
# TFC – Votação Electrónica

# 2001-2002

**TFC número: 147**

## Relatório do Trabalho Final de Curso

## Regime de Mestrado Integrado

**Licenciatura em Engenharia Informática e de Computadores**
**Departamento de Engenharia Informática**
**Instituto Superior Técnico**

**Orientador: Prof. André Zúquete**

---

**Co-Orientador: Prof. Paulo Ferreira**

---

**Rui F. L. Joaquim**

---

***LEIC-PSI, IST (44065)***
***GSD – INESC-ID***
***rjoaquim@gsd.inesc-id.pt***

**Trabalho realizado sob a orientação do**
**Prof. Doutor André Ventura da Cruz Marnôto Zúquete**

**Professor Auxiliar do Departamento de**
**Engenharia Informática do**
**Instituto Superior Técnico**

**e co-orientação do**
**Prof. Doutor Paulo Jorge Pires Ferreira**

**Professor Auxiliar do Departamento de**
**Engenharia Informática do**
**Instituto Superior Técnico**

# Resumo

Hoje em dia os Sistemas de Votação Electrónica (SVE) estão cada vez mais em voga. Um pouco por toda a parte multiplicam-se experiências com vista a ganhar a confiança necessária para a utilização dos SVE em novas e mais importantes eleições. Aqui encontra-se uma análise aos diversos tipos de SVE para utilização em redes de grande escala como a Internet. Por fim é apresentado um SVE que pretende responder a algumas necessidades ainda existentes nomeadamente a tolerância a faltas.

**Palavras-chave:** Sistema de Votação Electrónica (SVE), tolerância a faltas.

Aos meus pais

**Vicente Maria Joaquim e Olívia Lopes Pires**

À minha namorada

**Rosa Henriques**

# Agradecimentos

Ao Prof. Doutor André Zúquete pela exemplar orientação ao longo de todo o TFC. Agradeço também as excelentes discussões e revisões que contribuíram de forma decisiva para este trabalho.

Ao Prof. Doutor Paulo Ferreira pela orientação, ideias e revisões ao longo da elaboração deste TFC.

A todos os colegas da sala pelo bom ambiente de camaradagem, pelas ideias trocadas e ajuda nos momentos mais difíceis.

A todos os colegas de curso pelo apoio prestado ao longo de todos estes anos, nomeadamente ao Hugo Pais, Ivo Brandão, José Reis, Luís Sousa, Nuno Fernandes, Paulo Grave, Pedro Cardoso, Rui Martins, Rui Luís.

A todo o Grupo de Sistemas Distribuídos pelo apoio disponibilizado ao longo deste trabalho.

Ao INESC-ID pela cedência das infra-estruturas sem as quais a realização deste trabalho não teria sido possível.

Por fim, mas não por último, um agradecimento muito especial aos meus pais e namorada que sempre acreditaram nas minhas capacidades e me apoiaram nos momentos mais difíceis.

# Índice

# Índice de Figuras

# Índice Tabelas

# 1 Introdução

O primeiro Sistema de Votação Electrónico (SVE) foi proposto em 1869 [1] por Thomas A. Edison, a sua primeira patente. O sistema consistia num dispositivo electromecânico para os congressistas votarem.

Foram precisos cerca de vinte anos para que surgisse um sistema para uso público. Jacob H. Myers desenvolveu um sistema mecânico que foi usado pela primeira vez em 15 de Abril de 1892 em Lockport, New York.

Hoje em dia designa-se SVE a qualquer sistema de votação que utilize meios electrónicos nas fases de votação ou contagens dos votos. Estão a ser cada vez mais utilizados sistemas como urnas electrónicas e leitores ópticos de boletins para facilitar a contagem dos votos. Nas eleições do partido democrata, em 11 de Março de 2000, para as presidenciais primárias dos Estados unidos foi, primeira vez, utilizada a votação pela Internet com carácter vinculativo.

As experiências com SVE pela Internet estão a surgir como forma de permitir uma maior mobilidade por parte dos eleitores, procurando-se por este meio atacar o problema da abstenção. Entre os casos mais recentes de eleições encontramos a eleição para a primeira associação europeia de estudantes e as eleições locais em Sheffield e Liverpool, Inglaterra. No caso desta última também foram feitos testes de votação por SMS e televisão digital, bem como através de outros dispositivos electrónicos nas assembleias de voto.

Neste trabalho são analisadas as abordagens mais utilizadas pela comunidade científica, apontando os defeitos e virtudes de cada uma delas.

Um SVE é por natureza uma aplicação distribuída, logo deve-se ter em conta a tolerância a faltas. O problema de várias propostas científicas de SVE é a falta de implementações das mesmas. Isto leva a que aspectos como o da tolerância a faltas sejam tratados como aspectos secundários. Este trabalho, além de fazer uma análise aos SVE existentes, propõe um SVE tolerante a faltas e bastante configurável.

No segundo capítulo serão analisados os problemas e requisitos dos SVE, bem como os problemas existentes e a contribuição deste TFC na resolução desse problema. De seguida, no terceiro capítulo é apresentado o trabalho que se tem feito nesta área. O quarto capítulo apresenta a arquitectura proposta e o quinto trata da sua realização prática.

# 2 Problemas/Requisitos

Neste capítulo, em primeiro lugar, é feita análise aos aspectos sociais levantados pela utilização de SVE. De seguida são apresentados os requisitos que devem ser respeitados por um SVE. Segue-se uma comparação, no que respeita ao funcionamento nas diversas fase de um votação, entre um sistema de votação normal em papel e um SVE. Por último é feito um levantamento dos problemas existentes, e apresentadas as contribuições deste trabalho na sua resolução.

## 2.1 Aspectos Sociais

Embora este trabalho não tenha como objectivo ser um estudo social sobre SVE, é necessário saber o que os eleitores esperam pois estamos a falar de um aspecto fundamental para o bom funcionamento das democracias existentes.

Numa síntese das necessidades/obrigações dos eleitores Michael Shamos [2] definiu seis mandamentos:

**I. Que a escolha de cada eleitor seja mantida como um segredo inviolável**

**II. Que cada eleitor elegível possa votar uma só vez e só nas eleições para as quais está autorizado a votar.**

**III. Não tentarás corromper o Sistema nem vendeis vosso voto**

**IV. Que todos os votos sejam contados correctamente**

**V. O Sistema deve estar disponível durante toda a eleição**

**VI. Que seja possível detectar ataques contra os Mandamentos II-IV, mas que não violem o Mandamento I.**

Não é simples respeitar todos estes mandamentos num meio como o da Internet, descrito por Ford e Baum (1997) como sendo:

> *"...uma entidade extraterritorial, não controlada nem controlável por qualquer governo ou organização, mas em vez disso, opera exclusivamente numa base de mútua cooperação. A Internet pode melhor ser descrita como caos controlado."*

Olhando para o sistema eleitoral tradicional, em papel, chegamos facilmente à conclusão que não é perfeito, mas toleramos que possam existir erros na contagem dos votos, mais um menos um voto é indiferente se a diferença entre candidatos for na ordem das centenas. Toleramos também que existam pessoas capazes de venderem os seus votos ou, mais grave, que possa existir coacção por parte de candidatos menos honestos, mas continuamos a acreditar que o número de votos afectados por este processo menos lícito é insignificante, com a excepção de alguns casos pontuais (como são as eleições na Sicília, Itália).

O mesmo princípio rege os SVE pois a sua utilização em ambientes abertos, como a Internet, trás a vantagem da mobilidade e o inconveniente de permitir a coacção pessoal, pois nada impede quem coage de estar presente na altura em que o eleitor vota. Assim é natural que os SVE façam compromissos entre as qualidades desejáveis de um processo eleitoral e a adaptabilidade às características do meio onde irão funcionar.

## 2.2 Requisitos de Sistemas de Votação Electrónica

Tendo em conta tudo o que foi dito anteriormente vamos agora apresentar os requisitos que os autores dos diversos SVE escolheram para os mesmos.

Começamos pelos quatro que todos os autores referem como essenciais:

**Exactidão:** Um SVE diz-se exacto se (1) não é possível que um voto válido seja alterado, (2) não é possível que um voto válido seja eliminado da contagem, e (3) não é possível que um voto inválido faça parte do resultado final.

Um SVE é **totalmente exacto** se não for possível introduzir incorrecções ou se estas poderem ser corrigidas, e **parcialmente exacto** se as incorrecções poderem ser detectadas mas não necessariamente corrigidas.

**Democracia:** Um SVE diz-se democrático se (1) só permite que eleitores elegíveis votem e (2) cada eleitor só pode votar uma vez.

**Privacidade:** Um SVE diz-se privado se (1) nenhuma autoridade de voto nem ninguém consegue estabelecer ligação entre um boletim de voto e o eleitor que nele exerceu o seu direito de voto e (2) ninguém pode provar que votou de uma determinada maneira.

A segunda propriedade é indispensável para se conseguir a não coacção ou compra de votos. Algumas pessoas defendem que se temos o direito de escolher então também deveríamos poder vender o nosso voto. A verdade é que a maioria das pessoas não pensa assim e enveredando por esse caminho abrir-se-ia a porta à coacção, o que é de todo indesejável.

**Verificabilidade:** Um SVE diz-se verificável se todos podem, independentemente, verificar que todos os votos foram contados correctamente.

É usual encontrar duas definições de verificabilidade, a universal e a individual, em que por **verificabilidade universal** entende-se que se todos podem verificar que a contagem dos votos está correcta mas não é necessariamente possível verificar cada voto individualmente; por **verificabilidade individual** entende-se que cada votante possa verificar se o seu voto está correcto e efectuar a contagem dos restantes votos.

Nos sistemas actuais em papel qualquer pessoa entende o seu funcionamento. Por isso o requisito de verificabilidade não se impõe como ao nível dos SVE, em que se utilizam técnicas conhecidas apenas por um punhado de indivíduos. Neste cenário é necessário que os eleitores consigam obter uma prova do seu voto, para que possam confiar num sistema que não entendem totalmente. Com esta necessidade dos eleitores surge o problema da coacção nos SVE, abordado na secção de problemas existentes.

A estes quatro requisitos foram adicionados mais três por Lorrie F. Cranor e Ron K. Cytron [3].

**Conveniência:** Um SVE diz-se conveniente se o votante puder exercer o seu direito de voto rapidamente numa só sessão e com o mínimo de equipamento e competências.

**Flexibilidade:** Um SVE diz-se flexível se não restringir o formato dos boletins de voto.

Alguns SVE para poderem respeitar melhor outros requisitos, restringem o voto a um simples bit (Sim/Não).

**Mobilidade:** Um SVE diz-se móvel se só impuser restrições de natureza logística ao local onde cada eleitor pode exercer o seu direito.

Nos últimos tempos grande parte da comunidade científica que trabalha nesta área tem procurado desenhar SVE em que o eleitor não recebe nenhum recibo por parte do SVE a provar que votou. Esta propriedade é designada na literatura por **Receipt-free**. O desenho de tais SVE trás o problema de conciliar a propriedade de **Verificabilidade** com a de **Receipt-free**.

Num estudo feito por Pedro Antunes [4], em que se analisam vários SVE (a maioria utilizando suportes físicos) actualmente em funcionamento, é apresentada uma lista de dezoito outros requisitos. Muitos destes requisitos surgem das lições que se aprendem

na implementação e utilização dos diversos SVE, como por exemplo **Custo/Beneficio, Autenticação do operador, Integridade do SVE,…**
Mais uma vez é relevante sublinhar o facto de que para se obter um bom SVE deve-se ter em conta erros e falhas de SVE anteriores.

## 2.3 Fases de uma Votação

Uma votação normalmente tem as fases de **preparação, autenticação, votação, recolha de votos** e por fim a fase de **contagem** dos votos. A Tabela 1 descreve brevemente o que acontece em cada uma dessas fases. Os dados na tabela procuram pôr em evidencia as diferenças entre votações tradicionais e por SVE. São também apresentados diversos problemas específicos de cada uma das fases.

| Fases\Método | Tradicional | SVE | Problemas |
|---|---|---|---|
| **Preparação** | Registo nos cadernos eleitorais / elaboração dos boletins de voto | O tradicional mais a geração e distribuição de senhas | Confidencialidade de senhas |
| **Autenticação** | Apresentação do cartão de eleitor e BI | *Username* e senha | Personificação |
| **Votação** | Escolha num boletim de papel | Escolha num boletim electrónico | Erro no voto |
| **Recolha de votos** | Depósito dos votos na urna | Envio electrónico do voto | Recusa, eliminação ou substituição de votos |
| **Contagem** | Contagem manual voto a voto | Contagem electrónica dos votos (voto a voto ou resultados) | Eliminação ou introdução de votos; Fuga de resultados parciais |

**Tabela 1 – Fases de uma Votação**

## 2.4 Problemas Existentes nos SVE Actuais

### 2.4.1 Não Coacção/Verificabilidade

Actualmente vários autores, como Berry Schoenmakers em [5], defendem que a única solução para este problema é o recurso a SVE baseados em cifra homomórfica, uma vez que esta técnica permite decifrar o resultado da eleição sem decifrar os votos individuais. No entanto tais SVE trazem consigo algumas limitações derivadas da técnica utilizada, como o enorme tráfego gerado, o que impede actualmente a sua utilização em eleições de grande escala. Este tipo de SVE será abordado mais em detalhe no próximo capítulo.

### 2.4.2 Escalabilidade/Distribuição

Se existe uma ideia aceite por todos os que trabalham ou pensam neste assunto, ela é a de que um SVE para uma utilização geral tem de suportar eleições de grande escala.

Normalmente os investigadores concebem os seus SVE para serem eficientes e respeitarem o melhor possível todos os requisitos anteriormente descritos, assumindo que para eleições em larga escala não existirá só uma célula do sistema a funcionar mas sim várias com uma distribuição semelhante à que existe hoje nas eleições, i.e. por distritos ou concelhos. Riera, em [6], propôs uma realização de tal sistema com base em X.500.

A distribuição trás consigo um enorme ganho na robustez no sistema. Se houver um problema local como a corrupção de uma célula do sistema por qualquer entidade que leve à anulação da eleição, só a parte afectada é que tem de repetir o processo eleitoral. Tal é similar ao que acontece hoje em dia quando uma determinada população decide boicotar as eleições; só a mesma é afectada e não toda a população do país.

### 2.4.3 Transaccionalidade/Tolerância a Faltas

Outro problema, que ainda não foi suficientemente trabalhado, é o da transaccionalidade/tolerância a faltas. Em praticamente todos os SVE assume-se que o eleitor consegue realizar todo o processo de votação sem interrupções. Não se prevê que exista uma falha no equipamento terminal do eleitor ou na ligação entre este e as autoridades de voto no período de tempo que vai da autenticação ao depósito do voto. Normalmente a concretização do requisito de democracia é feita na autenticação, permitindo ao eleitor apenas uma autenticação válida. Após a mesma o eleitor ou deposita o voto ou, se o perder, este fica irremediavelmente perdido.
Este aspecto apenas foi abordado por M. Gonçalves e M. Ramos em [7].

### 2.4.4 Falta de Implementações

Apesar de já existir um número considerável de propostas para SVE, ainda existe uma grande falta de implementações. Existem empresas que têm produtos neste sector, mas só algumas planeiam distribui-los em open-source, como é o caso da empresa SAFEVOTE que se encontra a patentear o seu sistema. Existem algumas implementações a nível universitário, mas poucas, sendo as mais conhecidas o SENSUS de Lorrie F. Cranor e Ron K. Cytron, e o E-VOX de Mark Herschberg [8] que está em uso nas eleições da associação de estudantes do MIT. A GNU disponibiliza uma implementação open-source em java do seu SVE.

## 2.5 Contribuições

Este trabalho aborda o aspecto da tolerância a faltas, algo que tem sido um pouco olvidado nos vários SVE até hoje propostos.

Assim o SVE aqui proposto permite recuperar de uma falha no equipamento onde o eleitor está a votar. Assim se o eleitor não conseguir votar numa máquina devido a falha da mesma ou falha da rede pode sempre esperar até que se resolva a falha ou deslocar-se a outra máquina e retomar o processo que tinha sido interrompido.

É também um SVE altamente configurável. Podendo-se configurar o número de módulos do sistema face às restrições de segurança de cada eleição, e suportar várias eleições em simultâneo.

# 3 Trabalho Relacionado

Este capítulo pretende dar uma abordagem ao muito trabalho efectuado na área. Em primeiro lugar é apresentado um conjunto significativo de técnicas usadas na concepção de SVE. Seguidamente são introduzidos, com algumas referências específicas, os vários tipos de SVE existentes.

## 3.1 Técnicas mais Utilizadas em SVE

### 3.1.1 Cifra simétrica

A criptografia é uma técnica já muito antiga que consiste em alterar uma mensagem de modo a que só o emissor e o receptor consigam entender o seu conteúdo. Para atingir o objectivo é criada uma ou várias chaves que não são mais que um mapa das alterações a fazer para cifrar e decifrar as mensagens. Quando a mesma chave é utilizada para cifrar e decifrar a cifra diz-se simétrica.

$$C_k(M) = M'$$
$$D_k(M') = M$$

$C_k$ é a função de cifra chaveada com a chave $K$.
$D_k$ é a função de decifra chaveada com a chave $K$.
$M$ é a mensagem em claro
$M'$ é a mensagem cifrada que resulta da aplicação da função de cifra $C_k$ a $M$.

Aqui estão alguns dos mais conhecidos algoritmos de cifras simétricas: DES, IDEA, AES, Blowfish, RC4, RC5, A5.

Normalmente esta técnica é utilizada em SVE para cifrar a comunicação entre os vários elementos. Com este procedimento consegue-se estabelecer canais de comunicação seguros sobre redes inseguras.

### 3.1.2 Cifra assimétrica

Quando se utiliza uma chave para cifrar e outra diferente para decifrar a cifra diz-se assimétrica. Uma das chaves é dita pública e a outra é dita privada, o que é cifrado com a chave pública só pode ser recuperado com a chave privada e vice-versa.

$$C_k(M) = M' \quad \rightarrow \quad D_{k'}(M') = M$$
$$C_{k'}(M) = M'' \quad \rightarrow \quad D_k(M'') = M$$

***K*** é a chave pública

***K'*** é a chave privada

$C_k$ é a função de cifra chaveada com a chave ***K***.

$C_{k'}$ é a função de cifra chaveada com a chave ***K'***.

$D_k$ é a função de decifra chaveada com a chave ***K***.

$D_{k'}$ é a função de decifra chaveada com a chave ***K'***.

***M*** é a mensagem em claro

***M'*** é a mensagem cifrada que resulta da aplicação da função de cifra $C_k$ a ***M***.

***M''*** é a mensagem cifrada que resulta da aplicação da função de cifra $C_{k'}$ a ***M***.

Existem entidades, denominadas PKI, que gerem este tipo de chaves. Se uma entidade **X** requerer um par de chaves, estas ser-lhe-ão dadas, e a PKI emissora das chaves compromete-se a publicar num local público a chave pública e a guardar segredo da chave privada. Assim sempre que alguém queira contactar **X** com a certeza que só **X** vai entender a mensagem basta obter a chave pública de **X**, previamente afixada num local público, e enviar a mensagem cifrada pela chave que obteve.

Este método de cifra é mais pesado do que a cifra simétrica, logo não é o ideal quando se pretende trocar uma série de mensagens entre dois interlocutores. Normalmente, ao estabelecer-se um canal de comunicação segura, usa-se cifra assimétrica para combinar uma chave simétrica que depois é utilizada no restante diálogo, utilizando assim cifra simétrica que é muito mais eficiente.

Pode-se também fazer autenticação de mensagens recorrendo a cifra assimétrica. No caso de o emissor **X** da mensagem a assinar, cifrando-a com a sua chave privada,

todos os receptores que tenham em sua posse a chave pública de **X** podem decifrar a mensagem. Como só **X** tem acesso à sua chave privada só este a pode ter gerado, logo esta mensagem é autenticada como sido gerada por **X**.

Os algoritmos mais conhecidos de cifras assimétricas são o RSA e ElGamal.

No que respeita a SVE esta técnica é normalmente utilizada para garantir para a geração de canais de comunicação seguros entre as várias entidades. Existem também já muitas sugestões para utilizar a PKI como plataforma de base na autenticação dos eleitores.

### 3.1.3 Mix-Net

Quando um emissor **X** quer enviar uma mensagem anónima a um receptor **Y** de uma forma segura pode recorrer a estas construções.

O processo é simples, o emissor cifra sucessivas vezes a mensagem que quer enviar, depois envia essa mensagem cifrada para o primeiro mix. Este retira a primeira cifra e reordena várias mensagens que tenha recebido e envia-as para o próximo mix. O processo repete-se até a mensagem chegar ao receptor, Figura 1.

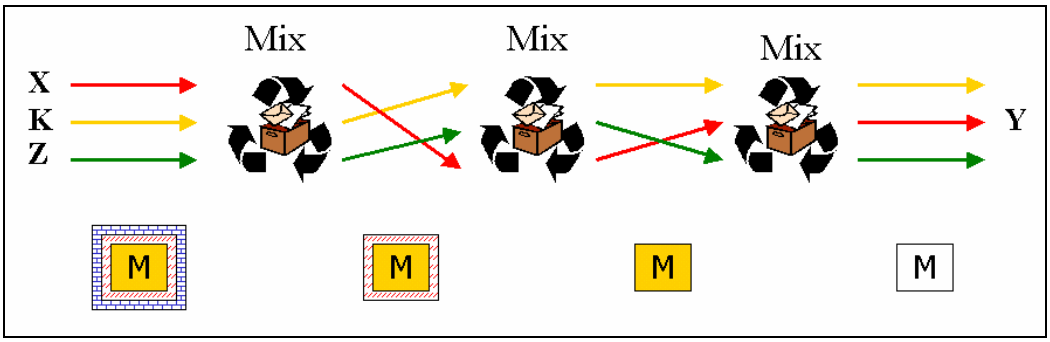


**Figura 1 – Mix-Net**

Estas construções são úteis a SVE para conseguirem a propriedade da privacidade, pois os interlocutores comunicam anonimamente. Normalmente este mecanismo é utilizado nos SVE para o depósito anónimo dos votos.

### 3.1.4 Canais anónimos

O conceito de canal anónimo foi introduzido por David Chaum [9] em 1981 e, como o nome indica, pretende-se um canal em que um emissor **X** possa enviar uma mensagem a um receptor **Y** sem que este consiga saber quem lhe enviou a mensagem mas ao mesmo tempo possa dar uma resposta.

Uma ideia para implementar um canal anónimo é utilizar uma mix-net alterada. Em vez de o emissor cifrar várias vezes a mensagem, este cifra-a só uma vez com a chave do primeiro elemento da mix-net, neste caso chamado anonymizer e envia-a para este. O mix que recebeu a mensagem decifra-a, e constrói uma nova mensagem juntando à mensagem original um identificador único. Esta nova mensagem é cifrada com a chave do próximo anonymizer e enviada para este. O processo é repetido até a mensagem chegar ao receptor, Figura 2.

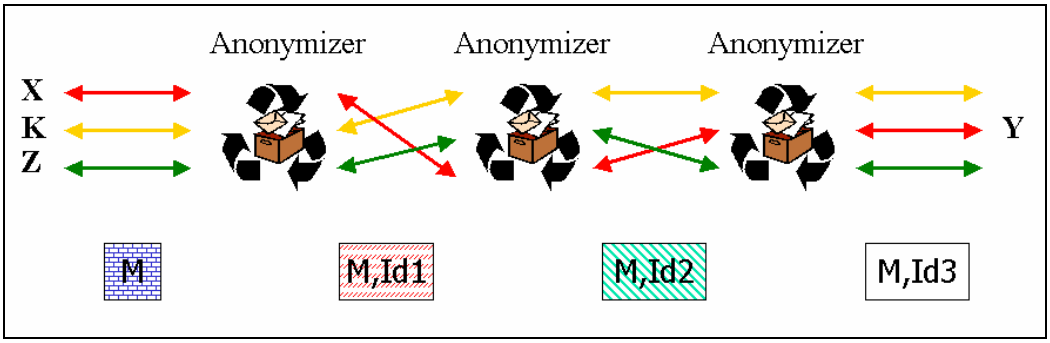


**Figura 2 – Canal anónimo baseado em Mix-Net**

O receptor então processa a mensagem e envia a resposta com o identificador que vinha com a mensagem. Depois o processo é invertido e assim consegue-se uma comunicação anónima e segura.

Como a Mix-Net, um canal anónimo pode ser utilizado para garantir a propriedade da privacidade. No entanto este mecanismo é mais poderoso que uma Mix-Net porque permite bi-direccionalidade. Com esta propriedade além de poder ser utilizado para depositar anonimamente os votos, pode também ser utilizado nas comunicações entre vários módulos eleitorais [7].

### 3.1.5 Assinaturas cegas

A técnica de assinatura cega permite que uma entidade **X** assine uma mensagem de **Y** sem saber o conteúdo da mensagem.

Informalmente o método consiste em **Y** enviar a **X** uma mensagem num envelope fechado, o que consiste em aplicar o factor de cegamento, o envelope está revestido no interior a papel químico, seguidamente **X** assina o envelope e a assinatura passa através do papel químico para a mensagem. **Y** ao receber de volta o envelope assinado retira a mensagem de dentro do mesmo, i.e. retira o cegamento, e pode verificar que a mensagem está assinada por **X**, Figura 3.

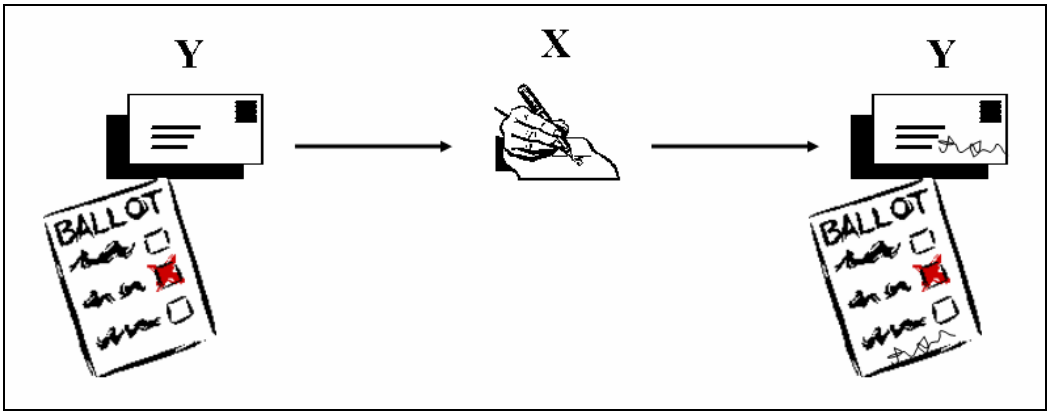


**Figura 3 – Assinaturas cegas**

O algoritmo que David Chaum apresentou é baseado no RSA e consiste em fazer com que o assinante assine um texto, cifrando-o com a sua chave privada, sem ter conhecimento do mesmo.

O Bob tem uma chave pública ***e***, uma privada ***d*** e o módulo público ***n***. A Alice quer que o Bob assine uma mensagem ***m***.

1. A Alice escolhe um factor de cegamento ***k*** entre 1 e ***n***. De seguida ela cega a mensagem envia-a para o Bob.

   $t = mk^e \bmod n$

2. O Bob assina a mensagem ***t*** e retorna-a à Alice.

   $t^d = (mk^e)^d \bmod n$

3. A Alice descega a mensagem $t^d$ calculando:

   $c = t^d/k \bmod n$

4. O resultado é a mensagem $m$ cifrada com a chave privada do Bob.

   $c = m^d \bmod n$

   O que se pode verificar facilmente.

   $t^d = (mk^e)^d \equiv m^d/k \pmod n$, então $t^d/k = m^dk/k \equiv m^d \pmod n$

Existe nos SVE uma entidade que controla o número de vezes que os eleitores votam (propriedade da democracia). Nos SVE baseados em assinaturas cegas é normal que a entidade que faz este controlo assine o voto do eleitor para depois se verificar que este é válido. Esta técnica permite a esses SVE assegurar a propriedade de democracia respeitando a privacidade.

### 3.1.6 Zero-Knowledge Proofs

Esta técnica é utilizada para a prova de que se conhece algo mas sem revelar esse algo. Um exemplo que se encontra geralmente na literatura é o da gruta de Ali Baba. O objectivo é o Ali Baba provar ao amigo João que conhece o segredo que lhe permite mover a pedra que bloqueia a gruta, Figura 4.

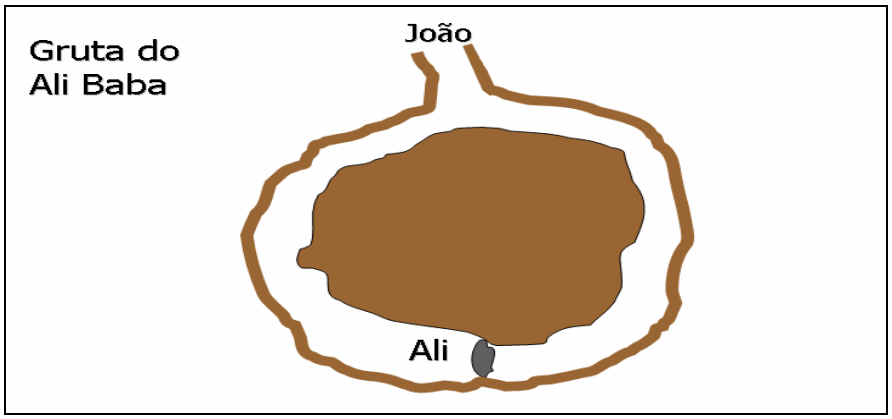


**Figura 4 – Gruta do Ali Baba**

O Ali Baba entra na gruta sem o João ver e escolhe um caminho, depois o João da entrada da gruta pede ao Ali Baba para sair por um dos túneis, esquerdo ou direito, se o Ali Baba sair por onde o João escolheu prova que conhece o segredo. Da primeira

vez a prova é parcial, isto é o João fica apenas com 50% de certeza que o Ali Baba conhece o segredo, mas podem repetir o processo até o João ficar convencido.

Estas provas servem para garantir a correcção de alguma confusão introduzida nos dados, como por exemplo a cifra com recifra aleatória. Os SVE garantem assim a propriedade de exactidão e verificabilidade.

### 3.1.7 Cifra de múltipla decifra (Threshold Decryption)

Este tipo de cifra consiste em arranjar uma chave pública para cifra e em vez de existir uma única chave privada existem várias, i.e. é gerada uma chave ***(t,n)*** em que ***t*** é o número mínimo de partes da chave necessários para a decifra, e ***n*** é o número de partes da chave que são geradas, assim uma mensagem cifrada pela chave pública só pode ser decifrada por quem possuir ***t*** partes da chave privada, Figura 5.

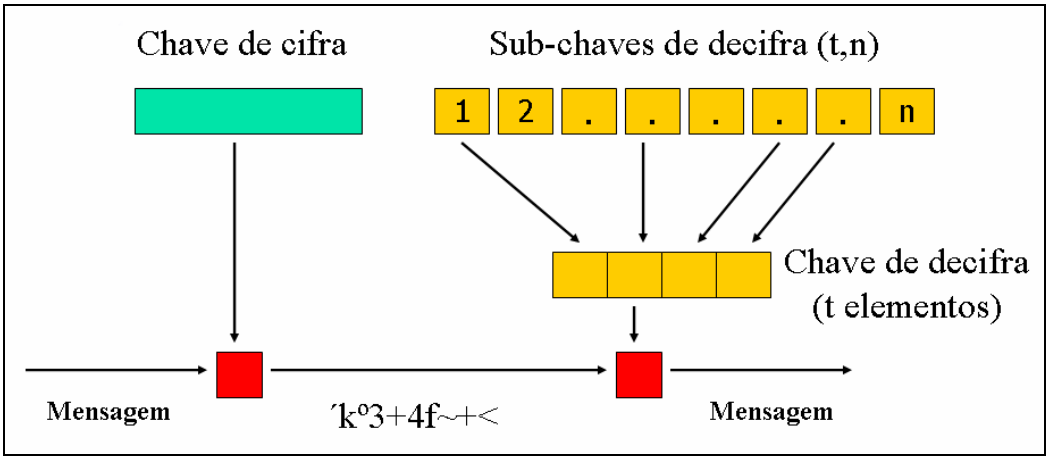


**Figura 5 – Cifra de múltipla decifra**

Normalmente este esquema é utilizado distribuindo as ***n*** partes da chave privada por ***n*** entidades distintas evitando assim que qualquer coligação de ***t*-1** entidades consiga quebrar a cifra e ter acesso à mensagem.

Os SVE que utilizam esta técnica conseguem obter um método que lhes permite verificar a correcção da contagem dos votos, visto que existem várias combinações possíveis de entidades para decifrar os votos. Introduz-se também uma melhor protecção contra fugas de resultados parciais pois o apuramento destes não depende só de uma entidade mas sim de ***t*** entidades.

### 3.1.8 Cifra com recifra aleatória

Considerando ***M'*** a cifra da mensagem ***M,*** esta técnica consiste em alterar ***M'*** para ***M''***, tal que ***M' ≠ M''*** mas verificando que a decifra de ***M''*** é ***M***, Figura 6.

$$C_k(M) = M' \quad \rightarrow \quad D_{k'}(M') = M$$
$$F(M') = M'' \quad \rightarrow \quad D_{k'}(M'') = M$$

***K*** é a chave pública

***K'*** é a chave privada

$C_k$ é a função de cifra chaveada com a chave ***K***.

$D_{k'}$ é a função de cifra chaveada com a chave ***K'***.

***F*** é a função de recifra.

***M*** é a mensagem em claro

***M'*** é a mensagem cifrada que resulta da aplicação da função de cifra $C_{k'}$ a ***M***.

***M''*** é a mensagem cifrada que resulta da aplicação da função ***F*** a ***M'***.

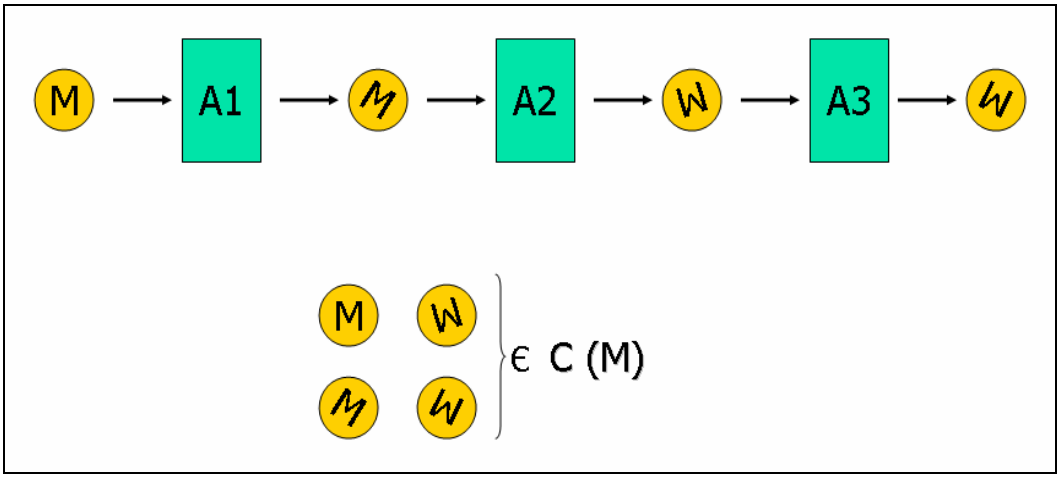


**Figura 6 – Cifra com recifra aleatória**

Esta técnica é utilizada em SVE para introduzir confusão no voto, alterando-o ao passar de entidade em entidade. Só o votante sabe qual é o voto certo pois cada entidade faz prova que a alteração foi correctamente efectuada, através de zero knowledge proofs. Normalmente nos SVE que utilizam esta técnica o eleitor vê o voto no início da cadeia de entidades que fazem a recifra e no final escolhe o seu voto. Só o eleitor sabe em quem está a votar pois só este teve acesso ás zero knowledge proofs.

Os SVE conseguem com esta técnica o garante das propriedades de verificabilidade e exactidão no processo eleitoral

### 3.1.9 Cifra homomórfica

Uma função ***f*** é dita homomórfica em relação à ⊗ (multiplicação) e ⊕ (adição) se para todo ***X*** e ***Y*** temos:

$$f(X) \otimes f(Y) = f(X \oplus Y).$$

As funções de cifra homomórfica, por exemplo a variação homomórfica da função de cifra ElGamal, podem ser usadas para revelar o resultado das eleições sem revelar cada voto em si, protegendo desta maneira o anonimato dos eleitores em relação ao seu voto.

O problema que surge na utilização deste tipo de cifra é a forte limitação ao formato dos boletins de voto, normalmente restringidos a votos Sim/Não. A limitação ao número de candidatos que um voto deste formato possui é ultrapassada normalmente recorrendo à utilização de um voto Sim/Não por candidato, mas nunca ultrapassando a desvantagem de só permitir votos de um só bit.

## 3.2 Abordagens ao Problema

Numa eleição tradicional, tal como num SVE é natural existir uma ou mais autoridades de voto. Assim, podem classificar os diversos SVE arquitecturalmente tendo em conta o número de autoridades de voto neles existentes. Existem SVE sem autoridades de voto em que os eleitores controlam-se uns aos outros, e no final todos contam os votos. Mas devido à complexidade e ao enorme tráfego gerado, este SVE não é viável para mais de meia dúzia de votantes. Encontram-se também SVE com uma única autoridade de voto, mas isso faz com que todo o poder esteja concentrado numa única entidade. O que não é desejável. Restam os SVE com múltiplas autoridades de voto que se controlam mutuamente e garantem uma maior confiança no SVE por parte dos eleitores.

É óbvio que os SVE mais interessantes tendo em vista uma utilização em eleições reais são os de múltiplas autoridades de voto. Segue-se uma análise tecnológica destes SVE que se dividem em SVE baseados em Mix-Nets, baseados em assinaturas cegas e canais anónimos e por fim os baseados em cifra homomórfica.

### 3.2.1 SVE baseados em Mix-Nets

Um SVE baseado em Mix-Nets, como se pode ver na Figura 7, tem normalmente três elementos, o Autenticador, a Mix-Net e um Contador.

Tudo começa no registo dos votantes. O votante começa por se autenticar perante o Autenticador, após uma autenticação bem sucedida o Autenticador dá um boletim de voto fechado ao votante. O boletim é único e é atribuído aleatoriamente a cada votante.

Quando a votação começa o Autenticador pública os segredos que permitem abrir os boletins. De seguida o votante vota e cifra **n** + 1 vezes o boletim, **n** é o número de elementos que compõem a Mix-Net. As cifras são feitas com as chaves públicas dos elementos da Mix-Net e com a chave pública do Contador. Resta ao votante depositar o voto no primeiro elemento da Mix-Net.

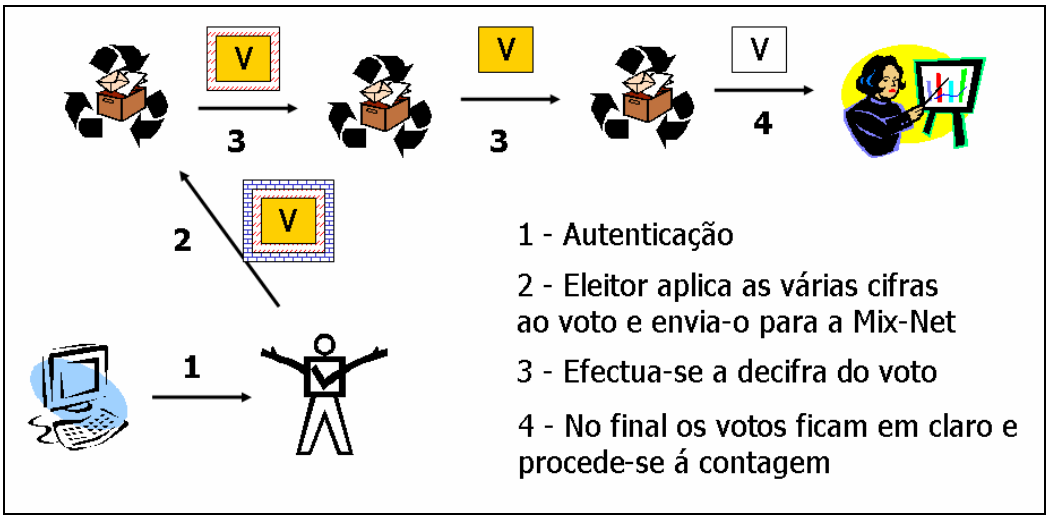


**Figura 7 – SVE baseado em Mix-Net**

Agora começa o processo de decifra e confusão efectuado pela Mix-Net. Chegando ao fim da Mix-Net o boletim apenas contém a cifra relativa ao Contador e já está bem baralhado com os restantes boletins.

Quando a votação termina o Contador recebe os boletins e efectua a última decifra dos mesmos. Por último é feita a contagem e afixação de resultados. Verificando a existência de boletins repetidos, verifica-se se alguém votou mais que uma vez.

Estes SVE produzem uma grande quantidade de informação, que é a sua principal desvantagem. A informação é produzida quando se faz a decifra e confusão em cada mix. Esta informação resulta das zero knowledge proofs utilizadas para provar que toda o processo efectuado no mix foi bem feito. Esta característica leva a que o processo de verificação da correcção de uma eleição seja muito pesado.

David Chaum propõe em [10] um SVE baseado em Mix-Nets. Park, Itoh e Kurosawa, em [11], defendem que o SVE proposto por Chaum não é justo. Dizem que se um voto for corrompido os restantes votos são sempre decifrados e publicados, logo a eleição não é justa para o votante que viu o seu voto corrompido. Assim propõem um esquema de divisão dos votos em duas partes. Na decifra dos votos cada parte é decifrada em separado, e só se todas as metades estiverem integras é que se juntam para formar o voto. Neste SVE basta um voto ser corrompido para que o resultado da eleição não seja apurado.

Um SVE receipt-free baseado em Mix-Nets foi proposto por Sako e Kilian em [12]. Neste SVE o processo de distribuição de votos foi alterado, passando a existir uma cadeia de entidades que participam na distribuição do boletim. O processo começa numa entidade que constrói dois votos para o votante, um sim e um não. É feito um bit-commitment aos boletins, sendo depois enviados para a entidade seguinte. Esta entidade baralha os votos e altera-os, fazendo com que a entidade seguinte não saiba qual dos votos é o sim e qual deles é o não. Todas estas alterações são mostradas ao votante. Estes passos repetem-se até se chegar ao fim da cadeia. Nessa altura o votante escolhe o voto que pretende e envia-o através de uma Mix-Net para o Contador.

O bit-commitment permite ao votante abrir o voto como quiser, iludindo quem o possa estar coagindo, mas o Contador só consegue abri-lo no sentido correcto.

Para conseguir a propriedade de receipt-free é necessário que o canal de comunicação, utilizado pelo votante na fase de escolha de voto, seja inviolável.

### 3.2.2 Canais Anónimos e Assinaturas Cegas

Estes SVE têm três elementos base, o Administrador, o Canal Anónimo e o Contador, Figura 8.

No início o votante assinala a sua opção no boletim de voto. De seguida o votante cega o boletim e envia-o para o administrador com a sua identificação.

O Administrador verifica a entidade do votante e verifica se este ainda não votou. Verificadas estas condições o Administrador assina o boletim e envia-o de volta para o votante.

O votante ao receber o boletim assinado retira o cegamento e envia a voto para o Contador através do Canal Anónimo.

Por último o Contador procede à contagem dos votos e à afixação dos resultados

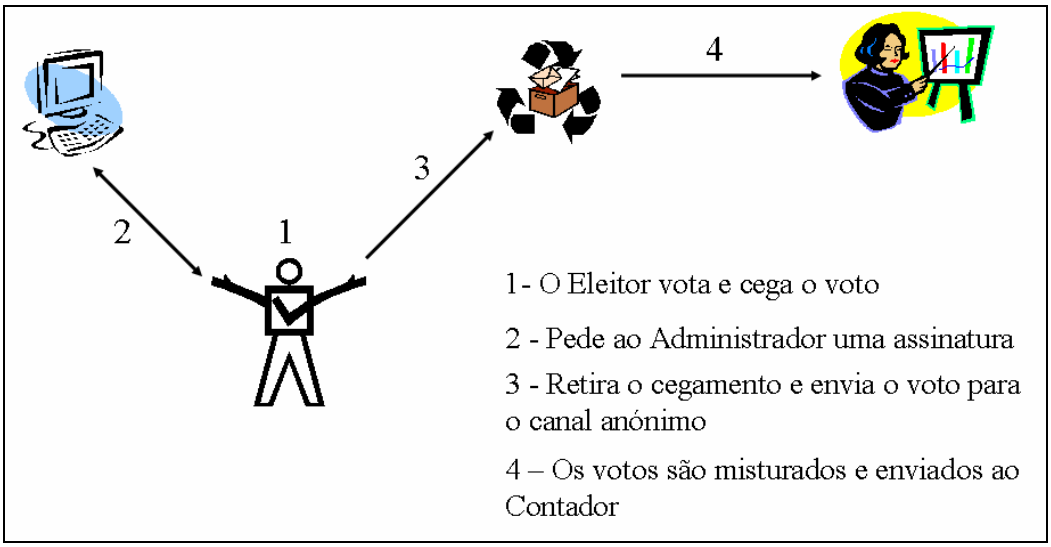


**Figura 8 – SVE baseado em assinaturas cegas e canais anónimos**

O SVE de referência nesta categoria foi proposto em 1992 por Fujioka, Okamoto e Otha [13], conhecido por protocolo de FOO. Neste SVE existem duas autoridades de voto. O Administrador que assina os votos, e o Contador efectua a contagem dos mesmos. O principal problema deste SVE é o facto de obrigar os eleitores no final da eleição a verificar se o seu voto chegou ao Contador, para enviarem a chave que permite ao Contador decifrar os votos.

Duas das mais conhecidas implementações de SVE baseados em assinaturas cegas baseiam-se no FOO. Uma delas, o SENSUS de Cranor e Cyntron [3], resolve o problema de FOO fazendo com que o Contador ao receber um voto emita um recibo. De seguida o eleitor ao receber o recibo envia a chave que vai permitir decifrar o voto, completando a acção do eleitor na eleição.

O outro SVE bastante conhecido baseado em FOO é o E-VOX, proposto por Herschberg [8]. A abordagem seguida neste SVE foi um pouco diferente, pois faz-se um bit-commitment com o voto antes deste ser assinado. Com este procedimento pode-se enviar o bit-commitment junto do voto para o Contador, poupando uma comunicação com o Contador. Assim, no final, o eleitor pode verificar se o seu voto foi contado verificando se o voto com o seu bit-commitment foi publicado.

Brandon DuRette em [14] propõe o E-VOX – Managed Administrator, uma versão mais robusta do E-VOX de Herschberg, que elimina a capacidade do Administrador de votar por eleitores que não o fizeram.

Os SVE, baseados em assinaturas cegas, que reclamam a propriedade receipt-free foram propostos por Okamoto em [15, 16]. Nestes SVE, na fase de preparação do boletim é feito uma trap-door. Esta trap-door permite ao Contador decifrar o voto e provar que este faz parte do conjunto de votos cifrados, através de uma zero knowledge proof. A trap-door ainda permite ao eleitor decifrar o seu voto do modo que quiser, iludindo deste modo quem o possa estar coagindo. Note-se que o Contador só consegue decifrar o voto num sentido e não nos dois como o eleitor.

Este SVE, para respeitar a propriedade receipt-free, pressupõe que existe um canal inviolável entre o votante e o Contador.

### 3.2.3 Cifra Homomórfica

Nestes SVE a ideia é a não revelação dos votos individuais garantindo no entanto que os resultados apresentados sejam correctos (verificabilidade universal), para isso utilizam-se as propriedades das funções homomórficas de modo que dos votos cifrados seja possível extrair o resultado final da eleição.

O processo de votação começa com a autenticação do utilizador perante um Administrador. Segue-se um processo interactivo entre o Administrador e o Votante através de um quadro público. Neste quadro o Administrador e cada votante têm o direito de escrever mas não de apagar.

No final do processo interactivo o votante colocou o seu voto cifrado com cifra homomórfica no quadro público e provou ao Administrador, através de zero knowledge proofs, que o voto é válido.

No final da eleição entra-se no processo de decifra do resultado. Primeiro é calculada a cifra da soma dos votos recorrendo ás propriedades da cifra homomórfica e feita prova da correcção da operação novamente com recurso a zero knowledge proofs. Na Figura 9 está representada a decifra do resultado final da eleição por várias entidades cooperantes, recorrendo à utilização usual de cifra homomórfica com propriedade de múltipla decifra.
Mais uma vez recorre-se a zero knowledge proofs para garantir que as entidades que decifraram o resultado o fizeram correctamente, isto acontece para não se divulgar a chave de decifra o que permitiria a decifra dos votos individuais, exactamente o que se pretende evitar neste tipo de SVE.

A principal desvantagem deste tipo de SVE é a usual limitação dos votos a votos do tipo Sim/Não, sendo esta limitação normalmente ultrapassada repetindo os passos mais pesados do SVE por cada candidato, o que seguramente não é desejável. Um problema desta solução, uma vez que nunca se decifra o voto, é como se impede o

eleitor de votar em mais que um candidato. Outra possível solução é recorrer a outras técnicas de cifra homomórfica que permitam múltiplas opções, como por exemplo a técnica de Paillier.

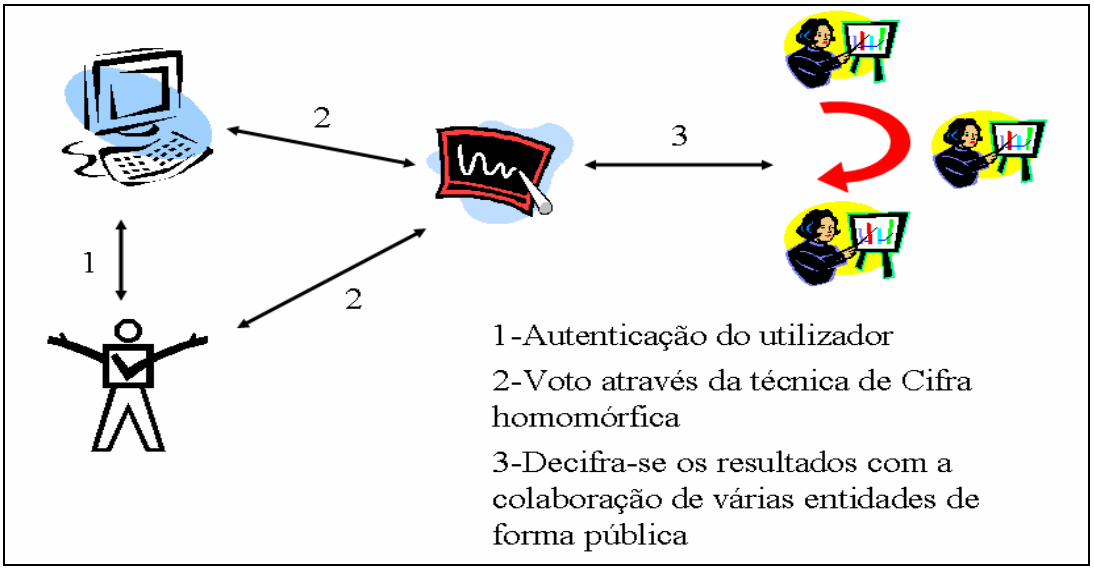


**Figura 9 – SVE baseado em cifra homomórfica**

É usual nestes SVE a existência de um canal público com memória (tipo um quadro onde todos podem escrever mas ninguém pode apagar) de modo a que seja possível provar que cada voto é válido.

Foi num SVE baseado em cifra homomórfica que pela primeira vez se introduziu a propriedade receipt-free. Benaloh e Tuinstra, em [17], apresentam um SVE receipt-free que faz uso de uma cabine de voto. Mais tarde, Hirt e Sako [18] provam que a proposta de Benaloh e Tuinstra não é receipt-free.

Cramer, Franklin, Schoenmakers e Yung apresentam, em [19], um SVE eficiente baseado em cifra homomórfica. A eficiência deste SVE advém de uma prova de validade eficiente para a cifra homomórfica. Neste SVE existe uma fase de preparação. Os boletins de voto são preparados pelo votante, dividindo-os pelas autoridades de voto que vão efectuar a decifra (Contador). Na fase de votação o votante só tem que enviar a parte do boletim que o torna num voto válido. Na fase de contagem cada Contador decifra a sua parte dos votos e depois todos juntos decifram

o resultado. A fase de preparação pode ser feita à priori, o que elimina bastante computação e comunicação durante a realização efectiva da eleição.

Cramer, Gennaro e Schoenmakers, em [20] propõem o protocolo, ainda mais eficiente que o anterior, que serve de base para o SVE receipt-free de Hirt e Sako [18]. Hirt e Sako no seu SVE utilizaram cifra de recifra aleatória para conseguirem a propriedade receipt-free. A grande alteração efectuada neste é que neste SVE o votante escolhe o voto em vez de o construir. O processo é feito em colaboração com várias entidades. Em primeiro lugar, todos os votos possíveis são construídos e feitos públicos. De seguida o votante entra em contacto com a primeira entidade e efectua o seguinte protocolo:

1. A entidade recifra os votos e baralha-os
2. A entidade fornece todas as provas ao votante para que este possa identificar os vários votos e verificar que a recifra foi bem feita
3. os votos são enviados para a entidade seguinte
4. volta-se a 1. até chegar à última entidade
5. O eleitor escolhe o voto e este é escrito num quadro público

No final os vários contadores produzem o resultado final da eleição com a informação que está no quadro público. Para que se garanta a propriedade receipt-free é necessário que o canal de comunicação utilizado pelo votante, na fase de escolha do voto, seja inviolável.

Em resposta ao problema de só existir a possibilidade de votos sim ou não, surgiu por Baudron, Fouque, Pointcheval, Poupard e Stern, em [21], um SVE que resolve esse problema. A resolução proposta assenta na substituição da técnica de cifra homomórfica utilizada. Os autores propõem um SVE baseado na técnica de cifra de Paillier em vez da que é normalmente utilizada, a técnica de ElGamal.

### 3.2.4 Análise comparativa dos SVE

Na Tabela 2, retirada de [22], podemos ver uma comparação dos vários tipos de SVE com múltiplas autoridades de voto. Os SVE baseados em assinaturas cegas, na sua

forma mais simples, têm uma enorme simplicidade. Esta simplicidade aliada ao facto de não restringirem o formato do voto torna-os SVE eficientes e flexíveis.

Os SVE baseados em Mix-Nets também não restringem o formato dos votos, mas em relação aos SVE baseados em assinaturas cegas são mais complexos, o que resulta numa maior carga computacional e comunicacional.

| Características\SVE | Homomórfico | Assinaturas cegas | Mix-Nets |
|---|---|---|---|
| **Estrutura matemática** | Muita | Pouca | Média |
| **Interacções votante – autoridades** | 1 | > 1 | ≥ 1 |
| **Contagem incremental** | Sim | Não | Não |
| **Verificabilidade** | Sim | ?? | ?? |
| **Custos por fase**<br>**Votação**<br>**Contagem**<br>**Verificação** | <br>Muito<br>Pouco<br>Pouco | <br>Pouco<br>Muito pouco<br>Apenas local | <br>Médio<br>Médio<br>Muito |

**Tabela 2 – Características de SVE**

Por último, os SVE baseados em cifra homomórfica são os que melhor protegem a privacidade do eleitor, pois não decifram os votos individuais mas sim o resultado final. O senão destes SVE é a sua grande complexidade matemática que restringe o formato dos votos, normalmente a respostas sim/não. O custo na fase de votação é também bastante elevado o que é claramente uma desvantagem.

Um dos problemas que todas as realizações de SVE têm de ter em conta é a autenticação dos votantes. Hoje em dia é usual encontrar sistemas com autenticação baseada em *username* e senha. Este sistema sobre de vários problemas como perdas da senha que impedem o eleitor de votar e ataques fáceis a senha fracas que permitem a personificação dos eleitores.

As PKI são boas possíveis soluções para o problema da autenticação. O grande problema com esta tecnologia é a sua fraca infiltração na comunidade de eleitores.

A propriedade receipt-free tem sido muito estudada nestes últimos anos. No entanto todos os SVE verificáveis que dizem que ter esta propriedade fazem recurso a dispositivos físicos, como canais de comunicação invioláveis e cabines de voto. Se estivermos a pensar em SVE que se utilizem a Internet, podemos facilmente chegar à conclusão que ainda não existem SVE receipt-free.

Uma possível solução para os dois problemas acima descritos é o recurso a dispositivos físicos, como smartcards [23]. A desvantagem do recurso a estes dispositivos é a necessidade de hardware especial que consiga lidar com os mesmos, e que normalmente não existe em casa dos eleitores.

# 4 Arquitectura

A arquitectura aqui apresentada tem como objectivos possibilitar um SVE bastante flexível e tolerante a faltas. Flexível na medida em que permite a realização de várias eleições simultâneas, a configuração personalizada de cada uma, e um formato flexível dos boletins de voto. Ao ser tolerante a faltas porque permite ao votante retomar o processo de votação a qualquer momento.

Existia a possibilidade de desenhar um SVE de início, ou aproveitar um já existente e expandi-lo de modo a respeitar os objectivos propostos. Após análise dos SVE existentes decidiu-se partir do E-VOX – Managed Administrators, proposto por Brandon DuRette [14]. A escolha deste SVE baseou-se no facto este já permitir alguma flexibilidade na configuração de eleições e no formato dos boletins de voto. Como foi proposto em 1999 já estão estudados os seus pontos fortes e fracos, o que facilita o desenvolvimento de um melhor SVE.

## 4.1 E-VOX – Managed Administrator

O E-VOX – Managed Administrator surge como evolução do SVE E-VOX, tese de mestrado de Mark Herschberg [8]. Este novo SVE tem como objectivo trazer uma maior robustez ao E-VOX original, Figura 10.

No E-VOX original o Administrator, por si só, podia introduzir votos pelos eleitores que não votaram. O risco de o Administrator ser apanhado era reduzido, uma vez que só era detectado se os eleitores que não votaram verificassem se alguém tinha votado em nome deles. E estar à espera que quem não votou tenha o incómodo de verificar que ninguém votou em seu nome é irrealista.

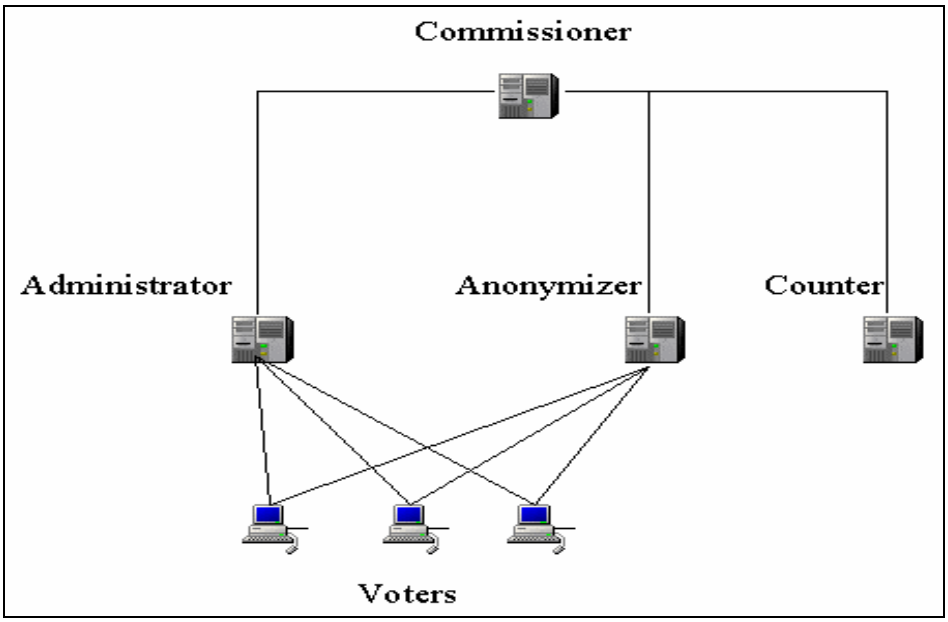


**Figura 10 – E-VOX original**

Na proposta de DuRette, Figura 11, em vez de um Administrator temos vários, e surge uma nova entidade o Manager. O procedimento da eleição pode-se dividir em várias fases, preparação, administração, anonimização, recolha e por último a fase contagem dos votos. O procedimento é o seguinte:

1. **Preparação**

   O eleitor contacta o Manager, identifica-se e recebe o boletim de voto.

2. **Administração**

   O eleitor após exprimir o seu voto no boletim envia-o cegado a um subconjunto ***t*** de Administrators. Cada Administrator assina o boletim e devolve-o ao eleitor. O eleitor ao receber as assinaturas dos Administrators descega-as e constrói uma lista das mesmas.

   Seguidamente o eleitor cega a lista e envia-a ao Manager. O Manager assina a lista e devolve-a ao eleitor. O eleitor ao receber a lista do Manager retira-lhe o cegamento.

3. **Anonimização**

   O eleitor envia o seu voto e a lista de assinaturas pelo Anonymizer.

4. **Recolha**

   Os votos chegam ao Counter que verifica todas as assinaturas. Se as assinaturas num voto estiverem todas correctas, o voto é aceite para contagem.

5. **Contagem**

   Por últimos são contados os votos que passaram na fase de recolha e publica-se o resultado da eleição.

O papel do Commissioner é de controlar a eleição. Recebe queixas por parte de todas as entidades eleitorais incluindo o eleitor. Se verificar que as falhas reportadas são graves procede a uma investigação para apurar as causas.

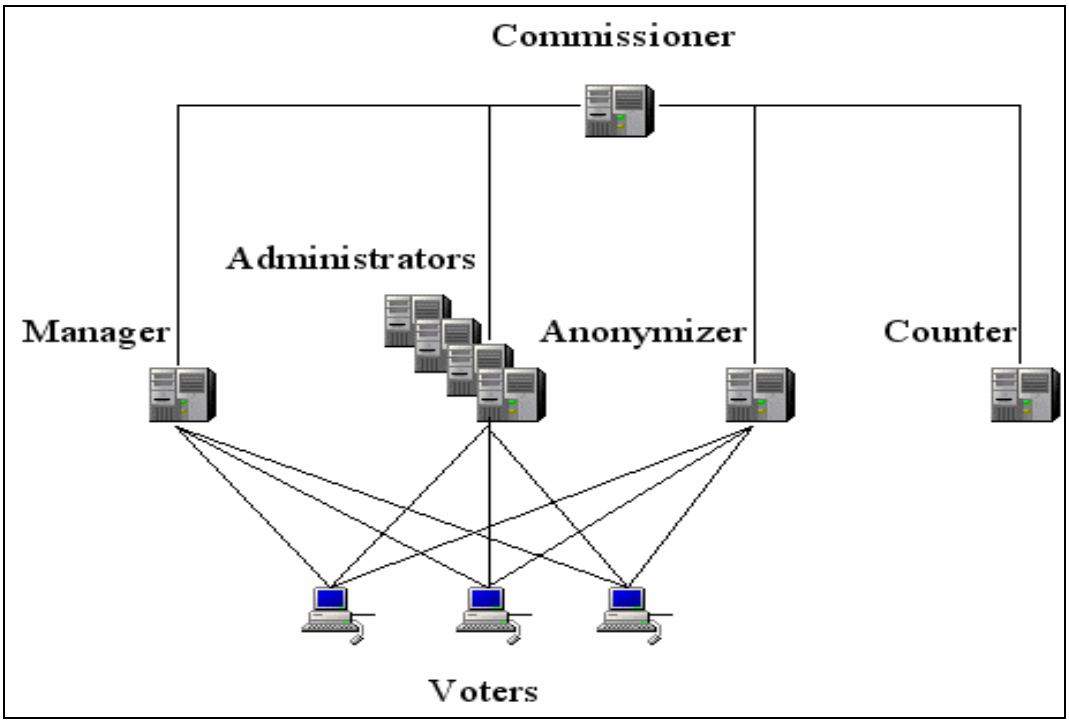


**Figura 11 – E-VOX Managed Administrator**

Supondo que existem $n$ Administrators e o eleitor, para que o seu voto seja considerado válido, tem que obter assinaturas de $t$ deles. Para se conseguir impedir que uma entidade por si só consiga introduzir votos, é necessário que $t \geq n/2 + 1$. Se tal não se verificar, o Manager consegue fazer passar-se por um eleitor que já tenha votado e obter uma lista válida de assinaturas, o que lhe permite introduzir votos.

Note-se que tanto o Manager como os Administrators só assinam uma vez por cada eleitor.

Esta possibilidade de personificação deve-se ao facto de, na implementação do SVE, o eleitor usar sempre a mesma senha para se identificar perante o Manager e os Administrators.

Outro problema nesta arquitectura é a possibilidade de perda, propositada ou não, de votos pelo Anonymizer e/ou Counter. Para evitar os danos causados pela perda de votos é sugerido, pelo autor, a replicação da cadeia Anonymizer-Counter.

O anonimato do eleitor é garantido por um único Anonymizer. Para o anonimato não ficar na posse de uma só entidade, o autor propôs que passasse a existir um caminho composto por vários Anonymizers entre o eleitor e o Counter.

A possibilidade de eleições múltiplas nesta implementação sofre de falhas graves. Pois apesar de o boletim para uma eleição ***X*** ser obtido no Manager perante uma autenticação válida, nada impede que um eleitor ***E1*** que o obteve de fazer uma cópia do seu boletim entregá-la a um outro eleitor ***E2*** que não tenha acesso à eleição ***X***.

O outro problema é que os servidores, Manager e Administrators, só guardam estado necessário para garantir que só assinam uma vez por eleitor. Assim se o Eleitor, por algum motivo, não conseguir prosseguir o protocolo depois de obter as ***t*** assinaturas requeridas dos Administrators já não conseguirá votar.

## 4.2 Arquitectura Proposta

Como já foi dito a arquitectura proposta, Figura 12, baseia-se no E-VOX – Managed Administrator. Em termos de entidades eleitorais surge uma nova entidade, o Distribuidor. O papel do distribuidor é, como o nome indica, distribuir os boletins de voto. Esta alteração é benéfica porque permite libertar o Manager de uma muito considerável quantidade de carga, distribuir um boletim a todos os votantes. É de notar que esta é a maior interacção com o votante em termos de dimensão da

mensagem trocada: além de todas as perguntas e das chaves públicas dos Anonimizadores e dos Contadores, vêm também as chaves que permitem verificar as assinaturas dos Administradores e do Manager.

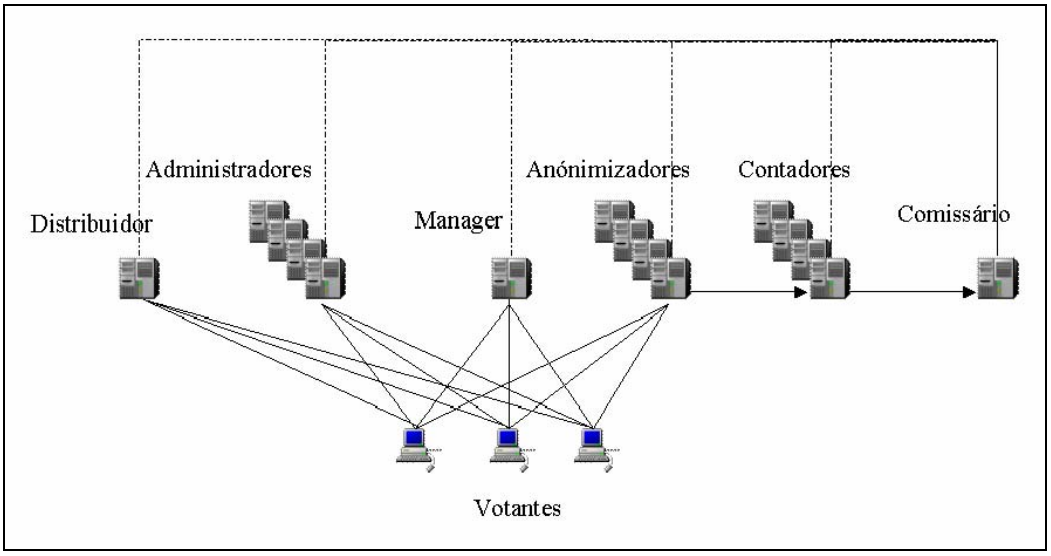


**Figura 12– SVE proposto**

Agora o Anonimizador e o Contador surgem replicados. Esta alteração vem combater o problema da perda, involuntária ou não, dos votos por parte do Anonimizador e do Contador.

## 4.3 Aspectos Implementacionais da Arquitectura

Na arquitectura aqui proposta, tal como em todo o TFC, houve uma preocupação para enquadrar a dupla óptica de TFC / mestrado integrado. Assim, apresenta-se neste TFC uma arquitectura que irá permitir resolver problemas como a tolerância a faltas, a perda de votos e suporte para várias eleições simultâneas. Como serão efectivamente resolvidos estes problemas será melhor estudado no trabalho de mestrado.

No aspecto de tolerância a faltas, há que manter o estado do processo eleitoral. Este estado é mantido pelo Votante, Administradores e Manager. O Votante tem de intervir pois é o único que pode recuperar a informação que passa no Manager e Administradores. Estes últimos têm de colaborar porque são eles que controlam a propriedade de democracia, cada eleitor tem um voto.

Para aumentar a garantia de anonimato e evitar a perda de votos há que rever a cadeia de anonimato (Anonimizadores – Contador). É possível apenas uma replicação da cadeia de anonimato mas também outra solução é ter uma cadeia de Anonimizadores em que não existe um caminho predefinido de Anonimizadores até aos Contadores.

Os Administradores e o Manager devem ter uma assinatura diferente por cada eleição. Assim obtém-se um suporte de eleições simultâneas que evita o problema da distribuição ilícita de boletins de voto.

Outro problema que se deve ter em atenção é a identificação de eleitor. No E-VOX utiliza-se sempre a mesma senha em todas as identificações do eleitor o que levanta graves problemas de segurança, nomeadamente uma possível personificação do eleitor por qualquer entidade em que este já se tenha identificado.

Os aspectos focados nos quatro parágrafos anteriores serão melhor estudados e desenvolvidos na tese de mestrado.

# 5 Realização

Como referido anteriormente, a realização prática deste TFC tem como objectivo fornecer uma boa plataforma para a futura tese de mestrado. Neste âmbito foi desenvolvido um SVE simples mas já com todos os módulos operacionais, Figura 13.

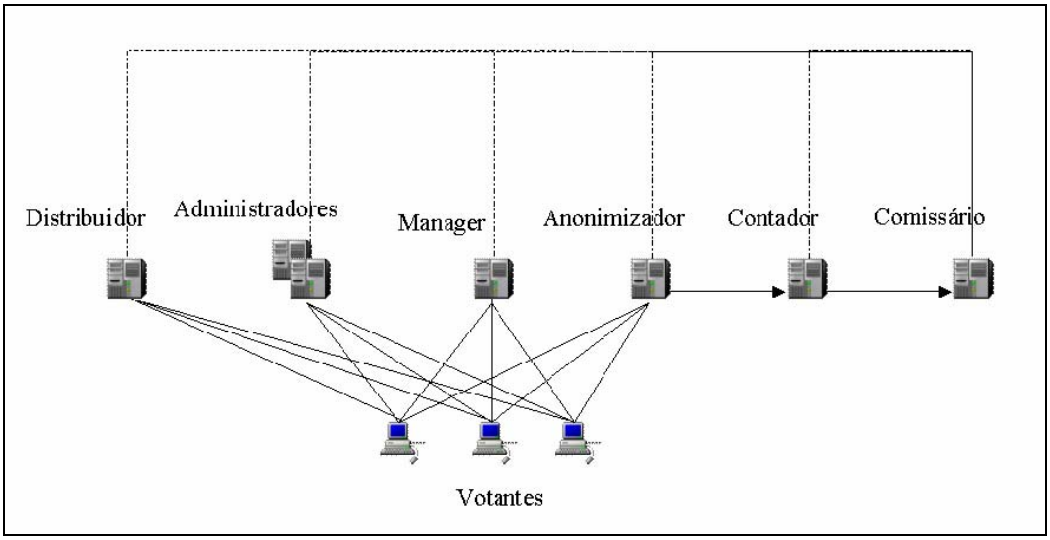


**Figura 13 – SVE desenvolvido no TFC**

Temos um Distribuidor, que pode ser replicado, para fazer a distribuição dos votos pelos Votantes. Existem dois Administradores que assinam o voto ao Votante, e o Manager que assina as assinaturas dos Administradores. A cadeia de anonimato é constituída por um Anonimizador e um Contador. Todos estes serviços foram implementados como XML Web Services. Um XML Web Service possui uma interface normalizada em XML, o que permite uma fácil ligação com outros módulos desenvolvidos, inclusivamente por outras pessoas. O Web Service é um serviço que corre num servidor HTTP, como tal o acesso por parte dos votantes aos mesmos é facilitado pela utilização do protocolo HTTP.

Os módulos Comissário e Votante foram implementados como aplicações Windows. O módulo Comissário permite fazer a configuração de toda a votação. O módulo Votante faz a interface com o votante.

Todos os módulos foram programados em C#. A comunicação com os XML Web Services é feita por SOAP (Simple Object Access Protocol [24]). A interface dos módulos implementados como XML Web Service está disponível em anexo.

Segue-se uma descrição uma descrição mais detalhada da funcionalidade de cada módulo.

## 5.1 Distribuidor

O módulo Distribuidor tem como função a distribuição dos boletins de voto. O seu funcionamento é simples: verifica a identidade do votante e se esta estiver correcta entrega-lhe um boletim de voto, produzido anteriormente pelo Comissário.

O boletim de voto, além de conter as perguntas e suas várias opções de resposta, contém toda a informação necessária para que o Votante consiga votar, nomeadamente o endereço das várias entidades que este tem de contactar.

A interface deste módulo está disponível no Anexo A.

## 5.2 Administrador / Manager

A funcionalidade destes módulos é a mesma, ambos verificam a identidade do votante e esta for correcta assinam o que o eleitor lhes pediu. Assim utilizam-se cópias de um mesmo módulo para os Administradores e Manager.

É de notar que o que difere a função dos Administradores e do Manager é o que estes assinam e não a forma como assinam. Como é o votante que constrói a mensagem que quer que seja assinada, esta é não passa de um conjunto de bytes aos olhos de quem a assina.

Como ainda não foi implementado um mecanismo de distribuição de chaves, existe uma função no módulo para escolher a chave a usar. Esta função, bem como a de

verificação da chave que está a ser utilizada, pode ser facilmente acedida num browser, bastando para isso aceder ao endereço onde se encontra o módulo.

Pode-se encontrar a descrição da interface deste módulo no Anexo B.

## 5.3 Anonimizador

Neste momento, como ainda não está definida a evolução da cadeia de anonimato, o único papel do anonimizador é fazer um *relay* no depósito do voto. Assim consegue-se esconder a origem do Voto. A descrição da interface do módulo está descrita no Anexo C.

## 5.4 Contador

Este módulo tem a seu cargo a operação de verificação e contagem dos votos. Quando recebe um voto verifica se este contém todas as assinaturas requeridas. Se todas as assinaturas forem válidas o voto é contado. A descrição da interface deste módulo encontra-se no Anexo D.

## 5.5 Comissário

O módulo Comissário tem a seu cargo a construção da eleição. Define a localização dos módulos Administradores, Manager, Anonimizador e Contador.

É também neste módulo que é elaborado o boletim de voto, definindo as várias perguntas e suas respostas relativas à eleição.

O módulo Comissário tem também a seu cargo o registo dos votantes que podem participar na votação.

## 5.6 Votante

Este é o módulo que faz toda a interacção com o votante. O seu funcionamento é o seguinte:

1. Primeiro pede ao votante a sua identificação e endereço do Distribuidor.
2. De seguida pede um boletim de voto ao Distribuidor.
3. Após receber o boletim, apresenta-o ao votante para este fazer a sua escolha.
4. Depois do votante exprimir o seu voto, o módulo Votante, pede as várias assinaturas requeridas dos Administradores.
5. Após receber as assinaturas dos Administradores, elabora uma lista com estas e pede ao Manager para a assinar.
6. Quando receber a assinatura do Manager envia-a junto com as assinaturas dos administradores e com o voto para o Anonimizador, concluindo assim o processo de votação.

## 5.7 Bases de Dados

Para armazenar todos os dados relativos à eleição foi criada uma base de dados, Figura 14. Por enquanto a base de dados é relativamente simples pois ainda não suporta várias eleições em simultâneo.

Existe uma tabela com a informação relativa à eleição, a tabela Eleicao. Para guardar os dados dos votantes foi criada a tabela Eleitores. No respeitante às perguntas existem duas tabelas, a tabela Perguntas que contém a questão e um identificador da pergunta, e a tabela Respostas que contém todas as respostas possíveis.

O Comissário tem acesso a todas as tabelas, porque efectua todo o trabalho de construção da eleição.

A tabela Eleitores é acedida pelos Administradores, Manager e Distribuidor para procederem à identificação dos votantes.

Os Administradores e o Manager têm acesso à tabela Eleicao para terem acesso às chaves com que assinam o voto.

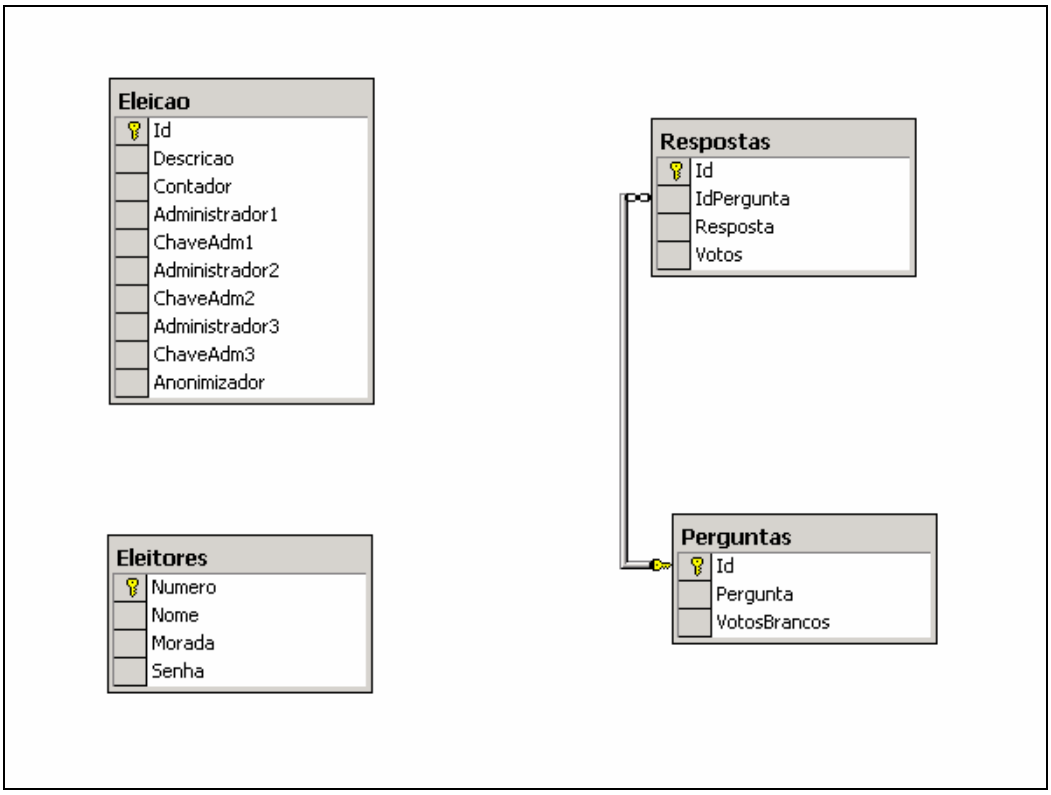


**Figura 14 – Base de Dados**

O Contador e o Distribuidor têm acesso às tabelas Perguntas e Respostas, o primeiro para poder proceder à contagem dos votos; o segundo para poder preencher o boletim que vai distribuir. Ambos os módulos têm também acesso à tabela Eleicao, o Distribuidor para retirar a informação relativa ao endereço dos vários módulos participantes na eleição, e o Contador para poder ter acesso às chaves com que se assinam os votos.

Cada módulo tem a sua base de dados que pode ser obtida efectuando uma exportação dos dados da base de dados do Comissário. Como a base de dados está implementada sobre SQL Server 2000, este é um procedimento bastante simples.

# 6 Conclusões e Trabalho Futuro

O objectivo deste TFC efectuado na dupla óptica de TFC / mestrado integrado foi fazer uma análise dos SVE existentes e efectuar a construção de um SVE modular que servisse de plataforma base para a tese de mestrado.

Da análise aos SVE existentes obteve-se uma lista de problemas a resolver na tese de mestrado:

- Tolerância a faltas – permitir que o processo de votação possa sempre ser reiniciado pelo votante.
- Anonimato e perda de votos – redesenhar a cadeia de anonimato de modo a ser mais robusta contra a perda de anonimato e votos.
- Múltiplas eleições – permitir a realização de eleições simultâneas.
- Identificação – realizar um método de identificação que não permita uma personificação trivial por parte das entidades onde o votante tem que se identificar.

Com base nesta mesma análise propôs-se uma arquitectura para um SVE capaz de lidar com os vários problemas detectados.

Com base na arquitectura proposta implementou-se um SVE modular e funcional, que vai servir de base para o trabalho futuro.

Tendo em conta o que foi proposto e o que foi feito, pode-se dizer que os objectivos foram completamente atingidos.

Como trabalho futuro, a ser realizado na área, é importante encontrar novas e melhores soluções para os seguintes problemas:

- Não Coacção / Verificabilidade
- Escalabilidade
- Tolerância a faltas

# Anexo A – Descrição da interface Distribuidor

### Código dos métodos:

```
[WebMethod]
public MensagemDistribuidorVotante ObtemBoletim(int numero, string
senha)
{
      MensagemDistribuidorVotante resposta = new
      MensagemDistribuidorVotante();
      resposta.erro = this.distribuidorBD.Autenticacao(numero,
senha);
      if(resposta.erro == MensagemErro.OK)
            resposta.infoEleicao = this.eleicao;
      return resposta;
}
```

### Descrição completa do serviço em XML:

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://localhost/Distribuidor"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://localhost/Distribuidor"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://localhost/Distribuidor">
      - <s:element name="ObtemBoletim">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
                name="numero" type="s:int" />
              <s:element minOccurs="0" maxOccurs="1"
                name="senha" type="s:string" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="ObtemBoletimResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
                name="ObtemBoletimResult"
                type="s0:MensagemDistribuidorVotant
                e" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:complexType
          name="MensagemDistribuidorVotante">
```

```
    - <s:sequence>
        <s:element minOccurs="1" maxOccurs="1"
          name="erro" type="s0:MensagemErro" />
        <s:element minOccurs="1" maxOccurs="1"
          name="infoEleicao" type="s0:InfoEleicao"
          />
      </s:sequence>
    </s:complexType>
  - <s:simpleType name="MensagemErro">
    - <s:restriction base="s:string">
        <s:enumeration value="OK" />
        <s:enumeration value="Erro_de_Autenticacao"
          />
        <s:enumeration value="Votante_Ja_Votou" />
        <s:enumeration
          value="Erro_no_Deposito_do_Voto" />
      </s:restriction>
    </s:simpleType>
  - <s:complexType name="InfoEleicao">
    - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="eleicao"
          type="s0:ConfiguracaoEleicao" />
        <s:element minOccurs="0" maxOccurs="1"
          name="boletim"
          type="s0:DescricaoBoletim" />
      </s:sequence>
    </s:complexType>
  - <s:complexType name="ConfiguracaoEleicao">
    - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="Descricao" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="Contador" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="Anonimizador" type="s:string" />
        <s:element minOccurs="0" maxOccurs="1"
          name="Administradores"
          type="s0:ArrayOfAnyType" />
      </s:sequence>
    </s:complexType>
  - <s:complexType name="ArrayOfAnyType">
    - <s:sequence>
        <s:element minOccurs="0"
          maxOccurs="unbounded" name="anyType"
          nillable="true" />
      </s:sequence>
    </s:complexType>
  - <s:complexType name="DescricaoBoletim">
    - <s:sequence>
        <s:element minOccurs="0" maxOccurs="1"
          name="Descricao" type="s:string" />
```

```
            <s:element minOccurs="0" maxOccurs="1"
               name="ListaPerguntas"
               type="s0:ArrayOfEstruturaPergunta" />
          </s:sequence>
        </s:complexType>
      - <s:complexType name="ArrayOfEstruturaPergunta">
        - <s:sequence>
            <s:element minOccurs="0"
               maxOccurs="unbounded"
               name="EstruturaPergunta"
               type="s0:EstruturaPergunta" />
          </s:sequence>
        </s:complexType>
      - <s:complexType name="EstruturaPergunta">
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
               name="idPergunta" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1"
               name="questao" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1"
               name="listaIdOpcoes"
               type="s0:ArrayOfInt" />
            <s:element minOccurs="0" maxOccurs="1"
               name="listaOpcoes"
               type="s0:ArrayOfString" />
          </s:sequence>
        </s:complexType>
      - <s:complexType name="ArrayOfInt">
        - <s:sequence>
            <s:element minOccurs="0"
               maxOccurs="unbounded" name="int"
               type="s:int" />
          </s:sequence>
        </s:complexType>
      - <s:complexType name="ArrayOfString">
        - <s:sequence>
            <s:element minOccurs="0"
               maxOccurs="unbounded" name="string"
               nillable="true" type="s:string" />
          </s:sequence>
        </s:complexType>
        <s:element name="MensagemDistribuidorVotante"
          type="s0:MensagemDistribuidorVotante" />
      </s:schema>
    </types>
  - <message name="ObtemBoletimSoapIn">
      <part name="parameters" element="s0:ObtemBoletim" />
    </message>
  - <message name="ObtemBoletimSoapOut">
      <part name="parameters"
        element="s0:ObtemBoletimResponse" />
    </message>
  - <message name="ObtemBoletimHttpGetIn">
      <part name="numero" type="s:string" />
```

```
    <part name="senha" type="s:string" />
  </message>
- <message name="ObtemBoletimHttpGetOut">
    <part name="Body"
      element="s0:MensagemDistribuidorVotante" />
  </message>
- <message name="ObtemBoletimHttpPostIn">
    <part name="numero" type="s:string" />
    <part name="senha" type="s:string" />
  </message>
- <message name="ObtemBoletimHttpPostOut">
    <part name="Body"
      element="s0:MensagemDistribuidorVotante" />
  </message>
- <portType name="DistribuidorSoap">
  - <operation name="ObtemBoletim">
      <input message="s0:ObtemBoletimSoapIn" />
      <output message="s0:ObtemBoletimSoapOut" />
    </operation>
  </portType>
- <portType name="DistribuidorHttpGet">
  - <operation name="ObtemBoletim">
      <input message="s0:ObtemBoletimHttpGetIn" />
      <output message="s0:ObtemBoletimHttpGetOut" />
    </operation>
  </portType>
- <portType name="DistribuidorHttpPost">
  - <operation name="ObtemBoletim">
      <input message="s0:ObtemBoletimHttpPostIn" />
      <output message="s0:ObtemBoletimHttpPostOut" />
    </operation>
  </portType>
- <binding name="DistribuidorSoap" type="s0:DistribuidorSoap">
    <soap:binding
      transport="http://schemas.xmlsoap.org/soap/http"
      style="document" />
  - <operation name="ObtemBoletim">
      <soap:operation
        soapAction="http://localhost/Distribuidor/ObtemB
        oletim" style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
- <binding name="DistribuidorHttpGet"
    type="s0:DistribuidorHttpGet">
    <http:binding verb="GET" />
  - <operation name="ObtemBoletim">
      <http:operation location="/ObtemBoletim" />
```

```
      - <input>
          <http:urlEncoded />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    </binding>
  - <binding name="DistribuidorHttpPost"
      type="s0:DistribuidorHttpPost">
      <http:binding verb="POST" />
    - <operation name="ObtemBoletim">
        <http:operation location="/ObtemBoletim" />
      - <input>
          <mime:content type="application/x-www-form-
            urlencoded" />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    </binding>
  - <service name="Distribuidor">
    - <port name="DistribuidorSoap"
        binding="s0:DistribuidorSoap">
        <soap:address
          location="http://localhost/Distribuidortfc/Distribui
          dor.asmx" />
      </port>
    - <port name="DistribuidorHttpGet"
        binding="s0:DistribuidorHttpGet">
        <http:address
          location="http://localhost/Distribuidortfc/Distribui
          dor.asmx" />
      </port>
    - <port name="DistribuidorHttpPost"
        binding="s0:DistribuidorHttpPost">
        <http:address
          location="http://localhost/Distribuidortfc/Distribui
          dor.asmx" />
      </port>
    </service>
  </definitions>
```

# Anexo B – Descrição da interface Administrador / Manager

### Código dos métodos:

```
[WebMethod]
public MensagemAssinada Assina(int numero, string senha, byte[] hash)
{
      if (MensagemErro.OK == this.admManagerBD.Autenticacao(numero,
senha))
      {
            MensagemAssinada resposta = new MensagemAssinada();
            resposta.erro = MensagemErro.OK;
            resposta.mensagem = this.admManagerBD.Assinatura(hash);
            return resposta;
      }
      else
            return new
      MensagemAssinada(MensagemErro.Erro_de_Autenticacao, null);
}

[WebMethod]
public string AcertaChave(int numero)
{
      if(this.admManagerBD.CarregaChave(numero))
            return "Chave carregada";
      else
            return "Nao foi possivel carregar a chave";
}

[WebMethod]
public int ConfirmaChave()
{
      return this.admManagerBD.ConfereChave();
}
```

### Descrição completa do serviço em XML:

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:s1="http://tempuri.org/AbstractTypes"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://tempuri.org/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://tempuri.org/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://tempuri.org/">
      - <s:element name="Assina">
        - <s:complexType>
          - <s:sequence>
```

```
            <s:element minOccurs="1" maxOccurs="1"
              name="numero" type="s:int" />
            <s:element minOccurs="0" maxOccurs="1"
              name="senha" type="s:string" />
            <s:element minOccurs="0" maxOccurs="1"
              name="hash" type="s:base64Binary"
              />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="AssinaResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
              name="AssinaResult"
              type="s0:MensagemAssinada" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:complexType name="MensagemAssinada">
      - <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
            name="erro" type="s0:MensagemErro" />
          <s:element minOccurs="0" maxOccurs="1"
            name="mensagem" type="s:base64Binary"
            />
        </s:sequence>
      </s:complexType>
    - <s:simpleType name="MensagemErro">
      - <s:restriction base="s:string">
          <s:enumeration value="OK" />
          <s:enumeration value="Erro_de_Autenticacao"
            />
          <s:enumeration value="Votante_Ja_Votou" />
          <s:enumeration
            value="Erro_no_Deposito_do_Voto" />
        </s:restriction>
      </s:simpleType>
    - <s:element name="AcertaChave">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="1" maxOccurs="1"
              name="numero" type="s:int" />
          </s:sequence>
        </s:complexType>
      </s:element>
    - <s:element name="AcertaChaveResponse">
      - <s:complexType>
        - <s:sequence>
            <s:element minOccurs="0" maxOccurs="1"
              name="AcertaChaveResult"
              type="s:string" />
          </s:sequence>
        </s:complexType>
```

```
        </s:element>
      - <s:element name="ConfirmaChave">
          <s:complexType />
        </s:element>
      - <s:element name="ConfirmaChaveResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
                name="ConfirmaChaveResult"
                type="s:int" />
            </s:sequence>
          </s:complexType>
        </s:element>
        <s:element name="MensagemAssinada"
          type="s0:MensagemAssinada" />
        <s:element name="string" nillable="true"
          type="s:string" />
        <s:element name="int" type="s:int" />
      </s:schema>
    - <s:schema
        targetNamespace="http://tempuri.org/AbstractTypes">
      - <s:complexType name="StringArray">
        - <s:complexContent mixed="false">
          - <s:restriction base="soapenc:Array">
            - <s:sequence>
                <s:element minOccurs="0"
                  maxOccurs="unbounded"
                  name="String" type="s:string" />
              </s:sequence>
            </s:restriction>
          </s:complexContent>
        </s:complexType>
      </s:schema>
    </types>
  - <message name="AssinaSoapIn">
      <part name="parameters" element="s0:Assina" />
    </message>
  - <message name="AssinaSoapOut">
      <part name="parameters" element="s0:AssinaResponse" />
    </message>
  - <message name="AcertaChaveSoapIn">
      <part name="parameters" element="s0:AcertaChave" />
    </message>
  - <message name="AcertaChaveSoapOut">
      <part name="parameters"
        element="s0:AcertaChaveResponse" />
    </message>
  - <message name="ConfirmaChaveSoapIn">
      <part name="parameters" element="s0:ConfirmaChave" />
    </message>
  - <message name="ConfirmaChaveSoapOut">
      <part name="parameters"
        element="s0:ConfirmaChaveResponse" />
    </message>
```

```
- <message name="AssinaHttpGetIn">
    <part name="numero" type="s:string" />
    <part name="senha" type="s:string" />
    <part name="hash" type="s1:StringArray" />
  </message>
- <message name="AssinaHttpGetOut">
    <part name="Body" element="s0:MensagemAssinada" />
  </message>
- <message name="AcertaChaveHttpGetIn">
    <part name="numero" type="s:string" />
  </message>
- <message name="AcertaChaveHttpGetOut">
    <part name="Body" element="s0:string" />
  </message>
  <message name="ConfirmaChaveHttpGetIn" />
- <message name="ConfirmaChaveHttpGetOut">
    <part name="Body" element="s0:int" />
  </message>
- <message name="AssinaHttpPostIn">
    <part name="numero" type="s:string" />
    <part name="senha" type="s:string" />
    <part name="hash" type="s1:StringArray" />
  </message>
- <message name="AssinaHttpPostOut">
    <part name="Body" element="s0:MensagemAssinada" />
  </message>
- <message name="AcertaChaveHttpPostIn">
    <part name="numero" type="s:string" />
  </message>
- <message name="AcertaChaveHttpPostOut">
    <part name="Body" element="s0:string" />
  </message>
  <message name="ConfirmaChaveHttpPostIn" />
- <message name="ConfirmaChaveHttpPostOut">
    <part name="Body" element="s0:int" />
  </message>
- <portType name="AdmManagerSoap">
  - <operation name="Assina">
      <input message="s0:AssinaSoapIn" />
      <output message="s0:AssinaSoapOut" />
    </operation>
  - <operation name="AcertaChave">
      <input message="s0:AcertaChaveSoapIn" />
      <output message="s0:AcertaChaveSoapOut" />
    </operation>
  - <operation name="ConfirmaChave">
      <input message="s0:ConfirmaChaveSoapIn" />
      <output message="s0:ConfirmaChaveSoapOut" />
    </operation>
  </portType>
- <portType name="AdmManagerHttpGet">
  - <operation name="Assina">
      <input message="s0:AssinaHttpGetIn" />
```

```
      <output message="s0:AssinaHttpGetOut" />
    </operation>
  - <operation name="AcertaChave">
      <input message="s0:AcertaChaveHttpGetIn" />
      <output message="s0:AcertaChaveHttpGetOut" />
    </operation>
  - <operation name="ConfirmaChave">
      <input message="s0:ConfirmaChaveHttpGetIn" />
      <output message="s0:ConfirmaChaveHttpGetOut" />
    </operation>
  </portType>
- <portType name="AdmManagerHttpPost">
  - <operation name="Assina">
      <input message="s0:AssinaHttpPostIn" />
      <output message="s0:AssinaHttpPostOut" />
    </operation>
  - <operation name="AcertaChave">
      <input message="s0:AcertaChaveHttpPostIn" />
      <output message="s0:AcertaChaveHttpPostOut" />
    </operation>
  - <operation name="ConfirmaChave">
      <input message="s0:ConfirmaChaveHttpPostIn" />
      <output message="s0:ConfirmaChaveHttpPostOut" />
    </operation>
  </portType>
- <binding name="AdmManagerSoap"
    type="s0:AdmManagerSoap">
    <soap:binding
      transport="http://schemas.xmlsoap.org/soap/http"
      style="document" />
  - <operation name="Assina">
      <soap:operation
        soapAction="http://tempuri.org/Assina"
        style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  - <operation name="AcertaChave">
      <soap:operation
        soapAction="http://tempuri.org/AcertaChave"
        style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  - <operation name="ConfirmaChave">
```

```
        <soap:operation
          soapAction="http://tempuri.org/ConfirmaChave"
          style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
  - <binding name="AdmManagerHttpGet"
      type="s0:AdmManagerHttpGet">
      <http:binding verb="GET" />
    - <operation name="Assina">
        <http:operation location="/Assina" />
      - <input>
          <http:urlEncoded />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    - <operation name="AcertaChave">
        <http:operation location="/AcertaChave" />
      - <input>
          <http:urlEncoded />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    - <operation name="ConfirmaChave">
        <http:operation location="/ConfirmaChave" />
      - <input>
          <http:urlEncoded />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    </binding>
  - <binding name="AdmManagerHttpPost"
      type="s0:AdmManagerHttpPost">
      <http:binding verb="POST" />
    - <operation name="Assina">
        <http:operation location="/Assina" />
      - <input>
          <mime:content type="application/x-www-form-
            urlencoded" />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
```

```
        </output>
      </operation>
    - <operation name="AcertaChave">
        <http:operation location="/AcertaChave" />
      - <input>
          <mime:content type="application/x-www-form-
            urlencoded" />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    - <operation name="ConfirmaChave">
        <http:operation location="/ConfirmaChave" />
      - <input>
          <mime:content type="application/x-www-form-
            urlencoded" />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
  </binding>
- <service name="AdmManager">
  - <port name="AdmManagerSoap"
      binding="s0:AdmManagerSoap">
      <soap:address
        location="http://localhost/AdmManager2/AdmMan
        ager.asmx" />
    </port>
  - <port name="AdmManagerHttpGet"
      binding="s0:AdmManagerHttpGet">
      <http:address
        location="http://localhost/AdmManager2/AdmMan
        ager.asmx" />
    </port>
  - <port name="AdmManagerHttpPost"
      binding="s0:AdmManagerHttpPost">
      <http:address
        location="http://localhost/AdmManager2/AdmMan
        ager.asmx" />
    </port>
  </service>
</definitions>
```

# Anexo C – Descrição da interface Anonimizador

**Código dos métodos:**

```
[WebMethod]
public MensagemErro DepositarVoto(byte[] voto)
{
      return this.ligacaoContador.DepositaVoto(voto);
}
```

**Descrição completa do serviço em XML:**

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:s1="http://tempuri.org/AbstractTypes"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://tempuri.org/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://tempuri.org/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://tempuri.org/">
      - <s:element name="DepositarVoto">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1"
                name="voto" type="s:base64Binary" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="DepositarVotoResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
                name="DepositarVotoResult"
                type="s0:MensagemErro" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:simpleType name="MensagemErro">
        - <s:restriction base="s:string">
            <s:enumeration value="OK" />
            <s:enumeration value="Erro_de_Autenticacao"
              />
            <s:enumeration value="Votante_Ja_Votou" />
            <s:enumeration
              value="Erro_no_Deposito_do_Voto" />
          </s:restriction>
        </s:simpleType>
```

```
        <s:element name="MensagemErro"
          type="s0:MensagemErro" />
      </s:schema>
    - <s:schema
        targetNamespace="http://tempuri.org/AbstractTypes">
      - <s:complexType name="StringArray">
        - <s:complexContent mixed="false">
          - <s:restriction base="soapenc:Array">
            - <s:sequence>
                <s:element minOccurs="0"
                  maxOccurs="unbounded"
                  name="String" type="s:string" />
              </s:sequence>
            </s:restriction>
          </s:complexContent>
        </s:complexType>
      </s:schema>
  </types>
- <message name="DepositarVotoSoapIn">
    <part name="parameters" element="s0:DepositarVoto" />
  </message>
- <message name="DepositarVotoSoapOut">
    <part name="parameters"
      element="s0:DepositarVotoResponse" />
  </message>
- <message name="DepositarVotoHttpGetIn">
    <part name="voto" type="s1:StringArray" />
  </message>
- <message name="DepositarVotoHttpGetOut">
    <part name="Body" element="s0:MensagemErro" />
  </message>
- <message name="DepositarVotoHttpPostIn">
    <part name="voto" type="s1:StringArray" />
  </message>
- <message name="DepositarVotoHttpPostOut">
    <part name="Body" element="s0:MensagemErro" />
  </message>
- <portType name="AnonimizadorSoap">
  - <operation name="DepositarVoto">
      <input message="s0:DepositarVotoSoapIn" />
      <output message="s0:DepositarVotoSoapOut" />
    </operation>
  </portType>
- <portType name="AnonimizadorHttpGet">
  - <operation name="DepositarVoto">
      <input message="s0:DepositarVotoHttpGetIn" />
      <output message="s0:DepositarVotoHttpGetOut" />
    </operation>
  </portType>
- <portType name="AnonimizadorHttpPost">
  - <operation name="DepositarVoto">
      <input message="s0:DepositarVotoHttpPostIn" />
      <output message="s0:DepositarVotoHttpPostOut" />
```

```
    </operation>
  </portType>
- <binding name="AnonimizadorSoap"
    type="s0:AnonimizadorSoap">
    <soap:binding
      transport="http://schemas.xmlsoap.org/soap/http"
      style="document" />
  - <operation name="DepositarVoto">
      <soap:operation
        soapAction="http://tempuri.org/DepositarVoto"
        style="document" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
- <binding name="AnonimizadorHttpGet"
    type="s0:AnonimizadorHttpGet">
    <http:binding verb="GET" />
  - <operation name="DepositarVoto">
      <http:operation location="/DepositarVoto" />
    - <input>
        <http:urlEncoded />
      </input>
    - <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
- <binding name="AnonimizadorHttpPost"
    type="s0:AnonimizadorHttpPost">
    <http:binding verb="POST" />
  - <operation name="DepositarVoto">
      <http:operation location="/DepositarVoto" />
    - <input>
        <mime:content type="application/x-www-form-
          urlencoded" />
      </input>
    - <output>
        <mime:mimeXml part="Body" />
      </output>
    </operation>
  </binding>
- <service name="Anonimizador">
  - <port name="AnonimizadorSoap"
      binding="s0:AnonimizadorSoap">
      <soap:address
        location="http://localhost/Anonimizador/Anonimiz
        ador.asmx" />
    </port>
```

```
  - <port name="AnonimizadorHttpGet"
      binding="s0:AnonimizadorHttpGet">
      <http:address
        location="http://localhost/Anonimizador/Anonimiz
        ador.asmx" />
    </port>
  - <port name="AnonimizadorHttpPost"
      binding="s0:AnonimizadorHttpPost">
      <http:address
        location="http://localhost/Anonimizador/Anonimiz
        ador.asmx" />
    </port>
  </service>
</definitions>
```

# Anexo D – Descrição da interface Contador

**Código dos métodos:**

```
[WebMethod]
public MensagemErro DepositarVoto(byte[] voto)
{
      if(this.contadorBD.DepositaVoto(voto))
            return MensagemErro.OK;
      else
            return MensagemErro.Erro_no_Deposito_do_Voto;
}
```

**Descrição completa do serviço em XML:**

```
<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:s1="http://tempuri.org/AbstractTypes"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://tempuri.org/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    targetNamespace="http://tempuri.org/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
  - <types>
    - <s:schema elementFormDefault="qualified"
        targetNamespace="http://tempuri.org/">
      - <s:element name="DepositarVoto">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="0" maxOccurs="1"
                name="voto" type="s:base64Binary" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:element name="DepositarVotoResponse">
        - <s:complexType>
          - <s:sequence>
              <s:element minOccurs="1" maxOccurs="1"
                name="DepositarVotoResult"
                type="s0:MensagemErro" />
            </s:sequence>
          </s:complexType>
        </s:element>
      - <s:simpleType name="MensagemErro">
        - <s:restriction base="s:string">
            <s:enumeration value="OK" />
            <s:enumeration value="Erro_de_Autenticacao"
              />
            <s:enumeration value="Votante_Ja_Votou" />
            <s:enumeration
              value="Erro_no_Deposito_do_Voto" />
```

```
            </s:restriction>
          </s:simpleType>
          <s:element name="MensagemErro"
            type="s0:MensagemErro" />
        </s:schema>
      - <s:schema
          targetNamespace="http://tempuri.org/AbstractTypes">
        - <s:complexType name="StringArray">
          - <s:complexContent mixed="false">
            - <s:restriction base="soapenc:Array">
              - <s:sequence>
                  <s:element minOccurs="0"
                    maxOccurs="unbounded"
                    name="String" type="s:string" />
                </s:sequence>
              </s:restriction>
            </s:complexContent>
          </s:complexType>
        </s:schema>
      </types>
    - <message name="DepositarVotoSoapIn">
        <part name="parameters" element="s0:DepositarVoto" />
      </message>
    - <message name="DepositarVotoSoapOut">
        <part name="parameters"
          element="s0:DepositarVotoResponse" />
      </message>
    - <message name="DepositarVotoHttpGetIn">
        <part name="voto" type="s1:StringArray" />
      </message>
    - <message name="DepositarVotoHttpGetOut">
        <part name="Body" element="s0:MensagemErro" />
      </message>
    - <message name="DepositarVotoHttpPostIn">
        <part name="voto" type="s1:StringArray" />
      </message>
    - <message name="DepositarVotoHttpPostOut">
        <part name="Body" element="s0:MensagemErro" />
      </message>
    - <portType name="ContadorSoap">
      - <operation name="DepositarVoto">
          <input message="s0:DepositarVotoSoapIn" />
          <output message="s0:DepositarVotoSoapOut" />
        </operation>
      </portType>
    - <portType name="ContadorHttpGet">
      - <operation name="DepositarVoto">
          <input message="s0:DepositarVotoHttpGetIn" />
          <output message="s0:DepositarVotoHttpGetOut" />
        </operation>
      </portType>
    - <portType name="ContadorHttpPost">
      - <operation name="DepositarVoto">
```

```
        <input message="s0:DepositarVotoHttpPostIn" />
        <output message="s0:DepositarVotoHttpPostOut" />
      </operation>
    </portType>
  - <binding name="ContadorSoap" type="s0:ContadorSoap">
      <soap:binding
        transport="http://schemas.xmlsoap.org/soap/http"
        style="document" />
    - <operation name="DepositarVoto">
        <soap:operation
          soapAction="http://tempuri.org/DepositarVoto"
          style="document" />
      - <input>
          <soap:body use="literal" />
        </input>
      - <output>
          <soap:body use="literal" />
        </output>
      </operation>
    </binding>
  - <binding name="ContadorHttpGet"
      type="s0:ContadorHttpGet">
      <http:binding verb="GET" />
    - <operation name="DepositarVoto">
        <http:operation location="/DepositarVoto" />
      - <input>
          <http:urlEncoded />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    </binding>
  - <binding name="ContadorHttpPost"
      type="s0:ContadorHttpPost">
      <http:binding verb="POST" />
    - <operation name="DepositarVoto">
        <http:operation location="/DepositarVoto" />
      - <input>
          <mime:content type="application/x-www-form-
            urlencoded" />
        </input>
      - <output>
          <mime:mimeXml part="Body" />
        </output>
      </operation>
    </binding>
  - <service name="Contador">
    - <port name="ContadorSoap" binding="s0:ContadorSoap">
        <soap:address
          location="http://localhost/Contadortfc/Contador.a
          smx" />
      </port>
```

```
  - <port name="ContadorHttpGet"
      binding="s0:ContadorHttpGet">
      <http:address
        location="http://localhost/Contadortfc/Contador.a
        smx" />
    </port>
  - <port name="ContadorHttpPost"
      binding="s0:ContadorHttpPost">
      <http:address
        location="http://localhost/Contadortfc/Contador.a
        smx" />
    </port>
  </service>
</definitions>
```

# Bibliografia

[1] Sequóia Voting Systems, www.sequoiavote.com

[2] Michael Shamos. Electronic Voting – Evaluating the Threat. Em *Proceedings of Computer, Freedom, and Privacy Conference (CFP'93)*, 1993.

[3] Lorrie F. Cranor e Ron K. Cytron. Sensus. A Security-Conscious Electronic Polling System for the Internet. Em *Proceedings of the Hawaii International Conference on System Sciences*, 1997.

[4] Pedro Antunes, Américo Monteiro, Natércia Soares, Rosa Maria Oliveira. Sistemas de Votação Electrónica. Faculdade de Ciências da Universidade de Lisboa, Departamento de Informática, *Technical Report TR-01-X,* Outubro 2001

[5] Berry Schoenmakers. Compensating for a lack of transparency. Em *Proceedings of the Tenth Conference on Computers, Freedom & Privacy (CFP'00),* 2000.

[6] Andreu Riera, Joan Borrell e Josep Rifà. A Protocol for Large Scale Elections by Coordinating Multiple Electoral Colleges. Em *Proceedings of Information Security in Research and Business (IFIP'97),* pp. 349-362, 1997.

[7] M. Gonçalves, M. Ramos. TFC Votação Electrónica. Instituto Superior Técnico, Setembro 2001.

[8] Mark Herschberg. *Secure electronic voting using the world wide web.* Master's thesis, Massachusetts Institute of Technology, Junho1997.

[9] David Chaum. Untraceable Electronic Mail, Return Address, and Digital Pseudonyms. Em *Communications of the ACM*, pp. 84-881994, 1981.

[10] David Chaum. Elections with Unconditionally-Secret Ballots and Disruption Equivalent to Breaking RSA. Em *Proceendins of Advances in Crytology – EUROCRYPT'89*, pp. 177-182, 1989.

[11] Choonsik Park, Kazutomo Itoh e Kaoru Kurosawa. Efficient Anonymous Channel and All/Nothing Election Scheme. Em *Proceedings of Advances in Cryptology – EUROCRYPT'93,* vol. 765 of *LNCS,* pp. 248-259. Springer-Verlag 1993.

[12] Kazue Sako e Joe Killian. Receipt-free mix-type voting scheme – A pratical solution to the implementation of a voting booth. Em *Proceedings of Advances in Cryptology – EUROCRYPT'95,* vol. 921 of *LNCS*, pp. 393-403. Springer-Verlag 1995.

[13] Atsushi Fujioka, Tatsuaki Okamoto e Kazuo Otha. A practical secret voting scheme for large scale elections. Em *Proceedings of Advances in Crytology – AUSCRYPT'92,* vol. 718 of *LNCS*, pp. 244-251. Springer-Verlag 1992.

[14] Brandon DuRette. *Multiple Administrators for Electronic Voting.* Bachelor's thesis, Massachusetts Institute of Technology, Maio 1999.

[15] Tatsuaki Okamoto. An Electronic Voting Scheme. Em *Proceedings of 14th World Computer Congress of International Federation for Information Processing (IFIP'96), Advanced IT Tools, Chapman & Hall*, pp. 21-30, 1996.

[16] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. EM *Proceedings of Workshop on Secutity Protocols '97,* vol. 1361 of *LNCS*, pp. 440-444. Springer-Verlag, 1997.

[17] Josh Benaloh e Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). Em *Proceedings of 26th ACM Symposium on theory of Computing (STOC),* pp. 544-553. ACM 1994.

[18] Martin Hirt e Kazue Sako. Em *Proceedings of Advances in Cryptology – EUROCRYPT'00*, vol. 1807 of *LNCS*, pp. 539-556. Springer-Verlag 2000.

[19] Ronald Cramer, Matthew Franklin, Berry Schoenmakers e Moti Yung. Multi-Authority Secret-Ballot Elections with Linear Work. Em *Proceedings of Advances in Crytology – EUROCRYPT'96,* vol. 1070 of *LNCS*, pp. 72-83. Springer-Verlag 1996.

[20] Ronal Cramer, Rosario Gennaro e Berry Schoenmakers. A Secure and Optimally Efficient Multi-Authority Election Scheme. Em *Proceedings of European Transactions on Telecommunications,* pp. 481-489, 1997.

[21] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard e J.Stern. Practical Multi-Candidate Election System. Em *Proceedings of the 20th ACM Symposium on Principles of Distributed Computing*, pp. 274-283. ACM 2001.

[22] Martin Hirt. Receipt-freeness in Electronic Voting. Em *Proceedings of Workshop on Trustworthy Elections (WOTE'01).* Agosto 2001.

[23] Andreu Riera, Joan Borrell, Josep Rifà. How Smartcards Helped to Solve a Cryptographic Paradox. Em *Actas de la V Reunión Española sobre Criptologia y Seguridad de la Información F.J. López, J. Pastor. J.M. Troya,* pp. 35-36. Universidade de Málaga, 1998.

[24] Simple Object Access Protocol (SOAP), http://www.w3.org/TR/SOAP