de erro, descarta o voto e continua a operação de escrutínio.

- Teste 4: Teste de carga. Escrutinador recebe um conjunto de 100 mil votos para ser contabilizado.

  Resultado: Escrutinador verifica cada um dos votos e os contabiliza, enviando os votos e o resultado de volta para o Quadro Negro.

6.6 Modo de Recuperação

- Teste 1: Modo de Recuperação é acionado com os arquivos de *backup* corretos.

  Resultado: Quadro Negro recarrega todos os votos dos arquivos e continua com a Fase de Votação.

- Teste 2: Votos são apagados do arquivo de *backup* antes da recuperação do sistema.

  Resultado: Pela diferença entre o número de votos armazenados e o número de eleitores contidos no arquivo encriptado de eleitores que já votaram, o sistema percebe a fraude e interrompe a eleição.

- Teste 3: Teste de carga. O Quadro Negro é acionado para carregar um arquivo com 100 mil votos.

  Resultado: Os votos são carregados corretamente e a votação continua.

6.7 Sinopse

Neste capítulo descreveram-se os testes executados para validar o funcionamento do software protótipo e provar sua segurança. Executaram-se tanto testes que representam etapas previstas no funcionamento do protocolo, quanto testes envolvendo ações tomadas por um possível agressor. O sistema provou-se robusto e condizente com sua especificação em todos seus resultados.

Capítulo 7

COMPARAÇÃO COM O HELIOS

Neste capítulo descreve-se o funcionamento de um sistema de votação eletrônica disponível para uso na Internet, o Helios Voting System. Este sistema é bastante utilizado por diversos usuários e já está em sua terceira versão, consistindo em uma boa base para comparação com o sistema proposto neste trabalho. Assim, destacam-se aqui as diferenças nos requisitos alcançados por cada um desses sistemas.

7.1 Helios Voting

Helios Voting System é o primeiro sistema de votação baseado pela web com audição aberta (HELIOS VOTING SYSTEM, 2010). Neste tipo de sistema, é possível ao eleitor auditar o processo inteiro.

Num sistema de votação online há um alto risco de coerção, pois um indivíduo pode ser influenciado por qualquer um que estiver por perto vendo o eleitor votar. O problema da coerção não é resolvido pelo sistema Helios, logo este não deve ser usado em eleições de grande importância. O sistema foi projetado para eleições de baixo risco de coerção, como as de clubes locais, de grupos online, de organizações estudantis, e outros.

O protocolo de votação prioriza a integridade em relação à privacidade. Isto significa que a implementação se dá de modo a garantir que mesmo que todos os gerentes de uma eleição sejam corruptos eles não conseguiriam falsificar um resultado de eleição. O eleitor deve confiar no Helios para garantir a sua privacidade.

A preparação do voto e a votação funcionam da seguinte maneira:

- O eleitor inicia o processo de votação indicando em qual eleição ele deseja participar.
- O eleitor escolhe então seu voto dentre as opções da cédula. Essa etapa é controlada pelo sistema BPS (*Ballot Preparation System*).
- BPS encripta a escolha do eleitor.

- O eleitor pode então auditar a cédula. O BPS mostra a mensagem cifrada e a aleatoriedade usada para criá-lo. Se o eleitor desejar, ele pode gerar uma nova encriptação do seu voto.
- Alternativamente, o eleitor pode escolher selar sua cédula. Desse modo, o BPS descarta os fatores de aleatoriedade e a mensagem não-cifrada, armazenando apenas o voto encriptado.
- O eleitor é então autenticado. Se a autenticação for válida, o voto encriptado é salvo como o voto daquele eleitor.

7.1.1 Quadro de Votos

Um quadro de votos é previsto pelo sistema Helios. Nesse quadro de votos qualquer um pode verificar a integridade através da presença de seu voto encriptado.

7.1.2 Rede de mistura

Para garantir o anonimato, uma rede de mistura é usada pelo protocolo Helios. Os textos cifrados são embaralhados e re-randomizados. O protocolo utilizado pelo Helios para a rede de mistura é o Sako-Kilian (ADIDA, 2008).

O funcionamento geral do Helios segue os procedimentos a seguir, no qual Alice deseja votar em um determinado processo eleitoral:

1. Alice prepara e audita tantas cédulas quanto queira, assegurando que todas as cédulas auditadas são consistentes. Quando se dá por satisfeita, Alice deposita uma cédula encriptada, o que requer que ela autentique-se.
2. O quadro de votos do Helios disponibiliza o nome de Alice e sua cédula encriptada. Qualquer um, incusive Alice, pode verificar o quadro e encontrar o seu voto encriptado.
3. Quando a eleição termina, o Helios embaralha todas as cédulas encriptadas e produz uma prova não interativa do correto embaralhamento.
4. Depois de um razoável período para os auditores poderem verificar o embaralhamento, o Helios decripta todas as cédulas embaralhadas, provê uma prova de decriptação para cada cédula e executa a computação dos votos.
5. Um auditor pode pegar todos os dados da eleição e verificar o embaralhamento, as decriptações e a computação dos votos.

7.2 Comparação do VoteRemote com o Helios

O protocolo Helios não satisfaz de maneira ampla os requisitos elencados no capítulo 2, enquanto que o VoteRemote satisfaz a maioria. Abaixo é feita uma comparação entre os dois protocolos, mostrando os requisitos não atendidos pelo Helios e o porquê não são atendidos.

- Requisitos de exatidão
  - * Um voto não pode ser repetido, apagado ou alterado.
    Alguns protocolos permitem que um eleitor possa sobrescrever seu voto se foi coagido (ADIDA, 2008). No caso do Helios e do VoteRemote isso não é possível, e esse requisito é atendido pelos dois protocolos.
  - * Todos os votos válidos são contabilizados.
    Ambos protocolos atendem a esse requisito.
- Democracia
  - * Somente os eleitores cadastrados podem votar e cada eleitor tem direito a lançar no máximo um voto.
    Tanto o Helios como o VoteRemote atendem a esses dois requisitos de democracia.
- Confidencialidade
  - * Anonimato: Não é possível associar um voto com o eleitor que o emitiu.
    Esse requisito é atendido tanto pelo Helios quanto pelo VoteRemote, porém naquele com a condição que deve-se confiar no servidor de Helios.
  - * Equidade: Todos os votos permanecem em segredo até o final do período de votação.
    Esse requisito é atendido por ambos protocolos.
  - * Incomprovabilidade: Nenhum eleitor pode provar que votou de maneira determinada.
    No caso do protocolo Helios esse requisito não é atendido, pois com os fatores de randomicidade guardados o eleitor consegue provar seu voto.
  - * Incoagibilidade: Nenhum eleitor pode ser forçado a votar de maneira determinada.
    Esse requisito não é atendido pelo Helios, que não tem um enfoque na incoagibilidade. Já o protocolo VoteRemote dependendo de como for im-

plementado (por exemplo, no caso de votação online) pode ter os mesmos problemas do protocolo Helios.

- Verificabilidade
  - * Universal: Qualquer um pode verificar que todos os votos válidos foram contados.
  - * Individual: Cada eleitor pode verificar que seu voto foi considerado válido.

Tanto o VoteRemote como o sistema Helios atendem os requisitos de Verificabilidade acima. Porém, enquanto o Helios fornece uma maior Verificabilidade para auditores, que podem verificar todos os passos do processo eleitoral, o VoteRemote permite uma maior Verificabilidade para qualquer usuário do sistema, seja ele eleitor ou auditor.

7.3 Sinopse

Neste capítulo descreveram-se as características e funcionamento do sistema de votação eletrônica Helios Voting System, aberto para uso por qualquer usuário ou organização que deseje organizar uma eleição *online*. Compararam-se então os requisitos atendidos pelo Helios e o sistema proposto neste trabalho. Apesar dos dois sistemas atenderem amplamente os principais requisitos de segurança, existem algumas diferenças entre eles, como o fato do Helios não atender completamente o requisito de Incomprovabilidade, e o VoteRemote possuir uma Verificabilidade maior sob a visão do eleitor.

Capítulo 8

CONSIDERAÇÕES FINAIS

8.1 Análise dos Resultados

Neste trabalho, implementou-se com sucesso um sistema de votação eletrônico seguro baseado no protocolo VoteRemote.

Inicialmente, estudou-se a fundo o protocolo, analisando-se seus pontos fortes. Baseado nessas observações, pode-se projetar um sistema que fosse capaz de atender a todas as premissas deste protocolo, garantindo sua segurança; implementasse corretamente seu funcionamento; e, ainda, apresentasse fácil distribuição e baixas exigências em termos de *hardware* especializado, de forma que pudesse ser usado por usuários normais da internet para realizações de diversos tipos de votações.

Além do mais, o sistema apresenta uma grande flexibilidade, podendo ser implementado de variadas formas, além de ser possível adicionar ou remover determinadas medidas de modo a se adaptar ao nível de segurança desejado. Um exemplo dessa flexibilidade é a forma como a senha do eleitor pode ser armazenada, que pode variar desde um *hardware* criptográfico ou um leitor biométrico, até uma senha de quatro dígitos a ser decorada pelo eleitor.

8.2 Possibilidades de Melhorias

O sistema proposto neste trabalho apresenta uma grande flexibilidade em termos de implementação. O protótipo aqui descrito é apenas um exemplo de implementação, apresentando um formato parecido com o do processo eleitoral brasileiro baseado na urna eletrônica, e restrito apenas ao uso de tecnologias simples, encontradas na casa da maioria dos usuários brasileiros (computadores domésticos e *pen drives*, por exemplo).

Outra possível implementação que seria muito interessante e útil atualmente envolveria, por exemplo, a troca da interface gráfica para uma página Web, de forma que os usuários pudessem votar de suas próprias casas. Para eleições de grande porte, poderia-se também distribuir o processamento do servidor em múltiplas Redes de

Mistura, Quadros Negros e Validadores, minimizando a carga sobre cada um desses componentes. Os módulos votadores seriam então divididos em grupos, cada um controlado por um Quadro Negro. Um conjunto de Quadro Negros enviaria seus votos para uma Rede de Mistura específica. Por fim, todas as Redes de Mistura enviariam seus votos embaralhados para um único Escrutinador, responsável por contabilizar o resultado da eleição e publicá-lo em todos os Quadros Negros.

Além disso, caso os partidos não confiem nos módulos fornecidos pelo organizador oficial da eleição, cada um deles poderia fornecer sua própria Rede de Mistura. Nesse caso, o processo de embaralhamento seria feito sequencialmente, começando sempre pela Rede Mistura oficial da eleição e passando, em seguida, pelas Redes de Mistura de cada partido, em uma ordem que poderia ser tanto previamente sorteada, ou ainda seguindo, por exemplo, o número do partido. Essa solução diminuria o desempenho do sistema, mas aumentaria seu nível de segurança e a confiança nele depositada por seus usuários.

Referências

ADIDA, B. Helios: Web-based Open-Audit Voting. In: *Proceedings of the Seventeenth Usenix Security Symposium (USENIX Security 2008)*. San Jose, CA, USA: USENIX Association, 2008. p. 335–348. Disponível em: <http://www.usenix.org/events/sec08/tech/full_papers/adida/adida.pdf>.

APACHE. *Jakarta Commons Configuration Website*. 12 2008. Acesso em 29 nov. 2010. Disponível em: <http://commons.apache.org/configuration/>.

____. *Apache MINA Website*. 11 2010. Acesso em 29 nov. 2010. Disponível em: <http://mina.apache.org/features.html>.

____. *Jakarta Commons Configuration Website*. 12 2010. Acesso em 29 nov. 2010. Disponível em: <http://commons.apache.org/configuration/>.

____. *LOG4J Website*. 03 2010. Acesso em 29 nov. 2010. Disponível em: <http://logging.apache.org/log4j/1.2/index.html>.

CHAUM, D. Blind signatures for untraceable payments. In: *CRYPTO*. New York: Plenum Press, 1982. p. 199–203.

HELIOS VOTING SYSTEM. *Helios Website*. 08 2010. Acesso em 29 nov. 2010. Disponível em: <http://heliosvoting.org/>.

LANGER, L. et al. Ein PKI-basiertes protokoll für sichere und praktikable onlinewahlen. In: *Proceedings of EDEM 2009*. Austria: OCG, 2009. p. 243–253.

MAMBO, M.; ZHENG, Y. (Ed.). *Information Security, Second International Workshop, ISW'99*, v. 1729 de *Lecture Notes in Computer Science*, (Lecture Notes in Computer Science, v. 1729). Kuala Lumpur, Malaysia: Springer, 1999. ISBN 3-540-66695-8.

NIST. *NIST Website*. Gaithersburg, Maryland: NIST, 12 2010. Acesso em 29 nov. 2010. Disponível em: <http://www.nist.gov/index.html>.

RIVEST, R. L.; SHAMIR, A.; ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, ACM, New York, NY, USA, v. 21, n. 2, p. 120–126, February 1978. ISSN 0001-0782. Disponível em: <http://dx.doi.org/10.1145/359340.359342>.

SPECIFICATION for the Advanced Encryption Standard (AES). 2001. Federal Information Processing Standards Publication 197. Disponível em: <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

THE LEGION OF THE BOUNCY CASTLE. *Bouncy Castle Website*. 02 2010. Acesso em 29 nov. 2010. Disponível em: <http://www.bouncycastle.org/>.