Système de vote électronique

Nicolas Bonvin
nicolas.bonvin@epfl.ch

Informatique

Projet de Master

Janvier 2005

Responsable
Prof. Serge Vaudenay
serge.vaudenay@epfl.ch
EPFL / LASEC

Superviseur
Gilbert Robert
robert@prolibre.com
ProLibre Sàrl

Sommaire

1 Introduction

Le document présent traite de l'étude et de la réalisation d'un système de vote électronique par Internet. Ce travail fait office de projet de Master, et a été réalisé en collaboration avec *ProLibre Sàrl*[1] à Carouge.

1.1 Généralités

Le vote électronique fait beaucoup parler de lui ces derniers temps, ne serait-ce que par le fait que la Suisse compte plusieurs projets pilotes. La commune de Carouge a ainsi déjà procédé à plusieurs votations par ce biais. Mais pourquoi vouloir remplacer ou compléter le vote traditionnel ? Serait-ce pour voir la population plus présente dans la vie politique suisse ? Pour faciliter le processus de vote ou encore le rendre plus attractif envers une partie de la population ? Ou bien est-ce le fait de certaines personnes influentes voulant profiter de l'engouement pour le « e-governement » ? Voici une explication fournie par le canton de Genève[2] :

- les citoyens suisses votant quatre à cinq fois par an, même parfois plus, il est nécessaire de s'assurer que la procédure de vote soit agréable et peu contraignante. Offrir une alternative au vote traditionnel va dans ce sens ;
- la démocratie directe se prête au vote par Internet, non seulement parce qu'elle entraîne de nombreux scrutins, mais aussi pour les vastes compétences données au peuple ;
- selon l'Office fédéral de la statistique, 65% de la population suisse a accès à Internet, à la maison ou sur son lieu de travail. Un Suisse sur trois surfe quotidiennement sur le Web. Offrir un moyen de voter supplémentaire utilisant un outil adopté par la population peut se montrer pertinent ;
- des 5,6 millions de citoyens suisses (pour une population totale d'environ 7 millions d'individus), 580 000 vivent à l'étranger. Il est important

[1] www.prolibre.com
[2] www.geneve.ch/evoting

qu'ils disposent aussi d'un système de vote efficace et simple. Cette remarque est également valable pour des personnes ayant une mobilité réduite vivant en Suisse ;
– le service public doit s'adapter aux nouveaux modes de vie pour aller à la rencontre de son public, y compris sur Internet.

Le projet pilote de Genève a cependant soulevé plusieurs critiques de la part de personnes soucieuses du respect de la démocratie. Le *GULL*[3], par exemple, dénonce entre autres un manque de communication de la part de l'État. Le processus de vote se doit d'être réalisé dans un contexte de transparence totale. Il revendique l'accessibilité aux programmes utilisés afin de pouvoir se convaincre de la fiabilité et de la sécurité du système. Le *GULL* rappelle également le fait qu'aucune application informatique ne s'est montrée sûre à 100%. Le fait de pouvoir auditer librement le code source de l'application est souhaitable et encouragé par la majorité des experts en sécurité informatique.

L'application décrite dans ce document est libre et ouverte. Le code source est soumis à la *GNU GPL*[4].

1.2 Cahier des charges

Suite aux « Accords de Genève », un projet s'est formé afin de demander l'avis des populations israéliennes et palestiniennes. Le vote électronique s'avère être un moyen pratique pour réaliser une votation de ce type. Selon le groupe de travail du projet, la votation est supportée par la Confédération helvétique et par des groupements modérés des deux pays. Les gouvernements locaux ne sont pas impliqués dans le projet, et peuvent même se montrer hostiles. Ceci dénote bien le contexte particulier pour lequel cette application doit être réalisée : faire voter les citoyens de deux pays en proie à des rivalités ancestrales, dont les gouvernements respectifs peuvent s'avérer réticents.

Il est nécessaire de prendre en compte les conditions particulières suivantes :
– le sujet de la consultation est extrêmement sensible au niveau politique ;
– il n'est possible de faire confiance qu'aux personnes impliquées dans le projet et nullement aux autorités locales ;
– il y a de fortes possibilités qu'il y ait des tentatives pour saboter la consultation ;

[3] Groupe des Utilisateurs Linux du Léman. www.linux-gull.ch/evote
[4] www.gnu.org/copyleft/gpl.html

– les listes électorales seront « probablement » disponibles ;
– il sera impossible de vérifier physiquement l'identité de tous les votants ;
– il sera impossible de distribuer quoi que ce soit dans les lieux publics. Les services de téléphonie et la poste peuvent être sollicités mais pas dans toutes les zones des pays ;
– il faut garantir une participation intéressante pour assurer une légitimité au vote ;
– il faut s'efforcer de garantir l'anonymat total (mise en péril de l'intégrité du votant).

En résumé, on ne peut avoir confiance en personne. Alors comment réaliser une application de vote électronique par Internet si même le gouvernement local se montre hostile ?

On peut imaginer que ce dernier contrôle les fournisseurs d'accès à Internet. Rien ne l'empêche de couper momentanément[5] l'accès à Internet afin d'empêcher la population de prendre part au vote, ou de falsifier les réponses aux requêtes *DNS*, par exemple, dans le but de rediriger les votants vers un serveur hostile, capable d'identifier les citoyens et de connaître leur choix. Il existe en outre toute une palette d'attaques auxquelles il faut prévoir une réponse adéquate. Par exemple, il faut songer aux attaques directes contre les serveurs de vote, comme un déni de service distribué, ou encore les attaques visant les faiblesses des démons tournant sur les serveurs. Il est également nécessaire de s'assurer que personnne ne puisse écouter le trafic circulant sur le réseau, les fournisseurs d'accès pouvant mettre en place des filtres pour capturer certains paquets spécifiques. Ce dernier problème est essentiellement résolu en utilisant du chiffrement. Cependant, la mise en place de chiffrement n'assure pas totalement la sécurité des données transitant sur le réseau, encore faut-il que ces mesures cryptographiques soient déployées de manière judicieuse.

La sécurité physique des serveurs doit également faire l'objet d'attentions particulières. À quoi bon blinder un serveur de manière logicielle, si un attaquant peut disposer physiquement de la machine ? Les serveurs devront donc, dans l'idéal, être hébergés dans un centre de calcul ultra-sécurisé situé dans un pays neutre et sûr, qui ne risque pas de subir la pression des deux pays concernés.

En poussant la logique sécuritaire encore plus loin, il pourrait se révéler judicieux que les personnes impliquées dans ce projet se fassent les plus discrètes possible. On peut aisément imaginer les pressions qu'elles pourraient subir.

[5] Une coupure de plusieurs jours serait inacceptable et très mal tolérée par la population.

2 Scenarii

Plusieurs scenarii ont été définis, visant à explorer différentes possibilités de déploiement du système de vote en conditions réelles. Ils varient essentiellement en fonction de la première phase du vote : l'enregistrement. La seconde partie, celle du vote par Internet, reste sensiblement la même, dès le moment où le votant possède un bulletin valide et autorisé à voter. L'idée est qu'à la fin de la phase d'enregistrement, chaque personne habilitée à prendre part au vote ait un bulletin en papier valide. C'est sur cette hypothèse que se base le système de vote électronique décrit dans les pages suivantes.

Le premier scénario fait état de la distribution des bulletins via le système postal local. Le second scénario propose de distribuer les bulletins par la presse indépendante. Un troisième scénario explore la voie du tout-électronique.

Plusieurs options pour s'enregistrer sont envisageables, comme l'enregistrement via Internet, par téléphone, ou même en se rendant dans un lieu dédié à cette tâche. Ceci dépend essentiellement des conditions réelles dans les deux pays, ainsi que des moyens financiers et de l'aide que l'on peut attendre des autorités locales.

Ces scenarii s'efforcent de mesurer la facilité avec laquelle il serait possible de détourner des votes de manière massive. Il est clair vu le contexte que des fraudes auront lieu, cependant il est vital qu'aucun gouvernement ne puisse influencer de manière significative l'issue du scrutin.

2.1 Processus de vote

Une personne possédant le droit de vote doit passer par plusieurs étapes pour pouvoir donner son avis :

1. Récupérer un bulletin de vote (poste, journaux)[1].
2. S'enregistrer auprès de l'autorité compétente, afin de valider son bulletin, et de prouver qu'elle a le droit de vote. La phase d'enregistre-

[1]Cette étape n'est pas requise dans le cas du scénario « tout-électronique » (*cf.* sect. 2.5).

ment peut s'étaler sur une période de plusieurs semaines. Des contrôles d'identité par téléphone peuvent être effectués. A la fin de cette étape, le votant dispose d'un bulletin de vote validé.

3. Voter par Internet en se munissant de son bulletin validé. La période de vote peut durer plusieurs semaines.
4. Attendre la fin de la période de vote, et vérifier que son vote a bien été pris en compte.
5. Prendre connaissance des résultats.

2.2 Bulletin papier

Sur chaque bulletin (voir fig. 2.1), distribué par la poste, ou récupéré dans un journal, se trouvent trois numéros[2] générés de manière aléatoire[3]. Ces numéros peuvent prendre l'aspect suivant :

`2A43B-5C77D-9V9G2-3CSCD-7TAD5`

Les caractères possibles composant ce numéro sont les lettres de l'alphabet, ainsi que les chiffres. Cependant, afin d'éviter toute confusion de la part des utilisateurs, les caractères similaires ont été retirés, comme le '0' et le 'o' ou le 'i', '1' et 'l' par exemple.

Le premier de ces numéros, le *SecT1*[4] est lié au second, le *SecT2*. Ceci est nécessaire pour le processus de vote, comme indiqué au chapitre 7. Le dernier numéro (*SecT3*) est quant à lui indépendant.

Lors de la phase d'enregistrement, le votant doit prouver qu'il possède un bulletin valide en fournissant le premier numéro inscrit sur ce dernier. L'office chargée de l'enregistrement des personnes autorisées à voter vérifie les informations personnelles du votant afin de confirmer qu'elle possède bien le droit de vote. Si tel est le cas, alors son bulletin sera validé, c'est à dire que le *SecT1* sera à même de prendre part au vote.

Au moment du vote, il faudra fournir en plus le troisième numéro. Ce numéro représente à lui seul « une voix » lorsqu'il est accompagné d'une signature. L'utilité du second numéro est expliquée en détail au chapitre 7. Il sert essentiellement à vérifier que le votant communique bien avec les bons

[2] Ce ne sont pas réellement des numéros, car ils contiennent des lettres. On devrait plutôt parler de « codes d'activation ».

[3] Ceci sera probablement réalisé à l'aide d'un générateur de nombre aléatoire quantique.

[4] *SecT* signifie *Secure Token*.

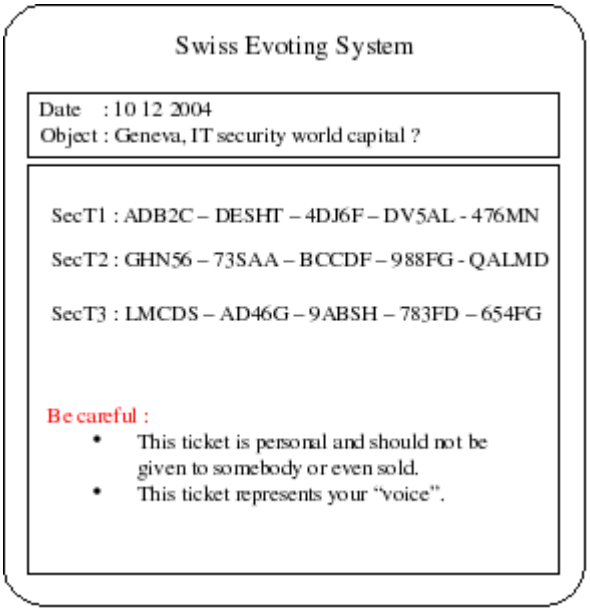

FIG. 2.1 – Exemple d'un bulletin papier contenant les trois *Secure Token*.

serveurs. Il est donc très important de ne pas communiquer ces numéros à un tiers, dès lors que l'enregistrement a été effectué.

Dans le cas du scénario « tout-électronique » (*cf.* sect. 2.5), le votant ne possède pas de bulletin papier, et par conséquent aucun numéro. Le processus est de fait légèrement différent. Le votant doit s'enregistrer sur un site Web en fournissant différentes informations personnelles. Ces informations seront comparées à des données officielles. Si la personne possède le droit de vote, alors il lui sera remis trois *Secure Token*.

2.3 Scenario n° 1 : le système postal

2.3.1 Hypothèses

- Les bulletins sont distribués par courrier postal. Vu qu'ils contiennent des données importantes, il est vital que les gouvernements en place ne puissent pas en avoir connaissance. Ceci pour éviter un détournement en masse des votes ;
- chaque personne désirant voter doit se procurer un bulletin, et doit s'enregistrer auprès d'une autorité compétente afin de le valider ;
- la période d'enregistrement peut s'étendre sur plusieurs semaines. Cette

période est nécessaire pour régler les problèmes d'usurpation d'identité.

2.3.2 Analyse

La distribution doit être réalisée de manière à ce que les gouvernements en place n'aient pas accès à une quantité importante de bulletins. Il faut donc être certain de l'honnêteté de la poste. Si cet objectif est atteint, alors ce système présente nombre d'avantages :

- le détournement en masse de bulletins devient très difficile ;
- chaque votant reçoit un bulletin. Les problèmes d'usurpation d'identité seront fortement réduits, car un bulletin valide est nécessaire pour pouvoir s'enregistrer.

Les bulletins étant imprimés à l'étranger[5], les gouvernements auront plus de peine à effectuer un détournement de masse. De plus, ils doivent recopier ces numéros « à la main », ce qui limite fortement le niveau de virulence de leur malveillance.

2.3.3 Coûts

- Le coût pour générer et stocker les numéros aléatoires est inconnu ;
- la distribution des bulletins par le service postal est évaluée à 400 000 €;
- le nombre minimum de serveurs requis par le protocole est de quatre. Ce nombre peut doubler, ou être revu à la hausse en fonction du besoin en redondance et disponibilité. Estimation du prix par serveur : 4 000 €;
- l'hébergement des serveurs doit se faire dans des centres de calcul haut de gamme. Ces centres doivent être en mesure de parer à une attaque par saturation distribuée[6] d'ampleur moyenne à grande, et d'absorber un trafic important. Dans l'idéal, les serveurs devraient être répartis dans des lieux géographiques différents. Le coût par serveur de ce genre de prestation en Suisse est inconnu.

La totalité des frais fixes se monte à environ 550 000 €.

[5] En supposant que les douanes nous autorisent à importer plusieurs millions de bulletins.

[6] Cette technique est connue sous le nom de *DDoS*, ou *Distributed Denial of Service*. Seules des architectures bien pensées peuvent y faire face de manière efficace.

2.3.4 Conclusion

Cette solution, bien que la plus onéreuse s'avère être la plus efficace pour éviter un détournement en masse des votes. À condition bien sûr que le service postal ne soit pas corrompu, ou fasse l'objet de pressions de la part de l'état.

2.4 Scenario n° 2 : la presse indépendante

2.4.1 Hypothèses

- Les bulletins sont distribués par la presse à grand tirage. Les journaux en question ne doivent pas être sous contrôle de l'état ou subir des pressions de la part de ces derniers. Il est également possible d'utiliser des journaux édités à l'extérieur du pays. Il est important que l'état ne puisse pas avoir accès à la base de numéros aléatoires ;
- chaque personne désirant voter doit se procurer un bulletin en achetant un journal, et doit s'enregistrer auprès d'une autorité compétente afin de le valider ;
- la période d'enregistrement peut s'étendre sur plusieurs semaines. Cette période est nécessaire pour régler les problèmes d'usurpation d'identité.

2.4.2 Analyse

Toute la sécurité du vote, est basée sur ces numéros aléatoires. Se les faire voler signifierait un détournement massif des votes. Afin de pouvoir éditer les journaux, il faudra leur fournir la base de données contenant ces numéros. Ceci peut mettre fortement en péril la sécurité du système. Une confiance absolue est nécessaire dans les journaux choisis. Une solution existe pour éviter de leur fournir la base de donnée : il suffit d'annexer à chaque journal un bulletin de vote déjà imprimé, sous forme de dépliant[7] par exemple.

Une autre faiblesse par rapport au premier scénario est à relever : une personne peut acquérir aisément plusieurs bulletins. Cette personne peut donc être plus facilement tentée de se faire passer pour son voisin. Les problèmes d'usurpations d'identité risquent de devenir non-négligeables.

2.4.3 Coûts

- Le coût pour générer et stocker les numéros aléatoires est inconnu ;

[7] Appelé également *flyer*.

– les coûts concernant l'achat des serveurs ainsi que leur hébergement restent les mêmes que pour le scénario précédent ;
– le coût pour imprimer les bulletins dans différents journaux ou annexer des dépliants est inconnu.

La totalité des frais fixes est inférieure au premier scénario. Estimation : 400 000 €.

2.4.4 Conclusion

Ce scénario se montre moins onéreux, cependant il présente une faille critique au niveau du détournement massif des votes. En se mettant à la place de l'attaquant, cette solution serait la plus intéressante. L'utilisation de dépliant permet de limiter le problème.

2.5 Scenario n° 3 : le tout-électronique

2.5.1 Hypothèses

– Les votants utilisent Internet pour s'enregistrer et acquérir le droit de participer au vote ;
– aucun document en papier n'est requis, hormis peut-être pour imprimer (ou noter) les *Secure Token* qui feront office de sésame lors du vote ;
– la période d'enregistrement peut s'étendre sur plusieurs semaines. Cette période est nécessaire pour régler les problèmes d'usurpation d'identité.

2.5.2 Analyse

Ce scénario offre l'avantage de ne pas devoir utiliser directement des infrastructures potentiellement surveillables par les états. Tout se passe via des serveurs hébergés à l'extérieur des pays concernés. L'authentification se fait donc en ligne. Toute personne désirant voter doit fournir des informations à son sujet. Ces informations doivent être comparées avec une base de données fiables. Le taux de tentative d'usurpation d'identité risque d'être élevé. N'importe qui peut essayer de se faire passer pour son voisin. Il est pratiquement impossible de s'assurer que la personne qui s'enregistre est celle qui correspond aux informations fournies. Cependant des vérifications par téléphone peuvent être envisagées. Un détournement massif est envisageable, à condition de posséder un nombre important d'*IP*, les informations sur la population, ainsi que de la main d'oeuvre pour faire l'enregistrement manuellement.

2.5.3 Coûts

- Le coût pour générer et stocker les numéros aléatoires est inconnu ;
- les coûts concernant l'achat des serveurs ainsi que leur hébergement restent les mêmes que pour le scénario précédent ;
- le coût de la vérification des enregistrements (via le téléphone, ...) est inconnu. Estimation : 100 000 €.

Cette solution est donc largement la moins onéreuse. Estimation : 200 000 €.

2.5.4 Conclusion

Ce scénario se révèle être le moins cher. Il offre un bon compromis entre le coût et le détournement potentiel de vote.

2.6 Synthèse

Le premier scénario est sans aucun doute le meilleur en supposant que le service postal ne soit corrompu, mais aussi le plus cher. S'il l'on veut s'assurer davantage de l'identité des votants, il est également possible de faire des vérifications téléphoniques. Cette remarque est valable pour tous les scenarii présentés.

Le second scénario présente un risque énorme lors de l'impression des journaux et est par conséquent inacceptable. Cependant, s'il s'avère possible de leur annexer des dépliants, cette solution peut se montrer intéressante. Moins sécuritaire que la première, elle permet de distribuer des bulletins de manière plus ou moins fiable pour un coût probablement inférieur à celui du service postal. Là aussi, le détournement massif de vote devient difficile. Cependant, usurper l'identité de son voisin est plus aisée.

Le dernier scénario possède quelques atouts ; financièrement abordable, il souffre seulement de problèmes d'usurpation d'identités. En supposant que l'on renforce le système de vérification post-enregistrement, ce scénario est tout à fait acceptable.

3 Systèmes de votes disponibles

3.1 Travaux existants et contributions

Trois approches dominent la littérature scientifique concernant le vote électronique. Plusieurs méthodes sont basées sur les *mix-nets* introduits par David Chaum [7]. Ils permettent de dissimuler le lien entre le votant et son bulletin au travers de nombreuses permutations. Plusieurs applications sont présentées dans les articles suivants : [35, 36, 39, 28, 2, 19, 26, 21].

D'autres méthodes se basent sur le protocole des signatures aveugles proposé par Chaum [8]. Le votant chiffre et masque son bulletin avant de le présenter à une autorité électorale pour validation. Son bulletin est accompagné de la preuve de son droit de vote. Après que l'autorité responsable ait validé le bulletin, le votant enlève le masque sur le message chiffré et signé afin de récupérer un bulletin signé qui ne peut pas être associé au message chiffré original (*cf.* [9, 18, 20, 33, 14, 32]).

Les méthodes basées sur le chiffrement homomorphique (*cf.* [4, 5, 38, 12, 13, 19, 3, 25, 15, 16]) font usage de certaines propriétés des cryptosystèmes probabilistes, où l'on a démontré l'existence de correspondances entre les opérations sur un certain groupe dans l'espace des *messages en clair* et les opérations sur le groupe correspondant dans l'espace des *messages chiffrés*. Étant donné que de nouveaux cryptosystèmes efficaces avec d'intéressantes proprietés homomorphiques ont été proposés dans la littérature (El Gamal [17], Naccache and Stern [29], et Paillier [34, 15]), les systèmes de vote électronique basés sur cette méthode ont gagné en attention. Dans la littérature, ces propriétés homomorphiques ont le plus souvent été utilisées pour compter les votes comme des *aggrégats*, sans déchiffrer un seul vote (ce qui assure la confidentialité ; *cf.* [13]), ou pour combiner les parties du bulletin d'un votant (*cf.* [5]).

Des approches hétérodoxes du vote électronique ont également été pro-

posées. L'article [24] présente une méthode probabiliste où la robustesse au sens fort n'est pas atteinte, du fait que les résultats sont seulement valides de manière probabiliste. Un autre article [37] met en avant une variation du protocole basé sur les signatures aveugles. Cependant avec cette méthode, la collusion entre les autorités électorales peut compromettre l'absence de reçu (*receipt freeness* : se réferre à l'incapacité de prouver à une personne tiers un certain vote). Le texte [27] propose un protocole qui ne se base pas sur des primitives cryptographiques conventionnelles. Ce protocole offre la confidentialité, mais pas l'absence de reçu. La méthode décrite dans [23] atteint le secret parfait au niveau du bulletin, mais encore une fois l'absence de reçu n'est pas garantit.

Nombre de ces protocoles atteignent beaucoup des propriétés les plus importantes et les plus désirées dans un système de vote électronique. Cependant, certaines exigences importantes ont de la peine à être satisfaite.

Un premier challenge pour les protocoles électroniques est le format des bulletins. Plusieurs protocoles parmi les plus robustes et plus pratiques ont été conçus à la base pour de simples choix binaires. Avec le temps, ils ont été modifiés pour supporter les choix multiples. Plus récemment, ces protocoles ont commencé à prendre en compte les questions ouvertes. Il est également souhaitable de garantir l'absence de reçu et le fait que le système prévienne les comportements coércitifs (menaces envers le votant, corruption, achat ou vente de voix, ...). Beaucoup de protocoles ont atteint ces propriétés en s'aidant d'hypothèses physiques (comme des canaux anonymes ou privés, des cartes à puce, des boîtes noires offrant du chiffrement, ...) ou de partis tiers honnêtes en qui l'on a confiance. Les questions ouvertes augmentent encore la difficulté de concevoir un système empêchant toute coercition. Une personne désirant acheter la voix d'un votant, ou une personne menaçant un votant peut demander à ce dernier d'inclure dans son vote une chaine de caractères identifiable de manière unique, de façon à ce que le vote puisse être reconnu plus tard.

Le protocole le plus récent proposé par David Chaum [10] satisfait plusieurs proprietés souhaitées. Le votant remplit son bulletin à l'aide d'une machine à voter[1] et reçoit un reçu imprimé, dont le chiffrement est basé sur des concepts de cryptographie visuelle. Le reçu peut être authentifié, et l'absence de reçu est garantie par le fait que le contenu de ce dernier est chiffré.

[1]Cette méthode s'éloigne des modèles de vote électronique par Internet. Elle requiert une machine spécifique, et non un simple ordinateur.

Dans l'article [31], une nouvelle méthode efficace basée sur le protocole de l'auteur, *shuffle mix-net protocol* [30], est proposée. Elle autorise les questions ouvertes, mais dépend cependant de conditions physiques et procédurales : le votant doit être surveillé par une autorité électorale afin de ne pas porter à l'extérieur de l'isoloir un *codebook* qui confirme la correspondance entre les codes électorales et les préférences du votant. Si le votant parvient à l'extraire de l'isoloir, alors il sera en mesure de prouver son choix à une tierce personne.

Une grande partie des protocoles théoriques se sont concentrés sur le concept de vérifiabilité universelle plutôt que sur le comptage pratique, vérifiable par le votant. Néanmoins, plusieurs critiques récentes font ressentir le besoin de bulletins physiques, de traces auditables dans les implémentations des systèmes électroniques utilisés, ainsi que de méchanismes compréhensibles par un être humain.

3.2 Mix Nets

3.2.1 Généralités

Les *Mix Nets* s'efforcent de permuter les votes de manière aléatoire afin d'éliminer la correspondance entre le texte chiffré, et son pendant déchiffré. Il est impossible de retrouver le message chiffré à partir du message en clair.

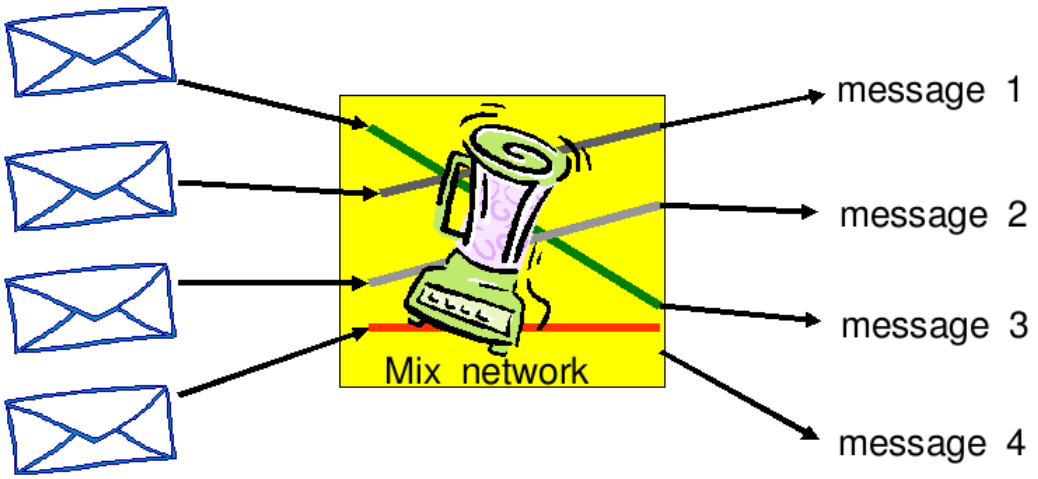

FIG. 3.1 – Permute les votes aléatoirement et déchiffre les entrées.

Parmi les applications possibles, on trouve les tableaux d'affichage « anonymisant » : chacun peut déposer un message sur le tableau d'affichage. A partir du message, il n'est pas possible d'en retrouver l'auteur. Ce système est utilisé par nombre de protocoles théoriques de vote.

3.2.2 Fonctionnement

Pour en illustrer le fonctionnement, imaginons un *Mix Net* basique composé de trois ordinateurs. Chaque machine possède une paire de clefs asymétriques. Le votant va chiffrer son vote avec les clefs publiques de chaque machine prenant part au *Mix Net*. Une fois son vote correctement chiffré, l'électeur va déposer son vote dans le *Mix Net*. La figure 3.1 en illustre le principe.

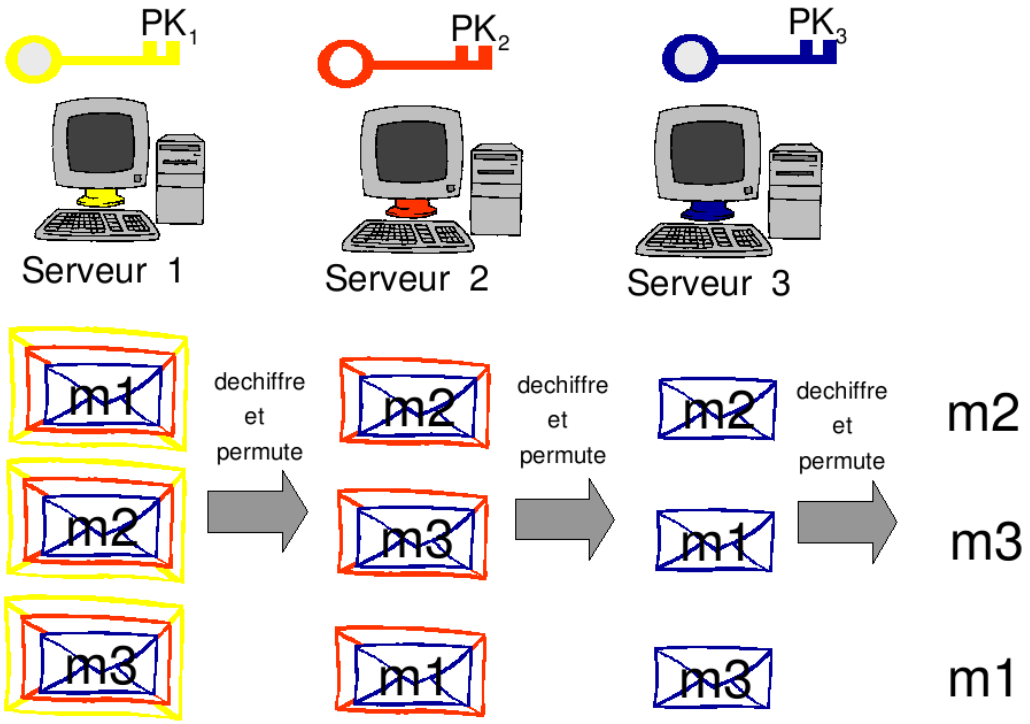

FIG. 3.2 – Basic Mix, introduit par David CHAUM en 1981.

Le serveur 1 reçoit trois messages $m1$, $m2$ et $m3$, chiffrés avec les clefs PK_1, PK_2 et PK_3. Chaque message est déchiffré avec la clef privée du serveur 1, puis les messages sont permutés de manière aléatoire avant d'être transmis au serveur suivant du *Mix Net*. Chaque serveur fait de même jus-

qu'à ce que le message soit totalement déchiffré.

La confidentialité est assurée tant qu'au moins un serveur reste honnête. Que se passe-t-il si un serveur venait à tomber ? Aucun message ne pourrait plus être déchiffré. Une solution envisageable est de séparer les clefs privées de chaque serveur, et de distribuer ces parties aux autres serveurs du *Mix Net*. Ainsi, si un serveur tombe, sa clef privée peut être reconstruite à l'aide des autres serveurs. Dès lors, la confidentialité n'est assurée que lorsqu'une majorité de serveurs est honnête.

Il est également possible qu'un serveur en vienne à falsifier un bulletin. Des solutions existent pour pallier à ce problème : chaque serveur doit prouver qu'il a permuté et déchiffré de manière correcte. Cette preuve doit être signée digitalement et transmise avec le bulletin vers le prochain serveur.

3.3 Chiffrement homomorphique

Un algorithme de chiffrement à clef publique est dit homomorphique s'il satisfait la propriété suivante :

$$E(v_1) \times E(v_2) \times E(v_3) \times \cdots \times E(v_n) = E(v_1 + v_2 + v_3 + \cdots + v_n).$$

Si l'on « multiplie » n bulletins chiffrés (pour la même clef publique), on obtient le chiffrement de la « somme » des bulletins originaux. De cette manière, déchiffrer le produit des bulletins chiffrés donne le résultat de la votation sans avoir eu besoin de déchiffrer chaque bulletin individuellement. La figure 3.3 illustre le principe.

Ce type de protocole présente l'inconvénient de pas pouvoir prendre en compte les questions de type ouvertes.

3.4 Signatures aveugles

Afin d'illustrer ce que sont les signatures aveugles, introduites par David Chaum (*cf.* [8]), imaginons que Bob ne fasse pas confiance à Alice. Par contre, il a confiance en Trent qui, lui, fait confiance à Alice. Bob est d'accord de recevoir un message d'Alice, seulement dans le cas où Trent signe le message. Malheureusement, le message est d'ordre personnel, et Alice ne veut pas que Trent puisse le lire. Grâce aux signatures aveugles, Trent peut signer le message sans pouvoir en lire le contenu.

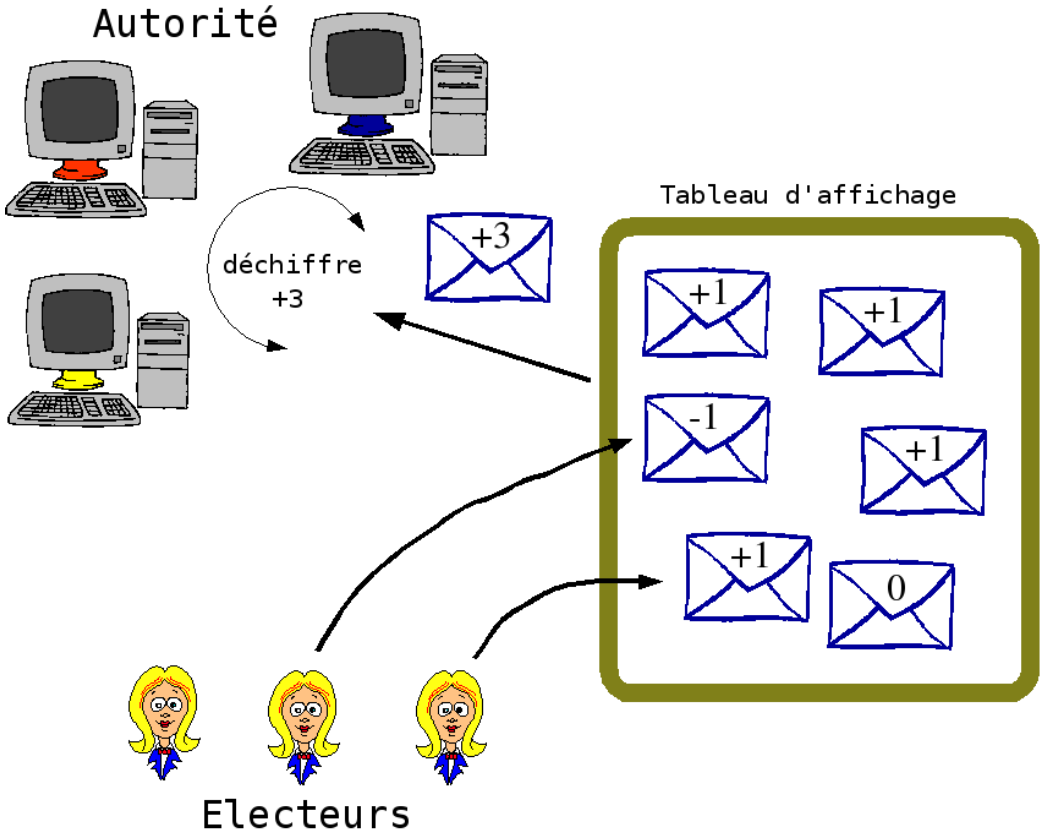

FIG. 3.3 – Principe d'un protocole de vote basé sur le chiffrement homomorphique.

Bruce SCHNEIER donne une bonne analogie de la solution. Dans le monde réel, Alice peut glisser son message dans une enveloppe dont l'intérieur est recouvert de papier carbone. Trent peut alors apposer sa signature sur l'extérieur de l'enveloppe ; elle se verra reproduite sur le message d'Alice. Trent ne peut donc pas prendre connaissance du message personnel. Alice peut maintenant sortir son message de l'enveloppe, et donner le message signé à Bob, qui peut vérifier la signature de Trent.

Sur la figure 3.4, le dépôt du message signé se fait grâce un canal anonyme (en traitillé). Le dépouillement est public.

Le système décrit dans ce document est basé sur les signatures aveugles.

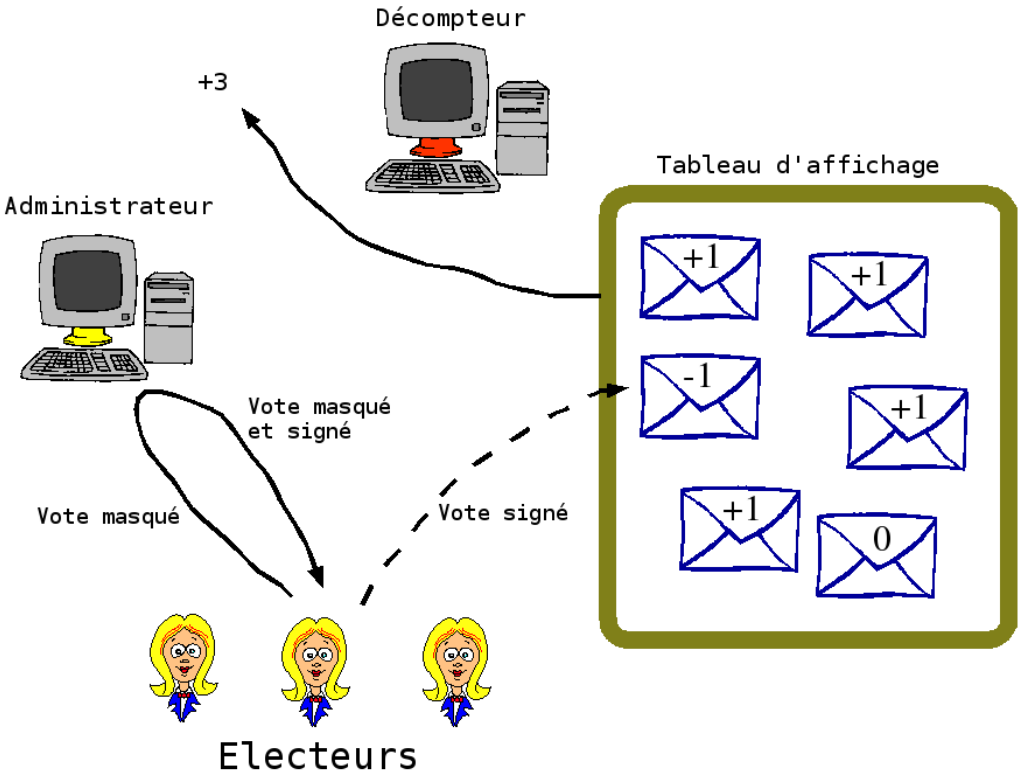

FIG. 3.4 – Principe d'un protocole de vote basé sur les signatures aveugles.

4 Propriétés du vote électronique

Plusieurs propriétés sont requises afin de garantir le bon fonctionnement de l'application de vote :

1. **Précision :**
   - il est impossible de modifier un vote ;
   - il est impossible d'éliminer un vote valide ;
   - il est impossible qu'un vote invalide soit compté.
2. **Démocratie :**
   - seules les personnes ayant le droit de vote peuvent prendre part à la votation ;
   - une personne ayant le droit de vote ne peut voter qu'une seule fois.
3. **Confidentialité :**
   - personne, pas même l'état, est dans la capacité de lier un bulletin de vote avec celui qui l'a émis ;
   - le votant ne peut pas prouver avoir voté d'une certaine manière et se laisser corrompre.
4. **Vérifiabilité :**
   - un votant peut vérifier que son propre vote a bien été pris en compte ;
   - chacun peut vérifier de manière indépendante si tous les votes ont été comptabilisés correctement.

   Les propriétés de précision, de démocratie et de vérifiabilité font parties des systèmes traditionnels du fait de la présence de représentants des différents partis politiques. La propriété de confidentialité est assurée par les isoloirs. La nature des votes par Internet étant particulière, il faut rajouter des propriétés de robustesse :

5. **Résistance à la collusion :**
   - aucune entité électorale (serveur participant au vote), ou aucun groupe d'entités, ne peut introduire de votes ou empêcher un citoyen de

voter. Si toutes les entités font partie de la conspiration, alors cette propriété n'est pas assurée.

6. **Disponibilité :**
   - le système fonctionne correctement durant toute la période de vote ;
   - chaque votant peut accéder au système durant toute la période de vote.
7. **Capacité de reprise :**
   - le système permet à un votant ayant interrompu le processus de vote de reprendre où il en était, ou de recommencer toute la procédure.

5 Eléments de cryptographie

Les protocoles de vote électronique reposent sur des structures cryptographiques. Ces structures, seules ou combinées, répondent aux diverses propriétés souhaitées par les systèmes de vote.

Ce chapitre présente essentiellement les aspects informatifs. Les détails nominatifs se trouvent à la section 8.3.

5.1 Signature digitale

Les signatures digitales sont le pendant électronique des signatures manuscrites, c'est-à-dire un objet « attaché » à un autre (par exemple un fichier), associant de manière irréfutable l'objet à la personne l'ayant signé.

Une signature doit avoir trois propriétés :

1. **Unicité :** la signature de deux objets différents doit être différente.
2. **Inforgeabilité :** Alice ne doit pas pouvoir recréer la signature de Bob.
3. **Vérifiabilité :** chaque personne doit pouvoir en confirmer l'authenticité.

Le système décrit dans ce document utilise le cryptosystème *RSA*, qui fonctionne de la manière suivante : Alice possède une clef publique e, une clef privée d, et un modulo n. Elle signe un objet M en le chiffrant à l'aide de sa clef privée.

$$S = M^d \bmod n$$

Étant donné qu'elle est la seule personne à connaître sa clef privée, elle est aussi la seule à pouvoir générer la signature S pour l'objet M.

Chacun peut dès lors vérifier que S est bel et bien sa signature de l'objet M en chiffrant S avec sa clef publique.

$$M = S^e \bmod n = (M^d)^e \bmod n = M \bmod n$$

Pour ce projet, la taille des clefs est de 2048 *bits*.

5.2 Signature aveugle

Trent est en possession d'une clef publique e, d'une clef privée d et d'un modulo n. Dans un premier temps, Alice masque son message M à l'aide d'une valeur aléatoire k comprise entre 1 et n.

$$B = Mk^e \bmod n$$

Ensuite, Trent le signe.

$$S' = B^d \bmod n = (Mk^e)^d \bmod n = M^d k \bmod n$$

Alice peut désormais retirer le masque et récupérer la signature de l'objet M, faite par Trent.

$$S = (S'/k) \bmod n = M^d \bmod n$$

S étant une signature digitale normale, elle peut être vérifiée selon le principe vu à la section 5.1. Alice doit impérativement vérifier la signature avant d'en faire usage.

5.3 Hash (chiffrement à sens unique)

Un hash est une fonction mathématique. Soit h le hash produit par la fonction de hachage H.

$$h = H(M)$$

Une fonction de hachage possède les propriétés suivantes :

- quelle que soit la taille de M, la taille du hash, h, est constante ;
- l'inverse, H^{-1}, est difficile à calculer, de sorte qu'en ayant h et H, il est difficile de trouver un M tel que $H(M) = h$;
- étant donné M, il est difficile de trouver un autre message M' tel que $H(M) = H(M')$;
- il est difficile de trouver deux message, M et M', tel que $H(M) = H(M')$.

L'algorithme employé pour ce projet est *SHA-1*[1], Secure Hash Algorithm. Il est capable de prendre en entrée n'importe quelle donnée inférieure à 2^{64} *bits* et produit en sortie 160 *bits*.

Les fonctions de hachage sont souvent utilisées avec les signatures digitales. Signer un document volumineux peut se montrer coûteux en terme de calcul. Le hash d'un objet est unique et souvent de taille moindre que l'objet original. C'est pourquoi il est plus pratique de signer le hash de l'objet, plutôt que l'objet lui-même.

[1] www.faqs.org/rfcs/rfc3174

5.4 Engagement

Pour illustrer ce qu'est l'engagement[2], prenons un exemple : Alice doit faire un choix qu'elle désire garder secret. Une foix ce choix réalisé, il lui est interdit de le changer. Bob doit pouvoir être en mesure de vérifier cette condition.

Alice enferme donc le message contenant son choix dans un coffre nécessitant deux clefs. Alice conserve la première clef, et donne la seconde à Bob. Alice ne peut pas ouvrir le coffre pour changer le message sans l'aide de Bob, et ce dernier ne peut pas lire le message sans l'aide d'Alice.

Dans ce projet, un chiffrement à sens unique est utilisé pour assurer cette fonction. Alice génère deux chaines de *bits* aléatoires, R_1 et R_2. Elles les insère dans la fonction de hachage ainsi que le message M, et envoie le tout à Bob avec une seule clef, disons R_1.

$$C = H(R_1, R_2, M)$$

Bob ne peut pas calculer M depuis C grâce aux propriétés de la fonction de hachage, H. De manière similaire, Alice ne peut pas trouver une paire message - chaîne de *bits* (M', R') telle que

$$C = H(R_1, R', M')$$

Ainsi, il lui est impossible de changer son message par M' sans que Bob ne puisse le détecter. En gardant R_2 secret, Alice empêche Bob de passer toutes les chaînes de *bits* possibles[3] avec R_1 dans la fonction de hachage afin de retrouver le message contenant le choix pour lequel s'est engagé Alice.

De manière spécifique à ce projet, l'engagement est utilisé pour confirmer le choix du votant : une fois son choix fait, il ne peut plus en changer, sans que ce ne soit détectable. Le chiffrement à sens unique utilisé est *HMAC-SHA1* :

$$h = H(R_1 \oplus opad, H(R_2 \oplus ipad, M))$$

avec :
- H : fonction de hachage *SHA-1* ;
- *ipad* : suite du byte 0x36 répété 64 fois ;
- *opad* : suite du byte 0x5C répété 64 fois ;
- R_1, R_2 : suite aléatoire[4] de 20 caractères.

D'une longueur de 160 *bits*, le résultat de ce chiffrement, h, sera envoyé, une fois masqué, à l'autorité électorale pour être signé.

[2] *bit commitment* en anglais.
[3] Cette attaque est connue sous le nom de *brute force attack*.
[4] Entropie : $\log_2(20^{62}) \approx 268$ *bits*.

6 Détails du système de vote

Ce chapitre explique de manière précise le fonctionnement du système de vote implémenté dans le cadre de ce projet.

6.1 Généralités

Le système réalisé se base sur le principe des signatures aveugles (*cf.* 5.2). On suppose ici que l'utilisateur a été correctement identifié, et possède un bulletin papier (*cf.* 2.2) validé par l'autorité chargée de l'enregistrement des votants. Des étapes listées à la section 2.1, seule l'étape 3 sera détaillée dans ce chapitre, ainsi que dans le suivant.

Le système de vote présenté dans ce document utilise quatre entités différentes afin de garantir une certaine robustesse : un administrateur, un commissaire, un anonymiseur et un décompteur. Ces entités sont des serveurs autonomes et indépendants les uns des autres. Ils communiquent entre eux via un réseau informatique et ne doivent pas se trouver nécessairement au même endroit physique. La figure 6.1 montre les interactions entre les différentes entités :

1. Le votant envoie un message contenant le choix pour lequel il s'est engagé. Ce message prend la forme d'un hash masqué, accompagné par le *SecT1* du votant.
2. L'administrateur envoie le *SecT1* reçu au commissaire pour vérification.
3. Le commissaire renvoie le *SecT2* dans le cas où le *SecT1* était correct. Dans le cas contraire, il renvoie un message d'erreur.
4. L'administrateur renvoie au votant le hash masqué après l'avoir signé, accompagné du *SecT2*, s'il n'a reçu aucun message d'erreur de la part du commissaire. Si tel est le cas, alors l'administrateur renvoie uniquement le message d'erreur au votant.

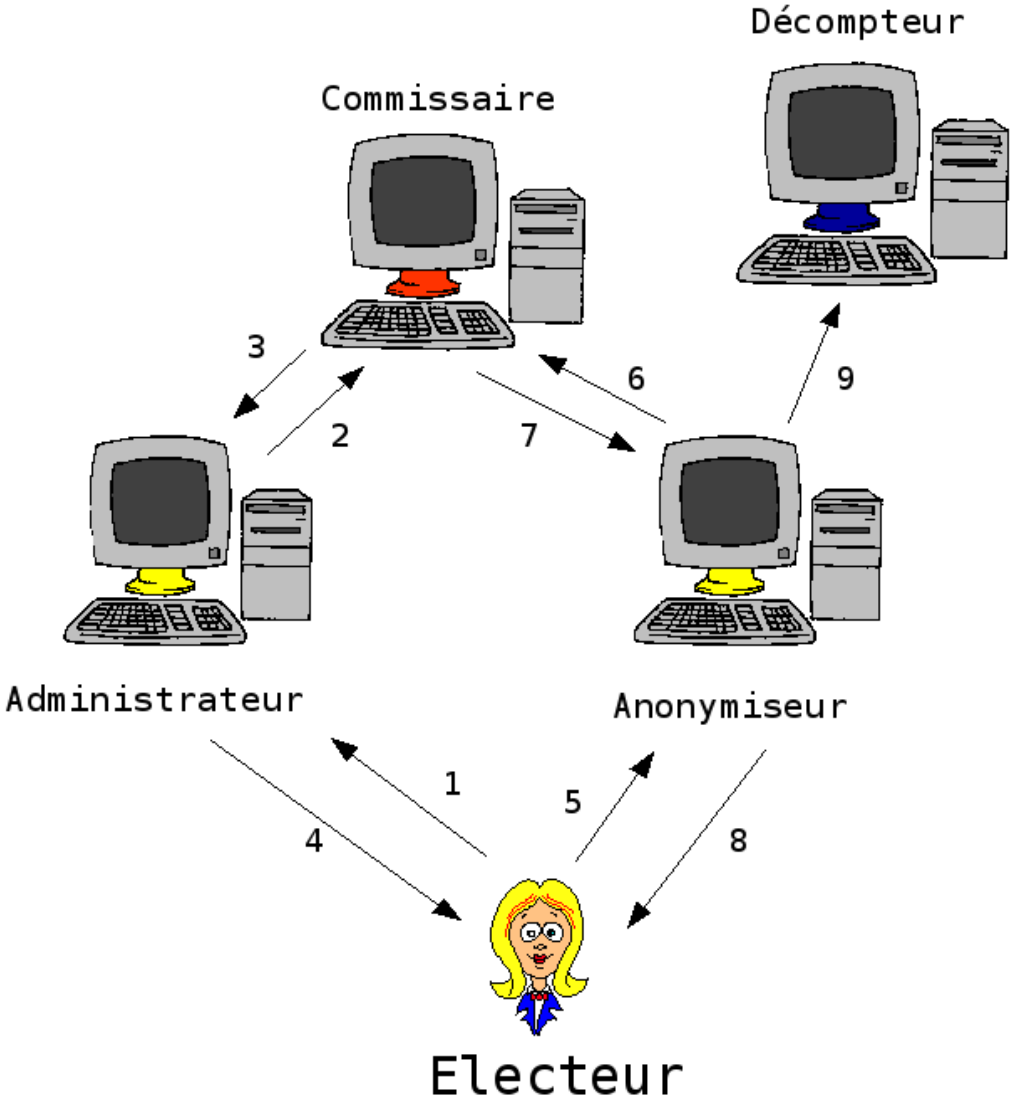

FIG. 6.1 – Vue globale du système de vote.

5. Après avoir vérifié que le *SecT2* corresponde avec celui présent sur le bulletin en papier, le votant démasque la signature, la vérifie et envoie à l'anonymiseur son choix en texte clair, la signature du commissaire, les clefs qu'il a utilisé lors de son engagement, ainsi que son *SecT3*. Ceci est chiffré avec le clef publique du décompteur. L'anonymiseur ne peut donc pas en prendre connaissance. L'électeur envoie également le *SecT1*, chiffré pour l'anonymiseur.

6. L'anonymiseur envoie le *SecT1* au commissaire pour le vérifier.
7. Le commissaire renvoie le *SecT2* si le *SecT1* est valable et n'a pas encore été utilisé. Dans le cas contraire, il renvoie un message d'erreur.
8. Si l'anonymiseur n'a pas reçu de message d'erreur, il enregistre le vote et envoie au votant une confirmation du bon déroulement de l'opération, ou un message d'erreur, accompagné du *SecT2*.
9. L'anonymiseur envoie à la fin de la session de vote[1] l'ensemble des votes au décompteur, afin que ce dernier puisse procéder au dépouillement et compter les voix.

La combinaison *SecT1* - *SecT2* est utile pour garantir au votant qu'il est bien en train de communiquer avec un des serveurs officiels. L'utilisateur fournit le *SecT1* avec les messages envoyés à l'administrateur et à l'anonymiseur, et reçoit en retour le *SecT2* correspondant au *SecT1*. Il doit vérifier que le numéro reçu correspond bien à celui inscrit sur son bulletin papier. Dans le cas où les numéros ne correspondent pas, le votant doit cesser immédiatement la procédure de vote et alerter les autorités responsables de la votation, car une tentative de détournement[2] est en cours.

Ces numéros étant générés aléatoirement, il est presque impossible de pouvoir deviner le *SecT2* correspondant au *SecT1*. De cette manière, il devient très difficile de se faire passer pour un serveur officiel sans avoir connaissance des couples *SecT1* - *SecT2*.

Le *SecT3* est quant à lui totalement indépendant des deux autres numéros. Il est impossible d'associer le *SecT1*, ou *SecT2*, au *SecT3*.

Le vote d'un utilisateur sera valide et compté uniquement s'il satisfait ces deux conditions :

1. Le vote doit avoir été signé par l'administrateur.
2. Le vote doit être accompagné d'un *SecT3* correct.

Chaque serveur possède une paire de clefs cryptographiques. L'utilisateur utilise la clef publique de l'entité avec laquelle il désire communiquer. Les serveurs utilisent entre eux du chiffrement symétrique afin de gagner en efficacité : une clef de session renouvellée à intervalle régulier (*cf.* sect. 7.3) permet de sécuriser les échanges de messages.

[1] La session peut durer plusieurs semaines.
[2] *Spoofing attack* en anglais.

6.2 Administrateur

L'administrateur est l'entité chargée de signer le vote d'un utilisateur. Il reçoit de la part de ce dernier un hash masqué et son *SecT1*. Avant de signer le hash, il doit vérifier que l'utilisateur possède bien un bulletin papier valide et qu'il est en droit de voter. Pour cela, il envoie le *SecT1* reçu au commissaire. Ce dernier le vérifie, et si le *SecT1* est valide et n'a pas encore été utilisé, alors il renvoie à l'administrateur le *SecT2* correspondant.

Si l'administrateur reçoit un message d'erreur de la part du commissaire, alors il ne signe pas le hash et transmet le message d'erreur à l'utilisateur. Si, au contraire, il reçoit un numéro, alors il signe le hash masqué, et le renvoie au votant avec le *SecT2*.

Durant le processus de vote, l'administrateur a accès aux informations suivantes :

- *SecT1* ;
- *SecT2* ;
- le vote pour lequel s'est engagé l'électeur (un hash) masqué ;
- le vote pour lequel s'est engagé l'électeur (un hash) masqué et signé ;
- la clef publique du commissaire ;
- sa clef privée ;
- sa clef publique ;
- la clef de session avec le commissaire ;
- la clef publique servant à signer les votes ;
- la clef privée servant à signer les votes.

Au terme de la session de vote, l'administrateur publie la liste de toutes les signatures qu'il a délivrées.

6.2.1 Malveillances

L'administrateur joue un rôle important dans le processus de vote : c'est lui qui rend un vote « valide » en y apposant sa signature. Que se passerait-il s'il décidait de forger des votes, de les signer et de les donner à l'anonymiseur ?

Pour qu'un vote soit compté, il faut en plus un *SecT3* valide. Or, en aucun cas l'administrateur ne peut avoir accès à cette information, et il est illusoire qu'il puisse deviner des *SecT3* valides. De plus, pour que l'anonymiseur accepte son bulletin, il doit fournir un *SecT1* valide et pas encore utilisé.

D'autre part, il lui est impossible de connaître le choix du votant, grâce aux propriétés des signatures aveugles et au fait que ce choix est hashé avec des données aléatoires (*cf.* 5.4).

6.3 Commissaire

Le commissaire garantit aux électeurs qu'ils communiquent bien avec un serveur officiel. Il détient tous les couples *SecT1* - *SecT2*, ainsi que tous les *SecT3* hashés, c'est à dire toute la « sécurité » du système.

Il doit s'assurer qu'une personne ne puisse voter qu'une seule fois : dès qu'il reçoit un *SecT1* valide provenant de l'anonymiseur, il le marque comme utilisé.

Afin de limiter le pouvoir du commissaire, les *SecT3* sont chiffrés de manière irréversible[3]. Lors du dépouillement, le commissaire passe le *SecT3* reçu en clair dans la fonction de hachage, et recherche le hash obtenu dans sa base afin d'en vérifier la validité.

Pendant le processus de vote, le commissaire a accès aux informations suivantes :

- *SecT1* ;
- *SecT2* ;
- *SecT3* hashé ;
- sa clef publique ;
- sa clef privée ;
- la clef publique de l'administrateur ;
- la clef publique de l'anonymiseur ;
- la clef de session avec l'administrateur et l'anonymiseur.

Au terme de la session de vote, il reporte les éventuels problèmes survenus.

6.3.1 Malveillances

Le commissaire n'ayant pas accès aux *SecT3* en clair pendant la période de vote, il ne peut donc pas insérer de vote. Il n'a également aucunement accès au choix de l'électeur.

Cependant, la base des *SecT3* doit être chiffrée directement après avoir imprimé les bulletins en papier. A un moment donné, cette base est en clair ; par conséquent on est obligé de devoir se reposer sur une entité, et d'avoir confiance en elle. Il est important de s'assurer que des malversations n'aient pas lieu à ce moment là. Ceci est un des points critiques du système. On peut imaginer des solutions satifaisant tous les partis en cause, par exemple en invitant un représentant de chaque parti à venir assister au chiffrement et à la destruction des *SecT3*. Cette étape n'est pas nécessaire si les partis font confiance à une tierce personne.

[3] A l'aide de *SHA-1*.

6.4 Anonymiseur

Cette entité permet de protéger l'anonymat de l'électeur. En effet, si le votant remettait son vote directement au décompteur, il serait possible à ce dernier d'associer un vote à un électeur, en se servant par exemple de l'adresse *IP* utilisée. Ce qui romprait la propriété de confidentialité.

L'anonymiseur reçoit plusieurs informations de la part de l'électeur :
- le *SecT1*, qui est envoyé au commissaire pour vérification ;
- une clef de session chiffrée avec la clef publique du décompteur ;
- un message chiffré avec la clef de session, destiné au décompteur.

D'autre part, il a connaissance des informations suivantes :
- *SecT2* ;
- sa clef publique ;
- sa clef privée ;
- la clef publique du commissaire ;
- la clef de session avec le commissaire.

L'anonymiseur stoque la clef de session, ainsi que le message chiffré dans un fichier, lorsqu'il a reçu la confirmation du commissaire comme quoi le *SecT1* est valide. Les fichiers contenant les votes portent un nom aléatoire, et leur date de création est régulièrement changée[4], afin de ne pas garder d'informations temporelles. De cette manière, on ne peut pas isoler un vote en sachant qu'il a été émis à telle heure. Le contenu d'un tel fichier peut ressembler à ceci :

```
<ballot>
<sessionkey>
uNt2eW4hWAUceElA2442qiq+Ko4daz1ZmudC/DBqYobJKSfRBMx0FTJSEQSCaiXLdBuObNjyrMth
6Vx/ZoPa77OLO9p/o96ArnGXHkge2b58lZnCnIbSh50CtasmjlM8r+bEo6Rc6YTG651w9mb4n2Yd
B8F70E6JUDGf1SRU3CXyBv6VK4s7nMcRpMNp+AYjX7WQeSgautFRC68fxoA1d2p9UPbXOyz72xZA
cDLTA0JxfRovuG3BPWRX0WKehg2eHi/OTaTGLsr4f4Gwbr7Yi9XoRgTKOF8bxI9x97DTtC66s9fU
McybzATABU64crXHqpAI0DT93m+7tEaLp/Q0KaAEcuQiAlDxouWWEehmn/SVs23FVLsAqM5XjcwK
LlN7vMF2P9j1lC3nQNEH1mCYnXEiO8L482R0ZhPZ3dB7Qs6BG1vO1ffN9m16lbvGTlpPQKdVgM1E
Mt1jCpn0yDjttd/U1zUC54sDzcAg4Dtkwpp4glKRgjRtOKm7E2wAPBd7z3S/x1CS7CdLLDl7bjja
CLYecfYlBy3wmmSzww5HXkID/OIc9JO9TbgsYxQcUp7ev65XELf65ivZTvaOUkcGtK9uZdHYdjvE
TuShdzToBRZt2ckS0xpMdjg33hhIztMnJTK1GWUWEjwzecYLlMbGcJ7H2H3tSdWMUqA1fsgDDQ==
</sessionkey>
<vote>
fNPSMu3M1C0oTpe1o5Y8hQXCBvHVBq5UioTPfWufFmB9fZr7ZFkLcOBrkHfSF9vT7mgBikKX7j7b
w7DRQlcz+g1T2axRAQKUZUennxZL+vYknUzFHGwxfbk7Bj94JWSpxuiso59BzcdtlcJdVikv59Zt
Iz1Ao6RNPPKdIgEFFeLz41ODsOJzLkwKk8Xmkg4zL3NafrMV6AlFtUJT5iEG0oibLWclzIsJ3TlM
DKjmEcNDASmZ0nj+dAcwhSZ+spB6gsP6bk+Nw+787prrCBC2HK6K1B+u2kfwnz6+NpnDrZyIeVC4
CtVsJvXszE1znqg2UqiJ+UI4hWmwHz/pgg/sOPn4LpnoO4Xu0S8ciD6XNGo+OmQ0+yF6CmxfTdxn
2HIZr3UMNvVY983EP2+zcm2kvj4LDL7ASBvHkXDXX6oMl7zrRXpUohripUR0zvKQAQaK6axzX9su
tx9wop6oxt+FwicuMpWWwlXpY80DClwL18T/ml3z6pTO1/LG+ICH5mPdne8Fvok4SHUuNk6rWS99
hcf2q1cwejrPMC33l1zQlbnVuc+US5Dv7OMIlQ2Lu8ylDokpYemzaG1KY/5BXJaU+7b2IHz/jLp5
```

[4]Ceci est réalisé à l'aide d'un simple *cron job* utilisant la commande `touch`.

8yt7vzHoidKSrAk3AsAQd0gg8wcaC6GDikvyrZ7HQx7Gwr1Q+BgrFpjUTVQ62szaSLW2v+/Hi8+3
0zSSHUrS/ccFu97ccE4CFXMLaVeu+UjI+3tU+pIeC5RGsFFYSnaEyGfDiiNJNKaCHfsOsW0I/F+0
zKAcnlBfd6Sbrxqu8QK5Dbd7iQeiVTVCYhyrkoEmT5p8PMeFzKueKvzD1wvVLo/Idj6Tk2rPulnq
RXwFRU20GYhZqNf0DowRNjwiPYX1P9g8OW0Z/vI+qI5weLABuLxGu8FLaqKBYETXLj64R4Krb7Xo
/HHe2qpV5nXHa5qN1dWNcQxAOrlNg/+Eraf3CYFyo+P3g42GDyzsSoWaDbOuUJj0yIrxKwsFMBwZ
r7hhnxeSLgHa/JnDtiIq0OODSJLNmeU3imHfJwCnv2JdV37Mq/Yrbw+1XC+MiQQGbaERvJestUex
JDaaTM3U69mhKi47mhImEhYVQV+uE5XC2vhCvzm30dERCc80JQZdZdKKet/T8VwjiO4/iPSz0tep
Qn/BsnrwYwHhqJpZxHZN+uBvw7tTw3Q80xA/2vcWc/s+scfuGLDpUSqCs9TNOyTTpwTEzNSd7xAG
JWP/sbcPfysWYmuXrpOX2fUuJXfrPDgXcS8U7fHybIHNure2Ifoxz2Kxe/4gf5R4e8m8z1xBfNEt
vOGDgGa8uTQ7tBrcrTF7Kmx5YewCPl7c3ZWhq4nz9T+cz99aOdTreG9ip9IivWgOB1V/bN0=
</vote>
</ballot>

A la fin de la session de vote, les fichiers contenant les votes sont envoyés au décompteur. L'anonymiseur rend public les *SecT1* des votes qu'il a stoqués, à des fins de transparence.

6.4.1 Malveillances

Là aussi, l'anonymiseur n'a aucun moyen de récupérer un *SecT3*, et donc aucun moyen d'injecter des votes. Il ne peut pas prendre connaissance des votes, car ils sont chiffrés.

Par contre, il peut associer une personne à son vote avec l'aide du décompteur. La confidentialité est donc rompue si ces deux entités coopèrent.

Point important à relever, l'anonymiseur peut aisément supprimer des votes. Une parade possible est d'utiliser plusieurs couples anonymiseur - décompteur. Si un anonymiseur supprime des votes, cela sera détecté grâce aux résultats des autres couples.

6.5 Décompteur

Le décompteur n'est pas actif durant la session de vote. Son travail ne commence qu'après la fin de la session. Il est chargé de déchiffrer les votes fournis par l'anonymiseur et de comptabiliser les résulats. Il a accès aux informations suivantes :

- le vote en texte clair ;
- le vote dont s'est engagé l'électeur ;
- le vote dont s'est engagé l'électeur signé ;
- les clefs d'engagements ;
- le *SecT3* ;
- sa clef publique ;
- sa clef privée ;
- la clef publique servant à signer les votes.

La marche à suivre pour chaque fichier contetant un vote est la suivante :

1. Déchiffrer la clef de session à l'aide de sa clef privée.
2. Déchiffrer le vote à l'aide de la clef de session.
3. Vérifier l'intégrité du message en recalculant le *HMAC-SHA1*.
4. Reconstruire le hash (le vote pour lequel s'est engagé l'électeur) à l'aide des clefs d'engagement.
5. Vérifier la signature de ce hash.
6. Vérifier que le *SecT3* est valide.
7. Vérifier qu'aucun vote n'a déjà été comptabilisé pour ce *SecT3*.
8. Comptabiliser la voix.

Lorsque le dépouillement est achevé, il publie les résultats, les votes en texte clair, les clefs d'engagements, ainsi que le le vote signé. De cette manière, chacun peut recompter les voix, vérifier la signature de l'administrateur, et s'assurer que son vote a bien été pris em compte, en recherchant parmi les clefs d'engagements.

Rendre public toutes ces informations rend la coercition plus aisée. Un électeur peut prouver ce qu'il a voté en fournissant ses clefs d'engagement. Pour diminuer le risque de coercition, il existe deux possibilités :

1. Ne pas publier les clefs d'engagement. L'électeur ne peut plus prouver son choix. Par contre, il ne peut plus vérifier que son vote a bien été pris en compte.
2. Ne pas publier le vote en texte clair. Là aussi, le votant ne peut pas prouver son vote. Cette fois, il peut vérifier que son vote a bien été pris en compte. Revers de la médaille, il n'est plus possible de vérifier que le décompteur a comptabilisé correctement les votes.

6.5.1 Malveillances

La période de vote étant achevée, il n'est plus possible de contacter l'administrateur et donc d'injecter de nouveaux votes correctement signés.

Le décompteur peut par contre délibérément calculer de manière érronée. Cette malveillance peut être endiguée par la multiplication des couples anonymiseur - décompteur, et par le fait que tout un chacun peut calculer à la main le résultat de la votation. En outre, chacun peut vérifier, à l'aide du code source, que le programme réalise bien les tâches pour lesquelles il a été programmé.

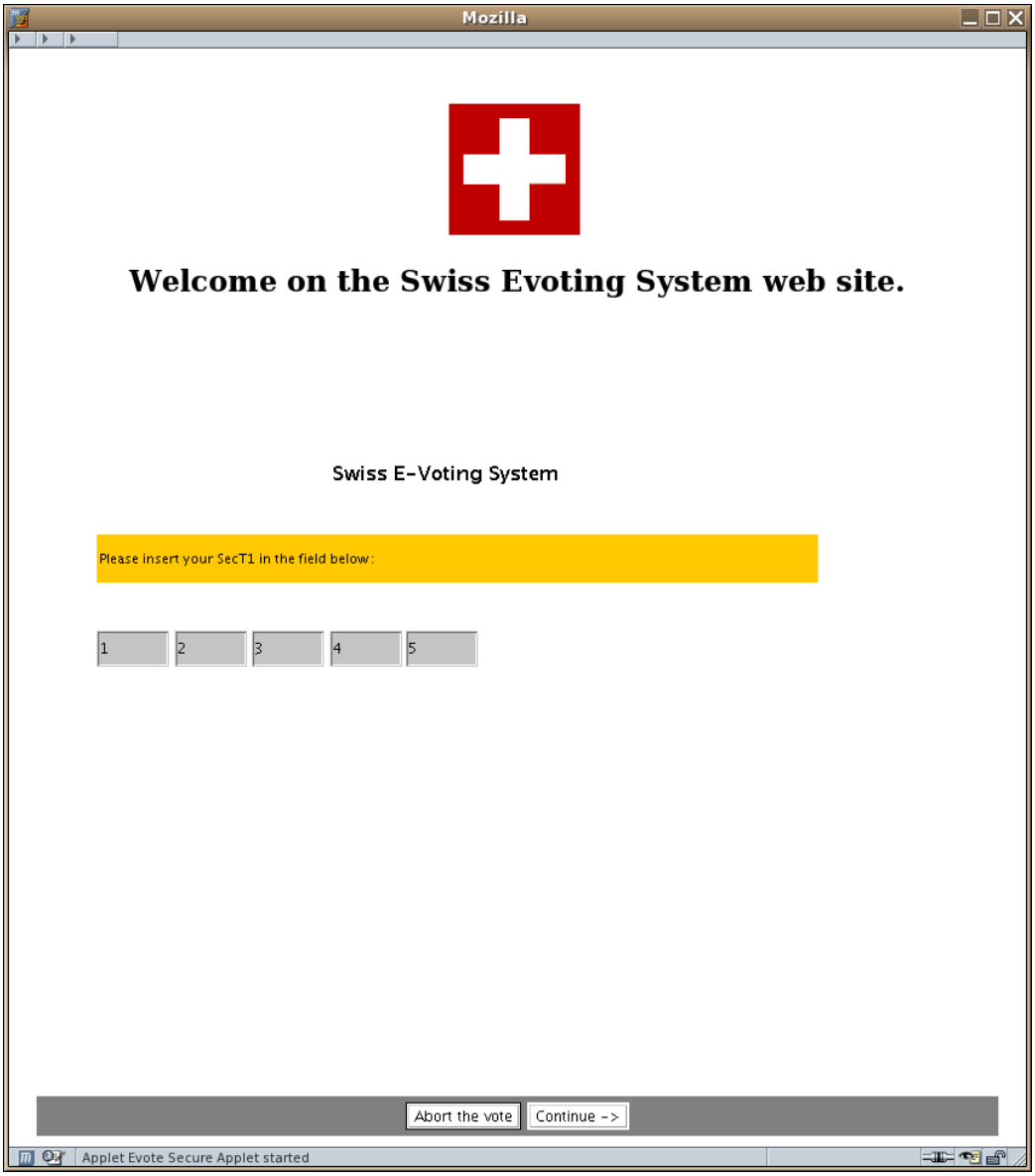

FIG. 6.2 – Saisie du *SecT1*.

6.6 Electeur

L'utilisateur utilise une applet *Java* pour communiquer avec les serveurs chargés de la votation. Il doit se rendre sur un serveur officiel, télécharger l'applet et réaliser les étapes suivantes, après s'être assuré d'avoir un bulletin en papier valide :

1. Entrer le *SecT1* (*cf.* figure 6.2).
2. Répondre à la question de la votation (*cf.* figure 6.3).

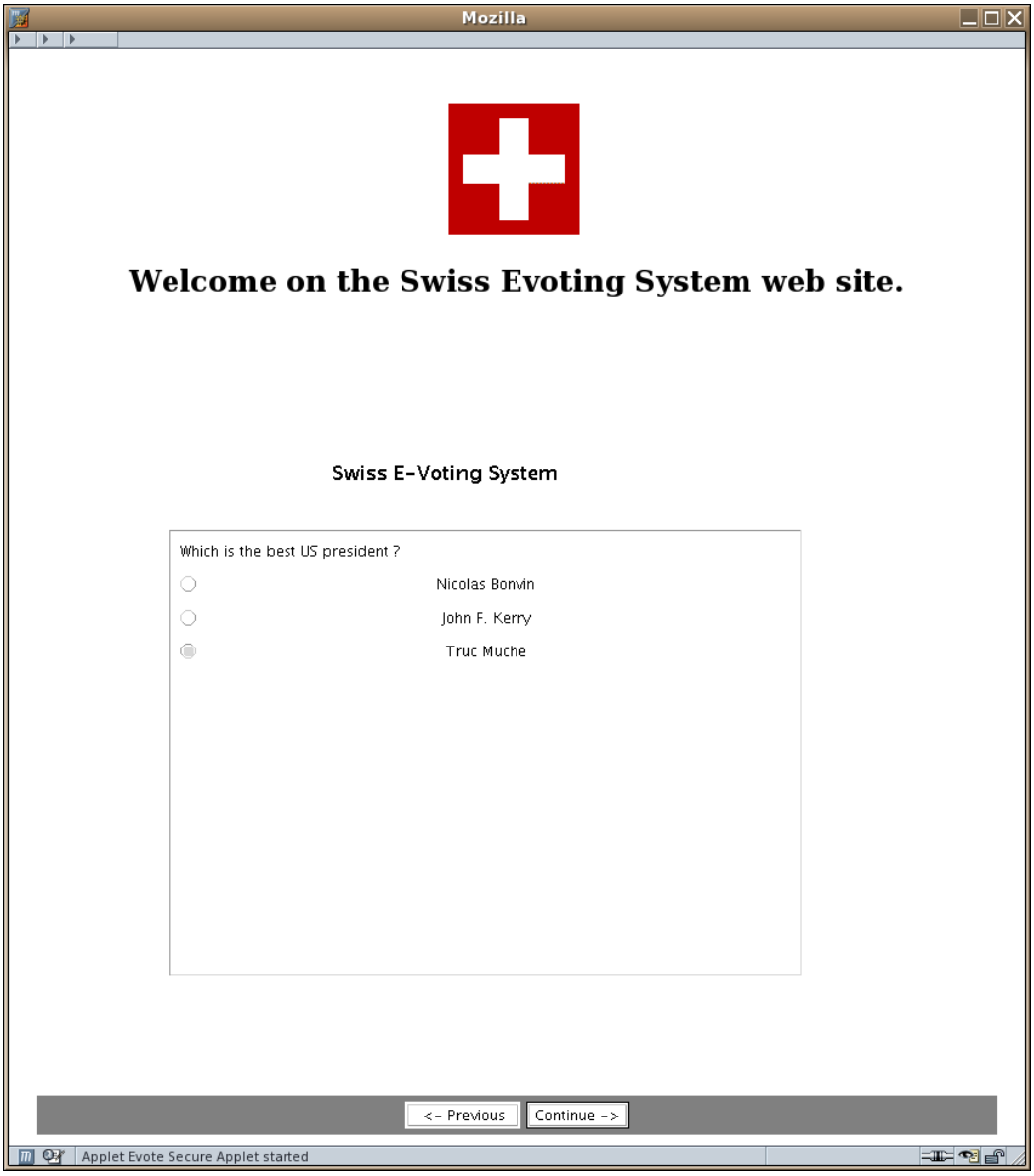

FIG. 6.3 – Question à choix unique.

3. Vérifier les informations saisies. En cas d'erreur, il suffit de revenir en arrière (*cf.* figure 6.4).
4. L'administrateur vient de renvoyer le *SecT2*. Vérifier qu'il corresponde à celui inscrit sur le bulletin en papier. Si tel n'est pas le cas, il impératif

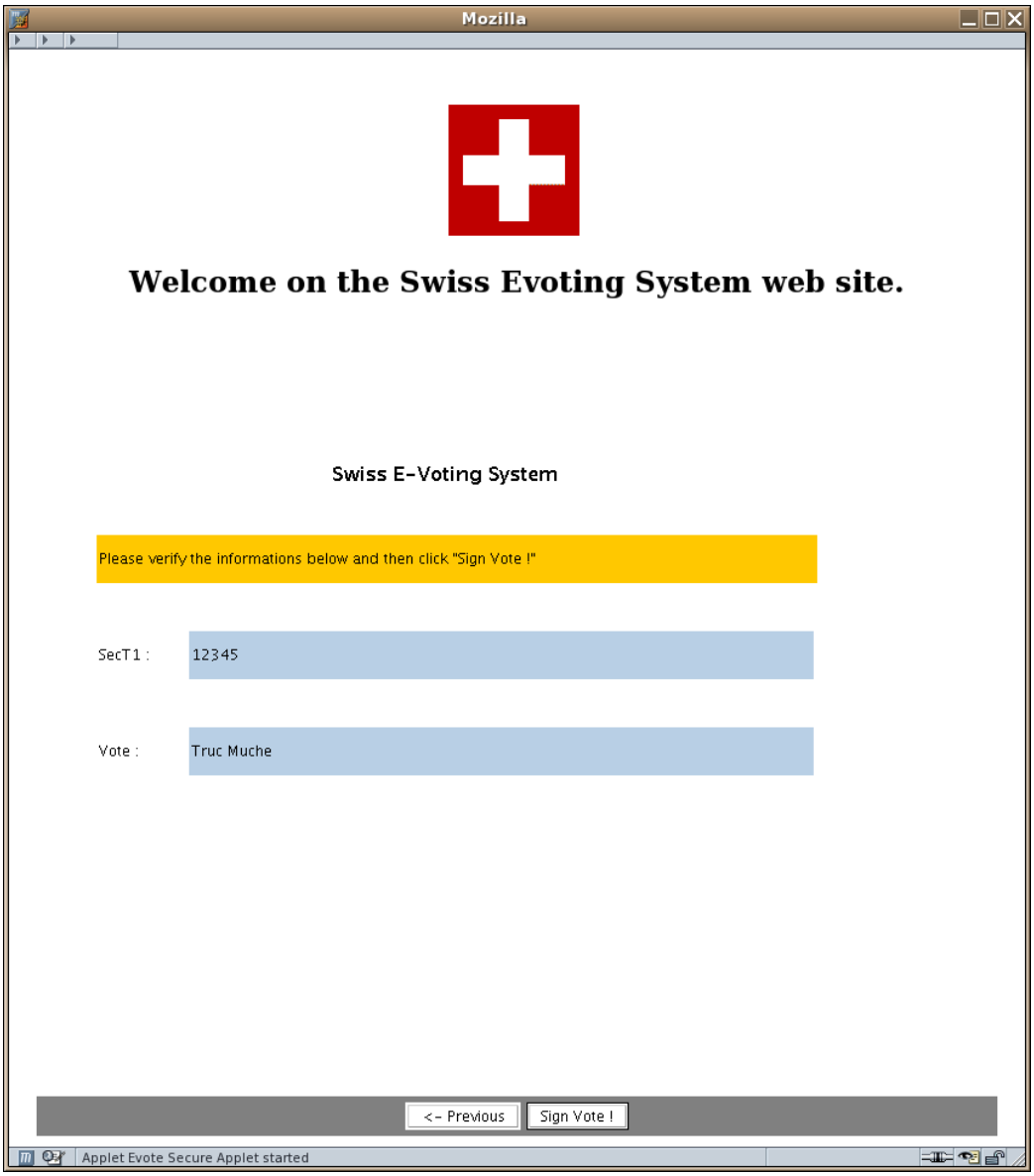

FIG. 6.4 – Vérification des informations saisies.

de clore la procédure de vote, car le serveur ayant répondu n'est pas un serveur officiel (*cf.* figure 6.5).

5. Entrer le *SecT3* (*cf.* figure 6.7).
6. Vérifier les informations saisies. En cas d'erreur, il suffit de revenir en arrière (*cf.* figure 6.8).
7. L'anonymiseur vient de renvoyer le *SecT2*. Vérifier qu'il correspond à celui inscrit sur le bulletin en papier. Si tel n'est pas le cas, il impératif de clore la procédure de vote, car le serveur ayant répondu n'est pas un serveur officiel (*cf.* figure 6.9).
8. La procédure de vote est terminée. Prendre note des clefs d'engagements afin de pouvoir vérifier que le vote a bien été pris en compte lors du dépouillement (*cf.* figure 6.10).

Avant l'étape 7, l'utilisateur peut interrompre à tout moment la procédure de vote, et la reprendre plus tard. À partir de l'étape 7, le vote étant pris en compte, le bulletin en papier de l'utilisateur n'est plus valide. Par conséquent, il ne pourra plus recommencer.

6.6.1 Sécurité de l'applet

L'utilisateur sait s'il dialogue avec un serveur officiel grâce au système de numéros aléatoires. Cependant, comment peut-il s'assurer qu'il utilise bien une applet officielle ? Une fraude possible serait de rediriger l'utilisateur vers un site ressemblant au site officiel, et de lui faire télécharger une applet identique visuellement, mais au comportement interne différent. Par exemple, l'applet pirate pourrait agir exactement comme l'originalle, au détail près qu'elle enverrait le vote en clair, ainsi que l'identité du votant à des personnes malintentionnées.

Il existe la possibilité de signer les applets afin d'en assurer l'authenticité. Cette solution présente l'inconvénient que seuls les utilisateurs expérimentés et concernés par la sécurité font chercher à vérifier la signature de l'applet.

Le système utilisé parvient à protéger l'utilisateur « malgré lui ». Lors du chargement de l'applet depuis un serveur officiel, un numéro aléatoire et totalement indépendant[5] est généré et passé en argument à l'applet. Ce numéro est transmis dans les messages envoyés par le votant, de manière à ce que les serveurs puissent vérifier l'authenticité de l'applet.

Ces numéros aléatoires sont générés par le serveur Web[6] servant la page contenant l'applet. Ils sont insérés dans une base de données avec un temps

[5] Il n'existe aucune relation avec l'identité de l'utilisateur, ou avec les différents *SecT*.
[6] Les numéros sont générés par le langage de script *PHP*.

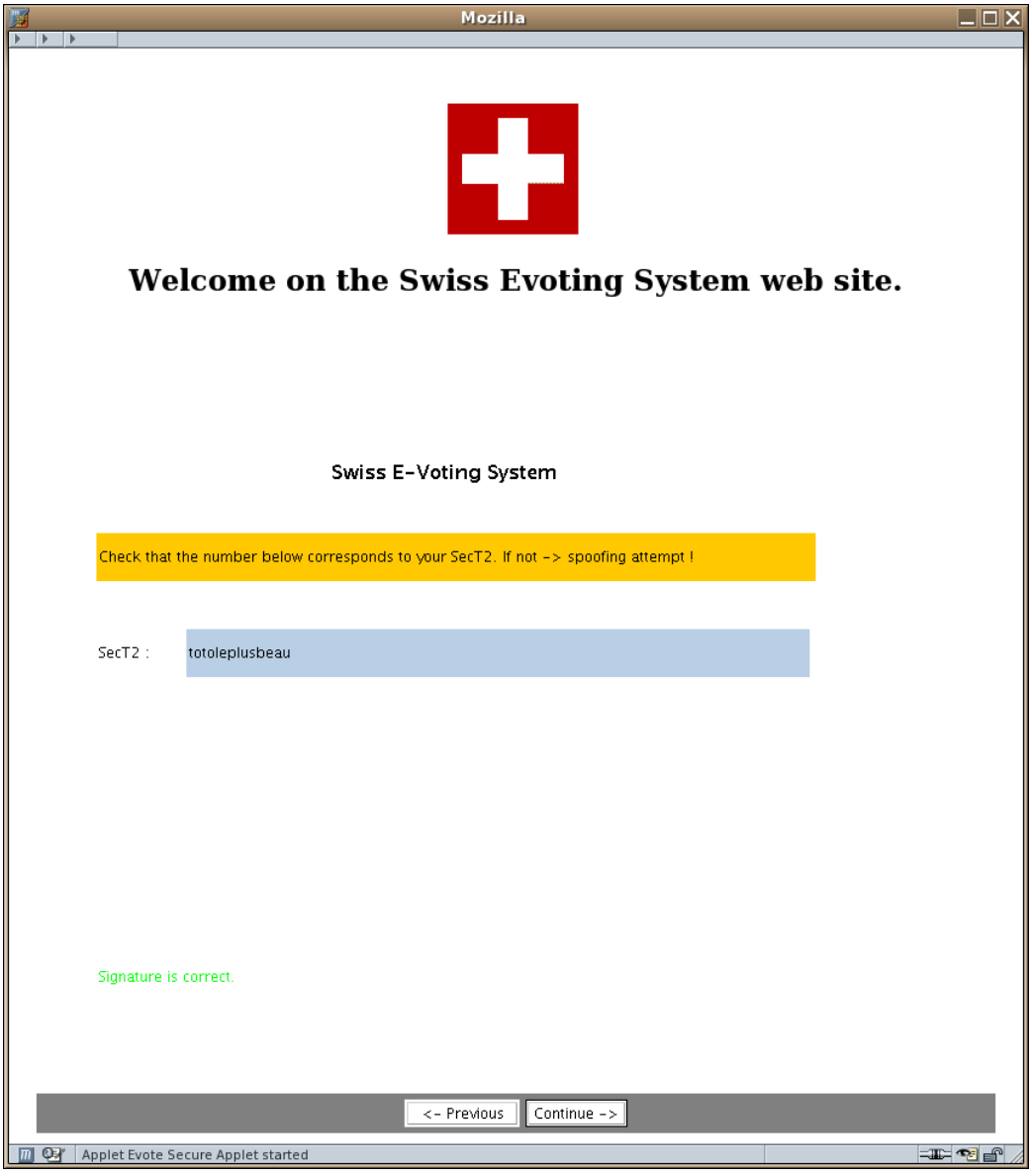

FIG. 6.5 – Réception du hash signé par l'administrateur et du *SecT2*.

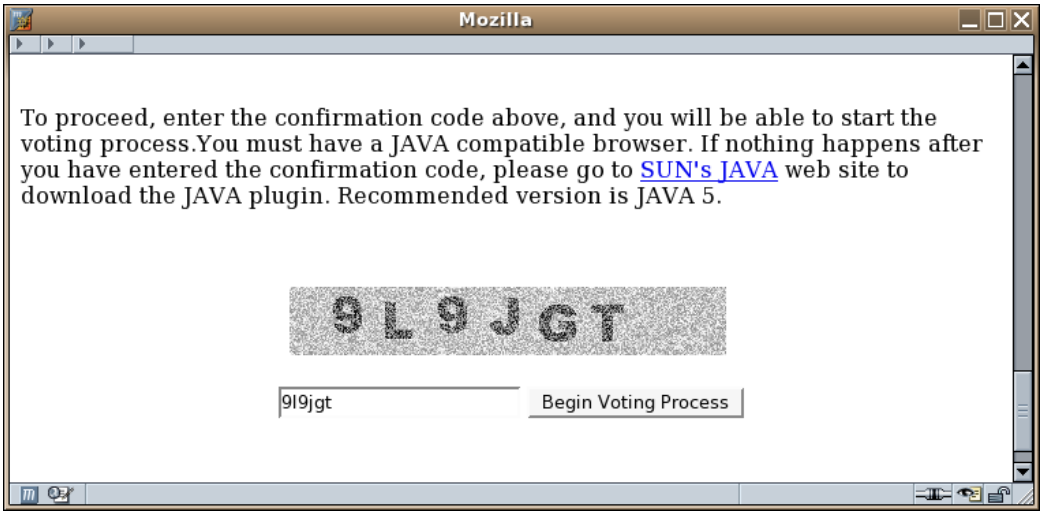

FIG. 6.6 – L'utilisateur doir reconnaître les caractères et les insérer dans le formulaire afin d'être autorisé à télécharger l'applet.

de vie limite. Cette base de donnée est accédée à son tour par les serveurs voulant vérifier une applet.

Les paramètres d'une applet *Java* sont inscrits directement dans le code *HTML* de la page servant l'applet. Il faut donc empêcher qu'un robot vole ces numéros et les utilise sur un site pirate. Pour répondre à cela, l'utilisateur doit reconnaître une suite de caractères présents sur une image (*cf.* figure 6.6). Ceci est en principe très difficile à rélaiser à l'aide d'un programme informatique. De plus, le serveur Web utilise *SSL/TLS* pour protéger les communications.

Plusieurs autres menaces peuvent mettre en péril le bon déroulement de la procédure de vote. Une applet tourne du côté client, à l'intérieur d'un navigateur. Aucun contrôle sur le poste client n'est possible, et il faut donc faire confiance à plusieurs briques matérielles ou logicielles, comme le système d'exploitation utilisé ou la machine virtuelle *Java* par exemple. Ces menaces sont communes pour à toutes applications informatiques.

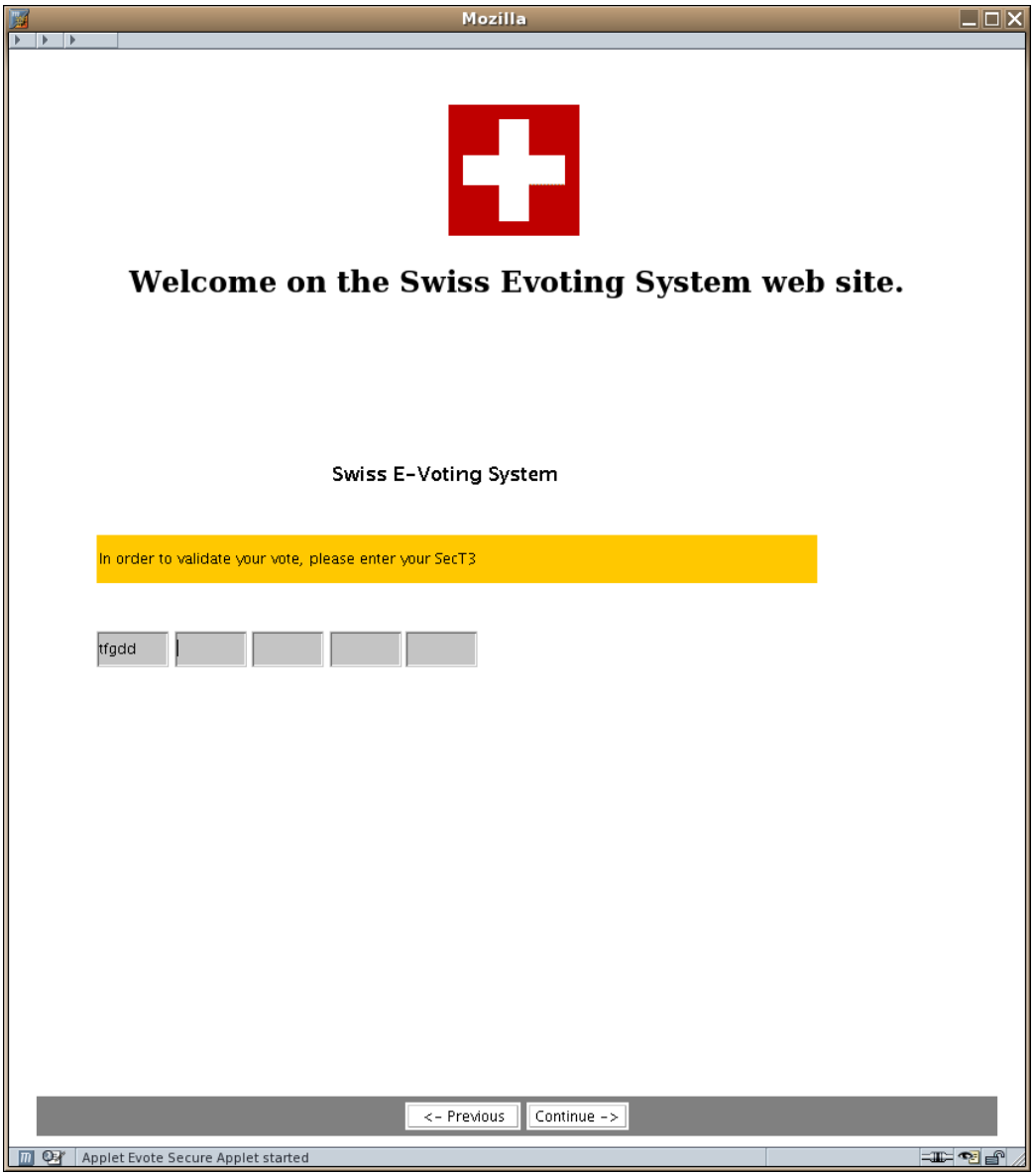

FIG. 6.7 – Saisie du *SecT3*.

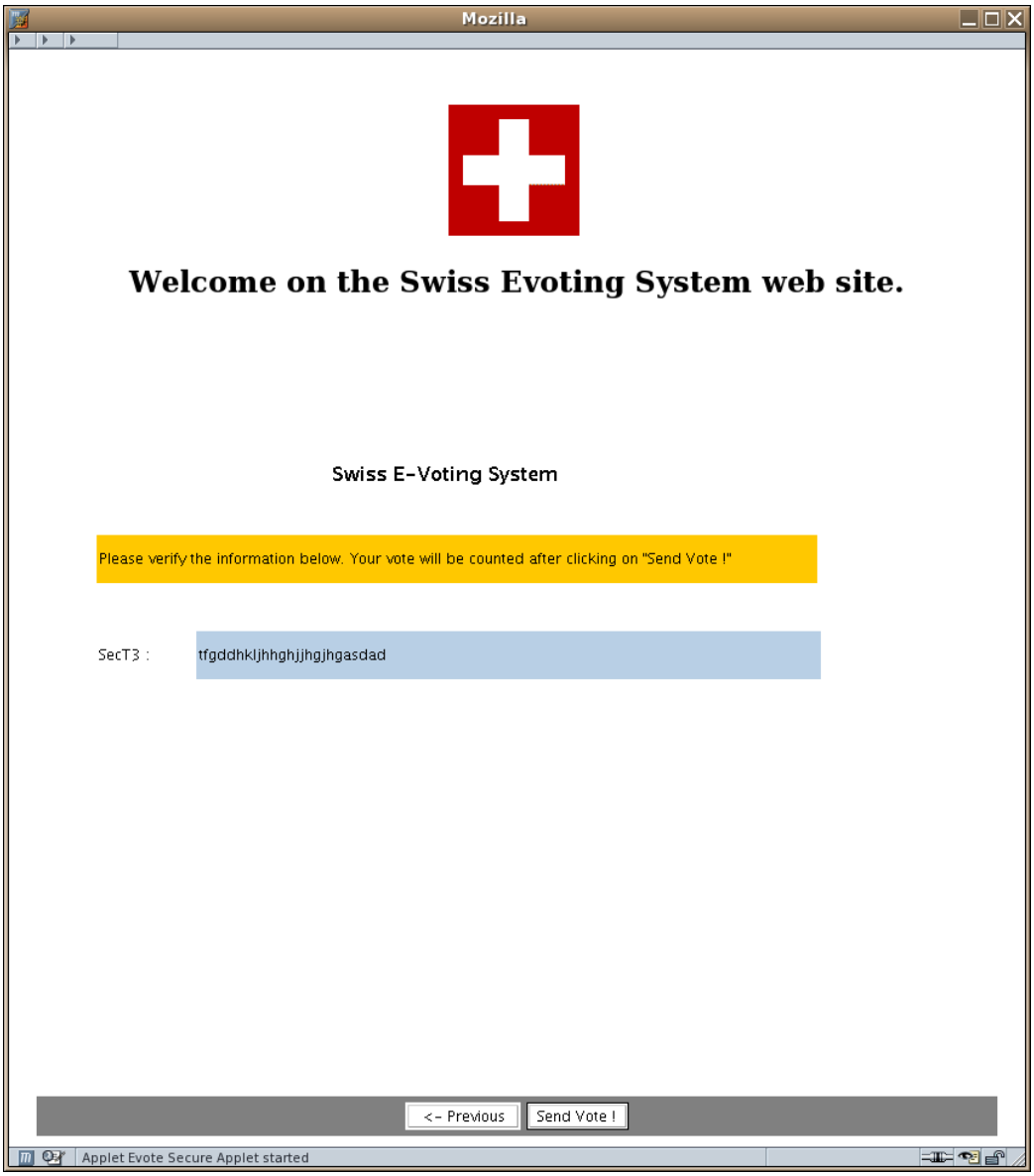

FIG. 6.8 – Vérification des informations saisies.

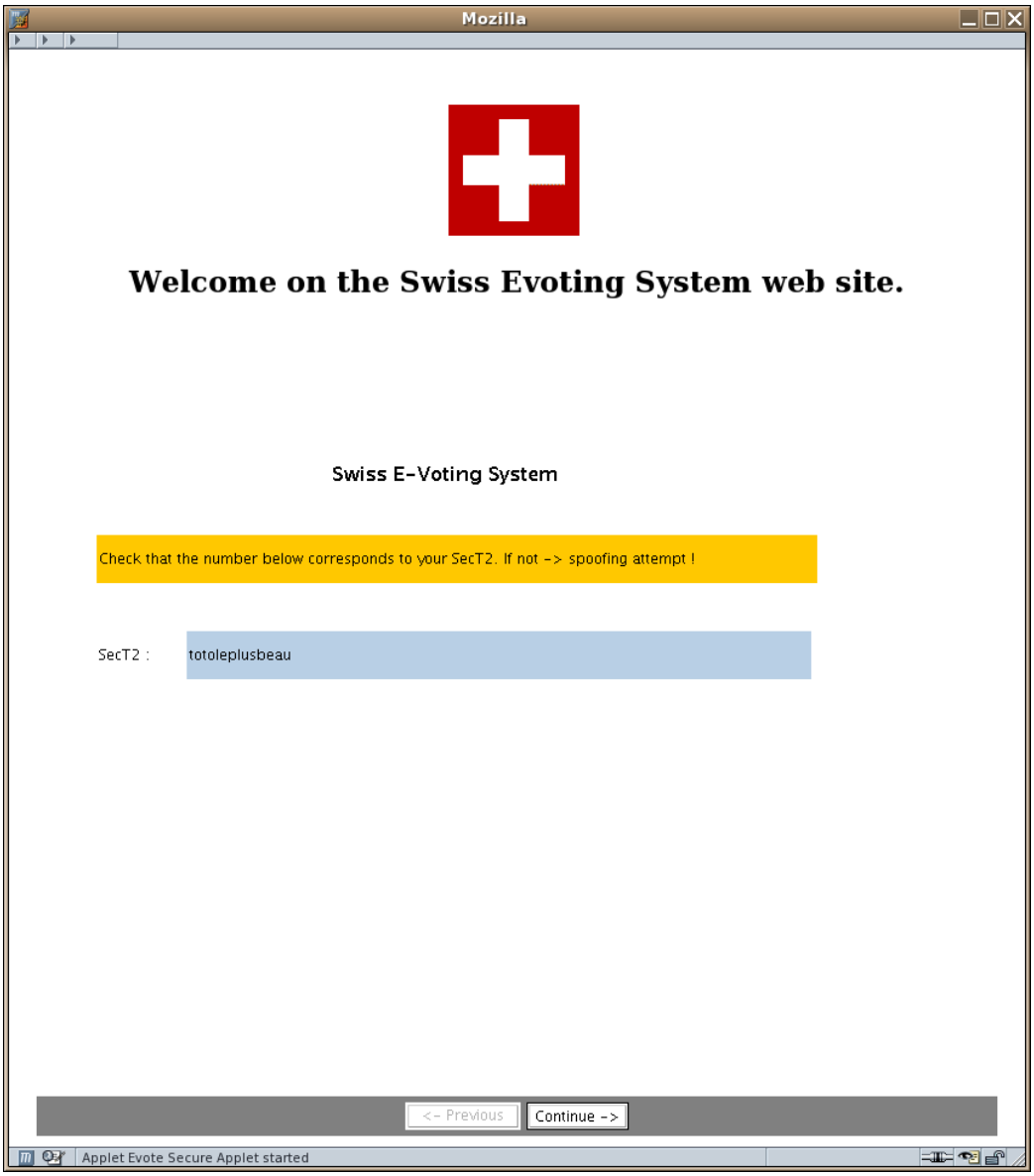

FIG. 6.9 – Vérification du *SecT2*.

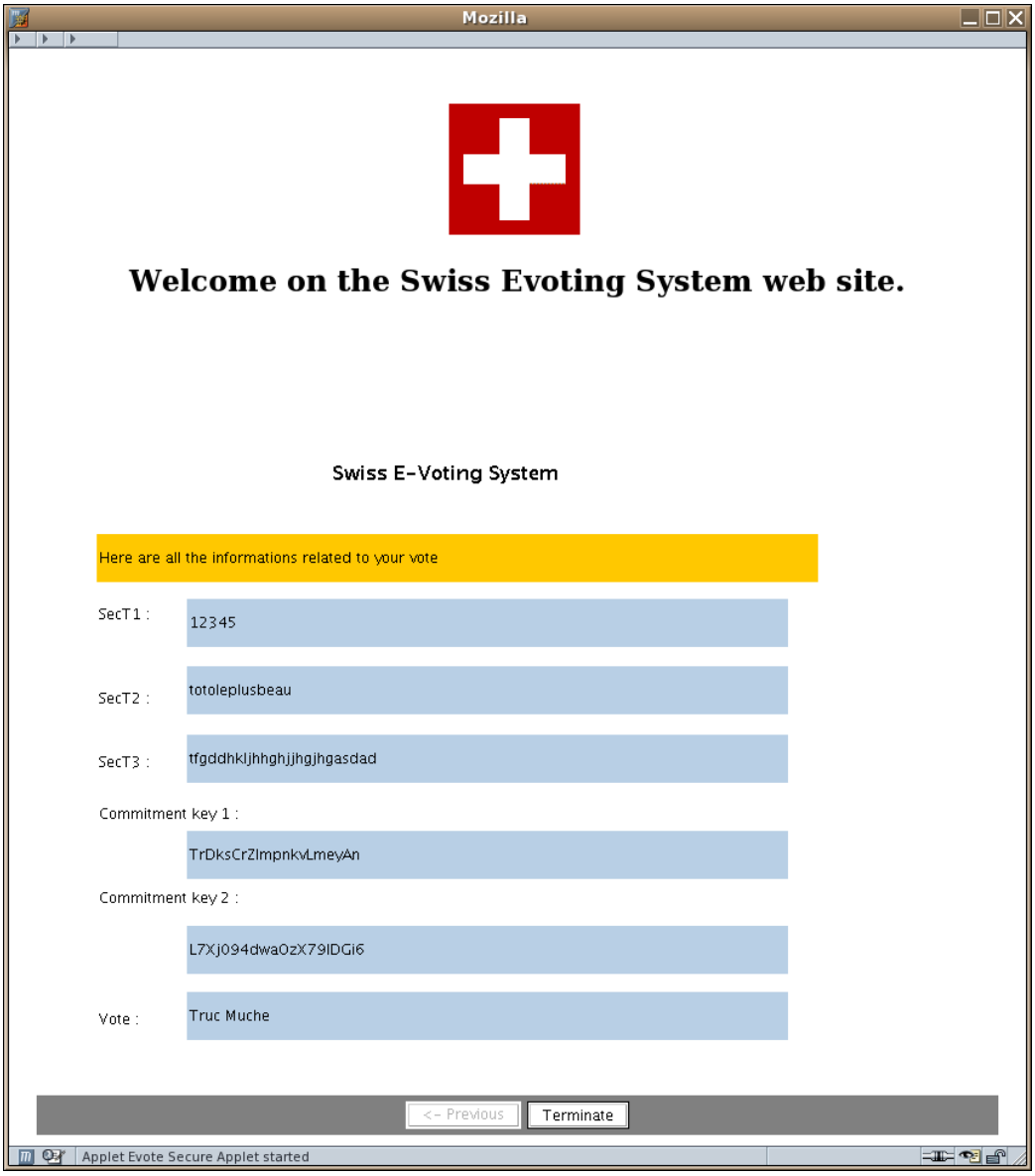

FIG. 6.10 – Résumé des informations saisies et reçues. Les clefs d'engagement permettent de vérifier que le vote a bien été décompté.

7 | Protocole

7.1 Format des messages

Deux formats de messages sont utilisés au niveau applicatif dans ce projet. Le premier concerne les messages envoyés par l'applet vers un serveur de vote, ainsi que les messages nécessaires au renouvellement de la clef de session (*cf.* sect. 7.3) utilisée par les serveurs de vote entre eux. Ces messages, M, prennent la forme suivante :

$$M = E_{PK}\{S\}\#E_S\{[type]\#[data]\#[padding]\#[hmac([type][data][padding])]\#[alea]\}$$

avec

$E_{PK}\{\cdot\}$: chiffrement avec la clef publique PK du destinataire ;
$E_S\{\cdot\}$: chiffrement avec la clef de session S ;
$\#$: séparateur ;
$type$: type du message émis ;
$data$: donnée du message ;
$padding$: bytes aléatoires ;
$hmac(\cdot)$: *HMAC-SHA1* à l'aide de la clef de session ;
$alea$: bytes aléatoires.

La deuxième sorte de messages est utilisée pour communiquer entre les serveurs, et lorsqu'un serveur renvoie des données à l'applet. Ce format est similaire au premier, hormis que seul le chiffrement symétrique est utilisé :

$$M = E_S\{[type]\#[data]\#[padding]\#[hmac([type][data][padding])]\#[alea]\}$$

avec

$E_S\{\cdot\}$: chiffrement avec la clef de session S ;
$\#$: séparateur ;
$type$: type du message émis ;
$data$: donnée du message ;
$padding$: bytes aléatoires ;
$hmac(\cdot)$: *HMAC-SHA1* à l'aide de la clef de session ;
$alea$: bytes aléatoires.

Le padding du message en clair a lieu avant son authentification et son chiffrement comme recommandé dans l'article [41]. Tous les messages sont encapsulés dans un format de plus bas niveau, nécessaire pour la reconstruction des paquets au niveau du réseau :

$$M' = [STX][M][ETX]$$

avec

M' : message envoyé sur le réseau ;
M : message généré et chiffré au niveau applicatif ;
STX : *Start of Text*, byte `0x02` ;
ETX : *End of Text*, byte `0x03`.

7.1.1 Type

Le contenu de chaque message varie en fonction de l'expéditeur et du destinataire. À chaque message correspond un type :

0. Message de l'applet vers l'administrateur pour faire signer le vote.
1. Message de l'administrateur au commissaire afin de vérifier le *SecT1*.
2. Message du commissaire à l'administrateur qui contient le *SecT2* correspondant ou un message d'erreur.
3. Message de l'administrateur à l'applet contenant le vote signé, ou un message d'erreur.
4. Message de l'applet à l'anonymiseur afin de déposer le vote.
5. Message de l'anonymiseur au commissaire afin de vérifier le *SecT1*.
6. Message du commissaire à l'anonymiseur qui contient le *SecT2* correspondant ou un message d'erreur.
7. Message de l'anonymiseur à l'applet contenant la confirmation de la prise en compte du vote, ou un message d'erreur.

8. Message[1] de l'anonymiseur au décompteur afin de transférer les votes à la fin de la session.
20. Message de l'anonymiseur ou de l'administrateur au commissaire afin de demander le renouvellement de la clef de session.
21. Message du commissaire à l'administrateur contenant la future nouvelle clef de session.
22. Message du commissaire à l'anonymiseur contenant la future nouvelle clef de session.
23. Message de l'anonymiseur ou de l'administrateur au commissaire confirmant l'utilisation de la nouvelle clef de session.

7.1.2 Data

Les données transférées par les messages dépendent de leur type :

0. [SecT1]%[hash (engagé) masqué]%[spoofID].
1. [SecT1].
2. [SecT1]%[réponse].
3. [hash (engagé) masqué et signé]%[réponse].
4. $[E_{PK\_décompteur}\{S\}]\%[E_S\{\cdot\}]\%$[SecT1].
5. [SecT1].
6. [SecT1]%[réponse].
7. [réponse].
8. [hash (engagé) signé]%[vote en texte clair]%[hash (engagé)]%[clef d'engagement 1]%[clef d'engagement 2]%[SecT3].
20. [alea A]%[signature de A].
21. [nouvelle clef de session]%[signature].
22. [nouvelle clef de session]%[signature].
23. [nouvelle clef de session]%[réponse session].

où

- *spoofID* est le numéro aléatoire généré au moment du téléchargement de l'applet (*cf.* sous-sect. 6.6.1) ;
- *réponse* vaut « SecT2 correspondant », « Incorrect SecT1 » ou « SecT1 already used » ;
- *réponse session* vaut « OK-Admin » ou « OK-Ano ».

[1] Ce type de message n'est pas implémenté dans la version actuelle, car les votes seront probablement transférés d'une manière plus efficace au décompteur. L'utilisation de *SSH* peut, par exemple, se montrer pertinente.

7.1.3 Padding

Il est important que la longueur des messages échangés soit constante afin de ne révéler aucune information sur leur contenu. Une longueur fixe est choisie pour chaque type de message échangé. Si le message original est inférieure à la taille choisie, on y insère des caractères aléatoires jusqu'à obtenir la taille désirée. Ces caractères sont générés de la même manière que l'aléa.

Il est utile de noter que le *padding* décrit dans cette sous-section diffère de celui présent lors du chiffrement. Le *padding* utilisé pour les opérations de chiffrement est conforme au standard *PKCS#1 v1.5*.

7.1.4 Hmac

Le champ [*hmac*] contient le résulat de la fonction de chiffrement à sens unique *HMAC-SHA1* qui prend comme argument le type et les données. La clef utilisée avec cette fonction est celle utilisée pour chiffrer le message. Ce champ est recalculé à la réception du message et comparé afin de pouvoir s'assurer de l'intégrité du message.

7.1.5 Alea

Lors de questions de type binaire, où l'on attend une réponse du genre « oui » ou « non », les messages chiffrés sont plus vulnérables aux attaques, car le texte chiffré est connu. Il est donc important d'ajouter lors du chiffrement une composante aléatoire.

Pour l'implémentation actuelle, ce champ est composé de 60 caractères appartenant à un jeu de 62 caractères. L'entropie vaut donc :

$$\log_2(60^{62}) \approx 366 \ bits$$

7.2 Échange de messages

Pour réaliser la procédure de vote dans sa totalité, l'électeur doit envoyer seulement deux messages. La figure 7.1 illustre le premier message envoyé par l'applet, ainsi que l'interaction entre l'administrateur et le commissaire. Les lettres de la figure représentent les actions effectuées, et les chiffres correspondent aux données échangées :

A :

- génère une clef de session S ;
- s'engage pour le vote ;

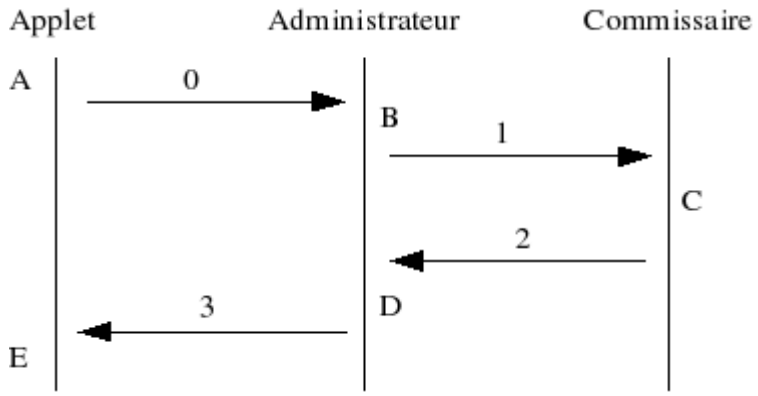

FIG. 7.1 – Premier message envoyé par l'applet.

– masque le vote ;

B :

– vérifie le *HMAC-SHA1* du message ;

C :

– vérifie le *HMAC-SHA1* du message ;
– vérifie que le *SecT1* soit valide et puisse encore être utilisé pour voter ;
– si tout est correct, envoie le *SecT2* correspondant ;

D :

– vérifie le *HMAC-SHA1* du message ;
– signe le hash (engagé) et masqué

E :

– vérifie le *HMAC-SHA1* du message ;
– retire le masque du hash ;
– vérifie la signature de l'administrateur ;
– vérifie le *SecT2* ;

0 :

– $[E_{PK\_administrateur}\{S\}]\#[E_S\{[0]\#[[SecT1]\%[\text{hash (engagé) masqué}]\%[spoofID]]\#[padding]\#[hmac]\#[alea]\}]$;

1 :

– $E_{SessServer}\{[1]\#[SecT1]\#[padding]\#[hmac]\#[alea]\}$;

2 :

– $E_{SessServer}\{[2]\#[SecT2]\#[padding]\#[hmac]\#[alea]\}$;

3 :

– $E_S\{[3]\#[[\text{hash (engagé) signé}]\%[SecT2]]\#[padding]\#[hmac]\#[alea]\}$;

Les interactions provoquées par l'envoi du second message sont décrites par la figure 7.2 :

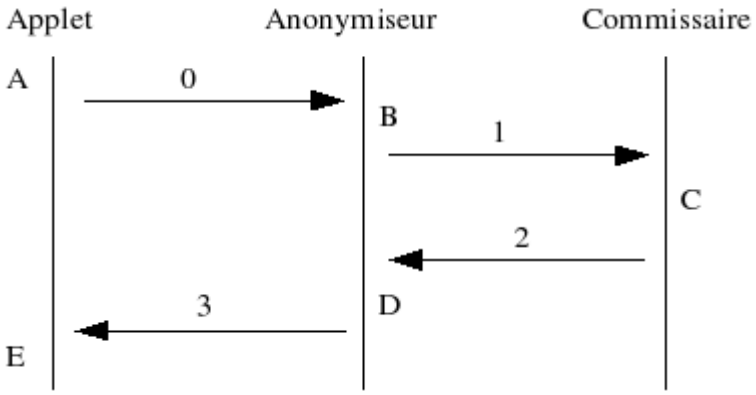

FIG. 7.2 – Second message envoyé par l'applet.

A :
– génère deux clefs de session S_A et S_B ;
B :
– vérifie le *HMAC-SHA1* du message ;
C :
– vérifie le *HMAC-SHA1* du message ;
– vérifie que le *SecT1* soit valide et puisse encore être utilisé pour voter ;
– marque le *SecT1* comme ayant voté ;
– si tout est correct, envoie le *SecT2* correspondant ;
D :
– vérifie le *HMAC-SHA1* du message ;
– enregistre le vote ;
– brasse les votes ;
E :
– vérifie le *HMAC-SHA1* du message ;
– vérifie le *SecT2* ;
0 :
– $[E_{PK\_anonymiseur}\{S_A\}]\#[E_{S_A}\{[4]\#\ [E_{PK\_décompteur}\{S_B\}]\#[E_{S_B}\{[$[hash (engagé) et signé]%[vote en texte clair]%[hash engagé]%[clef d'engagement 1]%[clef d'engagement 2]%[SecT3]]#[padding]#[hmac]#[alea]$\}]$ #[padding]#[hmac]#[alea]$\}]$;
1 :
– $E_{SessServer}\{[1]\#[SecT1]\#[padding]\#[hmac]\#[alea]\}$;
2 :

– $E_{SessServer}\{[2]\#[SecT2]\#[padding]\#[hmac]\#[alea]\}$;

3 :

– $E_{S_A}\{[3]\#[[OK]\%[SecT2]]\#[padding]\#[hmac]\#[alea]\}$;

7.3 Renouvellement des clefs de session

L'administrateur, l'anonymiseur et le commissaire utilisent une clef de session symétrique pour diminuer le temps de calcul nécessaire au traitement de chaque message. Pour assurer une certaine sécurité au système, cette clef de session est renouvelée chroniquement.

Les clefs de session sont générées par le commissaire et transmises à l'administrateur et à l'anonymiseur en utilisant leur clef publique. Le commissaire a la responsabilité de la mise à jour de la clef, mais les deux autres serveurs peuvent demander explicitement un renouvellement. Le (re-)démarrage d'un serveur quel qu'il soit entraine le renouvellement de la clef de session.

La figure 7.3 propose une illustration du renouvellement périodique de la clef :

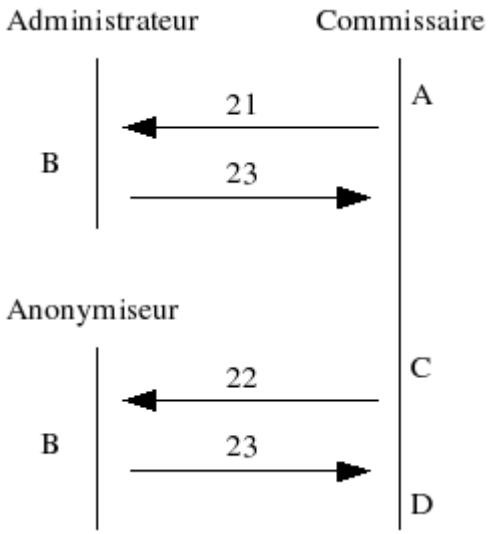

FIG. 7.3 – Renouvellement périodique de la clef de session.

A :

– génère deux clefs de session S_1 et $SessServer$;

– signe $SessServer$;

B :

– vérifie le *HMAC-SHA1* du message ;
– vérifie la signature ;
– enregistre $SessServer$;
– supprime la clef de session précédente ;

C :
– génère une clef de session S_2 ;

D :
– supprime la clef de session précédente, lorsque les deux serveurs ont confirmé l'utilisation de la nouvelle clef ;

21 :
– $[E_{PK\_administrateur}\{S_1\}]\#[E_{S_1}\{[21]\#[[SessServer]\%[signature]]\#[padding]\#[hmac]\#[alea]\}]$;

22 :
– $[E_{PK\_administrateur}\{S_2\}]\#[E_{S_2}\{[22]\#[[SessServer]\%[signature]]\#[padding]\#[hmac]\#[alea]\}]$;

23 :
– $E_{SessServer}\{[23]\#[[SessServer]\%[OK]]\#[padding]\#[hmac]\#[alea]\}$;

L'administrateur ou l'anonymiseur peuvent demander le renouvellement de la clef, après un redémarrage par exemple. Seul un message initial supplémentaire est envoyé, comme le montre la figure 7.4 :

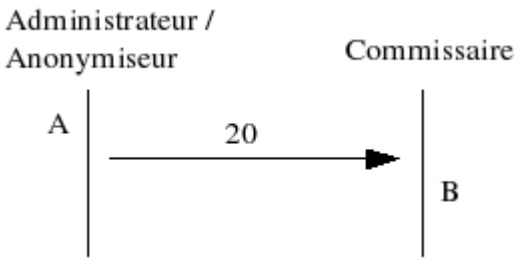

FIG. 7.4 – Demande de renouvellement de la clef de session.

A :
– génère une clef de session S_1 ;
– signe un aléa A ;

B :
– vérifie le *HMAC-SHA1* du message ;
– vérifie la signature de A ;

20 :
– $[E_{PK\_commissaire}\{S_1\}]\#[E_{S_1}\{[20]\#[[A]\%[signature]]\#[padding]\#[hmac]\#[alea]\}]$;

Le reste de la procédure est celui expliqué à la figure 7.3. Des messages pouvant intervenir entre le début et la fin de la procédure de renouvellement, le commissaire doit conserver l'ancienne clef jusqu'au moment où les deux autres serveurs ont confirmé l'utilisation de la nouvelle clef.

7.4 Propriétés du système de vote

Les propriétés requises par un système de vote électronique ont été exprimées au chapitre 4. Une preuve informelle du système vis-à-vis de ces propriétés peut être explicitée :

1. **Précision :**
   - le vote ne peut être altéré sans détruire la signature de l'administrateur ;
   - un électeur peut vérifier si son vote n'a pas été pris en compte lors du dépouillement en examinant la liste des votes publiés par le décompteur, et peut dès lors en informer le commissaire. Le vote peut encore être comptabilisé s'il est valide ;
   - l'utilisation de plusieurs décompteurs rend la suppression de votes largement moins aisée que lorsque seul un décompteur est actif. Tenter de supprimer un vote implique de devoir le détruire chez tous les décompteurs ;
   - un vote invalide ne peut être comptabilisé lors du dépouillement final car les signatures, vérifiables par n'importe qui, sont publiées au terme de la session de vote. En outre, un vote valide doit être accompagné par un *SecT3* correct.
2. **Démocratie :**
   - un électeur ne peut voter qu'un seule fois, pour autant qu'il ne soit en possession que d'un seul bulletin en papier valide ;
   - il est impossible de générer un bulletin en papier valide, et seules les personnes autorisées[2] à prendre part à la votation le reçoivent et peuvent le faire valider.
3. **Confidentialité :**
   - tant que les hypothèses formulées restent vraies, personne, pas même les autorités électorales, sont en mesure de lier un vote à une personne ;

[2]Dans le cas qui nous concerne, ceci est très difficile à réaliser.

– selon la variante choisie (*cf.* sect. 6.5), le votant peut prouver son choix, et par conséquent l'objectif de non-corruptibilité n'est pas atteint[3]. Il suffit au votant de fournir ses clefs d'engagement et la signature de l'administrateur démasquée pour prouver ce qu'il a voté.

4. **Vérifiabilité :**
   – chacun peut vérifier les signatures et recompter les votes. Chaque électeur est en mesure de vérifier si son propre vote a bien été pris en compte en utilisant les clefs d'engagement[4], et peut considérer que les autres votes sont corrects du fait de la signature.
5. **Résistance à la collusion :**
   – un vote n'est valide qu'accompagné d'une signature et d'un *SecT3*. Personne durant la session de vote, hormis le votant, n'a accès au *SecT3* en clair. Aucun serveur n'est donc en mesure de fabriquer des votes valides[5].
6. **Disponibilité :**
   – le système fonctionne correctement tant qu'un nombre minimal de serveur reste opérationnel. Il faut au minimum un administrateur, un commissaire, un anonymiseur et un décompteur. Cette propriété dépend fortement de la configuration du système lors du déploiement.
7. **Capacité de reprise :**
   – l'électeur peut prendre part à la procédure de vote plusieurs fois tant que son *SecT1* n'est pas marqué comme utilisé ;
   – seul un vote est comptabilisé[6]. Le votant doit attendre la confirmation de l'anonymiseur pour être certain que son bulletin en papier ne puisse plus être utilisé par une tierce personne.
8. **Robustesse :**
   – le système est capable de mettre en échec les tentatives d'usurpation d'identité des serveurs, ainsi que les tentatives d'utilisation d'applets non officielles.

7.5 Format des questions

Le système de vote présenté dans ce document prend en compte trois sortes de questions :

[3] Ceci est malheureusement vrai pour une majorité des systèmes de vote.
[4] Cela dépend encore une fois de la variante choisie (*cf.* sect. 6.5).
[5] Ceci n'est vrai que si l'on suppose que le processus de chiffrement et de destruction de la base de données des *SecT3* s'est faite sans incident (*cf.* sect. 6.3.1).
[6] En fait, seul un vote par SecT3 est pris en compte.

1. Les questions à choix unique, où une seule réponse est autorisée.
2. Les questions à choix multiple, où plusieurs réponses sont tolérées.
3. Les questions ouvertes, où il est possible d'entrer directement sa réponse au clavier.

L'interface de l'applet est générée de manière automatique en fonction de la question. Cette dernière est chargée lors de l'initialisation de l'applet. Elle prend la forme d'un simple fichier *XML*. Voici un exemple de fichier décrivant une question à choix unique (*cf.* figure 6.3) :

```
<question>
        <type>
                single
        </type>
        <name>
                us-election
        </name>
        <description>
                Which is the best US president ?
        </description>

        <answer>
                Nicolas Bonvin
        </answer>

        <answer>
                John F. Kerry
        </answer>

        <answer>
                Truc Muche
        </answer>
</question>
```

Un fichier *XML* pour une question à choix multiple ressemble à ceci :

```
<question>
        <type>
                multiple
        </type>
        <name>
                Men
        </name>
        <description>
                If you can choose, what whould your be ?
        </description>

        <answer>
                clever
        </answer>

        <answer>
                beautiful
        </answer>

        <answer>
                wealthy
        </answer>
```

```
</question>
```

Pour finir, voici le contenu d'un fichier décrivant une question ouverte :

```
<question>
        <type>
                open
        </type>
        <name>
                us-election
        </name>
        <description>
                What do you think about US elections ?
        </description>

</question>
```

8 Implémentation

8.1 Généralités

Plusieurs critères rendent le choix des technologies plutôt restreint. Tout d'abord, les votes doivent être chiffrés au niveau du client. Rien de sensible ne doit pouvoir être vu en clair par un serveur de vote. Autre critère important à respecter : le système doit être disponible pour la majorité des plateformes informatiques contemporaines.

Par exemple, le fait d'utiliser *SSL/TLS* ne résoud en rien le problème. Les messages ne transitent certes pas en clair sur le réseau, mais le serveur de destination est capable de lire toutes les données. Il est donc judicieux de chiffrer les informations directement au niveau du client. Pour ce faire, il existe plus ou moins quatre solutions :

1. Utiliser un programme dédié, qui peut prendre par exemple la forme d'un *CD-ROM* bootable.
2. Utiliser *JavaScript*, compris dans la majorité des navigateurs.
3. Utiliser *ActiveX*, présent uniquement sur plateforme *Windows* avec *Internet Explorer*.
4. Utiliser une applet *Java*, disponible pour la majorité des navigateurs et systèmes d'exploitation.

La première idée fut écartée du simple fait qu'il est peu envisageable techniquement et financièrement de distribuer des *CD-ROM* dans les deux pays concernés.

La seconde solution se montre intéressante, car les composantes logicielles nécessaires pour voter sont déjà présentes chez la majorité des électeurs. Cependant chaque navigateur possède sa propre implémentation de *JavaScript* et écrire du code compatible avec tous les navigateurs n'est pas chose aisée. Un point important est à souligner : peu de bibliothèques cryptographiques sont disponibles en *JavaScript*. De plus , il paraît sage d'éviter d'implémenter les versions académiques des algorithmes de chiffrement. Par conséquent, cette solution fut écartée.

La technologie *ActiveX* se voit rejetée d'emblée, du fait de son indisponibilité sous un autre système que *Windows*.

La technologie retenue est donc *Java*. Le système est prévu pour fonctionner avec le plugin *Java 1.2*. Même si *Java 5* est recommandé, nul besoin d'une version récente.

8.2 Serveur

Un programme faisant office de serveur se voit traditionnellement implémenté à l'aide de *threads*. Par exemple, un *thread* principal écoute sur un port donné. Lorsqu'il accepte une connexion, il crée un nouveau *thread* et lui transmet la requête. Réaliser un système performant à l'aide de *threads* peut très vite devenir complexe. De plus, sous forte charge, le passage d'un *thread* à l'autre peut se montrer pénalisant en terme de performance. Rien qu'avec une centaine de *threads*, la majeure partie des cycles du processeur est utilisée pour le changement de contexte.

Dès *Java 1.4*, une alternative plus performante est apparue : les *sockets* non bloquants[1]. Grâce à cette approche, un seul *thread* peut gérer un nombre arbitraire de *sockets*. Ceci permet aux serveurs de garder un nombre peu élevé de *threads*, même lors du traitement de plusieurs milliers de *sockets*, ce qui améliore de manière significative les performances.

A titre de comparaison, les solutions utilisant des *threads* connaissent une forte dégradation des performances lorsque le nombre de clients augmente. Dans la littérature, on parle souvent d'une limite de 3 000 clients simultanés, alors que les serveurs basés sur les *sockets* non bloquants arrivent à en gérer 10 000 sans perte de performance notable.

La solution retenue dans le cadre de ce projet est celle basée sur les *sockets* non bloquants.

8.3 Cryptographie

8.3.1 Chiffrement

Dans le cadre de ce projet, l'algorithme retenu pour le chiffrement symétrique est *Blowfish*[2]. Il a été conçu par Bruce Schneier et ne fait l'objet d'aucun brevet. *Blowfish* est un algorithme de chiffrement par bloc de 64 *bits* et utilise une taille de clef variable : de 32 *bits* à 448 *bits*.

[1] java.sun.com/j2se/1.4.2/docs/guide/nio

[2] www.schneier.com/blowfish.html

L'implémentation actuelle du système de vote utilise la taille de clef maximale, soit 448 *bits*, dans le mode *CFB*.

Le padding utilisé est décrit par *PKCS#1 v1.5*[3]. Plusieurs attaques connues contre *PKCS#1 v1.5* existent. La première fut présentée lors de *Crypto '98* par Daniel Bleichenbacher (*cf.* [6]). Deux autres attaques sont décrites dans [11]. Il est donc nécessaire de remplacer dans une version future du système de vote le méchanisme de *padding* par *OAEP*, décrit par la version 2.1 de *PKCS#1*.

Le chiffrement asymétrique est réalisé à l'aide de *RSA*. Chaque serveur possède un jeu de clefs. L'administrateur possède une seconde paire de clefs destinée à signer les votes. La taille des clefs est de 2 048 *bits*.

8.3.2 Signature aveugle

L'empreinte masquée est calculée à partir d'une empreinte normale grâce à une fonction de masquage. Cette empreinte masquée peut être signée à l'aide d'une clef de signature en utilisant un algorithme correspondant à la fonction de masquage. La signature peut être démasquée par la personne ayant masqué l'empreinte. La signature démasquée est vérifiable directement depuis l'empreinte originale.

Dans le cas des signatures aveugles avec *RSA*, l'administrateur possède une clef de signature avec un exposant public e, un exposant secret d et un modulo n. Lorsque l'électeur désire faire signer son vote, les actions suivantes sont entreprises :

- l'applet génère deux clefs d'engagement de manière aléatoire, R_1 et R_2. Ces clefs ont une longueur de 20 caractères choisis parmi un jeu de 62 caractères ;
- le vote (en texte clair) ainsi que les clefs d'engagement sont introduits dans la fonction *hmac* afin de procéder à l'engagement. Le résultat, f, est une empreinte de 160 *bits* :
$$f = hmac(R_1, R_2, vote)$$
- un facteur aléatoire de masquage r est créé ;
- l'empreinte est masquée à l'aide du facteur de masquage :
$$f' = fr^e$$
- l'empreinte masquée est envoyée à l'administrateur et est signée de manière conventionnelle par ce dernier :
$$s' = f'^d = (fr^e)^d = f^d(r^e)^d = f^d r$$

[3] www.rsasecurity.com

– une fois renvoyée à l'électeur, cette empreinte masquée et signée peut être démasquée afin de récupérer la signature s :

$$s \ = s'/r \ = f^d$$

Ce processus est sûr, car la multiplication avec un élément du groupe du modulo est une fonction bijective et car il existe un r capable de convertir s' en une signature pour toute empreinte f. Mais ceci est impossible à réaliser sans connaître le modulo secret d. Cela signifie que l'administrateur ne peut pas savoir ce qu'il est en train de signer, et cela signifie également que l'électeur ne peut pas utiliser la signature pour une autre empreinte.

D'autre part, l'utilisation d'une fonction de hachage h garantit que la propriété multiplicative de *RSA* ne pose pas de problème. Si s_1 et s_2 valent :

$$s_1 = f_1^d \text{ et } s_2 = f_2^d$$

alors

$$s = s_1 s_2 \ = f_1^d f_2^d \ = f^d \text{ pour } f \ = f_1 f_2$$

Ceci peut conduire à de graves failles de sécurité si les messages sont signés directement. C'est pourquoi l'utilisation de la fonction *hmac* est rigoureusement nécessaire.

8.3.3 Générateur de nombre aléatoire

Générer une nouvelle clé de session, par exemple, nécessite un générateur de nombre aléatoire cryptographiquement sûr.

Les *bits* aléatoires sont lus directement depuis le fichier `/dev/urandom` ou `/dev/random`, si ce dernier est lisible, comme sur un système *GNU/Linux*. Ces *bits* ne font pas l'objet de vérification quant à leur propriété aléatoire.

Dans le cas contraire, si le fichier n'est pas lisible, un générateur de nombre pseudo-aléatoire basé sur *MD5* est mis en oeuvre. Ce générateur doit être initialisé et fourni par un autre générateur de nombre aléatoire.

Les *octets* aléatoires sont générés par blocs de 16 *octets*. Le bloc i vaut :

$$r_i = H(s_0 \cdots s_i)$$

s_0 est la graine initiale qui est permutée à chaque étape afin de former $s_i = s_{i-1} + r_i$. H est la fonction de hachage *MD5*, où l'étape finale qui consiste à ajouter la longueur du message est omise.

s_0 et chaque r_i sont pris de la source d'entropie, le second générateur. Le nombre de *bits* de ces valeurs doit être suffisamment grand pour s'assurer d'une quantité appréciable d'entropie. Le générateur de nombre pseudo-aléatoire est initialisé avec 256 octets et injecte 4 octets d'entropie à chaque étape.

Ce générateur aléatoire est comparable à l'utilisation de *MD5* dans le mode *OFB* avec un vecteur initial *IV* secret. À chaque étape, r_i vaut :

$$r_i = H(s_0 \cdots s_i) = H(s_0 \cdots s_{i-1}) + h(H(s_0 \cdots s_{i-1}), s_i)$$

où h est la fonction d'étape de *MD5*. La sécurité de ce générateur de nombre pseudo aléatoire repose sur la difficulté de prévoir un *bit* de $h(x, y)$ où x est connu et de la forme expliquée ci-dessus.

L'iniatialisation peut donc prendre du temps, selon la source d'entropie. C'est pour cela qu'elle est réalisée à l'aide de son propre *thread*. Si des *bits* aléatoires sont demandés avant la fin de l'initialisation, alors la requête sera bloquée.

Le second générateur utilise un *thread* qui compte indéfiniment pour générer un *octet* de donnée. Après un certain temps, le *thread* est tué et les 8 *bits* de poids faible du compteur sont retournés.

La période de rotation est choisie de telle manière que le compteur atteigne au moins 1 024 lorsqu'un *octet* est généré. Cependant, sous forte charge, le compteur peut ne pas dépasser 256. Le résultat de ce générateur de nombre aléatoire n'a pas de bonnes propriétés statistiques, mais chaque *octet* de donnée contient tout de même un ou deux *bits* d'entropie. C'est pourquoi il n'est pas utilisé directement, mais sert à fournir de l'entropie à l'autre générateur.

La robustesse de ce générateur n'étant pas garantie, il semble judicieux de le remplacer dans une future version du système de vote par un générateur cryptographiquement sûr, tel *ANSI X9.17* (*cf.* [1]), basé sur *Triple-DES* ou *Yarrow-160* (*cf.* [22]) proposé par Counterpane System[4].

8.4 Bibliothèques et codes sources utilisés

Tout code source provenant d'une source externe utilisé dans le cadre de ce projet est soumis à la *GNU GPL*[5].

8.4.1 Serveur

Les programmes faisant office de serveur, le commissaire (*cf.* sect. 6.3), l'administrateur (*cf.* sect. 6.2) et l'anonymiseur (*cf.* sect. 6.4), se basent sur la solution présentée dans l'article [40] de Nuno Santos. Cet article explique comment réaliser en *Java* un serveur alliant robustesse et performances élevées. Il détaille notamment le fonctionnement des *sockets* non bloquants.

[4] www.counterpane.com
[5] www.gnu.org/copyleft/gpl.html

8.4.2 Cryptographie

Java ne fournit de bibliothèques cryptographiques en standard qu'à partir de la version 1.4. Auparavant, l'extension *JCE*[6] était distribuée de façon autonome. Selon les contraintes fixées par le cahier des charges, l'applet doit être compatible avec la version 1.2 et supérieure de *Java*. Cette solution a donc été écartée et remplacée au profit de classes[7] écrites par Logi Ragnarsson. Le paquetage *logi.crypto 1.1.2* est compatible avec la version 1.1 de *Java* et permet d'utiliser de nombreux algorithmes de chiffrement à sens unique (hash), symétrique et asymétrique :

- *Blowfish* ;
- *DES* ;
- *Triple DES* ;
- *RSA*, y compris les signatures et les signatures aveugles ;
- *MD5* ;
- *SHA-1*.

Il permet d'utiliser ces algorithmes dans différents modes (si applicable) :

- *CBC* ;
- *CFB* ;
- *ECB* ;
- *OFB*.

Le paquetage implémente également plusieurs sortes de *padding* :

- *zero padding* ;
- *PKCS#5* ;
- *PKCS#1 v1.5*.

Cette bibliothèque doit faire l'objet d'un audit rigoureux avant d'utiliser le système de vote en condition réelle.

[6] java.sun.com/products/jce
[7] www.logi.org/logi.crypto/devel

9 Déploiement

9.1 Généralités

Déployer une application peut se révéler délicat, surtout dans les conditions mentionnées. Chaque composante doit être sécurisée de manière optimale, que ce soit physiquement ou de manière logicielle. Pour commencer, il est recommandé de placer les serveurs dans des centres de calcul hautement sécurisés qui doivent fournir les services suivants :

- contrôle d'accès ;
- redondance électrique ;
- climatisation ;
- protection anti-feu ;
- redondance des connexions réseau ;
- garantie d'*uptime* élevé ;
- ...

La figure 9.1 illustre un déploiement possible de l'application. L'électeur ne communique pas directement avec les serveurs de vote, il doit d'abord passer par un *firewall*. Son rôle est de filtrer les demandes en empêchant, par exemple, qu'une même *IP* fasse plus qu'un nombre fixé de requêtes simultanément. Il doit également faire de la redirection statique de port : l'applet se connecte par exemple sur le port 9 000 du *firewall* pour pouvoir dialoguer avec l'administrateur. Ceci implique que l'applet soit téléchargée directement depuis le *firewall*, et donc que le port 443[1] de ce dernier soit ouvert.

L'accès au réseau virtuel privé[2] n'est possible qu'en passant par le *firewall*, les adresses *IP* des serveurs de vote ne sont donc pas publiques. Un attaquant qui a signé une applet afin d'être capable d'ouvrir des ports vers un serveur distant autre que le sien n'est pas en mesure de mener à terme son attaque sans connaître ces adresses. Cette attaque échoue également par le fait que le serveur de vote vérifie l'authenticité de l'applet. L'avantage d'utiliser un réseau virtuel privé est le fait que chaque serveur du système de vote peut se

[1] Port sur lequel écoute un serveur *HTTPS*.
[2] *VPN* en anglais.

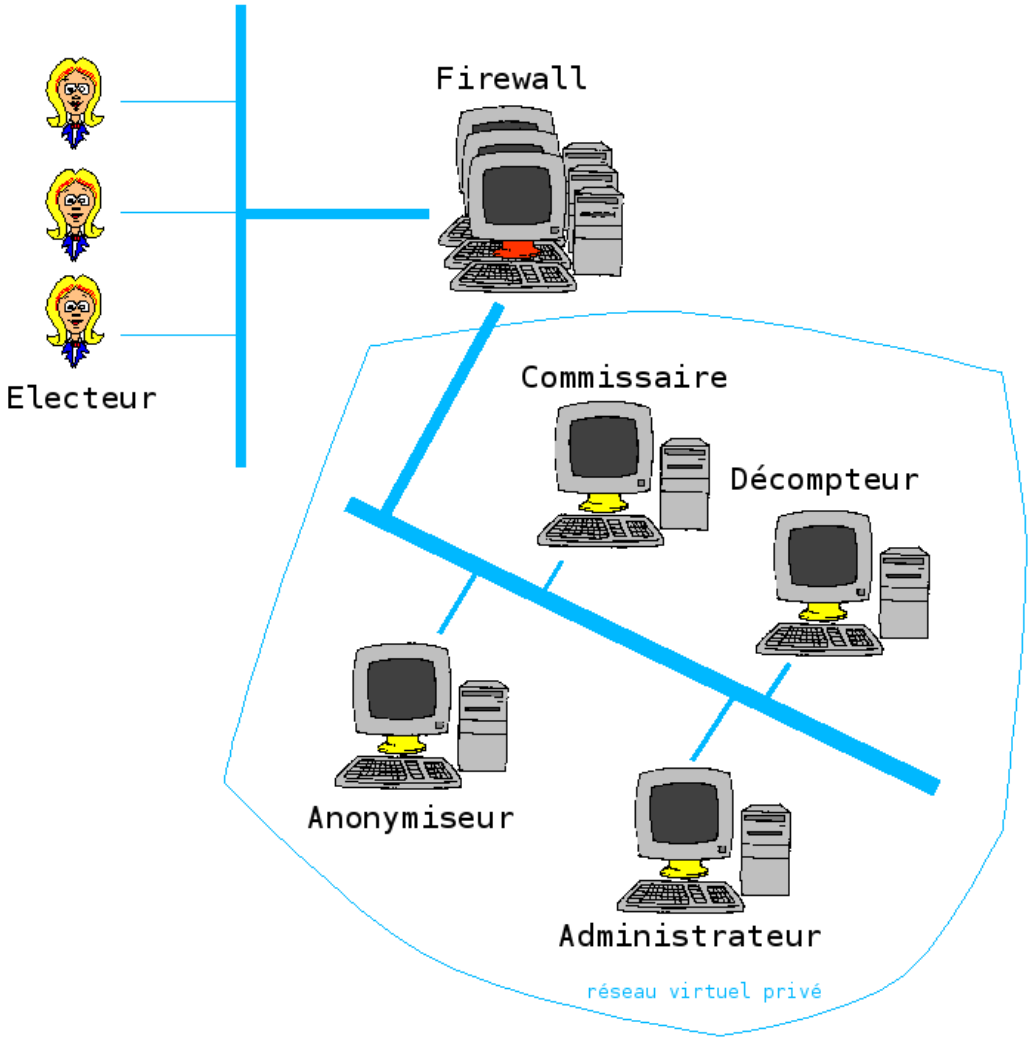

FIG. 9.1 – Déploiement du système de vote.

trouver dans un pays différent, par exemple, tout en gardant la sécurité et les avantages d'un réseau privé.

Toutes les composantes du système de vote, c'est-à-dire le commissaire, l'administrateur, l'anonymiseur et le décompteur doivent être en haute-disponibilité[3]. Idéalement, chaque composante est hébergée dans un centre de calcul différent, géographiquement distant.

Le goulet d'étranglement de la solution illustrée par la figure 9.1 est clairement le *firewall*. Ce dernier doit être en mesure de supporter une attaque par

[3] *Failover* et *load balancing*.

saturation de moyenne à grande ampleur. L'infrastructure et les compétences nécessaires pour résister à cette attaque peuvent engendrer un surcoût non négligeable.

Un déploiement en grandeur nature suppose que chaque composante se trouve en mains différentes. Dans le cas qui nous concerne, l'appui de pays autres que la Suisse pourrait donner un gage de cédibilité supplémentaire. Les serveurs pourraient être répartis de la manière suivante :

- la Confédération Helvétique se chargerait du commissaire, de l'administrateur ainsi que du décompteur ;
- la France aurait la charge d'un anonymiseur ;
- la Grande-Bretagne s'occuperait d'un autre anonymiseur ;
- l'Allemagne prendrait également en charge un anonymiseur.

Les serveurs seraient hébergés dans les pays de leurs détenteurs respectifs. Le fait d'impliquer la France, la Grande-Bretagne et l'Allemagne rend la destruction volontaire de votes difficile. Il faudrait que les trois pays coopèrent afin de supprimer le même vote sur les trois serveurs. Même en supposant que telles seraient leurs intentions, la tâche ne serait pas aisée car chaque vote est stocké dans un fichier portant un nom totalement aléatoire et est chiffré de façon à ce que les anonymiseurs n'aient aucune information quant à son contenu.

Au terme de la session de vote, les anonymiseurs envoient les votes au décompteur. Une délégation des deux pays concernés, ainsi que toute personne intéressée, peut prendre part au dépouillement. Le décompteur vérifie que les données fournies par chaque anonymiseur concordent, règle les différends s'il y en a, et publie les résultats.

9.2 Utilisation

Par défaut, le nombre maximal de fichiers que peut ouvrir un processus sur une plateforme *Unix* est fixé en général à 1 024. Cette limite implique que le nombre de clients simultanés ne peut dépasser 500. Pour augmenter le nombre de requêtes simultanées possibles à 5 000, il faut posséder les droits *root* et entrer la commande suivante dans un *shell* :

```
ulimit -n 10240
```

9.2.1 Administrateur

Pour lancer le programme faisant office d'administrateur, il suffit de taper la commande suivante dans un *shell* :

```
java -jar -server administrator.jar port
        [--host-commisioner h]
        [--port-commisioner p]
```

avec

- **port :** port sur lequel écoute l'administrateur ;
- **h :** adresse du commissaire (optionnel). Par défaut, l'adresse est *localhost* ;
- **p :** port sur lequel écoute le commissaire (optionnel). Par défaut, le port est 8 000 ;

Il est d'usage d'attribuer le port 9 000 à l'administrateur, comme le montre l'exemple suivant :

```
java -jar -server administrator.jar 9000 --host-commisioner
                        commi.test.ch
```

L'archive *jar* contient tous les fichiers nécessaires au bon fonctionnement du programme, y compris les différentes clefs asymétriques.

9.2.2 Anonymiseur

L'anonymiseur se lance de manière similaire :

```
java -jar -server anonymiser.jar port
        [--host-commisioner h]
        [--port-commisioner p]
```

avec

- **port :** port sur lequel écoute l'anonymiseur ;
- **h :** adresse du commissaire (optionnel). Par défaut, l'adresse est *localhost* ;
- **p :** port sur lequel écoute le commissaire (optionnel). Par défaut, le port est 8 000 ;

Le port préconisé pour l'anonymiseur est le 10 000 :

```
java -jar -server anonymiser.jar 10000 --host-commisioner
                        commi.test.ch
```

9.2.3 Commissaire

Le lancement du commissaire nécessite de connaître l'adresse et le port des deux autres serveurs :

```
java -jar -server commisioner.jar port
        [--host-administrator h1]
        [--port-administrator p1]
        [--host-anonymiser h2]
        [--port-anonymiser p2]
```

avec

- **port :** port sur lequel écoute le commissaire ;
- **h1 :** adresse de l'administrateur (optionnel). Par défaut, l'adresse est *localhost* ;
- **p1 :** port sur lequel écoute l'administrateur (optionnel). Par défaut, le port est 9 000 ;
- **h2 :** adresse de l'anonymiseur (optionnel). Par défaut, l'adresse est *localhost* ;
- **p2 :** port sur lequel écoute l'anonymiseur (optionnel). Par défaut, le port est 10 000 ;

Le commissaire est traditionnellement lancé sur le port 8 000 :

```
java -jar -server commisioner.jar 8000 --host-administrator
        admin.test.ch --host-anonymiser ano.test.ch
```

9.2.4 Applet

Utiliser l'applet ne nécessite aucune action particulière, du moment que le *plugin Java* est installé correctement. Il suffit de se connecter à l'aide de son navigateur préféré sur le port 443 du serveur de vote officiel. Ce serveur est en fait l'unique point d'entrée du réseau virtuel privé : le *firewall*.

10 Conclusion

10.1 Généralités

Le système de vote ne sera très certainement pas déployé comme prévu à l'origine, du fait de certaines décisions diplomatiques. Cependant, il est tout à fait envisageable d'utiliser ce logiciel pour d'autres types de votations.

On peut imaginer que l'association des étudiants de l'EPFL[1] décide de renouveler son comité en proposant à tous les étudiants de voter pour les représentants de leur choix. Il suffirait d'envoyer un email contenant les trois numéros aléatoires à chaque étudiant pour qu'il puisse prendre part à l'élection. Il serait même envisageable d'utiliser le service d'authentification[2] de l'EPFL pour s'assurer que chaque étudiant ne puisse avoir accès de manière sécurisée qu'à un seul jeu de numéros.

10.2 Améliorations possibles

Le système présenté dans ce document peut faire l'objet de nombreuses améliorations, et il reste encore plusieurs tâches à effectuer avant d'envisager un réel déploiement :

- tester de manière rigoureuse le comportement des différents serveurs sous forte charge, vérifier qu'ils agissent conformément aux prévisions ;
- prévoir des méchanismes pour contrer des attaques spécifiques, comme celles basées sur paquets malformés, ou encore celles tentant de saturer les tampons des *sockets* ;
- gérer une plus grande partie de la sécurité au niveau applicatif, comme la vérification des *IP* connectées. Ce dernier problème est en partie traité en amont par le *firewall* ;
- gérer la persistance de la base des numéros aléatoires. Dans la version actuelle, ces numéros sont uniquement chargés en mémoire. Arrêter le commissaire entraine la perte des changements survenus. L'état courant

[1] agepoly.epfl.ch
[2] tequila.epfl.ch

n'est pas conservé ;
- mettre en place la base de donnée permettant la vérification de l'authenticité de l'applet ;
- développer un site Web visuellement attractif et professionnel ;
- revoir l'ergonomie de l'applet ;
- s'assurer de la robustesse de l'application ;
- utiliser le padding décrit dans *PKCS#1 v2.1* ;
- actuellement, la clef utilisée pour calculer le *HMAC* est la même que la clef de session. Ceci est fortement déconseillé et doit donc faire l'objet d'une mise à jour ;
- remplacer le générateur de nombre aléatoire par un générateur cryptographiquement sûr ;
- ...

Il faut également ajouter qu'un tel système ne peut être déployé en conditions réelles avant d'avoir été rigoureusement audité par des experts indépendants. Aucun logiciel n'est exempt d'erreurs. Par conséquent aucune garantie ne peut être donnée quant à la sécurité du système.

10.3 Impressions personnelles

Premier véritable projet réalisé dans le cadre de mes études, ce travail de diplôme m'a permis d'explorer plusieurs mondes qui m'attirent. Mélanger des concepts de sécurité et de réseau fut vraiment enrichissant.

Un aspect agréable de ce projet de Master réside dans le fait qu'il a été possible de faire un compte-rendu de l'état de l'art dans le domaine du vote électronique. La première partie de mon projet fut exclusivement consacrée à lire diverses sources, comme des articles écrits par des informaticiens, des mathématiciens ou même des journalistes généralistes, ainsi que plusieurs rapports critiquant[3] les systèmes actuels.

Il m'a également été possible de tester différentes technologies, notamment concernant l'implémentation des serveurs et les fonctions de chiffrement.

En guise de conclusion, je tiens à remercier M. Gilbert Robert de m'avoir permis de réaliser ce travail en entreprise, ainsi que Prof. Serge Vaudenay pour ses précieux conseils.

[3] De manière positive ou négative.

Références

[1] *Ansi x9.17. american national standard institute. financial institution key management (wholesale). asc x9 secretariat, american bankers association.*, (1986).

[2] M. Abe, *Universally verifiable mix-net with verification work independent of the number of mix-servers*, EUROCRYPT '98, (1998), p. 437–447.

[3] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard et J. Stern, *Practical multi-candidate election system*, PODC '01, (2001), p. 274–283.

[4] J. Benaloh et D. Tuinstra, *Receipt-free secret-ballot elections*, STOC '94, (1994), p. 544–553.

[5] J. C. Benaloh, *Verifiable secret-ballot elections*, PhD Thesis, Yale University, Department of Computer Science, (1987).

[6] D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the rsa encryption standard*, Advances in Cryptology : Proceedings of CRYPTO '98, (1998), p. 1–12.

[7] D. Chaum, *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the ACM, (1981), p. 84–88.

[8] ——, *Blind signatures for untraceable payments*, Advances in Cryptology - Crypto '82, (1983), p. 199–203.

[9] ——, *Elections with unconditionally-secret ballots and disruption equivalent to breaking rsa*, EUROCRYPT '88, (1988), p. 177–182.

[10] ——, *Secret-ballot receipts and transparent integrity*, (2002).

[11] J. Coron, M. Joye, D. Naccache et P. Paillier, *New attacks on pkcs#1 v1.5 encryption*, Prooceedings of EUROCRYPT 2000, (2000), p. 369–381.

[12] R. Cramer, M. Franklin, B. Schoenmakers et M. Yung, *Multi-authority secretballot elections with linear work*, EUROCRYPT '96, (1996), p. 72–83.

[13] R. Cramer, R. Gennaro et B. Schoenmakers, *A secure and optimally efficient multi-authority election scheme*, EUROCRYPT '97, (1997), p. 103–118.

[14] L. Cranor et R. Cytron, *Sensus : A security-conscious electronic polling system for the internet*, Proceedings of the Hawaii International Conference on System Sciences, (1997).

[15] I. Damgård et M. Jurik, *A generalisation, a simplification and some applications of paillier s probabilistic public-key system*, Public Key Cryptography '01, (2001), p. 119–136.

[16] I. Damgård, M. Jurik et J. B. Nielsen, *A generalization of paillier s public-key system with applications to electronic voting*, (2003).

[17] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, CRYPTO '84, (1984), p. 10–18.

[18] A. Fujioka, T. Okamoto et K. Ohta, *A practical secret voting scheme for large scale elections*, AUSCRYPT '92, (1992), p. 244–251.

[19] M. Hirt et K. Sako, *Efficient receipt-free voting based on homomorphic encryption*, EUROCRYPT '00, (2000), p. 539–556.

[20] P. Horster, M. Michels et H. Petersen, *Blind multisignature schemes and their relevance to electronic voting*, Proc. 11th Annual Computer Security Applications Conference, (1995), p. 149–156.

[21] M. Jakobsson, A. Juels et R. L. Rivest, *Making mix nets robust for electronic voting by randomized partial checking*, USENIX '02, (2002), p. 339–353.

[22] J. Kelsey, B. Schneier et N. Ferguson, *Notes on the design and analysis of the yarrow cryptographic pseudorandom number generator.*, Selected Areas in Cryptography'99, (1999).

[23] A. Kiayias et M. Yung, *Self-tallying elections and perfect ballot secrecy*, PKC '02, (2002), p. 141–158.

[24] H. Kikuchiy, J. Akiyamaz, H. Gobio et G. Nakamuraz, *stochastic voting protocol to protect voters privacy*, (1998).

[25] B. Lee et K. Kim, *Receipt-free electronic voting scheme with a tamperresistant randomizer*, ICISC2002, (2002), p. 405–422.

[26] E. Magkos, M. Burmester et V. Chrissikopoulos, *Receipt-freeness in large-scale elections without untappable channels*, I3E, (2001), p. 683–994.

[27] D. Malkhi, O. Margo et E. Pavlov, *E-voting without 'cryptography'*, Financial Cryptography '02, (2002).

[28] M. Michels et P. Horster, *Some remarks on a receipt-free and universally verifiable mix-type voting scheme*, ASIACRYPT '94, (1996), p. 125–132.

[29] D. Naccache et J. Stern, *A new public-key cryptosystem*, EUROCRYPT '97, (1997), p. 27–36.

[30] C. A. Neff, *A verifiable secret shuffle and its application to e-voting*, Proceedings of the 8th ACM conference on Computer and Communications Security, (2001), p. 116–125.

[31] ——, *Detecting malicious poll site voting clients*, (2003).

[32] M. Ohkubo, F. Miura, M. Abe, A. Fujioka et T. Okamoto, *An improvement on a practical secret voting scheme*, ISW '99, (1999), p. 225–234.

[33] T. Okamoto, *Receipt-free electronic voting schemes for large scale elections*, Security Protocols Workshop, (1997), p. 25–35.

[34] P. Paillier, *Public-key cryptosystems based on composite degree residuosity classes*, EUROCRYPT '99, (1999), p. 223–238.

[35] C. Park, K. Itoh et K. Kurosawa, *All-nothing election scheme and anonymous channel*, EUROCRYPT '93, (1993), p. 248–269.

[36] B. Pfitzmann, *Breaking an efficient anonymous channel*, Lecture Notes in Computer Science, 950 (1995), p. 332–340.

[37] I. Ray, I. Ray et N. Narasimhamurthi, *An anonymous electronic voting protocol for voting over the internet*, Proceedings of the Third International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems, (2001).

[38] K. Sako et J. Kilian, *Secure voting using partial compatible homomorphisms*, CRYPTO '94, (1994), p. 248–259.

[39] ——, *Receipt-free mix-type voting scheme*, EUROCRYPT '95, (1995), p. 393–403.

[40] N. Santos, *Building highly scalable servers with java nio*, onjava.com, (2004).

[41] S. Vaudenay, *Security flaws induced by cbc padding - applications to ssl, ipsec, wtls...*, Advances in Cryptology - Eurocrypt 2002, (2002).