SIGEleição – Um Novo Jeito Seguro de Votar

Jadson Santos[1], Clarissa Lins[1], Marcos Madruga[1]

[1]Superintendência de Informática – Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59078-970 – Natal– RN – Brasil

{jadson, clarissa, madruga}@info.ufrn.br

Abstract. *The effort and cost of setting up several electronic voting machines or the work for the manual counting of hundreds or thousands of paper ballots led to emerge a demand for implementation of a web system to manage elections. This required that the system had extra safety features not present in the other systems developed by UFRN. This paper describes the SIGEleição voting system, highlighting the security, accessibility, availability and performance mechanisms that have been implemented to provide elections for the various management positions inside UFRN.*

Resumo. *O esforço e custo de se configurar diversas urnas eletrônicas ou o trabalho para contagem manual de centenas ou milhares de votos em papel fez com que surgisse uma demanda para implementação de um sistema web para realizar votações. Nesse sentido, a UFRN desenvolveu o sistema de votação SIGEleição para que se pudesse realizar pleitos para os diversos órgãos que compõem a instituição. Este artigo descreve o sistema SIGEleição, destacando os mecanismos de segurança, acessibilidade, disponibilidade e desempenho que foram implementados.*

1. Introdução

As instituições de ensino superiores brasileiras necessitam realizar periodicamente eleições para a escolha dos seus cargos de direção tais como: reitoria, diretoria de centros, diretoria de departamentos ou coordenações de cursos.

Normalmente essas eleições são realizadas presencialmente em várias localidades da instituição. Algumas vezes até em localidades geograficamente distintas. Isso acarreta um custo considerado para realizar essas eleições, tanto financeiro, como de recursos humanos para organizar e para trabalhar na eleição.

Nesse cenário, de busca de melhoria organizacional, efetividade e eficiência dos recursos públicos, surgiu a demanda de se desenvolver um sistema eleitoral em que se pudesse registrar votos de qualquer localidade, sem a necessidade de montar uma estrutura física grande e custosa. A votação *on-line* vem maximizar a conveniência e acesso dos eleitores, permitindo o pleito eleitoral em qualquer lugar que tenha acesso à Internet.

Nesse sentido, a Universidade Federal do Rio Grande do Norte (UFRN) por meio da Superintendência de Informática [SUPERINTENDÊNCIA, 2015], órgão responsável pela política de TI da instituição, desenvolveu o sistema de eleição *on-line* com objetivo de reduzir esforço financeiro, logístico e humano além de garantir os processos e requisitos exigidos para segurança e confiabilidade da eleição. O presente

artigo pretende descrever os aspectos de segurança implementados para a versão 2.0 do SIGEleição para que ele pudesse ser utilizando nas diversas eleições ocorridas na UFRN.

O artigo está organizado da seguinte forma: Na Seção 2 são discutidas algumas características dos sistemas de votação atuais. Na Seção 3 são apresentadas as funcionalidades de segurança implementas para o SIGEleição. A Seção 4 é apresentada a funcionalidade de cabines registradas, que pode ser usada para restringir o universo de computadores que podem registrar votos. A Seção 5 apresenta a avaliação realizada sobre o sistema. Por último na Seção 6 as conclusões.

2. Sistemas Eletrônicos de Votações

Segundo [POST 2001] existem três tipos principais de sistemas eletrônicos de votação: (i) Máquina de votar de gravação eletrônica direta do voto, (ii) Contagem de cédulas realizada por máquina e (iii) Sistemas *on-line*. O SIGEleição se enquadra no terceiro tipo.

[Chaves e Mello 2014] realizaram uma busca por soluções de sistemas de votação *on-line* chegando a três nomes principais, como possíveis candidatos para serem usados na votação do CONSUP IFSC:

1. Sistema Aberto de Eleições Eletrônicas (SAELE)
2. SIGEleição
3. Helios versão 3

Após esse passo, definiram sete requisitos que deveriam ser atendidos por esses sistemas: (i) Só poderão votar os eleitores que forem considerados aptos pela comissão eleitoral; (ii) Cada eleitor só terá direito a um único voto por segmento que este estiver apto a votar (docente, discente e técnico-administrativo); (iii) A escolha do eleitor deve ser mantida em sigilo. Ninguém poderá saber em quem o eleitor votou, mesmo se este quiser revelar; (iv) A solução e o resultado das eleições devem ser auditáveis. A integridade dos votos deve ser garantida, ninguém poderá alterar, incluir ou remover votos; (v) A solução deve ser economicamente viável, tanto para sua aquisição ou implantação, quanto para realização do pleito; (vi) A solução deve ser de fácil uso por eleitores e pela comissão eleitoral; (vii) Não permitir a realização de apurações parciais antes do término da eleição.

Após uma análise dos três sistemas, [CHAVES e MELLO 2014] desconsideraram o SIGEleição por não atendar os requisitos iii e vii, porém esse trabalho avaliou a versão inicial do sistema.

Ademais, [MONTEIRO et. al., 2001] também cita sete requisitos necessários a um sistema de votação *on-line* para impedir a fraude: (i) não permitir a alteração do voto; (ii) não eliminar um voto válido; (iii) averiguar se o voto é claramente expresso; (iv) assegurar que só votaram as pessoas registradas; (v). assegurar que cada pessoa só votou uma vez; (vi) assegurar que o voto é secreto; (vii) verificar que os votos foram contados corretamente, isto é, que o total de votos obtidos coincide com o número de votantes.

O presente artigo pretende descrever as soluções de segurança implementadas na versão 2.0 do SIGEleição que permitiram que ele atendesse os requisitos de segurança citados por [MONTEIRO et. al., 2001] e por [CHAVES e MELLO 2014].

3. Soluções de Segurança do SIGEleição versão 2.0

Nesta seção são descritos como os mecanismos de segurança do SIGEleição foram implementados, de forma a atender aos requisitos de segurança mencionados da seção anterior.

3.1 Sigilo do Voto

A primeira propriedade que um sistema de votação eletrônico deve possuir é garantir que os votos registrados sejam sigilosos. Ou seja, ninguém, em tempo algum, nem mesmo o administrador da base de dados, pode obter a informação: "em quem o eleitor votou?" Para conseguir essa propriedade, o SIGEleição foi projetado para salvar as informações dos votos em duas tabelas distintas e não relacionadas entre si. A Tabela 1 mostra uma representação da tabela denominada "Voto" e a Tabela 2 mostra uma representação da tabela VotoEleitor.

Tabela 1. Tabela Voto

id_voto	id_eleicao	id_chapa	valor	hash (128 caracteres)
12345	398221	1202	V	sdfjslfj930238dsofj20wes...
12346	398221	1203	V	92nwds923ksf923f923323...
12347	398221		B	8sd39sfds9823030909wfe0...

Tabela 2. Tabela VotoEleitor

id_eleicao	data_votacao	id_pessoa	ip_eleitor	hash (128 caracteres)
398221	22/05/2015	23423	10.3.20.123	d4slkf2989udf23wedsddf...
398221	22/05/2015	5342343	10.4.20.122	kd92lksfj20j02jwl320wd...
398221	22/05/2015	192302	192.0.2.120	80ksdfl23we0sdf0sdf0d...

A tabela Voto guarda os votos que foram computados para as eleições (coluna id_eleicao), com a indicação do valor do voto (V = válido, B = Branco e N = Nulo) e para qual candidato esse voto foi computado (coluna "id_chapa"). Nessa tabela não existe nenhuma indicação do eleitor que realizou o voto. Para salvar essa informação foi criada uma outra tabela chamada VotoEleitor.

A funcionalidade da tabela VotoEleitor é guardar o eleitor que realizou o voto, identificado pela coluna "id_pessoa", a eleição que essa pessoa votou (coluna "id_eleicao"), a data em que o voto foi registrado (coluna "data_votacao") e algumas

informações a mais para auditoria como o endereço IP do computador que o eleitor utilizou para votar (coluna “ip_eleitor”).

Não existe nenhuma informação ou chave estrangeira que ligue uma tabela a outra. Tornando impossível identificar, mesmo com acesso ou banco de dados do sistema, que determinado eleitor votou em determinado candidato.

Vale ressaltar que ainda determinados cuidados foram tomados na tabela VotoEleitor para impedir a identificação do voto e aumentar a segurança do sistema: (i) A coluna “data_votacao” não guarda as informações do horário que o voto foi registrado, apenas o dia. (ii) Essa tabela também não possui um identificador sequencial que possa mostrar a ordem em que os votos foram realizados. (iii) Por último, a cada voto registrado, uma mistura aleatória na ordem dos votos é realizada. Assim a ordem em que os votos estão registrados nessa tabela na base de dados não é a mesma ordem em que os votos foram realizados, nem a mesma ordem dos votos registrados na tabela Voto, impossibilitando que alguém faça essa associação.

A Figura 1 mostra código parcial do método *shuffleVotosEleitor*() utilizado para essa tarefa. Esse método sorteia aleatoriamente um voto entre os 10 primeiros, a linha sorteada é então atualizada via um comando SQL. Ao fazer isso, o banco de dados PostgreSQL (versão 9.3.2), por uma questão de performance, internamente deixa a linha atual desativada e insere uma nova no final na tabela. Na prática funciona como se o voto fosse apagado entre os 10 primeiros e inserido no final da tabela.

```
private void shuffleVotosEleitor(VotoEleitorDao votoEleitorDao, VotoEleitor votoEleitorAtual, int idEleicao)

    String camposVotosEleitor = "pessoa.id, eleicao.id";

    Random r = new Random(System.currentTimeMillis());

    List<VotoEleitor> votosEleitor = votoEleitorDao.findIdentificadoresVotosEleitoresRegistrados(idEleicao, 1

    if(votosEleitor.size() > 1){  // apartir do segundo voto

        Integer indexVotoEscolhido =  r.nextInt( new Long(votosEleitor.size()-1).intValue() ); // sorteia 1 \

        VotoEleitor votoSorteado = votoEleitorDao.findByPrimaryKey( (votosEleitor.get(indexVotoEscolhido)).g

        /* ********************************************************************************
         * Realiza apenas um update na linha, ao realizar um update o POSTGRESQL mudar a ordem dos votos na t
         * pois ele apaga o anterior e insere um novo
         *
         * O voto sorteado passa a ser o ÚLTIMO voto retornado na tabela, ao se fazer um "select"
         ******************************************************************************** */
        votoEleitorDao.update("UPDATE eleicao.voto_eleitor SET id_pessoa = ? , id_eleicao = ? WHERE id_pesso
                votoSorteado.getPessoa().getId(), votoSorteado.getEleicao().getId(), votoSorteado.getPessoa()

    }
}
```

Figura 1. Código de embaralha a ordem dos votos do eleitor

3.2 Não Permitir Alteração de um Voto Válido

Toda eleição criada no SIGEleição deve possuir uma comissão eleitoral, cujo presidente tem como principal função gerar e guardar sigilosamente uma chave de segurança para a eleição.

A chave de segurança é uma sequência de 64 caracteres gerados aleatoriamente. O ponto primordial da segurança do SIGEleição é que essa chave de segurança não é persistida em nenhum lugar pelo sistema. Ela fica apenas na memória do servidor que

executa o SIGEleição. Ninguém que gerencia o sistema tem acesso à chave de segurança, mesmo que se possua acesso ao banco de dados.

Com a chave de segurança carregada na memória do servidor, ela é utilizada para gerar vários *hashes* [GAT UFRJ, 2015] que permitem ao sistema identificar se alguma alteração foi realizada nas tabelas auditadas.

A Figura 2 mostra o esquema de geração do hash da tabela voto.

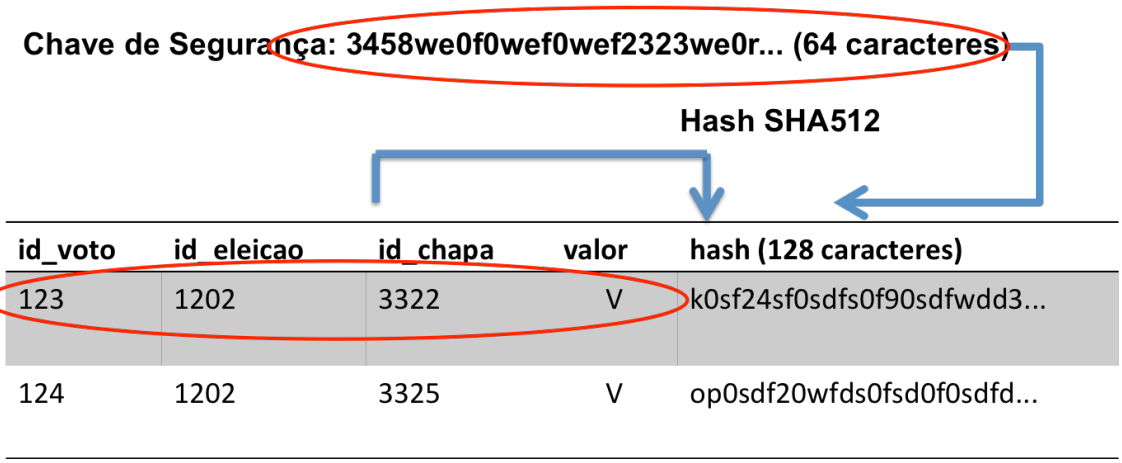


Figura 2. Esquema de Criação do Hash dos Votos

Para cada voto registrado, linha da tabela Voto, os dados da linha são concatenados juntos com a chave de segurança da eleição. Com o resultado dessa concatenação é então gerado um hash SHA512 de 128 caracteres para esse voto e salvo na coluna "hash" da linha correspondente ao voto.

Como ninguém que tem acesso à base de dados do sistema, tem acesso à chave de segurança, não é possível calcular um hash válido. Outra propriedade das funções de hashing é que elas são unidirecionais [GAT UFRJ 2015], não é possível a partir do valor dos hashes anteriormente salvos recuperar o valor da chave de segurança. Com isso, garantimos que ninguém pode registrar um novo voto válido na base de dados do sistema.

Também não é possível alterar um voto registrado, pois outra propriedade das funções de hashing é que se houver qualquer modificação de parte da informação usada na geração do hash, o hash obrigatoriamente também muda. Se alguma coluna da tabela for alterada, para o hash correspondente permanecer válido, ele teria que ser recalculado, o que seria impossível já que não se conhece a chave de segurança

A Figura 3 mostra o código de geração do hash da tabela voto.

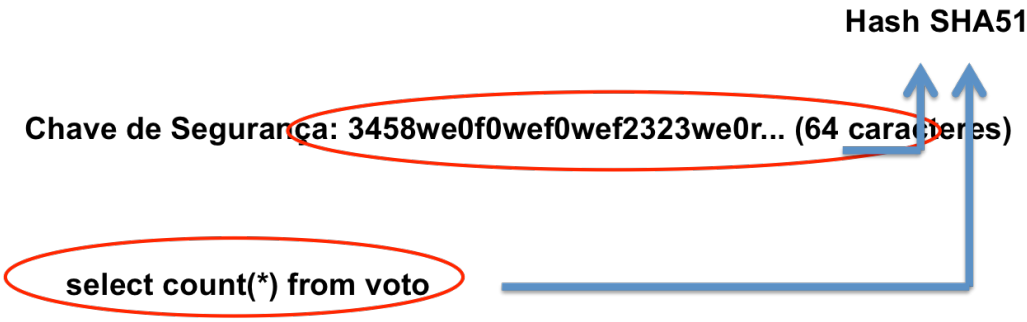
```java
/**
 * Gera o hash do voto para auditoria. Os dados o objeot devem estar populados se não vai dar
 * @param senhaAuditoria
 * @return
 *
 * <p> Criado em:  28/08/2014  </p>
 *
 */
public String gerarHash(String senhaAuditoria) {

    // pré-condição do método  //
    if( id <= 0 || eleicao == null || eleicao.getId() <= 0  || grupoEleitor == null  || grupoE
        throw new RuntimeException("Informações incompletas para gerar o hash");

    String dadosVoto = ""+id+""+eleicao.getId()+ ( candidatura != null && candidatura.getId()


    // adiciona a senha em memória para auditar se alguém alterou as informações no banco //
    // faz uma duplicação de informações, para pontencializar pequenas mudanças           //
    dadosVoto = dadosVoto + senhaAuditoria + dadosVoto + senhaAuditoria + dadosVoto;

    String tempHash = SHAHashingUtil.toSHA512(dadosVoto);

    // pós-condição do método //
    if( StringUtils.isEmpty(tempHash))
        throw new RuntimeException("Hash foi gerado errado.");

    return tempHash;
}
```

Figura 3. Código de Geração do Hash do Voto

No algoritmo de criação do hash, os dados do voto são concatenados, em seguida eles são combinados várias vezes com a chave de segurança para potencializar pequenas diferenças entre as informações dos votos e dificultar a colisão de hashes.

Por último, para garantir que um voto não seja apagado da base de dados, o SIGEleição além de gerar um hash para cada voto registrado no sistema, gera um hash geral para a eleição, que fica armazenado na tabela “eleicao”. Esse hash é gerado a partir da quantidade de votos registrados até o momento na eleição. A cada voto registrado, o sistema verifica se o hash atual é válido. Em caso positivo, uma nova contagem de votos é realizada, um novo hash é gerado para a nova quantidade de votos e esse campo é atualizado na tabela “eleicao”. Esse esquema é apresentado na Figura 4.

id_eleicao	inicio	fim	hash (128 caracteres)
1202	10/05/2015	22/05/2015	usd82wsdf92d230lsdlssdsd8...



id_voto	id_eleicao	id_chapa	valor	hash (128 caracteres)
123	1202	3322	V	k0sf24sf0sdfs0f90sdfwdd3...
124	1202	3325	V	op0sdf20wfds0fsd0f0sdfd...

Figura 4. Geração do hash da eleição com a quantidade de votos registrados

Dessa maneira, se por acaso um voto for apagado da base de dados, o hash da quantidade de votos da eleição não coincidirá com o hash calculado a partir da quantidade de votos registrado na tabela "voto". Não é possível tentar alterar esse número, porque novamente não se sabe a chave de segurança utilizada para se gerar o hash da tabela "eleição". Qualquer tentativa de alterar esse número, geraria um hash inválido que impugnaria imediatamente a eleição.

Vale ressaltar que o mesmo esquema de criação de hash é utilizado para a tabela VotoEleitor. Não sendo possível adicionar, alterar ou remover os dados dessa tabela que são utilizados para auditoria da eleição.

3.3 Auditar os Resultados de uma Eleição

Ao término da eleição, para homologar os resultados, o presidente deve informar novamente essa chave de segurança ao sistema, que então é utilizada para verificar se todos os votos foram registrados. Somente se todos os votos forem válidos e a quantidade de votos registrado for válida, o resultado da eleição é homologado e publicado.

Caso ocorra algum problema durante a eleição e o sistema seja reiniciado, a informação da chave de segurança é perdida. Cabe ao presidente da comissão eleitoral entrar no sistema e fornecer novamente a chave de segurança. Pois, nenhum voto pode ser registrado no SIGEleição enquanto a chave de segurança não estiver carregada na memória do servidor.

Como mostrado na Tabela 2, a tabela VotoEleitor mantém algumas informações para auditoria, como a endereço IP usado pelo eleitor para registrar o seu voto. Por questões de segurança não é possível obter informações do tipo: "Quantos votos

candidato X obteve em determinado setor", pois como dito anteriormente, não é guardada nenhuma referência em quem o eleitor votou.

3.4 Assegurar que cada pessoa só votou uma vez

Além um uma regra de negócio no código do sistema que verifica a tentativa de realizar um voto duplicado, a chave primária na tabela VotoEleitor é a combinação das colunas ("id_pessoa", "id_eleicao"), impedindo que um mesmo eleitor vote duas ou mais vezes para a mesma eleição.

3.5 Assegurar que só votaram as pessoas registradas

Outra propriedade do SIGEleição é assegurar a autenticidade do eleitor, ou seja, que ninguém possa votar no seu lugar e só pessoas que estão habilitadas a votar em uma eleição podem registrar o seu voto. Para isso, foram implementados alguns esquemas de autenticação dos usuários do SIGEleição.

3.5.1. Modo de Autenticação com Login e Senha

Esse é o modo padrão de autenticação da maioria dos sistemas e é um mecanismo obrigatório do SIGEleição. Todo eleitor para entrar no sistema deve informar pelo menos um *login* e uma senha que são pessoais e intransferíveis. A segurança desse método de autenticação está na segurança da senha do eleitor.

O SIGEleição conta com um sistema de bloqueio de *logins* inválidos. Na primeira tentativa inválida, o eleitor fica impedido de tentar novamente essa operação por um período de 1 minuto. Na segunda tentativa, 2 minutos; na terceira, 4 minutos e assim progressivamente. Esse bloqueio dificulta a tentativa de quebra da senha por força bruta.

3.5.2. Modo de Autenticação com Pergunta de Segurança

Além do *login* e senha, o SIGEleição solicita que o eleitor responda uma pergunta de segurança, selecionada aleatoriamente do banco de dados. Esse modo de autenticação é opcional e complementar. Essas perguntas de segurança tem por objetivo dificultar que usuários não humanos tentem acessar o sistema. Um exemplo de uma pergunta de segurança utilizada nesse módulo de autenticação: "Quantas letras tem a palavra 'casa'?".

A Tabela 3 mostra o tempo para se descobrir as perguntas de segurança em uma hipotética tentativa de invasão do sistema. Por questões de segurança não se vai divulgar quantas perguntas de segurança existem cadastradas e o tempo que se demoraria para se descobrir todas elas.

Tabela 3. Tempo de bloqueio a cada tentativa errada de se logar no sistema

Tentativas Inválidas de Login	Tempo para tentar novamente
1	1 minuto
2	2 minutos
3	4 minutos
4	8 minutos
5	16 minutos
10	512 minutos
20	524.288 minutos

3.5.3. Modo de Autenticação com Captcha

Outra estratégia opcional e complementar ao modo de autenticação por *login* e senha é exibir um campo de captcha [CAPTCHA 2015] junto ao campo de *login* e senha para impedir que usuários não humanos tentem acessar o sistema. Esse modo de autenticação é alternativo as perguntas de segurança.

3.5.4. Modo de Autenticação em Duas Etapas

Esse módulo de autenticação é opcional e complementar e pode ser utilizado em combinação com qualquer um dos métodos anteriores. O objetivo é dificultar ainda mais a quebra da senha do eleitor. Quando ativado, após autenticação com *login* e senha, e algum dos outros métodos de autenticação que também estejam ativados, o sistema gerará aleatoriamente uma segunda senha para o eleitor. Essa segunda senha é então enviada para o e-mail do eleitor registrado no sistema. Para ele conseguir se *logar* no sistema terá que acessar o seu e-mail, verificar a senha gerada e informá-la ao sistema em uma segunda tela de autenticação.

Para conseguir se passar pelo eleitor, neste caso, o suposto falsário teria que saber o *login* e a senha normais do sistema, a segunda senha gerada pelo sistema ou o e-mail e a senha do e-mail do eleitor. Uma tarefa muito mais difícil.

Para esse modo de autenticação é possível solicitar que o SIGEleição não solicite a segunda senha caso o eleitor considere que está em um dispositivo seguro. Neste caso é guardado um *cookie* no dispositivo do eleitor pelo navegador web para não mais solicitar a segunda senha enquanto ele acessar o sistema por esse dispositivo, melhorando a usabilidade do sistema.

3.5.5. Pergunta Pessoais de Segurança dentro da Cabine de Votação

Após passar por todos os mecanismos de autenticação do sistema é possível ainda configurar para o sistema solicitar ao eleitor dentro da cabine de votação perguntas pessoais de segurança para confirmação do voto. Neste caso, além dos demais mecanismos de autenticação baseados em senha, um suposto falsário teria que conhecer alguns dados pessoais do eleitor para conseguir registar um voto por ele.

Caso o eleitor erre as perguntas, ele ficará bloqueado não podendo mais utilizar o SIGEleição. A não ser por uma cabine de votação registrada explicada na próxima seção. O número de perguntas de segurança e tentativa erradas até que o eleitor seja bloqueada é configurado por eleição e depende do nível se segurança que a comissão eleitoral queira ter na eleição.

3.5.6. Consultas Específica para cada Eleição.

O sistema trabalha com o conceito de grupos de eleitores, um grupo de eleitor é uma consulta SQL que determina quais eleitores podem votar em uma determinada eleição. Cada eleição pode definir as consultas SQL dos seus grupos de eleitores. A partir da consulta SQL pode-se definir com precisão quais eleitores farão parte da eleição. Impedindo que eleitores não autorizados registrem um voto na eleição.

3.6. Disponibilidade

As duas últimas propriedades desejadas em um sistema de votação é que ele esteja disponível durante todo o período de votação; e ser acessível para usuário com necessidades especiais. Isso garante que quem queira votar possa utilizar o sistema.

Uma das implicações da estratégia de auditoria dos votos escolhida foi a perda da escalabilidade horizontal do sistema no nível do banco de dados. Como a estratégia de auditoria para detectar a remoção de votos foi a criação de um "hash global" por eleição, com a quantidade de votos registrados, não é possível que dois eleitores registrem seu vou ao mesmo tempo no sistema. Se não, um sobrescreveria a quantidade de votos do outro, tornando a eleição inválida. Dois eleitores podem até acessar o sistema ao mesmo tempo e tentar registrar o seu voto, mas no banco de dados um voto deve ser registrado poucos milissegundos depois do outro.

Para isso utilizou-se a estratégia de sincronização que impede que o mesmo trecho do código seja executo por *threads* diferentes ao mesmo tempo. Porém para suportar várias instancias do sistema em execução foi criada também uma estratégia de *lock* nas tabelas "eleicao" e "voto". Isso impede que o sistema seja escalado horizontalmente no nível do banco de dados. Para o universo de eleitores de uma instituição do porte da UFRN, ou até unas 10 vezes maior, isso não se reflete em um problema. Porém teria que se pensar em algum outra estratégia como banco de dados distribuído se for desejado utilizar o SIGEleição em eleição de porte nacional com milhões de eleitores hábitos a votar. A Figura 5 mostra o código de geração do hash da eleição que obriga apenas 1 eleitor por vez executar esse código com trecho de código sincronizados e consultas que realizam *locks* no banco de dados.

```
public void geraHashEleicao(int idEleicao, VotoDao votoDao)throws NegocioException, DAOException {

    synchronized(ProcessadorVotacao.class) {  // se for mesma instância da JVM, 2 pessoas não executam esse método ao mes

        /* ********************************************************************************************
         * LOCK PARA DOIS ELEITORES VOTANDO NA MESMA ELEIÇÃO AO MESMO TEMPO NÃO SOBRE ESCREVER O HASH
         * ********************************************************************************************/
        Eleicao eleicao = votoDao.findDadosEleicaoGerarHashLock(idEleicao);

        long qtdAtualDeVotosAtuais = votoDao.countVotosEleicaoComLock(eleicao.getId());

        // verifica o hash da quantidade atual, antes de escrever a novo quantidade para ter certeza que ninguém alterou
        // só verifica, logicamente, depois do primeiro voto, ou seja, se o hash da eleição não for nulo

        if( ! eleicao.getHash().equals(eleicao.gerarHash(new ChavesAuditoriaManager().getChavesAuditoriaEleicaoCarregada(
            throw new NegocioException("Quantidade de votos registrados para essa eleição não é válida.");
        }


        // se o hash tá válido, gera o novo hash para a quantidade atual de votos + o voto que está sendo registrado agor

        String novoHashEleicao = eleicao.gerarHash(new ChavesAuditoriaManager().getChavesAuditoriaEleicaoCarregada(eleica

        votoDao.updateField(Eleicao.class, eleicao.getId(), "hash", novoHashEleicao); // atualiza o hash da eleição para
    }

}
```

Figura 5. Código Sincronizado de Geração do Hash da Eleição

Como mencionado, toda eleição possui uma chave de segurança e essa chave é carregada apenas na memória do servidor. Para garantir uma maior disponibilidade do sistema era desejado executar o sistema em mais de um servidor. Foi necessário viabilizar alguma solução que permitisse que a chave de segurança gerada fosse compartilhada entre todas as instâncias que estivessem executando o sistema, sem que isso comprometesse a segurança do sistema.

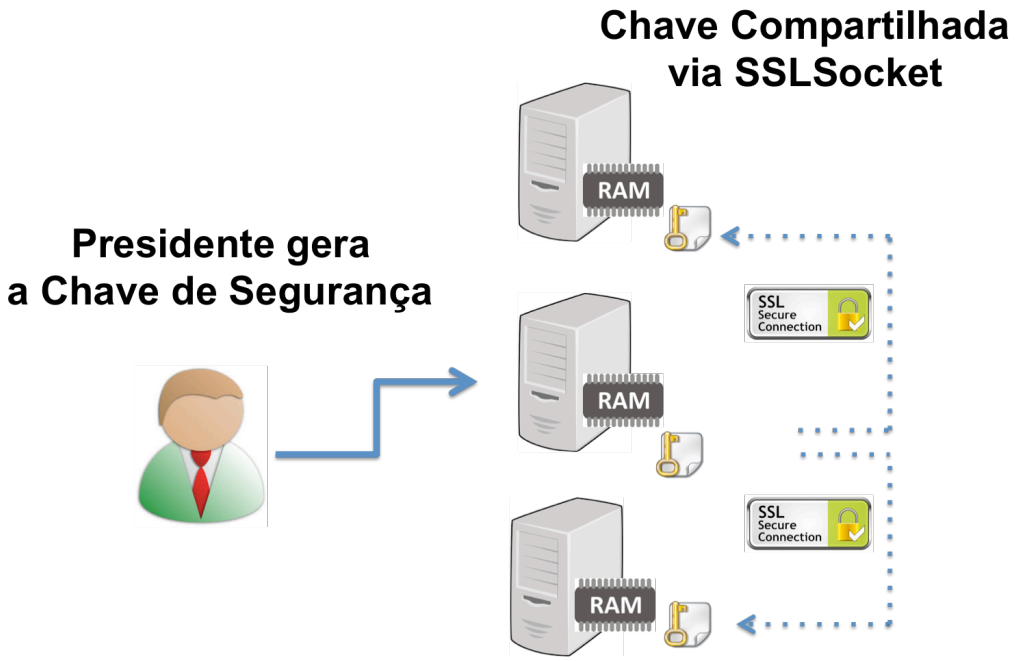


Figura 6. Compartilhamento da chave de segurança

Para isso foi implementado um esquema de cache distribuído exemplificado na Figura 6 utilizando SSLSocket [SSLSocket 2016]. Quando o presidente de uma eleição carrega a chave de segurança em uma instância do SIGEleição, essa chave é compartilhada via conexão TCP para as instâncias previamente cadastradas como sendo servidores do SIGEleição. Esse cadastro é realizado informando o endereço IP dos

servidores do SIGEleição. Segundo [SSLSocket 2016], o uso dos protocolos SSL sobre a conexão TCP providos pela biblioteca SSLSocket do Java garantem integridade, autenticação e confidencialidade, impedido que se possa descobrir a chave de segurança de uma eleição monitorando a rede durante esse processo de compartilhamento entre os servidores que executam o sistema.

3.7 Acessibilidade do Sistema

Por último as páginas do sistema foram testadas para garantir que deficientes visuais consigam votar na eleição, mediante o uso de um software próprio que leem as informações na tela do sistema.

4 Cabines de Votação Registradas

Outra funcionalidade suportada pelo SIGEleição é o registro de Cabine de Votação. Uma cabine de votação registrada funciona de maneira similar às urnas eletrônicas utilizadas pelo TSE para realizar as votações para cargos políticos no Brasil, e é composta por um computador de acesso restrito e controlado que acessa o servidor do SIGEleição utilizando um endereço IP fixo e previamente cadastrado no sistema.

O controle de acesso a cabine ocorre pelo "mesário" escolhido pela comissão eleitoral, que só permite um eleitor votar por vez mediante a apresentação de um documento oficial de identificação com foto. Neste caso, o usuário necessita apenas se autenticar na tela de *login* do sistema, sem precisar responder as perguntas de segurança, uma vez que as mesmas não são apresentadas dentro de uma cabine registrada.

Por meio dessa funcionalidade é permitido que eleitores que tenham sido bloqueados, por terem errado as perguntas de segurança em cabines não registradas, realizem o seu voto.

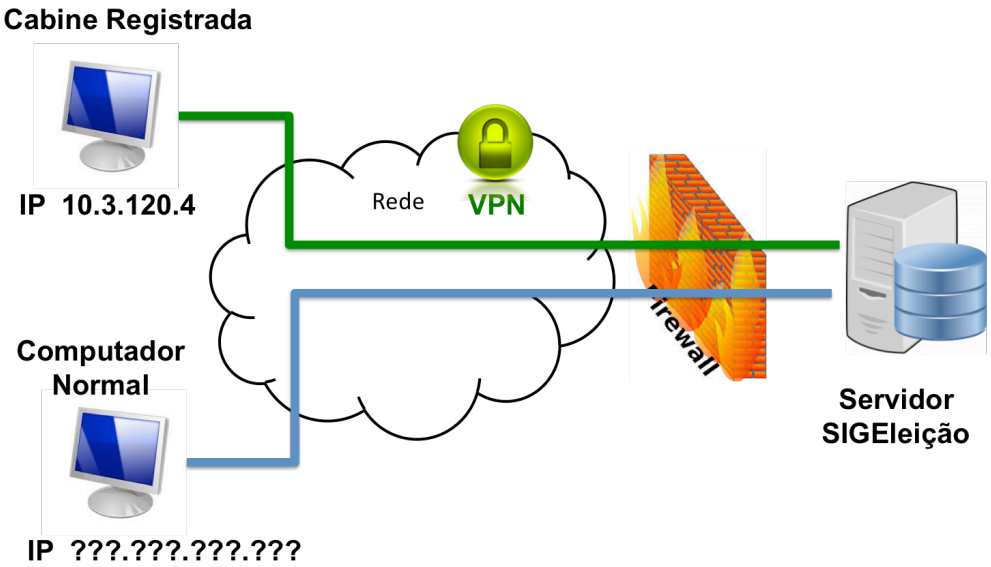


Figura 7. Infraestrutura para o suporte de cabines registradas no SIGEleição

Como o SIGEleição identifica as cabines de votação seguras através do endereço IP, é importante impedir que uma máquina não autorizada utilize algum dos endereços cadastrados como sendo dessas cabines, ou se passe por ele, através de uma técnica conhecida como IP Spoofing [TANASE, 2003]. Para isso, o protocolo IPSec [DORASWAMY, 2003] é empregado para criar uma VPN [FU, Zhi et al, 2001] entre a cabine segura e um firewall que protege o sistema, como pode ser observado na Figura 7.

Desse modo, pacotes nos quais o endereço IP pertença a lista dos endereços IP cadastrados como sendo de cabines seguras, só chegarão ao servidor do SIGEleição se realmente tiverem sido gerados por elas, caso contrário, serão descartados pelo firewall.

O firewall também implementa outras medidas de segurança, como, por exemplo, limitar o número de requisições por segundo vindas de uma mesma máquina, como proteção contra ataques de DOS [INFO WESTER 2015] aos servidores do sistema.

O sistema SIGEleição pode se configurado para só aceitar registro de votos de cabines registradas para uma determinada eleição. Assim a comissão eleitoral pode optar em utilizar o sistema para realizar uma eleição mais restrita, em que o usuário não consiga votar de qualquer computador.

5. Avaliação do SIGEleição

Nos aprimoramentos realizados na versão 2.0 do SIGEleição buscou-se otimizar o sistema para suporta a demanda de mais de 30.000 eleitores aptos a votar na eleição de Reitor da UFRN. Para isso foram realizados vários testes de *stress* do sistema com a ferramenta JMeter [APACHE JMETER 2015] conseguindo-se uma carga de 2.000 eleitores por minuto registrando votos em uma eleição. Esse número foi mais do que suficiente para atender ao universo de 30.000 eleitores esperados.

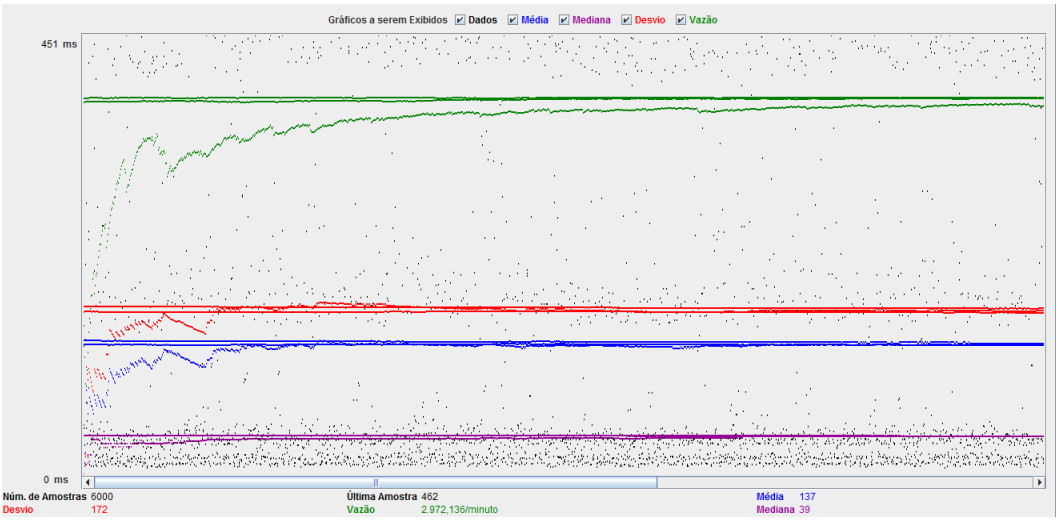


Figura 8. Testes de Performance do SIGEleição com o JMeter

A Figura 8 mostra a saída de um dos testes de performance a que o SIGEleição foi submetido. Nela é mostrado o tempo de resposta das requisições HTTP realizadas desde de a tela de *login* até o registro de voto, ao passar do tempo. O objeto desse testes

era mediar a vazão do sistema em relação a quantidade de eleitores tentando registrar votos em uma eleição.

Desde da implantação da versão 2.0 com as melhorias na segurança descrita nesse artigo, já ocorreram 162 eleições, com 36.473 votos registrados, 20.931 eleitores diferentes já acessaram o sistema e registraram os seus votos, em 266 chapas.

Até o momento, ocorreu um problema apenas em 1 eleição. Por um bug no mecanismo de *lock* no momento do cálculo do hash da quantidade de votos para a eleição. A quantidade ficou inválida, o que foi imediatamente identificado pelo mecanismo de auditoria baseado em hashes, interrompendo e invalidando a eleição. Era uma falha no sistema, mas comprovou que qualquer erro ou alteração das informações auditadas de uma eleição são imediatamente detectadas pelo sistema, não se permitindo que continue a votação. Essa falha já foi corrigida na versão atual do sistema.

6. Conclusões

O SIGEleição conseguiu atender aos critérios de confiabilidade e segurança para ser utilizado por uma instituição de ensino superior como a UFRN no pleito para a escolha dos representantes de diversos órgãos da universidade.

A versão atual conseguiu atender a maioria dos requisitos de segurança desejados para tais sistemas, como sigilo do voto, auditoria dos resultados por meio de uma chave de segurança que impede a alteração de votos na base de dados, vários mecanismos de autenticação para acessar o sistema, performance para atendar a demanda de registro de votos e acessibilidade para registar os votos.

Vários grupos de interesse têm se formado para debater o tema. Os proponentes do voto *on-line* acreditam que a nova tecnologia aumentará a participação dos eleitores, permitindo que o eleitorado obtenha maior eficiência e agilidade no pleito. Assim, torna-se mais disponível o acesso ao processo democrático.

Críticos da eleição pela *on-line* alegam que a tecnologia necessária para a devida autenticação dos eleitores e para garantir precisão e integridade ao sistema eleitoral não existe ou não está suficientemente difundida em nossa sociedade. Eles alegam também que a exclusão digital inclinaria ainda mais o poder político na direção das maiorias privilegiadas; e que as empresas privadas não são autoridades confiáveis na administração das eleições do setor público.

Existe ainda a questão de influência política no voto. Como em sistemas *on-line,* como o SIGEleição, o eleitor pode votar de qualquer computador com acesso à Internet, a tecnologia não tem como garantir que o eleitor não sofreu uma coação política no momento do voto. Considerando que não se use a funcionalidade, presente no SIGEleição, de permitir votos apenas em cabines registradas.

Ademais sistemas como o SIGEleição podem futuramente abrir um caminho para uma nova forma de democracia, mais justa e ampla do que a democracia representativa que temos hoje. Na qual o eleitor pode ser consultado diretamente sobre assuntos de importância para um país.

Referências

APACHE JMETER. Disponível em: http://jmeter.apache.org/. Último acesso em 23/05/2015

CAPTCHA. Completely Automated Public Turing Test To Tell Computers and Humans Apart. Disponível em: http://www.captcha.net/ Último acesso em 23/05/2015

CHAVES, Shirlei A., MELLO, Emerson R. O uso de um sistema de votação o on-line para escolha do conselho universitário. Anais do XIV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais — SBSeg 2014.

CRANOR, L. Design and Implementation of a Practical Security-Conscious Electronic Polling System. Department of Computer Science, Washington Universit, 1996.

DORASWAMY, Naganand; HARKINS, Dan. IPSec: the new security standard for the Internet, intranets, and virtual private networks. Prentice Hall Professional, 2003.

SSLSocket. Disponível em: https://docs.oracle.com/javase/7/docs/api/javax/net/ssl/SSLSocket.html. Último acesso em 08/04/2016

FU, Zhi et al. IPSec/VPN security policy: Correctness, conflict detection, and resolution. In: Policies for Distributed Systems and Networks. Springer Berlin Heidelberg, 2001. p. 39-56.

GAT UFRJ. Função Hashing. Disponível em: http://www.gta.ufrj.br/grad/09_1/versao-final/assinatura/hash.htm. Último acesso em 23/05/2015

INFO WESTER. Ataques DoS (Denial of Service) e DDoS (Distributed DoS) Disponível em: http://www.infowester.com/ddos.php. Último acesso em 26/05/2015

MONTEIRO, Américo; SOARES, Natércia; OLIVEIRA, Rosa M.; ANTUNES, Pedro. Sistemas Electrónicos de Votação. Lisboa: Faculdade de Ciências da Universidade de Lisboa, 2001.

POST, U. Online voting. In POSTNOTE, number 155. UK Parliamentary Office of Science and Technology. 2001.

SUPERINTENDÊNCIA de informática. , 2015. Disponível em: <http://https://info.ufrn.br/html/>. Acesso em: 29 jun. 2015

TANASE, Matthew. IP spoofing: an introduction. Security Focus, v. 11, 2003.

UFRN escolhe novo reitor por voto eletrônico nesta terça-feira, 11 Disponível em: http://www.sistemas.ufrn.br/portal/PT/noticia/14002551#.VWEhzee7zaY. Último acesso em 23/05/2015