UCLouvain

epl

École polytechnique de Louvain

Simulation d'un système de vote dans le cadre d'une démocratie liquide

Copie d'un vote anonymement

Auteur: **Alexis Cherpion**
Promoteur: **Olivier Pereira**
Lecteurs: **Henri Devillez, Thomas Peters**
Année académique 2021–2022
Master [120] en science des données, orientation technologie de l'information

Table des matières

Chapitre 1

Introduction

Le mouvement des gilets jaunes en France semble indiquer une envie populaire de renouveau démocratique, comme le montrent de nombreux articles d'actualité [18][16][11]. Le modèle de démocratie liquide peut correspondre à cette envie de changement. En effet, ce type de démocratie permet une implication plus personnelle des votants puisqu'elle permet à celui-ci de choisir entre un vote direct et un vote pour un représentant, lui déléguant ainsi sa voix, avec possibilité de la rétracter quand ils le veulent [24]. Il est ainsi pertinent de s'intéresser à la création d'un modèle de vote permettant des élections dans le cadre de la démocratie liquide.

La démocratie liquide, délégative ou encore par proxy peut être vue comme un mélange de la démocratie directe et de la démocratie représentative, essayant de profiter des forces des deux modèles [23]. Mais quel est l'apport de la démocratie liquide vis à vis des autres modèles démocratiques ? C'est la question à laquelle tentent de répondre Blum and Zuber en tentant de démontrer la supériorité de la démocratie liquide sur la démocratie représentative. A cette fin, ils prennent le modèle de démocratie représentative comme norme et en viennent à la conclusion que la démocratie liquide est plus avantageuse à la fois sur le plan épistémologique et égalitarien [9].

Cependant, l'un des problème que pose la démocratie liquide est celui du système de vote. En effet, le système classique utilisé dans une démocratie représentative n'est pas (ou trop difficilement) utilisable. Dans ce type de démocratie, le votant se rend directement sur place afin de justifier de son identité, puis écrit son vote sur un format papier, et le dépose dans l'urne anonymement. Cette méthode est viable étant donné qu'on demande au votant d'être présent uniquement lors de l'élection des représentants (une fois tous les 5 ans dans le cadre des élections fédérales belge, par exemple [1]). Dans le cadre d'une démocratie liquide, par contre, l'implication du votant est bien plus grande étant donné qu'il est dans la capacité de voter pour chaque loi, et/ou, lorsqu'il délègue sa voix à un représentant, doit être capable de la rétracter dès qu'il le veut. Or, il est très difficile voir impossible pour un

votant de devoir se déplacer sur place à chaque fois qu'il veut changer sa voix ou voter pour une loi. Cela pousserait les votants à déléguer leur voix et les changer plus rarement entrant en conflit avec l'objectif d'implication des votants de la démocratie liquide. De ce fait, il est nécessaire de développer un outil informatique permettant au votant de le faire en ligne afin que cela lui soit accessible dès qu'il en éprouve le besoin tout en garantissant son anonymat. En effet, les techniques de votes digitales apportent la praticité et l'efficacité nécessaire à la création d'une plateforme de vote [15].

D'ailleurs, certains partis pirates emploient la démocratie liquide et utilisent par conséquent des programmes développés dans ce cadre, comme LiquidFeedback ou Google votes [22][26]. Selon Paulin (2020), seuls ces deux outils sont compatibles avec la démocratie liquide et ont été testés. Cependant, selon Swierczek (2011) [26][6] et Hardt(2015) [19], l'anonymat des votants n'est pas assuré. Les votants ont au mieux un pseudo-anonymat, leur identité étant seulement couverte par un pseudonyme.

Afin de développer un modèle de vote garantissant l'anonymat, nous devons utiliser des méthodes cryptographiques. En effet, la cryptographie est nécessaire quand il s'agit de communiquer un message via un média non fiable, incluant de ce fait tous les réseaux, y compris internet. De plus, elle remplit 5 fonctions primaires : la confidentialité, l'authentification, l'intégrité, la non répudiation, et l'échange de clef, toutes nécessaires dans le cadre d'un système de vote en ligne [20]. En outre, le développement des méthodes de vérifications E2E permettent au votant d'avoir confiance au système de vote [5]. Des méthodes de e-voting adaptée à l'usage dans un navigateur ont été conçues à cette fin [25][3]. Dans le cadre de ce mémoire, nous allons nous inspirer d'Hélios. Hélios est un système de vote fiable, viable et utilisable dans le cadre d'élections. Il fonctionne par navigateur, ayant de ce fait une meilleure ergonomie. Enfin, étant donné qu'il existe des représentants dans le cadre de la démocratie liquide, le fait qu'il possède un système qui, à l'aide de légères modifications, permet de copier les votes en fait le meilleur choix.

L'objectif du présent mémoire est donc d'essayer de fournir un modèle valide de simulation de vote en conservant l'anonymat tout en étant utilisable dans le cadre d'une démocratie liquide. Pour ce faire, nous commencerons par définir et poser un modèle de démocratie liquide dont nous donnerons les caractéristiques. Ensuite, nous spécifierons le protocole en en définissant les acteurs, les adversaires types et les hypothèses de confiance ainsi que le modèle global utilisé pour le protocole et le cadre qu'il nécessite. Puis, nous établirons le protocole en précisant ses pré-requis mathématiques (chiffrement ElGamal et Zero Knowledge Proof), en explicitant les trois phases de l'élection (pré-élection, élection, et après-élection) et sa vérifiabilité End-to-End. Nous conclurons en proposant un résolution informatique autorisant la mise en place d'élection d'une démocratie liquide. Nous ferons une analyse du

code afin d'en établir les limites. Enfin, nous conclurons ce mémoire puis nous discuterons de pistes d'amélioration potentielles du programme.

Chapitre 2

Définition et modèle de démocratie liquide

2.1 Définition de démocratie liquide

Comme nous l'avons vu précédemment, la démocratie liquide est souvent définie comme une combinaison de démocratie directe et de démocratie représentative. Elle permet au votant de pouvoir à la fois voter directement pour une loi, selon le modèle de démocratie directe, mais aussi de déléguer sa voix à un représentant, de la même manière que dans une démocratie représentative [17].

L'idée initiale sous-tendant le modèle de démocratie liquide a été développée par Miller en 1969 sous le nom : "Un programme pour le vote direct et proxy" [23]. Cette théorie a depuis été reprise de nombreuses fois et sous différentes appellations comme démocratie délégative, vote par proxy ou encore démocratie liquide [9]. Il existe de nombreux exemples d'application d'un système de démocratie liquide. C'est notamment le cas de plusieurs partis pirates originaires d'Allemagne, d'Autriche, de Norvège, de France ou encore des Pays-Bas [28], par le biais de l'utilisation de LiquidFeedback. Google l'expérimente également au moyen d'un système de réseau social interne appelé Google Votes [19]. Enfin, dans un cadre plus politique, un parti d'Argentine, Democratia en Red, prône une redistribution du pouvoir politique ainsi qu'une participation plus grande du citoyen au système politique par la mise en place d'une démocratie liquide [2]. Ces quelques exemples, et plus particulièrement ceux à visée politique, montrent une volonté de remédier au système démocratique de l'intérieur en prenant la meilleure partie de la démocratie directe et représentative [9]. Cependant, dire que la démocratie liquide n'est qu'un mélange entre la démocratie représentative et la démocratie directe serait réducteur. Il n'existe néanmoins que peu d'études visant à conceptualiser ou théoriser un modèle construit de démocratie liquide, la plupart des études académiques se basant

davantage sur des recherches empiriques [28]. Elle possède pourtant ses propres caractéristiques qu'il convient d'énoncer.

2.2 Caractéristiques du modèle de démocratie liquide

Afin de donner une conceptualisation de la démocratie liquide (LD), nous nous baserons sur le modèle basique de Blum et Zummer [9]. La LD permet une prise de décision collective au moyen d'une participation démocratique directe mais aussi d'un système de représentant flexible. Il existe 4 grands critères dans une démocratie liquide vis à vis du droit de vote :

- Composant de démocratie directe : Un votant est en droit de voter directement pour n'importe quel problème politique, notamment une proposition de loi, mais aussi de participer aux débats concernant ces problèmes etc. Le votant peut pleinement participer à la vie politique.
- Délégation flexible : Un votant peut déléguer son vote à un représentant. Cette délégation est flexible puisqu'il peut déléguer ce vote pour un problème politique spécifique, un domaine politique entier ou encore pour toutes les décisions politiques prises.
- Metadélégation : Les personnes ayant reçu des votes sont en droit de déléguer ces votes à une autre personne [21].
- Rappel instantané : À n'importe quel moment, le votant doit pouvoir récupérer le vote qu'il a délégué pour pouvoir en faire usage lui-même ou le déléguer à une autre personne.

A partir du moment où une personne remplit les conditions choisies (âge, nationalité) par l'organisme ou le régime voulant utiliser une démocratie liquide, elle devient un votant. On remarque rapidement que dans ce modèle de démocratie tout le monde peut devenir un représentant, ce qui pose un problème au niveau de la répartition des rôles. En effet, selon les règles d'une démocratie liquide, dès qu'un citoyen volontaire pour devenir délégué reçoit une délégation de vote de la part d'un autre citoyen, il en devient un délégué, et de ce fait, se voit attribuer tous les pouvoirs et toutes les contraintes attenants à ce rôle [28]. La transparence concernant les décisions et les votes d'un délégué est donc nécessaire étant donné que les votants devraient pouvoir vérifier que le délégué suit bien son programme et vote bien ce qu'il avait prévu de voter afin de diminuer le risque de corruption et de pouvoir retirer leur vote en cas de malhonnêteté [27]. De plus, il est important

d'écarter un risque de corruption parmi les délégués publics de type achat de voix ou du fait de forcer certaines personnes à leur déléguer leur vote. C'est pourquoi les délégués ayant un nombre de vote insuffisant ne devraient pas être capable de connaître le nombre de personnes qui leur ont délégué leur vote [27]. Cela évitera qu'ils soient capable de comptabiliser et de vérifier si les votants à qui ils ont demandés des voix ont bel et bien voté pour eux. De ce fait, il convient de distinguer deux types de délégués :

- Les délégués privés : Délégué n'ayant pas été approuvé par l'administrateur comme étant fiable. De ce fait, son vote est public mais pas le nombre de personnes lui ayant délégué son vote.
- Les délégués publics : Délégué ayant été approuvé par l'administrateur comme étant fiable. De ce fait, à la fois leur vote et le nombre de personnes ayant voté pour lui sont rendus public.

Maintenant que le cadre et le modèle de démocratie liquide a été posé, il est nécessaire d'en faire la modélisation cryptographique afin de permettre aux citoyens de voter en toute confiance et en respectant ces limites. C'est pourquoi nous allons tout d'abord commencer par définir les rôles et fonctions des différents acteurs et adversaires dans le cadre d'un usage cryptographique.

Chapitre 3

Spécification de protocole

Dans cette section, nous allons vous présenter le protocole qui permettra de respecter les exigences d'une élection dans le cadre d'une démocratie liquide. Pour ce faire,nous allons introduire les différents acteurs, leurs rôles ainsi que les exigences de sécurité et de fonctionnement pour chaque acteur. Nous allons aussi introduire l'adversaire type et les hypothèses de confiance.

3.1 Acteurs

Dans notre protocole, nous considérons 5 acteurs différents. Voici la liste des acteurs de notre protocole.

- L'administrateur de l'élection : c'est l'acteur qui va organiser l'élection. Il définit la liste électorale, l'admissibilité des électeurs et le statut des délégués. Il est responsable de publier le résultat final de l'élection qui lui vient des curateurs.
- Les curateurs de l'élection : c'est un groupe d'individus indépendants qui est responsable de la confidentialité des bulletins et de l'intégrité des élections. Ils doivent procéder à la cérémonie de création des clefs avant les élections. Ils y génèrent chacun une paire privée-public Elgamal qu'ils partagent avec l'ensemble des curateurs. Ils publient ensuite la clé jointe. A la fin de l'élection, ils publient les parties de déchiffrement du décompte électoral.
- Les votants : ce sont les utilisateurs du système, ils vont utiliser le système pour voter en remplissant un bulletin de vote. Avant l'élection,ils reçoivent les informations nécessaires pour s'identifier et chiffrer leurs bulletins. Ils peuvent alors utiliser celles-ci pour voter ou pour tester leurs bulletins.

- Les délégués publics : ce sont les votants qui prennent la décision validée par l'administrateur de l'élection de rendre public leur voix et leur nombre de votants. Lors du décompte final de l'élection, leur nombre d'électeurs est ajouté à la décision de loi pour laquelle ils ont opté. Au sein de notre système de vote, il s'agit donc simplement de les implémenter comme options possibles dans le cadre d'un vote, c'est pourquoi nous ne les développerons pas plus.

- Les délégués privés : ce sont les votants qui prennent la décision validée par l'administrateur de se porter en représentant en rendant public leur vote mais pas leur nombre d'électeur. Nous l'implanterons dans notre système de vote en permettant aux autres votants et délégués de copier leurs bulletins de votes.

Votants	**Délégué public**	**Délégué privé**
peut voter et déléguer	peut voter et déléguer	peut voter et déléguer
voix privée	voix publique	voix publique
ne peut pas recevoir de voix	nb. de votants public	nb de votants privés

Table 3.1: Tableau récapitulatif sur la délégation

3.2 Adversaire type et hypothèses de confiance

Dans le cadre d'élections, les tentatives de corruption sont un soucis majeur qu'il faut tenter de limiter à défaut de pouvoir nullifier. C'est pourquoi, maintenant que les acteurs ont été explicités, il convient de définir les adversaires potentiels contre lesquels notre système de vote sera efficace et quelles présupposition concernant ceux-ci nous posons afin d'en connaître les limites.

Nous considérons qu'un adversaire peut corrompre les élections de deux façons différentes : en brisant l'intégrité de l'élection ou en rompant le secret des bulletins. Ils peuvent se trouver parmi les personnes participant à l'élection, qu'il s'agisse des votants, des délégués, des curateurs, ou encore de l'administrateur même des élections, mais il peut également s'agir d'une personne extérieure à l'élection et n'y participant pas. Nous partons du principe que ces adversaires sont discrets, c'est à dire que bien qu'ils veuillent porter atteinte aux résultats de l'élection ou à l'anonymat du vote, ils ne veuillent pas être détectés.

Nos hypothèses de confiances sont au nombre de trois. Premièrement, nous supposons que tous les adversaires sont limités en puissance de calcul de manière raisonnable et ne peuvent donc pas calculer un logarithme discret dans un groupe

cyclique, résoudre le problème DDH [10] ou briser un algorithme de hashage contemporain. Ils ne pourront donc pas déchiffrer les ciphertextes ElGamal sans clé ni générer de preuve ZKP non-interactive valide sans connaître les éléments nécessaires. Deuxièmement, nous assumons qu'au moins l'un des curateurs soit honnête afin de maintenir l'intégrité et le secret de l'élection. En effet, l'ensemble des clefs privées détenues par les curateurs est nécessaire afin de déchiffrer le contenu des bulletins de vote (tant individuels que lors du décompte final) et donc d'en briser le secret. Notre troisième et dernière hypothèse de confiance présume qu'un des client de vote fiable est disponible pour tous les électeurs à un moment donné. Cela permet aux votants de tester et d'envoyer leur bulletin de vote autant de fois qu'ils le veulent étant donné que seul le dernier envoi est comptabilisé. De ce fait, si le client de vote utilisé est corrompu, le votant peut le remarquer en changeant de client de vote. Sur ce nouveau client, il lui suffira de vérifier la fiabilité de son bulletin en le soumettant au challenge de Benaloh, et, s'il remarque une corruption de la machine, la signaler.

Par conséquent, nous pouvons d'ores et déjà constater plusieurs limites lorsqu'il s'agit de créer un système de vote insensible au corruption de l'intégrité des élections et de l'anonymat des bulletins de votes : il ne faut pas que les adversaires soit capable entre autres de résoudre le problème DDH, il faut qu'il existe au moins un curateur honnête et suffisamment de clients de vote fiable. Cependant, nous considérons que ces limites sont acceptables. Il est également important de souligner qu'un adversaire tentant de modifier les résultats de l'élection par contrainte physique sur les votants ou au moyen de pots-de-vin n'est pas couvert dans notre modèle cryptographique.

Maintenant que les acteurs, les adversaires types et les hypothèses de confiance ont été définis, nous allons expliciter la procédure d'organisation de vote générale prenant en compte ces différents acteurs. Cela permettra d'avoir une vision générale de la façon dont se déroule une élection avant d'en donner toutes les preuves mathématiques dans le chapitre suivant.

3.3 Organisation de l'élection

L'élection va être découpée en trois phases organisées selon la temporalité de celle-ci. Premièrement, nous allons voir les étapes nécessaires à la mise en place de l'élection avant même le commencement de celle-ci, que nous appellerons "pré-élection". Ensuite, nous détaillerons les différentes phases présentes lors de l'élection dans la partie "élection". Enfin, nous verrons rapidement la façon dont l'élection est clôturée et les vérifications nécessaires à cette fin dans la partie "après-élection".

3.3.1 Pré-Élection

La pré-élection contient trois grandes étapes : la création de l'élection par l'administrateur, la génération des clefs partielles par le curateurs et la cérémonie de fusion des clefs.

3.3.2 Création de l'élection par l'administrateur

Comme nous l'avons vu précédemment, l'administrateur a pour rôle d'organiser l'élection en soi. De ce fait, avant même le commencement de l'élection, l'admin va commencer par établir la liste des votants et des délégués. Ceux-ci seront définis selon les règles établies préalablement à l'élection (ie. âge, nationalité,...). Dans le cas d'un délégué, l'administrateur devra également les scinder en délégués publics et délégués privés, et ce, de nouveau selon des réglementations pré-établies (ie. volontariat, nombre de personnes ayant votés pour lui...). En effet, comme nous l'avons vu, les délégués privés ne peuvent savoir combien de personnes ont votés pour eux. Cela permet d'avoir des délégués n'ayant que peu de votants sans craindre une corruption trop facile étant donné le faible nombre de votant (ie. payer des votants et pouvoir vérifier que ceux-ci ont bien voté pour lui, ce système de corruption étant beaucoup plus simple à mettre en place pour un délégué n'ayant que 10 personnes ayant voté pour lui que pour un délégué possédant 10.000 votes)[28].

De plus l'administrateur a pour responsabilité d'établir la liste de question et de réponses possibles nécessaires à l'élection. Dans le cas d'une démocratie liquide, on peut imaginer qu'il s'agisse par exemple d'une ou plusieurs propositions de loi ainsi que des réponses "Pour - Contre".

3.3.3 Génération des clefs partielles par les curateurs

Les curateurs sont choisis au moment de la création de l'élection. Il est important que les curateurs soient choisis de façon à ce qu'ils soient le plus fiable possible. Cela permet d'éviter qu'ils collaborent entre eux afin de fausser l'élection étant donné que, comme nous l'avons vu, le système de vote nécessite au moins un curateur honnête pour fonctionner.

Une fois les curateurs choisis, une paire de clef partielle va être générée pour chaque curateur. Cette paire de clef contiendra une clef partielle privée et une clef partielle publique. Le curateur va devoir prouver qu'il connaît sa clef privée partielle afin de préserver le secret de l'élection et va partager sa clef publique avec les autres curateurs. Cela sera nécessaire à la mise en place de la troisième étape de la pré-élection.

3.3.4 Cérémonie de fusion des clefs

Une fois que les clefs publiques partielles de chaque curateur ont été partagées, on procède à la cérémonie de fusion des clefs. La fusion des clefs publiques partielles va constituer la clef publique de l'élection. Cette clef publique de l'élection permettra aux votants et aux délégués de chiffrer leur vote par la suite. De plus, la preuve de connaissance des clefs partielles envoyées par les curateurs va être vérifiée.

Voici un schéma permettant de résumer les trois grandes phases permettant d'instaurer une élection :

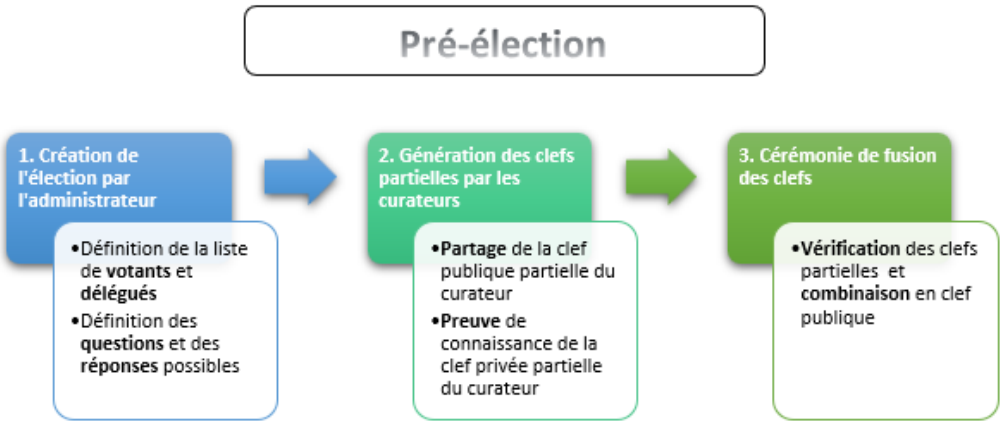

Figure 3.1: Schéma des étapes nécessaires à une pré-élection

Une fois toutes ces étapes mise en place, l'élection peut débuter. Les votants et les délégués vont pouvoir commencer à voter.

3.4 Élection

L'élection est découpée en deux grandes parties. Premièrement, les délégués votent et envoie la preuve de leur vote. C'est ensuite seulement que les votants pourront voter à leur tour et enverront également la preuve de leur vote.

3.4.1 Vote par un délégué

Dans le cadre de ce mémoire, nous implémenterons uniquement le cas des délégués privés. En effet, les délégués publics étant totalement public autant au niveau de leur nombre de voix que de leur voix en tant que tel, ils ne représentent que peu

d'intérêt en terme de représentation cryptographique. On pourrait, par exemple, inclure les délégués publics directement à la liste de réponses proposées afin que les votants puissent les choisir immédiatement étant donné que le vote des délégués publics ainsi que le nombre de personnes ayant voté pour eux est rendu public. Il est important de noter que dans le cadre d'une démocratie liquide, il est nécessaire que les votes de tous les délégués, publics ou privés, soient rendu publics. Cela permet une plus grande transparence lors de l'élection et aux votants de vérifier que l'intention de vote annoncée par le délégué a bien été respectée.

Nous utiliserons deux méthodes de vote direct et de copie de vote pour les délégués. La différence principale entre les deux méthodes est le moment auquel les votes des délégués vont être révélés. Lors de la méthode 1, le vote des délégués est rendu public au moment même de leur vote. Cela présente le même type d'inconvénient qu'un vote à main levée : le fait de voir les résultats varier au fur et à mesure de l'élection et de la phase de vote peut influencer le vote de certains délégués et de certains votants [13]. La raison pour laquelle le vote à main levée est plus couramment utilisé est qu'il permet une élection plus rapide, or, dans le cas de vote électronique, les deux méthodes de vote ont un temps quasiment équivalent. De plus, cela permettrait des corruptions actives afin de renverser le cours du vote en cours. Il nous a donc paru nécessaire de développer une deuxième méthode de vote dans laquelle les délégués ne révèlent leur vote qu'à la fin des élections en même temps que les votants.

Lors de la méthode 1, les délégués auront deux choix, comme les votants : voter directement pour une proposition de loi ou voter pour un délégué et lui déléguer son vote et donc le copier. Dans les deux cas, le délégué va partager l'aléatoire du chiffrement (cf. Chapitre 4) et, de ce fait, son vote sera immédiatement rendu public.

Lors de la méthode 2, le délégué possède également le choix de voter directement ou de copier son vote. Par contre, afin de ne pas rendre son vote public, il enverra tout d'abord son vote et ensuite la preuve que soit : son vote est valide (dans le cas où il vote directement) ou bien que son vote est la copie d'un vote valide (dans le cas où il choisit de copier un vote d'un autre délégué).

Une fois que les délégués ont votés et que leur vote ont été enregistrés, c'est au tour des votants de procéder au vote.

3.4.2 Vote par un votant

Le votant, comme le délégué, aura le choix de voter directement pour la question (ie. une proposition de loi) ou bien de déléguer son vote et donc de copier le vote d'un délégué. Dans le cadre de la démocratie liquide, il est bien sûr nécessaire que le vote des votants reste anonyme et ce, pour toute la durée de l'élection. Dans le

cas d'un vote direct, qu'il s'agisse de la méthode 1 ou de la méthode 2, le votant vote simplement pour une réponse. Dans le cas d'une copie de vote par le délégué, le processus diffère selon qu'il s'agisse de la méthode 1 ou 2. Dans le cas de la méthode 1, il enverra la preuve que son vote est valide. Dans la méthode 2, il enverra la preuve que son vote est la copie d'un des votes des délégués et ce afin de ne pas être en mesure de savoir pour quel délégué il a voté.

L'étape d'élection est la partie centrale de ce mémoire puisqu'elle est celle qui implémente la fonctionnalité de copie d'un vote de délégué privé nécessaire à la démocratie liquide. Comme pour la partie pré-élection, voici un schéma résumant les grandes étapes de celle-ci :

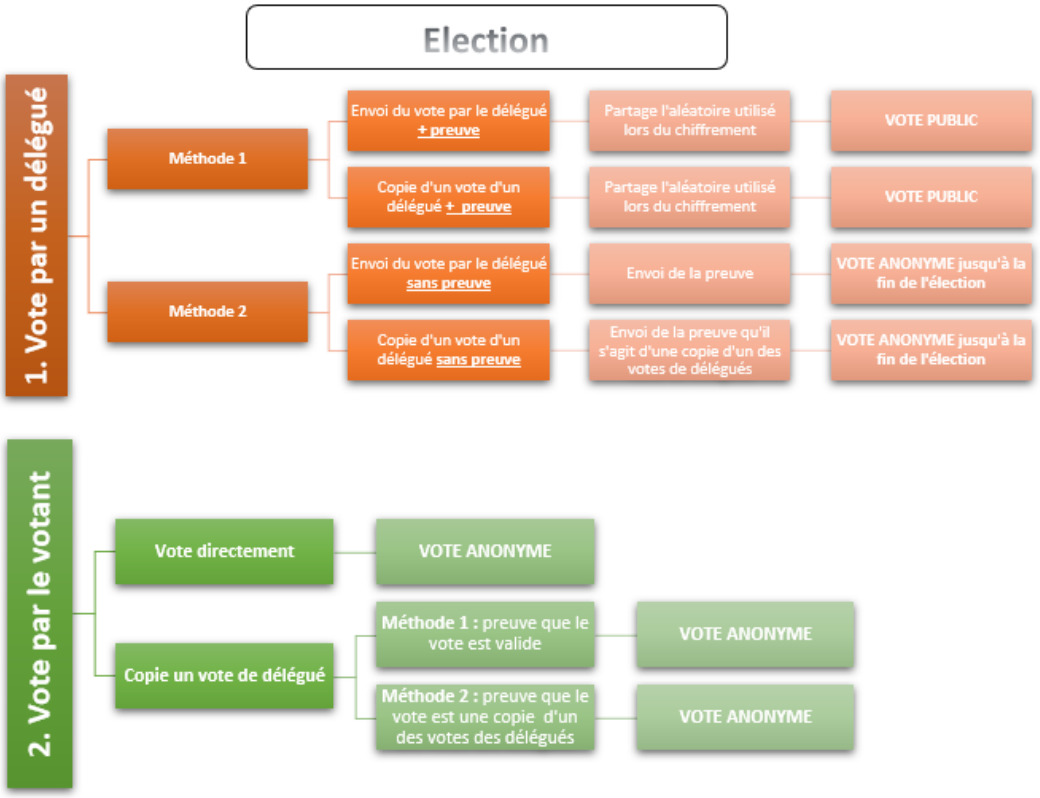

Figure 3.2: Schéma des étapes nécessaires à une élection

Nous pouvons maintenant aborder la dernière étape indispensable à l'organisation d'une élection : l'après-élection.

3.5 Après-élection

Une fois que tous les délégués puis les votants ont déposés leur ballots, il ne reste plus qu'à les comptabiliser. Cela se décompose en trois étapes : tout d'abord, on vérifie les bulletins de vote et on les additionne, ensuite on calcule les décomptes partiels, et enfin on calcule le résultat des élections.

3.5.1 Vérification des bulletins de votes puis addition des votes

Lors de cette étape, les votes des votants et des délégués sont vérifiés afin de contrôler que leurs bulletins de vote sont valide avant de les additionner.

3.5.2 Calcul des décomptes partiels par les curateurs

Les curateurs vont de nouveau se réunir à la fin de l'élection. Ils vont utiliser chacun leur clef privée partielle afin de déchiffrer un décompte partiel de l'élection. Une fois que ce déchiffrement partiel est effectué, ils devront envoyer la preuve que le décompte partiel ainsi obtenu est lui aussi valide.

3.5.3 Calcul du résultat de l'élection

Une fois que les décomptes partiels ont tous été envoyés et prouvés, ils vont être combinés afin qu'on obtienne le résultat final de l'élection. Dans le cadre d'une démocratie liquide, cela se traduirait par le nombre de vote pour chaque réponse proposée. Il est à noter que dans le cas de la méthode 2, c'est aussi à ce moment là que le vote des délégués privés serait rendu public.

Maintenant que toutes les étapes d'une élection ont été spécifiées et détaillées, voici un schéma les reprenant toutes afin de posséder une vue globale de l'organisation d'une élection :

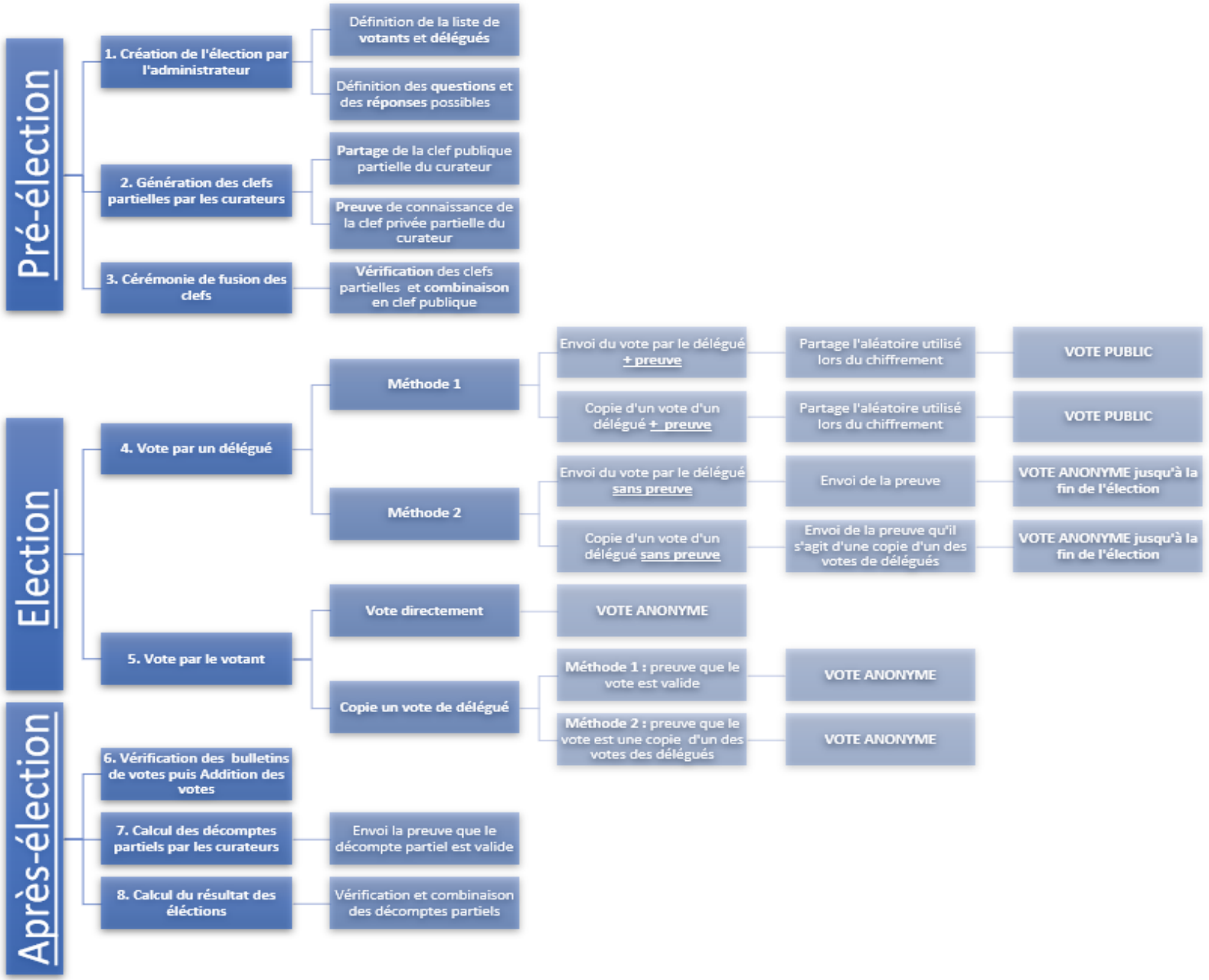

Figure 3.3: Schéma général des étapes nécessaires à une élection

Nous venons donc d'établir les spécificités de notre protocole. Pour ce faire, nous en avons définis les acteurs, nous avons soulignés les hypothèses de confiance et les adversaires types, et enfin, nous avons stipulés quelles seraient les étapes à mettre en place afin d'inscrire notre simulation de système de vote dans le cas d'une démocratie liquide. Nous pouvons donc désormais établir notre protocole à proprement parler et par conséquent toutes les preuves mathématiques nécessaires à son fonctionnement.

Chapitre 4

Protocole

Ce mémoire va utiliser Helios comme inspiration de système de vote et l'adapter dans le cadre d'une démocratie liquide. Helios Voting est un système de vote électronique End-to-End open-source sur navigateur. Il permet à la fois de voter dans le cadre d'élection mais aussi d'en créer. Il assure le secret des bulletin de vote en utilisant un chiffrement ElGamal homomorphe. Les bulletins de vote sont chiffrés et publics, ce qui facilite la vérification universelle. Ce système de vote est considéré comme à la pointe du développement [12]. Helios Voting a été initialement présenté par Adida en 2008 puis amélioré en Helios 2.0 dont le protocole a été détaillé par Adida, de Marneffe, Pereira et Quisquater en 2009 [4].

Le but du présent protocole sera donc d'introduire un design général du système de vote dans un cadre cryptographique. De ce fait, il faudra qu'il remplisse les conditions fixées lors de notre cadre théorique défini dans la section 2 mais aussi les spécificités de celui-ci définies dans la section 3. Mais pour commencer, nous allons expliciter les pré-requis mathématiques nécessaires afin d'aider à la compréhension générale de ce protocole cryptographique.

4.1 Pré-requis mathématiques

4.1.1 Chiffrement ElGamal

Le schéma de chiffrement ElGamal, qui est asymétrique ou encore à clé publique [14], est un algorithme en 3 parties: la génération de clé, le chiffrement et le déchiffrement (KeyGen, Enc, Dec). Il opère sur des groupes cycliques où le calcul du log discret est supposé difficile. Pour générer un tel groupe, il est d'usage de prendre un nombre premier p tel que $q = (p-1)/2$ est également premier. Le groupe généré $\mathbf{Z}_p^*$ a $p-1$ éléments [7], pour qu'il ne soit pas aisé de calculer les logarithmes discret, il est essentiel que $p-1$ aie un grand facteur premier[14]. On

prend alors un élement $g \in \mathbf{Z}_p^*$ d'ordre q, le sous-groupe $G :=< g >\subset \mathbf{Z}_p^*$ est aussi d'ordre q. Le schéma de chiffrement ElGamal vit dans G donné par les paramètres (p,q,g) [7].

Une paire de clé ElGamal est composée d'une clé privée et d'une clé publique. Lors de la génération de clé (KeyGen), on prend la clé privée, $sk \in \mathbf{Z}_q$, aléatoirement et $pk = g^{sk} \in G$ est la clé publique. La paire d'ElGamal s'écrit (sk,pk).

Pour chiffrer $m \in G$ avec $r \in \mathbf{Z}_q$ pris de manière uniformément aléatoire,

$$Enc_{pk}(m, r) = (m * pk^r, g^r) \in G \times G.$$

Le déchiffrement de e=(c,d) peut s'effectuer avec la clé secrète sk.

$$m \leftarrow Dec_{sk}(e) = c/d^{sk}$$

On peut voir que $\forall m \in G$ et $\forall r \in \mathbf{Z}_q$

$$Dec_{sk}(Enc_{pk}(m, r)) = m$$

Le chiffrement d'ElGamal est homomorphique par rapport à la multiplication.

$$Enc_{pk}(m_1, r_1) * Enc_{pk}(m_2, m_2) = Enc_{pk}(m_1 * m_2, r_1 + r_2)$$

4.1.1.1 Elgamal exponentiel

On peut modifier la méthode de chiffrement d'ElGamal afin de rendre le décompte des voix plus aisé. On modifie l'encodement du message pour qu'il devienne g^m, le chiffrement devient

$$Enc_{pk}(m, r) = (g^m * pk^r, g^r) \in G \times G.$$

Si l'on veut additionner les messages sans déchiffrer les messages individuellement, on peut multiplier les ciphertextes. On se retrouve alors avec une somme de tout les messages, comme $g^{m_1} * g^{m_2} = g^{m_1+m_2}$. On déchiffre alors le dernier message qui,de part la modification, sera sous la forme $g^{\Sigma m_i}$. Il faudra alors en calculer le log pour avoir le résultat. Dans le cas d'un question dichotomique, Σm_i est le nombre de voix, il est borné entre 0 et le nombre de votant, le résultat du log est donc calculable en quelques secondes sur un ordinateur portable classique [25].

Dans le cas d'une élection, on peut multiplier tous les bulletins de vote correspondant à une question dichotomique. Cela permet de calculer rapidement le nombre de personnes ayant répondu positivement.

4.1.2 Zero-Knowledge-Proof

Lors d'une élection, les votants ou les curateurs vont être obligés de fournir une preuve qu'ils respectent les conditions nécessaires au bon fonctionnement de l'élection sans dévoiler d'informations secrètes. Les preuves à divulgation nulle de connaissance, ou *Zero-Knowledge-Proof* (ZKP) sont des outils qui permettent de prouver que l'opération a été correctement effectuée sans révéler plus que ce fait [7]. Lors de la cérémonie de génération de clé, les curateurs vont devoir prouver qu'ils connaissent la clé privée liée à la clé publique qu'ils partagent tout en maintenant le secret de leur clé privée. Ils utiliseront alors les ZKP. Afin d'avoir une meilleure compréhension des ZKP, il est utile de présenter les protocoles Sigma qui représentent les schémas communs de toutes les ZKP utilisées dans Helios.

4.1.2.1 Protocole Sigma

Un protocole Sigma définit une interaction en trois temps entre un prouveur P et un vérifieur V.[25]. Ces trois temps se décomposent comme suit:

- P envoie un **engagement** (a) constitué d'une valeur aléatoire à V.
- V envoie à P un **challenge** (e) constitué d'un nombre aléatoire, ce nombre ne doit pas être connu de P avant son envoi.
- P envoie à V le **résultat** (f) composé d'une combinaison comprenant le challenge.

V peut alors vérifier si a, e et f sont cohérents par rapport à la preuve et accepter celle-ci le cas échéant.

Ces preuves sont dites interactives, elles nécessitent la présence d'un prouveur P et d'un vérifieur V. Cependant, il est à noter que dans une élection, les preuves doivent être disponibles pour vérification continue. Pour faciliter celle-ci, il est utile de rendre ces preuves non-interactives. Helios utilise la transformation de Fiat-Shamir afin d'effectuer la transformation d'interactive à non-interactive [8]. Cette transformation consiste à enlever le vérifieur. Ainsi, on remplace challenge (e) par un hash de l'engagement (a) pouvant être accompagné d'autres éléments nécessaires.

Nous définirons les ZKP utilisant la structure d'un protocole Sigma nécessaires à ce système de vote au moment où leur nécessité se présentera dans la suite de ce protocole.

Maintenant que les prérequis mathématiques ont été définis et démontrés, et que les paramètres, les adversaires et les hypothèses de confiance ont été explicitées dans les sections précédentes, nous sommes prêts à présenter les trois phases du protocole de façon détaillée.

4.2 Pré-élection

Durant la phase de pré-élection, on effectue toutes les tâches permettant la mise en place d'une élection. Elle est constitué de deux phases principales, la création de l'élection et la cérémonie de génération de clef que nous allons maintenant définir.

4.2.1 Création de l'élection

Pour commencer l'élection, l'administrateur doit effectuer une série de tâches afin de rendre son organisation possible. On identifie 5 tâches principales :

- L'administrateur de l'élection donne le nom à l'élection pour pouvoir l'identifier.
- Il définit ensuite la liste des délégués privés. Ce seront des votants spéciaux qui vont publier des bulletins copiables par les autres votants.
- L'administrateur doit ensuite définir la liste des questions et des réponses sujettes au vote, ainsi que le nombre de réponse que le votant à le droit de choisir.
- Il doit définir la liste des votants qui sont identifiables par une ID.
- L'administrateur sélectionnera les curateurs pour l'assister lors de cette élection. Les curateurs vont générer ensemble la clef publique de l'élection.

4.2.2 Génération des clés

Lorsque les curateurs sont sélectionnés, ils doivent procéder à la cérémonie de génération de clefs. Comme les bulletins de vote sont chiffrés en utilisant un chiffrement ElGamal et que le chiffrement de l'élection et de décompte des votes ne peut pas être confiée à une seule personne, cette tâche est confiée à un ensemble de personnes : les curateurs. Comme dit précédemment, l'ensemble des curateurs doit être corrompu pour pour que l'élection soit corrompue sans que cela ne soit détecté. En effet, il est nécessaire que tous les curateurs soient présent lors de la cérémonie de génération de clefs ainsi que pour le décompte final des votes.

4.2.2.1 Protocole de génération des clés

Chaque curateur C_i, va générer une paire de clef ElGamal (sk_i, pk_i) dont il partagera la clef publique pk_i pour former la clef publique de l'élection $pk_E = g^{sk_E} = \Pi g^{sk_i}$. La clef privée de l'élection sk_E reste inconnue de tout le monde. Les curateurs réutiliseront leur clef secrète pour faire un déchiffrement partiel de l'élection qui sera rassemblé suivant un procédé détaillé dans la partie déchiffrement de ce mémoire.

Il est essentiel que les curateurs connaissent la clef privée liée à la clef publique qu'ils partagent lors de la cérémonie. En effet, un curateur C_n malicieux pourrait attendre que tous les autres curateurs C_i postent leurs clefs publiques pour construire la sienne. Il ajouterait alors sa clef publique $y_n = \frac{y}{\Pi_{i=1}^{n-1} sk_i}$. Comme il connaît (sk, pk) et que la clef de l'élection est $pk_E = \frac{pk}{\Pi_{i=1}^{n-1} sk_i} \Pi_{i=1}^{n-1} sk_i = pk_n$, il connaîtrait alors la clef privé de l'élection et pourrait déchiffrer chaque vote individuellement.

Les curateurs vont donc être obligés de prouver la connaissance de leur clef secrète en utilisant la ZKP présentée ci-dessous.

4.2.2.2 ZKP utilisée lors de la génération des clefs

Nous allons d'abord présenter la **version interactive** du protocole avant d'en présenter la version non-interactive.

Voici un protocole prouvant la connaissance de la clef secrète sk liée à une clé publique pk formant donc la paire ElGamal(sk,pk).

Protocole:

- P sélectionne de manière uniformément aléatoire une autre paire (sk',pk') et envoie pk' comme engagement à V, a=pk'.
- V sélectionne de manière uniformément aléatoire le challenge (e) entre 0 et $n-1$ avec $n \leq q$ et l'envoie à P.
- P calcule $sk'' := sk' + e * sk (\text{mod q})$ et envoie la réponse f=sk" à V.

On peut calculer $pk'' = pk' * pk^e$ (mod p). V peut alors vérifier que $g^{sk''} = pk''$ (mod p). Si cette équation est correcte, V conclut en la connaissance de la clé secrète par P.

La solidité de cette preuve tient grâce à deux observations [25]:

- Si P est capable de produire une réponse (f) qui passe le test d'acceptation réalisé par V après qu'il ait remis un engagement (a), alors P est sûrement capable de le faire pour plus d'une valeur de challenge (e). Si P est capable de réussir le test pour une unique valeur de challenge (e), sa probabilité de réussir la preuve est de $1/q$, ce qui est improbable.
- Si nous pensons P capable d'envoyer la réponse (f) correcte pour deux challenges différents e_1 et e_2 au même engagement (a) alors nous pensons qu'il connaît sk. En effet, on peut retrouver

$$sk = \frac{(sk' + e_1 * sk) - (sk' + e_2 * sk)}{e_1 - e_2} = \frac{f_1 - f_2}{e_1 - e_2}$$

L'intuition derrière la propriété "Zero-Knowledge" de cette preuve est la suivante : V n'apprend rien de cette interaction (en dehors de la connaissance de sk par P) parce que V est capable de reproduire une interaction qui suit exactement la même distribution par lui-même. V choisirait (e) et (f) aléatoirement depuis $\mathbf{Z}_q$ et calculerait ensuite $a = \frac{g^f}{g^{xe}}$ [25].

Nous allons maintenant présenter la **version non-interactive** du protocole.

Le prouveur P veut prouver qu'il connaît la clef privée sk liée à la clef publique pk dans la paire ElGamal(sk,pk).

Protocole:

- P sélectionne de manière uniformément aléatoire une autre paire (sk',pk'); l'engagement est a = pk'.
- P calcule le challenge $e := H(pk, a)$ (mod q), avec H une fonction de hashage.
- P calcule le résultat $f := sk' + e * sk$ (mod q) et le partage.

La clé publique est pk et la preuve de sa justesse est $\Pi = (e, f)$. Le prouveur partage (pk, Π)

Tout observateur peut alors vérifier la justesse des ZKP non-interactives de chaque curateur et donc s'assurer que la clé publique a été générée de manière correcte.

Une fois toutes ces étapes effectuées, l'élection peut débuter et avec elle les tâches nécessaires à son bon fonctionnement.

4.3 L'élection

La phase d'élection est la phase principale du protocole. L'administrateur doit définir la date de début et de fin de cette phase. Les votants sont alors invités à voter.

Pour permettre la transitivité des votes, une faille d'Helios remarquée par Desmedt et Chaidos est utilisée [12]. Ils ont remarqué qu'avec l'accord d'une personne ayant voté, il était possible de copier le résultat de son vote sans que personne d'autre ne soit au courant. Les votes des délégués seront donc requis en premier afin de permettre leur utilisation future par les votants. L'élection commencera donc par une phase de vote de délégués où tous les délégués sont invités à voter. Ce choix de séparer la phase de vote en deux permet d'assurer aux votants de pouvoir voter quand ils veulent durant leur phase pour n'importe lequel des délégués sans qu'ils ne se retrouvent dans la situation où le délégué de leur choix n'a pas encore rendu disponible son bulletin sur le tableau des bulletins.

Lors de ces deux phases, les votants, qu'ils soient délégués ou non, ont deux options : soit ils votent directement pour un choix de la liste, soit ils reprennent un vote de délégué présent sur le tableau des bulletins et ils le copient. Pour commencer, la ZKP prouvant la validité du bulletin sera présentée. Nous définirons ensuite la théorie sous-jacente au copiage des votes. Nous finirons par une présentation du protocole suivi par les votants et les délégués lorsqu'ils votent.

Lors de ces votes, les votant peuvent modifier le résultat de l'élection en choisissant de chiffrer un résultat autre que 0 ou 1. Ils peuvent par exemple chiffrer 10000 ce qui augmentera de 10000 voix positives le décompte final. Il faut donc s'assurer qu'ils chiffrent un nombre autorisé. Nous utiliserons pour ce faire la ZKP présentée ci-dessous.

4.3.1 ZKP prouvant la validité du bulletin

Ce protocole ZKP est une version disjonctive du protocole de Chaum-Pedersen. Les votants doivent prouver que chacun des ciphertextes qu'ils envoient soient soit un chiffrement de 0 ou un chiffrement de 1. De plus, il faut que le produit de tous ces ciphertextes soit un chiffrement d'un nombre entier compris dans l'intervalle prescrite, indiquant qu'un nombre valide de réponses a été sélectionnée par le votant [25].

Il est possible de simuler un protocole sans connaître la clef secrète, car il est nécessaire de le faire lors de la preuve de Zero-knowledge du protocole, on sélectionne alors soit même le challenge. Pereira et al utilisent cette propriété pour demander au prouveur qu'il prouve à la fois qu'il a encodé 1 et qu'il a encodé 0, une seule des deux preuves est honnête et l'autre est simulée.

La première étape consiste à montrer la version modifiée du protocole de Chaum-Pedersen. Ce protocole cherche à prouver que le vote, qui a une forme de chiffré d'Elgamal exponentiel $(g^m * pk^r, g^r)$, chiffre bien m=v avec la paire d'Elgamal (sk,pk) et l'aléatoire r.

Protocole:

- P sélectionne de manière uniformément aléatoire $s \in \mathbf{Z}_q$ et envoie la paire $(a_1, a_2) = (g^s, pk^s)$ (mod p) comme engagement à V.
- V sélectionne de manière uniformément aléatoire le challenge e entre 0 et $n - 1$ avec $n \leq q$ et l'envoie à P.
- P calcule $f = s + e * r$ (mod q) et envoie la résultat f à V.

Le vérifieur accepte la preuve si $g^f = a_1 * (g^r)^e$ et $pk^f = a_2 * (\frac{g^m * pk^r}{g^v})^e$ (mod p). On voit qu'il est nécessaire que m=v pour que le vérifieur accepte la preuve.

Ensuite, voici la version disjonctive de ce protocole.

Ce protocole cherche à prouver que le chiffré d'ElGamal exponentiel $(g^m * pk^r, g^r)$ chiffre bien $m = v_0$ et $m = v_1$ avec la paire d'ElGamal (sk,pk) et l'aléatoire r. Le prouveur commence par simuler le résultat pour celui des deux cas qui n'est pas correct parmi $m = v_0$ et $m = v_1$.

Protocole:

- Simulation: P choisit e_{sim} et f_{sim} de manière uniformément aléatoire dans $\mathbf{Z}_q$ et calcule $a_{1sim} = \frac{g^{f_{sim}}}{g^{r*e_{sim}}}$ et $a_{2sim} = \frac{g^{sk*f_{sim}}}{g^{sk*r*e_{sim}}}$ (mod p)
- P sélectionne de manière uniformément aléatoire $s \in \mathbf{Z}_q$ et envoie à V les paires $(a_{1real}, a_{2real}) = (g^s, pk^s)$ (mod p) et (a_{1sim}, a_{2sim}).
- V sélectionne de manière uniformément aléatoire e entre 0 et $n-1$ avec $n \leq q$ et l'envoie à P.
- P calcule $e_{real} = e - e_{sim}$. Il calcule ensuite $f_{real} = s + e_{real} * r$ et envoie f_{real} et f_{sim} ainsi que e_{real} et e_{sim} à V.

Le vérifieur accepte la preuve si $g^f = a_1 * g^{re}$ et $pk^f = a_2 * (\frac{g^m * pk^r}{g^v})^e$ (mod p) pour les deux versions et si $e_{real} + e_{sim} = e$.

Comme les preuves sont symétriques, il est impossible de savoir laquelle des deux parties est simulée.

4.3.2 Vote copiable

Desmedt et Chaidos ont proposé une modification d'Helios qui permettrait l'émission d'un nouveau vote en copiant son vote et sa preuve avec la complicité du votant de manière à ce que cette manipulation soit indétectable et que le vote résultant soit indistinguable d'un vote classique [12]. Ce protocole est divisé en deux parties : la copie du vote et la copie de la preuve liée au vote.

4.3.2.1 Copie masquée du vote

Avec un vote $(\alpha, \beta) = (g^r, pk^r * g^v)$, où le vote v est égal à 0 ou 1, un copieur peut copier$(\alpha', \beta') = (g^{r+z}, pk^{r+z} * g^v)$. Il n'a besoin d'aucune connaissance supplémentaire que le vote présent sur le tableau des bulletins, il choisit z uniformément aléatoire dans $\mathbf{Z}_q$ et calcule,$(\alpha', \beta') = (\alpha * g^z, \beta * pk^z)$. On doit maintenant prouver que le vote ainsi généré est valide et indistinguable et pour qui il peut générer une preuve valide.

Validité du vote et de la preuve

Si le copieur a accès à une preuve pour (α, β), il peut la transformer en une preuve valide pour(α', β').

Preuve : Si (V,P) est un bulletin valide, avec (α, β) et $P = ((a_0, b_0, a_1, b_1), (e_0, f_0, e_1, f_1))$ une preuve valide comme détaillée dans l'explication de la ZKP ci-dessus, alors $((g^{r+z}, pk^{r+z} * g^v), (a_0, b_0, a_1, b_1), (e_0, f_0 + e_0 * z, e_1, f_1 + e_1 * z))$ est un ballot valide.

Nous avons alors:

Si pour i=0 et 1, l'équation

$$a_i = \frac{g^{f_i}}{\alpha^{e_i}} \; est \; correcte$$

Alors,

$$a_i = \frac{g^{f_i + e_i * z}}{(g^z * \alpha)^{e_i}} \; est \; correcte.$$

De manière similaire, si

$$b_i = \frac{pk^{f_i}}{((\beta / g^i)^{e_i}} \; est \; correcte.$$

Alors,

$$b_i = \frac{pk^{f_i + e_i * z}}{((pk^z \beta / g^i)^{e_i}} l'est \; aussi$$

.

La bulletin copié est donc valide.

Indistinguibilité du vote

Si z est choisi de manière uniformément aléatoire dans $\mathbf{Z}_q$, il s'en suit que $g^z * \alpha$ est uniformément aléatoire dans G. Il existe un s dans $\mathbf{Z}_q$ tel que $\alpha' = g^s$ et $\beta' / g^v = pk^s$. Comme la clef secrete de pk dans la paire (sk,pk), s et β' / g^v sont choisis de manière indépendante, $(pk, g^s, \beta' / g^v)$ est un problème DDH qu'un adversaire ne peut résoudre que si il peut distinguer (α, β) d'un autre vote. La preuve de vote P reste distinguable des autres votes de part sa similitude avec le vote copié.

4.3.2.2 Première méthode de copie de la preuve de vote

Dans la section précédente, on a vu qu'il était possible de copier un vote et de le rendre indistinguable sans interaction entre le copieur et le copié. Comme il n'est pas possible de créer de nouvelles preuves valides sans connaître l'aléatoire

utilisé dans le vote, la collaboration du copié est nécessaire. Ci-suit le protocole zero-knowledge pour créer une nouvelle preuve d'un ballot indistinguable avec la preuve du bulletin copié.

Protocole:

- Le copié choisit $w \in_{\mathbf{R}} Z_q$ et définit $a_{real} := g^w$ et $b_{real} := pk^w$. Pour la partie simulée, le copié choisit $e_{sim}, f_{sim} \in_{\mathbf{R}} \mathbf{Z}_q$ et définit $a_{sim} := \frac{g^{f_{sim}}}{\alpha^{e_{sim}}}$ et $b_{sim} := \frac{pk^{f_{sim}}}{(\beta/g^{sim})^{e_{sim}}}$ Le copié envoie l'engagement $(a_{real}, b_{real}, a_{sim}, b_{sim})$ au copieur.

- Le copieur choisit $\Delta_0, \Delta_1, k_0, k_1 \in_{\mathbf{R}} \mathbf{Z}_q$. Comme les deux parties sont simulées, il définit $A_i := \frac{a_i * g^{k_i}}{\alpha^{\Delta_i}}$ et $B_i := \frac{b_i * pk^{k_i}}{(\beta/g^i)^{\delta_i}}$ pour i=0 ,1. Nous avons $e = H(A_0, B_0, A_1, B_1)$ le challenge non interactif nécessaire à la preuve. Le copieur définit $E := e - \Delta_0 - \Delta_1$ et envoie E au copié comme challenge.

- Le copié calcule $e_{real} = E - e_{sim}$ et il calcule $f_{real} = w + r * e_{real}$, il envoie $(e_{real}, f_{real}, e_{sim}, f_{sim})$ au copieur.

- Le copieur vérifie que $E = e_0 + e_1$ et il vérifie que $a = \frac{g^f}{\alpha^e}$, $b = \frac{pk^f}{(\beta/g^v)^e}$ pour les deux votes. Si c'est correct, il calcule $E_i = e_i + \Delta_i$ et $F_i = f_i + k_i$. Le bulletin est alors $((\alpha, \beta), (A_0, B_0, A_1, B_1), (E_0, F_0, E_1, F_1)$. Il envoie ce bulletin sur le tableau des bulletins.

Il faut maintenant prouver que ce protocole est Zero-Knowledge mais aussi qu'il génère des bulletins corrects et indistinguables tout en ne dévoilant pas le vote du copié.

Preuve 1:

Le protocole de masquage de preuve est complet. De plus, si le copié est honnête, il sera accepté sur le tableau des bulletins.

Prouver que le protocole est complet est trivial.

Nous avons:

- $E = e_0 + e_1$ est valide comme le copié calcule $e_{real} = E - e_{sim}$

- La preuve simulée est valide par construction.

- Pour la preuve réelle, nous avons $a = \frac{g^f}{\alpha^e} = \frac{g^{w+r*e}}{\alpha^e}$ qui est correct comme $a = g^w$. Nous avons également $b = \frac{pk^f}{(\beta/g^v)^e} = \frac{pk^{w+r*e}}{(\beta/g^v)^e}$ qui est correcte comme $b = pk^w$ et $\frac{\beta}{g^v} = pk^r$.

Le bulletin de vote sera accepté si $E_0 + E_1 = (A_0, B_0, A_1, A_2)$ et si $A_i = \frac{g^{f_i}}{\alpha^{E_i}}$ et $B_i = \frac{pk^{F_i}}{(\beta/g^i)^{E_i}}$. Comme les égalités $a = \frac{g^f}{\alpha^e}$ et $b = \frac{pk^f}{(\beta/g^v)^e}$ sont correctes pour la preuve simulée et réelle, lorsqu'on remplace les variables (A_i, B_i, E_i, F_i) par leur définition, on remarque alors que le bulletin sortant est valide.

Preuve 2:

La preuve masquée créée par ce protocole est indistinguable de toute preuve valide créée de manière indépendante.

Nous avons $E_i = e_i + \Delta_i$ et $F_i = f_i + k_i$ avec Δ_i et k_i sélectionnés de manière uniformément aléatoire dans $\mathbf{Z}_q$. Le challenge et la réponse sont donc indépendants de ceux utilisés dans la preuve originale. Les valeurs de (A_0, B_0, A_1, B_1) sont uniquement déterminée par leur définition par (E_0, F_0, E_1, F_1) ($A_i := \frac{a_i * g^{k_i}}{\alpha^{\Delta_i}}$ et $B_i := \frac{b_i^p k^{k_i}}{(\beta/g^i)^{\delta_i}}$). Donc si un votant est capable de distinguer la preuve d'une autre preuve, il est capable de distinguer (E_0, F_0, E_1, F_1) qui est choisi de manière uniformément aléatoire.

Preuve 3:

Le protocole de masque de preuve possède la propriété de robustesse spéciale ("special soundness").

Si un copié peut produire une réponse à deux différents challenges E,E' pour un même engagement (a_0, b_0, a_1, b_1). Nous allons montrer que ce copié peut calculer l'aléatoire utilisé lors du chiffrement du vote. Comme $E \neq E'$, il doit exister $e_i \neq e'_i$ et $f_i \neq f'_i$ pour les deux réponses et challenges pour au moins un i, $i \in$0,1.

Nous avons,

$$a_i = \frac{g^{f_i}}{\alpha^{e_i}} \; et \; a_i = \frac{g^{f'_i}}{\alpha_i^{e'_i}}, alors$$

$$\frac{g^{f_i}}{\alpha_i^{e_i}} = \frac{g^{f'_i}}{\alpha_i^{e'_i}}, g^{f_i - f'_i} = \alpha^{e_i - e'_i}$$

on peut donc calculer

$$r = log_g \alpha = \frac{e_i - e'_i}{f_i - f'_i}$$

Preuve 4:

Sous l'hypothèse du "random oracle model", ce protocole de masquage est zero-knowledge pour un copieur qui suit le protocole.

Sous l'hypothèse du "random oracle model", et supposant le probleme DDH difficile à résoudre, nous allons décrire un simulateur pour ce protocole quand le copié est honnête. Le simulateur va devoir simuler les deux preuves, contrairement au prouveur honnête qui ne simule que la preuve incorrecte. Le simulateur va profiter de son contrôle de la fonction de hashage via le "random oracle model" pour faire correspondre les challenges. Le simulateur procède comme suit :

Protocole:

- Pour $i \in 0, 1$, choisir $e_i, f_i \in_{\mathbf{R}} Z_q$ et définir $a_i := \frac{g^{f_i}}{\alpha^{e_i}}$ et $b_i := \frac{pk^{f_i}}{(\beta/g^i)^{e_i}}$
- Choisir $\Delta_0, \Delta_1, k_0, k_1 \in_{\mathbf{R}} \mathbf{Z}_q$. Définir, pour $i \in 0, 1$, $A_i := \frac{a_i * g^{k_i}}{\alpha^{\Delta_i}}$ et $B_i := \frac{b_i * p^{k_i}}{(\beta/g^i)^{\Delta_i}}$ pour i=0 ,1. Poser $H(A_0, B_0, A_1, B_1) := e_0 + e_1 + \Delta_0 + \Delta_1$ et posons $e := H(A_0, B_0, A_1, B_1)$. Posons $E := e - \Delta_0 - \Delta_1$.
- $E_i = e_i + \Delta_i$ et $F_i = f_i + k_i$.
- Le simulateur simule l'interaction entre le copieur et le copié avec $((a_0, b_0, a_1, b_1), e, (e_0, f_0, e_1, f_1))$.
- L'interaction simulée entre le copieur et Hélios est $((\alpha, \beta), (A_0, B_0, A_1, B_1), (E_0, F_0, E_1, F_1))$.

Les adversaires qui ne savent pas résoudre le problème DDH ne peuvent pas différencier les engagements simulés des engagements réels. Ils ne peuvent donc pas différencier cette simulation d'un véritable protocole de masquage entre un copieur et un copié comme le simulateur suit les mêmes étapes.

Nous venons de prouver que ce protocole est zero-knowledge, qu'il génère des bulletins corrects et indistinguables tout en ne dévoilant pas le vote du copié.

Cependant ce protocole est interactif et nécessite donc pour chaque vote d'un votant souhaitant copier le vote d'un délégué d'avoir un échange interactif avec lui. Ces échanges interactifs pourraient révéler des informations sur le nombre de votants des délégués. Il nous faut alors remarquer que ces protocoles interactifs sont uniquement nécessaire pour maintenir le secret de vote des délégués. Par conséquent, étant donné que dans le cadre de la démocratie liquide le vote des délégués n'est pas privé, cet échange interactif n'est pas nécessaire. Nous allons après l'introduction de la méthode 2 de copie de vote , expliquer les phases de votes des délégués et des votants en utilisant ce protocole.

Un système de vote utilisant cette méthode de copie permet de procéder à une élection dans le cadre d'une démocratie liquide, il a l'avantage de présenter peu de différences par rapport à Helios et est donc simple à implémenter. Il nécessite

cependant de révéler le vote des délégués avant la fin de l'élection, dans le cas d'élection serrée cette information pourrait modifier le résultat de l'élection (cf.3.4). Il convient alors d'introduire une autre façon de copier les votes.

4.3.2.3 Deuxième méthode de copie de la preuve de vote

Pour éviter de faire la méthode interactive proposée précédemment, ou de perdre le secret de vote avant la fin des élections des délégués, les copieurs devront révéler l'information qu'ils copient un vote. Ils publieront la copie masquée du vote du délégué de leur choix accompagnée d'une preuve zero-knowledge qu'il s'agit bien d'une copie d'un vote d'un délégué existant sur le tableau des bulletins. Une version disjonctive du protocole de Chaum-Pedersen sera utilisée.

La première étape consiste à montrer la version interactive du protocole. Ce protocole cherche à prouver que le ballot $(A', B') = (g^m * pk^{r+r'}, g^{r+r'})$ est bien une copie masquée du ballot $(A, B) = (g^m * pk^r, g^r)$ avec $A' = A * pk^{r'}$ et $B' = B * g^{r'}$ et que le copieur connaît bien r'.

Protocole:

- P sélectionne $s \in_{\mathbf{R}} \mathbf{Z}_q$ et envoie la paire $(a, b) = (pk^s, g^s)$ (mod p) comme engagement à V.
- V sélectionne le challenge $e \in_{\mathbf{R}} \mathbf{Z}_q$ et l'envoie à P.
- P calcule $f = s + e * r'$ (mod q) et envoie la résultat f à V.

Le vérifieur accepte la preuve si $pk^f = a * (A'/A)^e$(mod p) et $g^f = b * (B'/B)^e$(mod p). On voit qu'il faut que $(B'/B) = g^{r'}$ et $(A'/A) = pk^{r'}$ pour que la preuve soit acceptée.

Voici maintenant la version disjonctive de ce protocole.

Le protocole consiste à effectuer le protocole de manière honnête pour le vote qui est réellement copié, et de simuler les preuves pour les autres votes.

Ce protocole cherche à prouver que le bulletin $(A', B') = (g^m * pk^{r+r'}, g^{r+r'})$ est bien une copie masquée d'un des ballots des délégués i $(A_i, B_i) = (g^{m_i} * pk^{r_i}, g^{r_i})$ avec $A' = A_i * pk^{r'_i}$ et $B' = B * g^{r'_i}$ et que le copieur connaît bien r' pour le ballot qu'il copie réellement.

Protocole:

- Simulation: Pour tout i $\in [1, n]$ différent de v où v est le vote copié, P choisit e_i et f_i de manière uniformément aléatoire dans $\mathbf{Z}_q$ et calcule $b_i = g^f/(\frac{B'}{B_i})^e$ et $a_i = pk^f/(\frac{A'}{A_i})^e$ (mod p)
- P sélectionne $s \in_{\mathbf{R}} \mathbf{Z}_q$ et calcule $(a_v, b_v) = (pk^s, g^s)$, il envoie $(a_1, b_1, ..., a_n, b_n)$ à V.

- V sélectionne $E \in_{\mathbf{R}} \mathbf{Z}_q$
- P calcule $e_v = E - \Sigma_{i \neq v} e_i$. Il calcule ensuite $f_v = s + e_v * r'_v$, il envoie ensuite $(e_1, f_1, ..., e_n, f_n)$ à V.

Le vérifieur accepte la preuve si pour tout i, $g^{f_i} = b_i * (\frac{B'}{B_i})^e$, $pk^{f_i} = a_i * (\frac{A'}{A_i})^e$ et $E = \Sigma_{i=1}^n e_i$

Comme les preuves sont symétriques, il est impossible de savoir quelle partie est réelle et quelle partie est simulée.

Il faut maintenant prouver que ce protocole est Zero-Knowledge mais aussi qu'il génère des bulletins corrects et indistinguables d'un autre vote copié tout en ne dévoilant pas le vote du copié.

Preuve 1:

Le protocole de masquage de preuve est complet.
Prouver que le protocole est complet est trivial.
Nous avons:

- $E = \Sigma_{i=1}^n e_i$ est valide comme le copié calcule $e_v = E - \Sigma_{i \neq v} e_i$
- La preuve simulée est valide par construction.
- Pour la preuve réelle, nous avons $a_v = pk^f / (\frac{A'}{A_v})^e = pk^{s+r'_v * e}$ qui est correcte comme $a_v = pk^s$ et $\frac{A'}{A_v} = pk^{r'}$. Nous avons également, $b_v = g^f / (\frac{B'}{B_v})^e = g^{s+r'_v * e}$ qui est aussi correcte avec $b_v = g^s$ et $\frac{B'}{B_v} = g^{r'}$.

Preuve 2:

La preuve masquée créée par ce protocole est indistinguable de toute preuve valide d'un vote copié avec cette méthode créée de manière indépendante.

Nous avons e_i et f_i sélectionnés de manière uniformément aléatoire dans $\mathbf{Z}_q$, pour toutes les preuves simulées. (a_v, b_v), $e_v = E - \Sigma_{i \neq v} e_i$ et $f_v = s + e_v * r'_v$ sont composé à partir de valeur sélectionnées de manière uniformément aléatoire. Le challenge et la réponse sont donc indépendants d'une autre copie de vote de délégué. Les valeurs de $(a_1, b_1, ..., a_n, b_n)$ simulées sont uniquement déterminée par leur définition par $(e_1, f_1, ..., e_n, f_n)$. Donc si un votant est capable de distinguer la preuve d'une autre preuve, il est capable de distinguer $(e_1, f_1, ..., e_n, f_n)$ qui est choisi de manière uniformément aléatoire.

Preuve 3:

Le protocole de masque de preuve possède la propriété de robustesse spéciale ("special soundness").

Si un prouveur peut produire une réponse à deux différents challenges E,E' pour un même engagement $(a_1, b_1, ..., a_n, b_n)$, nous allons montrer que ce prouveur peut calculer l'aléatoire utilisé lors du rechiffrement du vote. Comme $E \neq E'$, il doit exister $e_i \neq e'_i$ et $f_i \neq f'_i$ pour les deux réponses et challenges.

Posons $\frac{A'}{A_i} = \alpha_i$ Nous avons,

$$a_i = \frac{g^{f_i}}{\alpha_i^{e_i}} \; et \; a_i = \frac{g^{f'_i}}{\alpha_i^{e'_i}}, alors$$

$$\frac{g^{f_i}}{\alpha_i^{e_i}} = \frac{g^{f'_i}}{\alpha_i^{e'_i}}, g^{f_i - f'_i} = \alpha^{e_i - e'_i}$$

on peut donc calculer, pour tout les i,

$$r'_i \stackrel{?}{=} log_g \alpha_i = \frac{e_i - e'_i}{f_i - f'_i}$$

Pour $i = v$, Nous avons bien,

$$r'_v = log_g \alpha_v = \frac{e_v - e'_v}{f_v - f'_v}$$

V peut alors utiliser sa connaissance de r' pour vérifier quel vote rempli bien la relation, $(A'/A) = pk^{r'}$.

Preuve 4:

Sous l'hypothèse du "random oracle model", ce protocole de copiage masqué est zero-knowledge pour un prouveur qui suit le protocole.

Sous l'hypothèse du "random oracle model", et supposant le probleme DDH difficile à résoudre, nous allons décrire un simulateur pour ce protocole quand le prouveur est honnête. Le simulateur va devoir simuler toutes les preuves, contrairement au prouveur honnête qui ne simule que les preuves incorrectes. Le simulateur va profiter de son contrôle des challenges via le "random oracle model" pour faire correspondre les challenges. Le simulateur procède comme suit :

Protocole:

- Pour $i \in [1, n]$, choisir $e_i, f_i \in_{\mathbf{R}} \mathbf{Z}_q$ et définir $a_i = pk^{f_i}/(\frac{A'}{A_i})^e$ et $b_i = g^{f_i}/(\frac{B'}{B_i})^e$
- Poser $E = \Sigma_{i=1}^n e_i$

- Le simulateur simule l'interaction entre le copieur et le copié avec $((a_1, b_1, ..., a_n, b_n), E, (e_1, f_1, ..., e_n, f_n))$.

Les adversaires qui ne savent pas résoudre le problème DDH ne peuvent pas différencier les engagements simulés des engagements réels. Ils ne peuvent donc pas différencier cette simulation d'un véritable protocole de copie masquée de vote entre un prouveur et un vérifieur comme le simulateur suit les mêmes étapes.

Afin de rendre ce protocole non-interactif, on utilise la modification de Fiat-Shamir. Le challenge est alors remplacé par $E = H(a_1, b_1, ..., a_n, b_n)$. Le vérificateur doit donc vérifier que $E = H(a_1, b_1, ..., a_n, b_n)$ lors de sa vérification si la preuve est valide. Les preuves concernant le protocole non-interactif sont sensiblement les mêmes.

4.3.3 Phase de vote des délégués

La phase de vote commence par la phase de vote des délégués, afin de s'assurer que les votants peuvent voter pour le délégué de leur choix en copiant son bulletin sur le tableau des bulletins.

4.3.3.1 Méthode 1 de copie

Les délégués émettent leur vote en utilisant la preuve ZKP précédemment introduite (cf. 4.3.1). Ils publient le bulletin classique de la forme (P,V) auquel ils ajoutent leur ID et l'aléatoire r utilisé pour coder le chiffré. Ils brisent alors le secret de leur vote. Lorsqu'ils veulent copier le vote d'un autre délégué, ils peuvent prendre le vote du délégué et le déchiffrer. Après cela, ils envoient un bulletin de vote avec la voix déchiffrée, ils rajoutent leur aléatoire pour que le bulletin puisse être recopié également. Il est ainsi possible de savoir ce qu'ils ont voté.

4.3.3.2 Méthode 2 de copie

Afin de s'assurer que les preuves des votes copiés par les délégués soient correctes, il faut diviser la phase de vote des délégués en deux parties. Pour commencer, les délégués envoient uniquement leurs votes sans preuves sur le tableau des bulletins, ils vont donc générer un ciphertexte directement ou copier et masquer un ciphertexte d'un autre délégué. Ensuite, les ciphertextes sont tous enregistrés et les délégués doivent alors renvoyer les mêmes ciphertextes accompagnés de leurs preuves. Dans le cas d'un vote direct, ils envoient la ZKP classique introduite dans la section ZKP prouvant la validité du bulletin (cf. 4.3.1). Si ils copient le vote d'un autre délégué, ils devront utiliser la ZKP prouvant qu'ils ont bien copié un des ciphertextes provenant d'un des délégués (cf. 4.3.2.3). Il est alors possible de savoir si un

délégué a voté directement ou s'il a copié le vote d'un autre délégué. Lorsque les délégués fournissent la preuve liée à leur ciphertexte dans un nouveau bulletins de vote complet, on doit vérifier si le ciphertexte du nouveau bulletin est le même que celui du bulletin envoyé précédemment afin de s'assurer que les délégués copiant les votes puissent créer leurs preuves sans problèmes.

4.3.4 Phase de vote des votants

Lorsque la phase de votes des délégués est terminée, les votants peuvent alors émettre leurs bulletins de vote. Ils peuvent voter directement ou en reprenant le vote d'un délégué. Lorsqu'ils votent directement, les votants vont émettre un nouveau bulletin avec un ciphertexte chiffrant leur vote et la preuve ZKP prouvant la légitimité de celui-ci.

4.3.4.1 Méthode 1 de copie

Dans le cas de la méthode de copie introduite en premier et lorsqu'ils reprennent le vote d'un délégué, ils copient et masquent le vote du délégué et recréent une preuve grâce à la connaissance de r. Cette preuve est une ZKP que le vote est 0 ou 1 (cf. 4.3.1). Il peut être utile de rappeler que les votants ne partagent pas leur aléatoire afin de garder le secret de leur vote, il est donc impossible de copier la preuve associé à leur vote sans leur collaboration active.

4.3.4.2 Méthode 2 de copie

Dans le second cas, lorsqu'ils reprennent le vote d'un délégué, ils copient et masquent le vote du délégué. Ils fournissent ensuite la preuve ZKP qu'ils ont bien effecutée une copie masquée d'un des ciphertextes chiffrant le vote d'un des délégués (cf 4.3.2.3).

Nous venons de voir les nombreuses mais nécessaires tâches composant la phase d'élection. Il convient donc de clôturer ce protocole par la dernière des trois phases d'une élection : l'après-élection.

4.4 L'après élection

Les curateurs vont alors se rassembler pour le déchiffrement des bulletins de vote. Tous les curateurs doivent être présent lors du déchiffrement étant donné que toutes les clés partielles sont nécessaires. Afin de procéder au décompte, la propriété d'addition homomorphe d'ElGamal exponentiel est utilisée. Dans le cas d'une question dichotomique, les chiffrés des votes sont multipliés et le résultat est alors

la somme de tous les votes. Ce sera le seul ciphertexte déchiffré, permettant de maintenir le secret des votes individuels.

Pour le ciphertexte (c_1, c_2), avec c_1 le message masqué et c_2 l'aléatoire masqué, chaque curateur C_i calcule et publie sa part du déchiffrement $d_i = c_2^{x_i}$. Le résultat est alors $\frac{c_1}{\Pi d_i}$ Cette procédure utilisée par Helios est une version simplifiée de celle de Pedersen. Elle ne permet pas qu'il manque un curateur lors du dépouillement. En outre, elle présente l'avantage d'être plus simple à utiliser étant donné que les curateurs n'ont pas besoin de moyens de communication privés entre eux [25]. La variante exponentielle d'ElGamal est utilisée, le résultat du déchiffrement est sous la forme g^m et il faut donc calculer le log pour avoir le résultat final. Comme m est le nombre de voix, il est borné entre 0 et le nombre de votant, le résultat du log est donc calculable en quelques secondes sur un ordinateur portable classique [25].

Un curateur malicieux pourrait manipuler le résultat de l'élection en ne publiant pas une part de déchiffrement d correcte. Par exemple, si il publie $d'_i = \frac{d_i}{g^v}$, le résultat final devient $\frac{c_1}{\Pi d_i / g^v}$, ce qui réduit le nombre de voix de v. Il faut donc s'assurer que les curateurs envoient le déchiffrement partiel correspondant au ciphertexte. Pour ce faire nous utiliserons, les Zero-Knowledge-Proof introduites dans la partie ci-dessous.

4.4.1 ZKP utilisée lors du décompte

Un protocole de Chaum-Pedersen est utilisé dans cette situation. Voici le protocole sous sa forme interactive.

Ce protocole cherche à prouver que d est le bon déchiffrement partiel de (c_1,c_2), c_1 étant le message masqué et c_2 l'aléatoire masqué, utilisant la bonne clé privée sk liée à la clé publique pk de la paire ElGamal (sk,pk).

Protocole:

- P sélectionne aléatoirement $s \in \mathbf{Z}_q$ et envoie la paire $(a_1, a_2) = (g^s, c_2^s)$ (mod p) comme engagement à V.
- V sélectionne le challenge e entre 0 et $n - 1$ avec $n \leq q$ et l'envoie à P.
- P calcule $f = s + e * sk$ (mod q) et envoie le résultat f à V.

Le vérifieur accepte la preuve si $g^f = a_1 * pk^e$ (mod p)et $c_2^f = a_2 * d^e$ (mod p)

La solidité et la propriété ZKP de cette preuve sont justifiées selon les mêmes arguments que pour la justification lors de la ZKP concernant la génération de clé.

Tout observateur peut alors vérifier les preuves de chaque observateur ainsi que la justesse du produit des parts de déchiffrement afin de s'assurer que le bon résultat de l'élection est publié.

Lors du décompte, ce protocole est appliqué pour chaque question et le résultat final est ensuite publié par l'administrateur de l'élection. Dans le cas de la méthode 2 de copie, le vote des délégués reste secret tout au long de l'élection. Si le vote des délégués doivent être public à la fin de l'élection, les curateurs se rassemblent à nouveau pour déchiffrer les votes individuels des délégués.

4.5 "End-to-end" vérifiabilité

Les systèmes de vote informatiques étant soumis aux bugs et autres corruptions, les utilisateurs ne peuvent leur donner une confiance aveugle. Cela a amené les chercheurs à créer des systèmes de vote vérifiable de bout à bout, les systèmes de vote "End-to-end"[5]. Les système de vote "End-to-end" sont définis par trois propriétés :

- *cast as intended:* le vote sur le bulletin que le votant créée représente effectivement ce qu'il a voté.
- *recorded as cast:* le vote que le votant émet sur le serveur est sous la même forme que lorsqu'il a été créé.
- *tallied as recorded:* le votant peut vérifier que tous les votes valides, dont le sien, sont bien inclus dans le décompte.

4.5.1 Cast as intended

Dans l'hypothèse d'une machine corrompue, le votant pourrait se retrouver dans la situation où son vote n'est pas encodé correctement dans le bulletin de vote. Pour pallier à ce problème, avant la transmission du vote, le votant peut décider de vérifier son bulletin de vote. On appelle cela le Benaloh challenge. Comme cette vérification arrive après la complétion du bulletin de vote mais juste avant son émission, l'attaquant, en l'occurrence la machine corrompue, ne peut pas savoir si le bulletin va être réellement émis au moment du challenge. Dans le cas d'un challenge, le bulletin de vote ne peut pas être réutilisé car l'aléatoire utilisé pour encoder le code est divulgué par la machine pour vérifier si le vote encodé sur le bulletin correspond bien à l'intention de vote du votant. Il faut alors ré-encoder un nouveau ballot. Si le Benaloh challenge est demandé par une population assez large de votants, on peut s'attendre à ce que les tentatives de modification de résultat de l'élection par machine de vote corrompue soient détectées[25].

4.5.2 Recorded as cast

Maintenant que le votant peut se convaincre de la justesse du bulletin de vote généré, il peut vérifier qu'il correspond bien au bulletin de vote reprenant son ID sur le tableau des bulletins. L'accès au tableau des bulletins ne demandant aucune identification, il n'est donc pas possible pour un attaquant d'adapter le tableau par rapport au votant qui demande vérification.

4.5.3 Tallied as recorded

Les votants peuvent vérifier que tous les bulletins sont bien émis par des votants présents dans la liste des votants admis. Comme soulevé précédemment, tout le monde peut vérifier la validité des preuves de tous les bulletins sur le tableau et ensuite vérifier la véracité du décompte en utilisant la preuve du décompte présentée auparavant.

4.5.4 Schéma de l'utilisation des principes mathématiques au sein du modèle d'organisation de l'élection

Pour plus de lisibilité, nous allons maintenant introduire un schéma reprenant les principaux concepts mathématiques explicités dans la partie protocole afin d'avoir une vue globale de leur interaction avec l'organisation de l'élection, et donc de mieux comprendre le moment où leur utilisation est nécessaire. Cela permet également d'introduire un résumé de la partie protocole que nous venons de détailler.

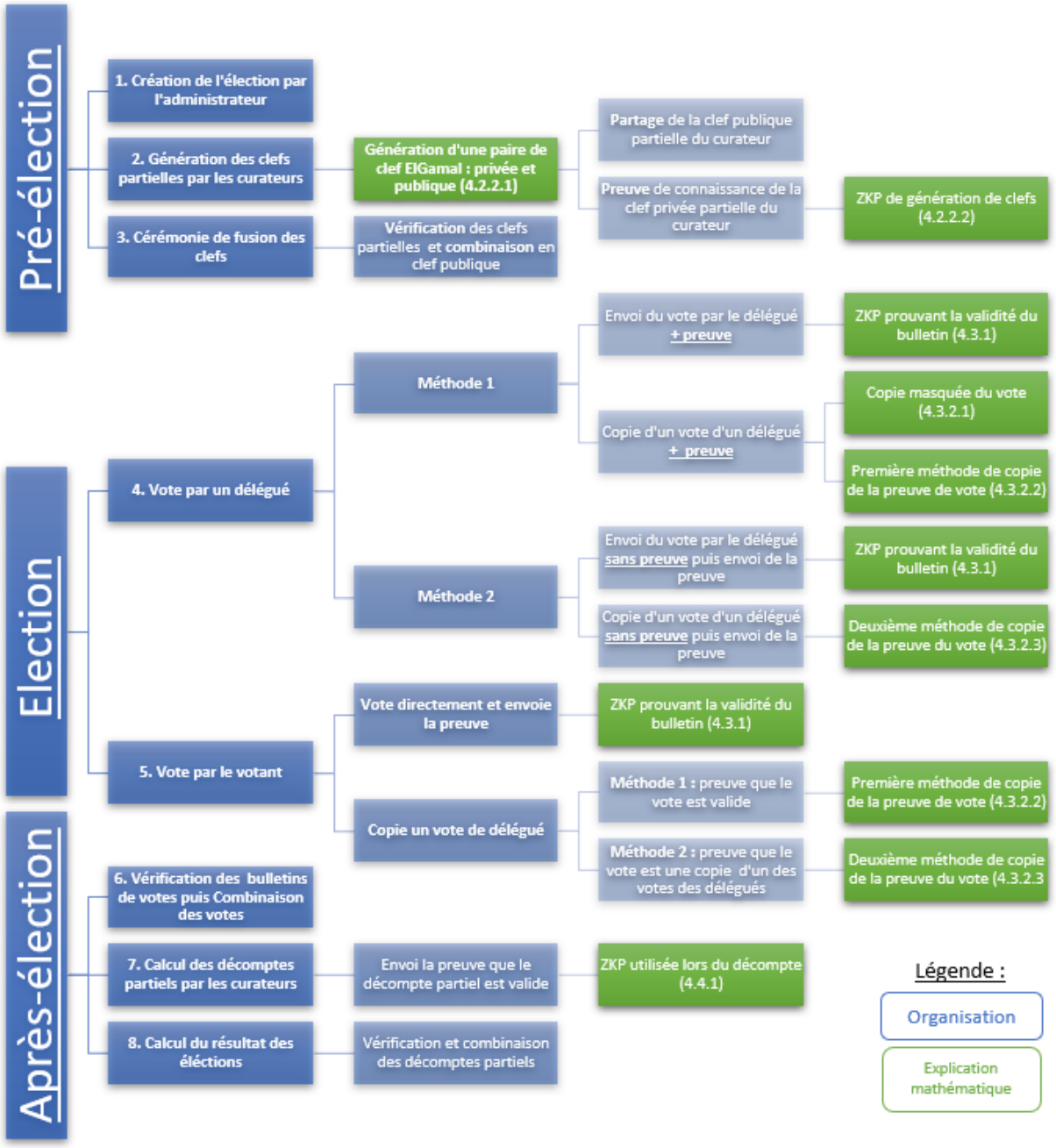

Figure 4.1: Schéma de l'organisation de l'élection avec l'emplacement des preuves mathématiques

Chapitre 5

Détail technique du code

Cette section va introduire et expliquer de façon rudimentaire les fonctions et structures utilisées afin de simuler une élection utilisant les outils introduits dans les sections précédentes.

5.1 Sources et bibliothèques utilisées

Les bibliothèques ElGamal , canonicaljson et numbers créées et modifiées pour le cours "LELEC2770 : Privacy Enhancing Technologies" par Olivier Pereira, Henri Devillez et Gaetan Cassiers sont utilisées dans le code de ce mémoire. Leur code servant à simuler un élection a également inspiré le code.

Les bibliothèque secrets et json de python sont utilisées dans le code. La bibliothèque numpy est également utilisée.

Pour générer des données nécessaires à l'analyse du code, les bibliothèques random et timeit sont utilisées.

5.2 Simulation d'élection

Dans cette première partie, nous parlerons des structures et fonctions qui sont communes aux deux méthodes de copiage. Nous introduirons les fonctions dans le cas d'une élection simple, c'est à dire une unique réponse booléenne à la question.

5.2.1 Creation de l'élection par l'administrateur

La fonction sert à créer le tableau des bulletins sous forme de Json. Dans le cadre d'une simulation, on simule également la génération clefs des curateurs et on les ajoutes avec leurs preuves sur le tableau des bulletins. On ajoute également sur

le tableau des bulletins, la clef publique de l'élection en combinant les clefs partielles.

Algorithm 1: $generate\_election(n\_curateurs, n\_vot, list\_del)$

Entrée:le nombre de curateurs, le nombre de votants, une liste de l'id des délégués
Sortie:le json du tableau des bulletins, les clefs privées des curateurs
$G \leftarrow GenereGroupElgamal()$
$bb \leftarrow Genere\_BB\_vide()$
$FOR\ EACH\ curateur$
    $clef\_curateurs" \leftarrow create\_clef\_curateur\_with\_proof(bb)$
pk$\leftarrow combine_clef_curateur(bb["group], bb["clef_curateurs"]).y$

5.2.2 Génération des clefs partielles par les curateurs

La fonction crée une paire ElGamal et la preuve zero-knowledge de la connaissance de la clef secrète. Elle poste la clef publique et la preuve sur le tableau des bulletins.

Algorithm 2: $create\_clef\_curateur\_with\_proof(bb)$

Entrée:le tableau des bulletins
Sortie:la clef secrète de la paire créée
$sk, pk \leftarrow GenerePaireElGamal()$
$engagement \leftarrow GenerePaireElegamal()$
$challenge \leftarrow Hash(engagement.pk)$
$réponse \leftarrow engagement.sk + challenge * sk\%p$

5.2.3 Fusion des clefs curateurs

La fonction vérifie la validité des preuves liée au clef partielles des curateurs ensuite elles les combines et ajoute la clef publique de l'élection sur le tableau des bulletins.

Algorithm 3: $combine\_clef\_curateur(G, Clefs)$

Entrée: le dictionnaire du groupe utilisé pour le chiffrement de l'élection, un dictionnaire des clefs des curateurs et leurs preuves
Sortie:La clef publique de l'élection sous forme d'objet
$FOR\ Each\ Clef$
$IF\ verify\_proof\_clef\_curateur() :$
    $pk* = pk\%p$

5.2.4 Vérification d'une clef curateur

La fonction vérifie la preuve de la clef partielle, rend True si le preuve est correcte.

Algorithm 4: $verify\_proof\_clef\_curateur(G, Clef)$

Entrée: le dictionnaire du groupe utilisé pour le chiffrement de l'élection, un dictionnaire de la clefs d'un curateur et sa preuve
Sortie:un booléen
$RETURN\ pow(g, réponse, p) == engagement * pow(pk, challenge, p)\%p$

5.2.5 Combinaison des bulletins de vote

La fonction vérifie la validité des preuves des bulletins et combine les votes.

Algorithm 5: $combine\_vote(bb)$

Entrée: le dictionnaire du tableau des bulletins
Sortie: la combinaison des votes sous forme d'objet ElGamalCiphertext
$FOR\ Each\ BulletinDeVote$
IF verify_proof_vote():
$DécompteChiffré \leftarrow AjoutHomomorphique(ChiffréDuBulletin)$

5.2.6 Calcul des décomptes partiels par les curateurs

La fonction créée la part de déchiffrement et sa preuve d'un curateur du résultat chiffré de l'élection. Elle poste cette part de déchiffrement et la preuve de sa validité sur le tableau des bulletins.

Algorithm 6: $cast\_part\_dec\_with\_proof(bb, ct\_res, sk\_curateur)$

Entrée: le dictionnaire du tableau des bulletins, le ciphertexte du résultat de l'élection sous forme d'objet ElGamalCiphertext, la clef secrète du curateur sous forme d'objet ELGamalSecretKey
$DéchiffrementPartiel \leftarrow pow(c1, ClefSecreteCurateur, p)\%p$
$s \leftarrow Aléatoire(0, q-1)$
$engagement \leftarrow [pow(g, s, p), pow(c1, ClefSecreteCurateur, p)]$
$challenge \leftarrow hash(engagement)$
$réponse \leftarrow s + challenge * ClefSecreteCurateur\%p$

5.2.7 Vérification des décomptes partiels

La fonction vérifie la preuve du décompte parteil et rend True sur la preuve est valide.

Algorithm 7: $verify\_proof\_dec(part\_dec, G)$

Entrée: le dictionnaire d'un déchiffrement partiel et de sa preuve, la dictionnaire du groupe utilisé dans l'élection
Sortie: le booléen de la validité de la preuve
$challenge = hash(engagement)$
RETURN
$pow(g, réponse, p)\ EST\ EGUAL\ engagement[0] * pow(pk, challenge, p)\%p$
AND $pow(c1, réponse, p)\ EST\ EGUAL$
engagement[1]*pow(PartDeDéchiffrement,challenge,p)% p

5.2.8 Combinaison des décomptes partiels

La fonction vérifie les preuves des parts de déchiffrement pour ensuite les combiner. Elle renvoie le résultat de l'élection.

Algorithm 8: $combine\_decryption\_factor(bb)$

Entrée: le dictionnaire du tableau des bulletins
Sortie: g exposant le résultat de l'élection
$RésultatChiffré \leftarrow combine\_vote(bb)$
$FOR\ EACH\ PartDeDéchiffrement$
$If\ verify\_proof\_dec(PartDeDéchiffrement, group)$
  $R* = PartDeDéchiffrement\%p$
$Résultat = RésultatChiffré.c2/R\%p$

5.3 Les parties spécifiques à la méthode 1 de copiage de vote

5.3.1 Le Json type du tableau des bulletins

Le Json du tableau des bulletins ressemble à ceci lorsqu'on utilise la première méthode de copie de vote.

```
<group>::= {"p": <p>,"g" : <g>}
<pk>::=<pk>

<bulletin_votant>::={
        "ct" : {
                "c1" : <c1>,
                "c2" : <c2>;
        }
        "zkpproof" : {
                "commit" : [<a0>,<b0>,<a1>,<b1>],
                "challenge" : [<e0>,<e1>],
                "response" : [<f0>,<f1>],
        },
        "aléatoire":<aléatoire>,
}
<clef_curateur>::={
        "pk": <pk>,
        "zkpproof"={
                "commit" : <a>,
                "response" : <f>,
        },
}
<part_dec>::={
        "pk" : <pk>,
        "c1" : <c1>,
        "part_dec" : <d>,
        "zkpproof" : {
                "commit" : [<a>,<b>],
                "response" : <f>,
        },
}
<id>::= <id>
<tableau_des_bulletins>::={
        "group" : <group>,
        "clefs_curateurs" : [<clef_curateur_0>,<clef_curateur_1>,...],
        "parts_dec" : [<part_dec_0>,<part_dec_1>,...],
        "bulletins_vot" : [<bulletin_0>,<bulletin_1>,...],
        "pk":<pk>,
        "ids_vot": [<id_0>,<id_1>,...]
        "ids_del" : [<id_0>,<id_1>,...]
}
```

Figure 5.1: Représentation du tableau des bulletins dans le code de la méthode 1

5.3.2 Envoi du bulletin dans le cas d'un vote direct

La fonction permet de créer un bulletin valide composé d'un vote chiffré par ElGamal exponentiel et de la ZKP de sa validité. Le bulletin de vote est ensuite posté sur le tableau des bulletins.

Algorithm 9: $cast\_vote\_with\_proof(bb, vote, id\_vot, delegue)$

Entrée: le dictionnaire du tableau des bulletins,le vote (un nombre en 0 et 1),l'identifiant du votant(sa position dans la liste des votants),un boolean spécifiant si le vote provient d'un délégué ou non

$c1, c2 \leftarrow ElGamalEncrypt(vote) with\ the\ randomness\ r$

$ChallengeSimulé \leftarrow pow(g, random(0, q-1), p)$

$RésultatSimulé \leftarrow pow(g, random(0, q-1), p)$

$EngamentSimulé[0] \leftarrow$
$pow(g, ResultatSimulé, p)/(pow(c1, ChallengeSimulé, p)\%p$

$EngamentSimulé[1] \leftarrow$
$pow(pk, ResultatSimulé, p)/(pow(\frac{c2}{pow(g,1-vote,p)}\%p, ChallengeSimulé, p)\%p$

$z = random(0, q-1)$

$EngagementRéel[0] \leftarrow pow(g, z, p)$

$EngagementRéel[1] \leftarrow pow(pk, z, p)$

$Challenge \leftarrow hash(commit)$

$ChallengeRéel \leftarrow Challenge - ChallegeSimulé$

$RésultatRéel \leftarrow r * ChallengeRéel + z\%q$

$IF\ delegue\ IS\ TRUE \quad RETURN\ r$

5.3.3 Envoi du bulletin dans le cas d'une copie

La fonction permet de créer un bulletin valide composé de la copie masquée d'un vote chiffré par ElGamal exponentiel et de la ZKP de sa validité. Le bulletin de vote est ensuite posté sur le tableau des bulletins. Seul les votes des délégués peuvent être recopié.

Algorithm 10: $copy\_vote\_with\_proof(bb, id\_copie, id\_vote, delegue)$

Entrée: le dictionnaire du tableau des bulletins,l'id du vote à copier,l'identifiant du votant(sa position dans la liste des votants),un boolean spécifiant si le vote provient d'un délégué ou non

$IF\ id\_copie\ IN\ IdsDélégués$
$vote \leftarrow ElgamalDéchiffreEnUtilisantR(VoteCopié)$
$cast\_vote\_with\_proof(bb, vote, id\_vote, delegue)$

5.3.4 Vérification de la preuve du bulletin

La fonction vérifie si la preuve liée au vote est bien valide.

Algorithm 11: $verify\_proof\_vote(G, pk, ballot)$

Entrée: le dictionnaire du groupe utilisé lors de l'élection, le dictionnaire de la clef publique de l'élection,un dictionnaire du bulletin de vote, contenant le bulletin de vote et sa preu
Un booléen concernant la validité de la preuve ZKP
$SommeChallenge \leftarrow hash(Engagement)$
$IF\ \Sigma Challenge\ IS\ NOT\ EQUAL\ SommeChallenge \rightarrow$
$RETURN\ FALSE$
$FOR\ 0\ AND\ 1$
$IF\ pow(g, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[0] * pow(c1, Challenge, p)\%p \rightarrow RETURN\ FALSE$
$IF\ pow(pk, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[1] * pow(\frac{c2}{pow(g,vote,p)}, Challenge, p)\%p \rightarrow RETURN\ FALSE$
$RETURN\ True$

5.4 Les parties spécifiques à la méthode 2 de copiage de vote

5.4.1 Le Json type du tableau des bulletins

Le Json du tableau des bulletins en utilisant la deuxième méthode de copie ressemble à ceci.

```
<group>::= {"p": <p>,"g" : <g>}
<pk>::=<pk>

<bulletin_votant>::={
        "ct" : {
                "c1" : <c1>,
                "c2" : <c2>;
        }
        "zkpproof" : <zkpproof_direct> ou <zkpproof_inderect>,
        "direct":<direct>
}
<zkpproof_direct>::= {
                "commit" : [<a_0>,<b_0>,<a_1>,<b_1>],
                "challenge" : [<e0>,<e1>],
                "response" : [<f0>,<f1>],
        },
<zkpproof_copie>::={
                "commit" : [[<a_0>,<b_0>],[<a_1>,<b_1>],...],
                "challenge" : [<e0>,<e1>,...],
                "response" : [<f0>,<f1>,...],
}

<clef_curateur>::={
        "pk": <pk>,
        "zkpproof"={
                "commit" : <a>,
                "response" : <f>,
        },
}
<part_dec>::={
        "pk" : <pk>,
        "c1" : <c1>,
        "part_dec" : <d>,
        "zkpproof" : {
                "commit" : [<a>,<b>],
                "response" : <f>,
        },
}
<id>::= <id>
<tableau_des_bulletins>::={
        "group" : <group>,
        "clefs_curateurs" : [<clef_curateur_0>,<clef_curateur_1>,...],
        "parts_dec" : [<part_dec_0>,<part_dec_1>,...],
        "bulletins_vot" : [<bulletin_0>,<bulletin_1>,...],
        "pk":<pk>,
        "ids_vot": [<id_0>,<id_1>,...]
        "ids_del" : [<id_0>,<id_1>,...]
}
```


Figure 5.2: Représentation du tableau des bulletins dans le code de la méthode 2

5.4.2 Envoi du bulletin sans preuve lors d'un vote direct pour les délégués

La fonction permet de créer un bulletin non valide incomplet composé d'un vote chiffré par ElGamal exponentiel. Le bulletin de vote est ensuite posté sur le tableau des bulletins.

Algorithm 12: $cast\_vote\_without\_proof(bb, vote, id\_vot)$

Entrée: le dictionnaire du tableau des bulletins,le vote (un nombre en 0 et 1),l'identifiant du votant(sa position dans la liste des votants)
Sortie: l'aléatoire utilisé lors du chiffrement
$c1, c2 \leftarrow ElGamalEncrypt(vote) with\ the\ randomness\ r$

5.4.3 Renvoi du bulletin avec preuve lors d'un vote direct pour les délégués

La fonction reprend le vote chiffré précédemment avec la méthode $cast\_vote\_without\_proof()$ et rajoute la preuve que le vote est valide dans le bulletin. Le bulletin est ensuite ajouté dans le tableau des bulletins.

Algorithm 13: $recast\_vote\_with\_proof(bb, vote, id\_vote, r)$

Entrée: le dictionnaire du tableau des bulletins,le vote (un nombre en 0 et 1),l'identifiant du votant(sa position dans la liste des votants,l'aléatoire utilisé lors du chiffrement du vote)
$c1, c2 \leftarrow RécupèreCtDepuisLeTDB(id\_vote)$
$ChallengeSimulé \leftarrow pow(g, random(0, q-1), p)$
$RésultatSimulé \leftarrow pow(g, random(0, q-1), p)$
$EngamentSimulé[0] \leftarrow$
$pow(g, ResultatSimulé, p)/(pow(c1, ChallengeSimulé, p)\%p$
$EngamentSimulé[1] \leftarrow$
$pow(pk, ResultatSimulé, p)/(pow(\frac{c2}{pow(g,1-vote,p)}\%p, ChallengeSimulé, p)\%p$
$z = random(0, q-1)$
$EngagementRéel[0] \leftarrow pow(g, z, p)$
$EngagementRéel[1] \leftarrow pow(pk, z, p)$
$Challenge \leftarrow hash(commit)$
$ChallengeRéel \leftarrow Challenge - ChallegeSimulé$
$RésultatRéel \leftarrow r * ChallengeRéel + z\%q$

5.4.4 Envoi du bulletin sans preuve dans le cas d'une copie pour les délégués

La fonction permet de créer un bulletin non valide incomplet composé de la copie masquée d'un vote chiffré par ElGamal exponentiel. Le bulletin de vote est ensuite posté sur le tableau des bulletins.

Algorithm 14: $copy\_vote\_without\_proof(bb, id\_copie, id\_vote)$

Entrée: le dictionnaire du tableau des bulletins,l'id du vote à copier,l'identifiant du votant(sa position dans la liste des votants),un boolean spécifiant si le vote provient d'un délégué ou non
Sortie: l'aléatoire utilisé lors du chiffrement
$IF\ id\_copie\ IN\ IdsDélégués$
$z = aléatoire(0, q-1)$
$c1, c2 \leftarrow RécupèreCtDepuisLeTDB(id\_copie)$
$copieC1 \leftarrow c1 * pow(g, z, p)\%p$
$copieC2 \leftarrow c2 * pow(pk, z, p)\%p$

5.4.5 Renvoi du bulletin avec preuve dans le cas d'une copie pour les délégués

La fonction reprend le vote chiffré précédemment avec la méthode $copy\_vote\_without\_proof()$ et rajoute la preuve que le vote est une copie d'un des votes des délégués. Le bulletin est ensuite ajouté dans le tableau des bulletins.

Algorithm 15: $recopy\_vote\_with\_proof(bb, id\_copie, id\_vote, z)$

Entrée: le dictionnaire du tableau des bulletins,l'id du vote à copier,l'identifiant du votant(sa position dans la liste des votants),l'aléatoire utilisé lors de la copie masquée du vote

$IF\ id\_copie\ IN\ IdsDélégués$

$Aprime, Bprime \leftarrow RécupèreCtDepuisLeTDB(id\_vote)$

$FOR\ EACH$ délégué $EXCEPT\ id\_copie$

$ChallengeSimulé \leftarrow pow(g, aléatoire(0, q-1), p)$

$RésultatSimulé \leftarrow pow(g, aléatoire(0, q-1), p)$

$EngagementSimulé[0] \leftarrow pow(g, RésultatSimulé, p)/pow(\frac{Aprime}{c1Délégué}, ChallengeSimulé, p)\%p$

$EngagementSimulé[1] \leftarrow pow(pk, RésultatSimulé, p)/pow(\frac{Bprime}{c2Délégué}, ChallengeSimulé, p)\%p$

$s \leftarrow aléatoire(0, q-1)$

$EngamentRéel[0] \leftarrow pow(g, s, p)$

$EngamentRéel[1] \leftarrow pow(pk, s, p)$

$Challenge \leftarrow hash(Engagement)$

$ChallengeRéel \leftarrow Challenge - \Sigma\ ChallengeSimulé$

$RésultatRéel \leftarrow z * ChallengeRéel + s\%q$

5.4.6 Envoi du bulletin avec preuve dans le cas d'un vote direct pour les votants

La fonction crée un nouveau bulletin comprenant le vote du votant avec la preuve de sa validité. Elle envoie alors ce bulletin sur le tableau des bulletins.

Algorithm 16: $cast\_vote\_with\_proof(bb, vote, id\_vote)$

Entrée: le dictionnaire du tableau des bulletins,le vote (un nombre en 0 et 1),l'identifiant du votant(sa position dans la liste des votants

$z = cast\_vote\_without\_proof(bb, vote, id\_vote)$

$recast\_vote\_with\_proof(bb, vote, id\_vote, z)$

5.4.7 Envoi du bulletin avec preuve dans le cas d'une copie pour les votants

La fonction crée un nouveau bulletin comprenant la copie du vote choisi avec la preuve que cette copie est bien une copie du vote d'un des délégués. Elle envoie alors ce bulletin sur le tableau des bulletins.

Algorithm 17: $copy\_vote\_with\_proof(bb, id_copie, id\_vote)$

Entrée: le dictionnaire du tableau des bulletins,l'id du vote à copier,l'identifiant du votant(sa position dans la liste des votants),l'aléatoire utilisé lors de la copie masquée du vote

$z = copy\_vote\_without\_proof(bb, id_copie, id\_vote)$

$recast\_vote\_with\_proof(bb, id_copie, id\_vote, z)$

5.4.8 Vérification de la preuve du bulletin

La fonction vérifie si la preuve liée au vote est bien valide.

Algorithm 18: $verify\_proof\_vote(G, pk, ballot, bb)$

Entrée: le dictionnaire du groupe utilisé lors de l'élection, le dictionnaire de la clef publique de l'élection,un dictionnaire du bulletin de vote, contenant le bulletin de vote et sa preu,la dictionnaire du tableau des bulletins

Un booléen concernant la validité de la preuve ZKP

$IF\ Direct\ IS\ EQUAL\ TRUE$

$SommeChallenge \leftarrow hash(Engagement)$

$IF\ \Sigma Challenge\ IS\ NOT\ EQUAL\ SommeChallenge \rightarrow$

$RETURN\ FALSE$

$FOR\ 0\ AND\ 1\ AS\ vote$

$IF\ pow(g, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[0] * pow(c1, Challenge, p)\%p \rightarrow RETURN\ FALSE$

$IF\ pow(pk, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[1] * pow(\frac{c2}{pow(g,vote,p)}, Challenge, p)\%p \rightarrow RETURN\ FALSE$

$RETURN\ True$

$ELSE$

$Aprime, Bprime \leftarrow RécupèreCtDepuisLeTDB(id\_vote)$

$SommeChallenge \leftarrow hash(Engagement)$

$IF\ \Sigma Challenge\ IS\ NOT\ EQUAL\ SommeChallenge \rightarrow$

$RETURN\ FALSE$

$FOR\ EACH\ Délégué$

$c1Délégué, c2Délégué \leftarrow RécupèreCtDepuisLeTDB(id\_délégué)$

$IF\ pow(g, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[0] * pow(\frac{Aprime}{c1Délégué}, Challenge, p)\%p \rightarrow RETURN\ FALSE$

$IF\ pow(pk, Résultat, p)\ IS\ NOT\ EQUAL\ Engagement[1] * pow(\frac{Bprime}{c2Délégué}, Challenge, p)\%p \rightarrow RETURN\ FALSE\ Return\ TRUE$

5.5 Résumé

Dans ce chapitre, nous avons présenté les fonctions intéressantes et nécessaires à la création des deux systèmes de vote présentés dans la cas d'un élection simple où la réponse à la question posée est un simple booléen. Toujours dans ce cadre d'élection simple, nous avons présenté les fonctions permettant des créer et vérifier des bulletins de vote valides lors d'un vote direct ou d'une copie dans le cas de la méthode 1 de copie et de la cas de la méthode 2 de copie. Afin de permettre la mise en place d'élection plus complexe où de multiples réponses sont possibles, nous avons adapté ces fonctions pour la partie analyse de performance de code qui suit. L'implémentation de cette augmentation est assez directe, il s'agit de transformer le tableau des bulletins et de modifier les fonctions ci-dessus afin que la tableau des bulletin puisse recevoir des matrices de bulletins de votes et de déchiffrement partiels où la ligne correspond au numéro de la réponse et la colonne à l'identifiant du votant ou à l'identifiant du curateur respectivement.

L'entièreté du code définit dans cette partie se trouve dans le répertoire GitHub disponible à l'adresse suivante https://github.com/cherpiona/memoire

Dans le chapitre suivant, nous allons mettre à l'épreuve le code des systèmes de votes des deux méthodes de copie.

Avant cela, et afin d'induire une meilleure compréhension et une vision globale de ce travail dans son entièreté, nous allons introduire un schéma mettant en lien à la fois les étapes de l'organisation de l'élection, les explications mathématiques qui lui sont nécessaire, et enfin, les algorithmes informatiques permettant de simuler notre élection.

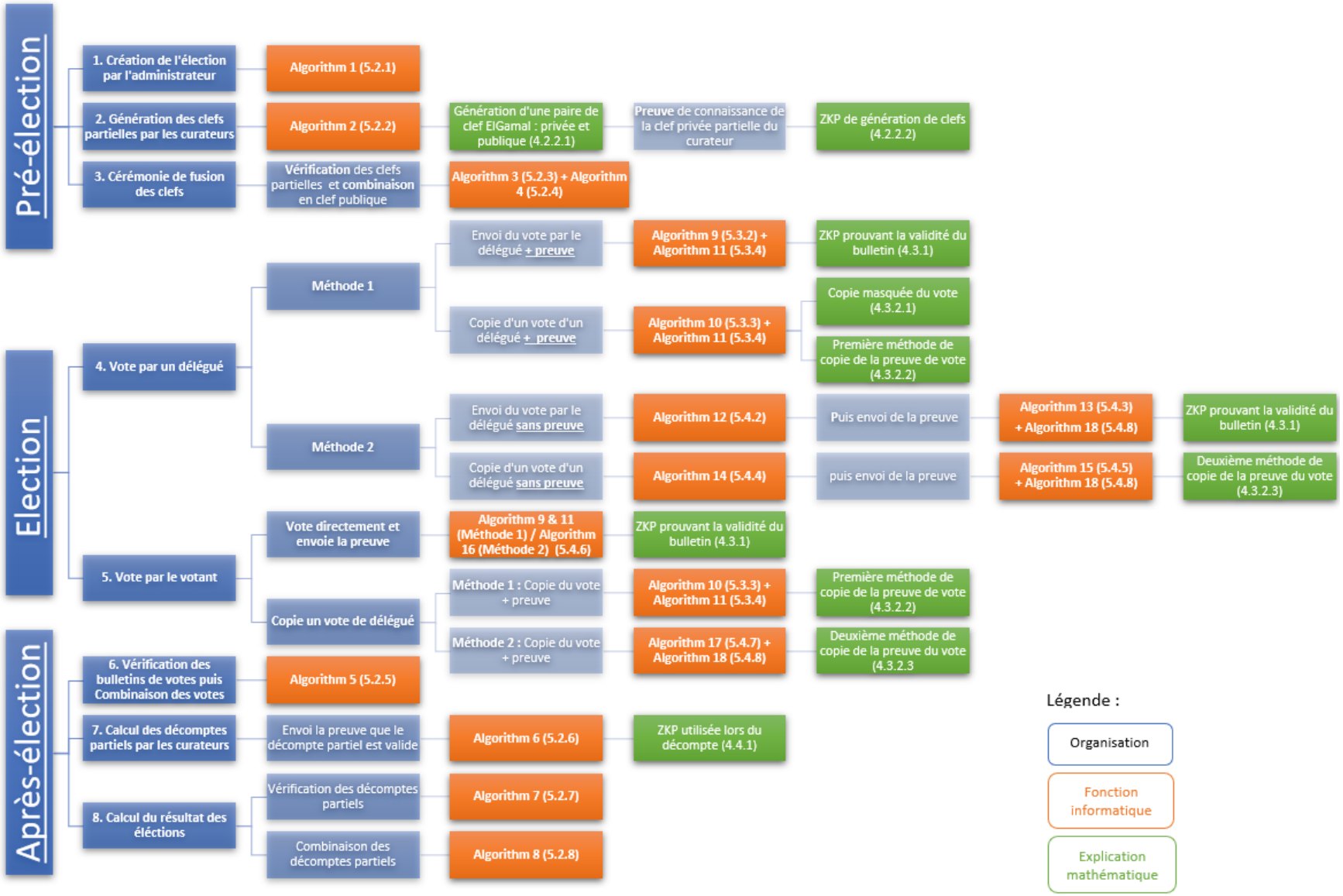

Figure 5.3: Schéma de l'organisation de l'élection avec l'emplacement des preuves mathématiques et les fonctions informatiques utilisées

Chapitre 6

Analyse performance

Cette section va tenter de vérifier si ces systèmes des votes et plus précisément ses méthodes de copie sont applicables de le cadre d'une élection réelle. Afin de les tester, nous effectuerons des simulations d'élection. Toutes les étapes nécessaires au bon fonctionnement vont être simulées:

- La génération du tableau des bulletins ainsi que la cérémonie de création de clef sont simulées.
- Les votes des délégués et votants sont simulés et aléatoires, ainsi que si le votant vote directement ou en déléguant sa voix.
- Les curateurs et leurs déchiffrements des parts de décompte sont également simulés.

Afin d'analyser nos résultats, nous allons donc procéder à différentes simulations en contrôlant différentes variables de tests. Pour ce faire, les différentes variables de test envisagées dans ce mémoire sont : le nombre de votants classiques, les votants qui ne sont pas des délégués, le nombre de délégués, la proportion de votants directs, le nombre de réponses à la question posée lors de l'élection et le nombre de curateurs. Nous allons tester sur les deux méthodes les sujets ci-dessous:

- l'impact du nombre de délégués et du nombre de votants sur le temps de calcul des simulations d'élections.
- l'impact du nombre de délégués et de la proportion de votant direct sur le temps de calcul des simulations d'élections.
- l'impact du nombre de réponses sur le temps de calcul des simulations d'élections.

- l'impact du nombre de curateurs sur le temps de calcul des simulations d'élections.

Les données de temps de calcul ont été récoltées sur un PC fixe avec un processeur Rysen 5 5500 (6 coeurs, fréquence de 3,6 GHz) et possédant 16 Go de RAM sur Windows 11 version 10.0.22000. Elles ont été obtenues en utilisant la fonction timeit de la bibliothèque timer de python.

Nous allons commencer cette analyse en étudiant l'impact du nombre de délégués et du nombre de votants sur le temps de calcul de l'élection.

6.1 Première analyse: nombre de votants et nombre de délégués

Lors de ces simulations, la proportion de votant direct pour les délégués et les votants est fixée à 0.5, la moité des votes sont directs. Le nombre de réponses est fixé à 3 et le nombre de curateurs à 5. Les variables testées sont donc le nombre de délégués qui varie entre 1 et 300 et le nombre de votant classique qui varie entre 1 et 3000. Nous commencerons par analyser les résultats des simulations d'élection utilisant la méthode 1 de copie de vote.

6.1.1 Méthode 1

Afin de se représenter l'influence des variables testées, observons ce graphique où le nombre de délégués est en abscisse, le nombre de votants classique en ordonnée et le temps de calcul en secondes en côte.

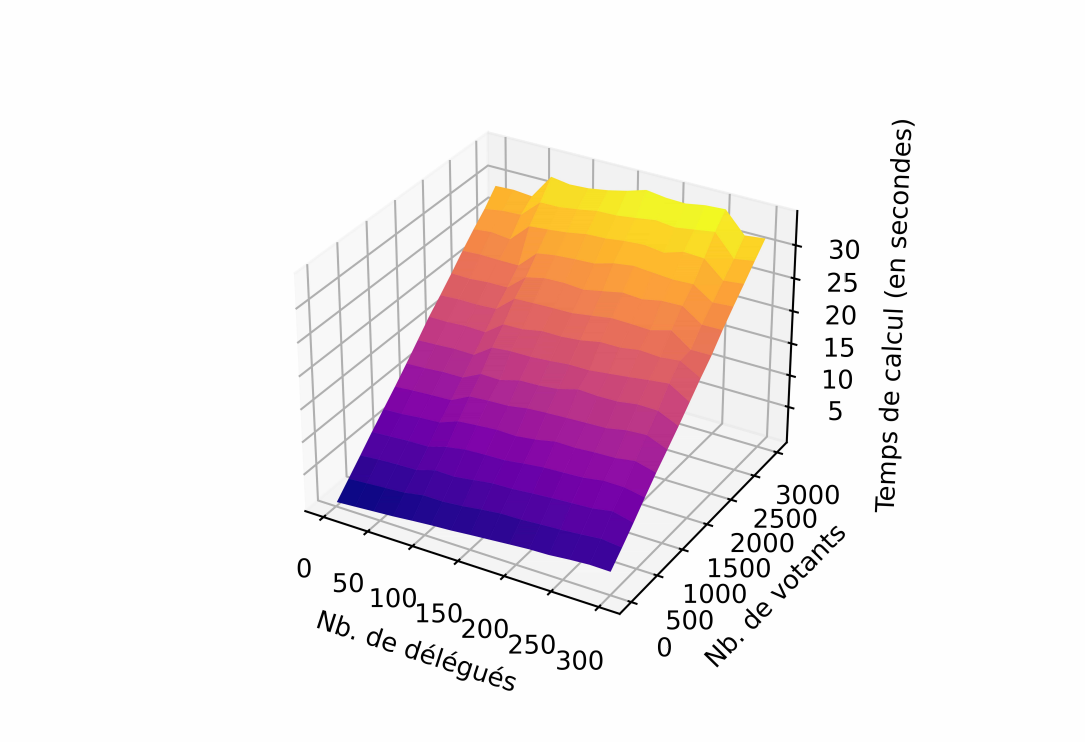

Figure 6.1: Graphique du temps de calcul (z) de l'élection en fonction du nombre de délégués (x) et du nombre de votants (y)

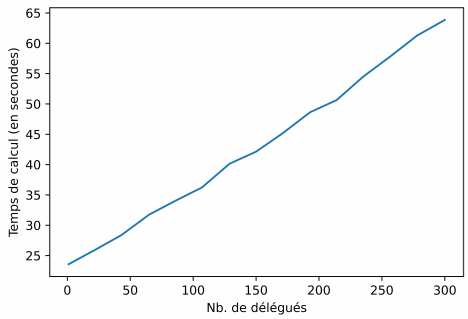

Figure 6.2: Projection en 2 dimensions sur la courbe de niveau où y=3000 votants

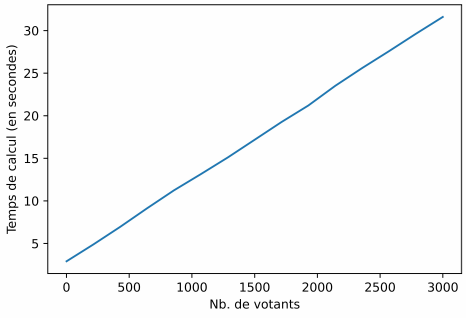

Figure 6.3: Projection en 2 dimensions sur la courbe de niveau où x=300 délégué

L'analyse des graphique en 3d et 2d permettent de réaliser que le nombre de délégués et le nombre de votants influence tous deux linéairement le temps de calcul.

Le nombre de délégué et le nombre de votants ne semblent pas avoir d'impact différent et semble être interchangeable dans le graphe.

Il n'est donc pas nécessaire d'apporter une attention particulière au nombre de délégués lors d'une élection créée par un système de vote utilisant le méthode 1 de copie de vote.

6.1.2 Méthode 2

Afin de se représenter l'influence des variables testées, observons ce graphique où le nombre de délégués est en abscisse, le nombre de votants classique en ordonnée et le temps de calcul en secondes en côte.

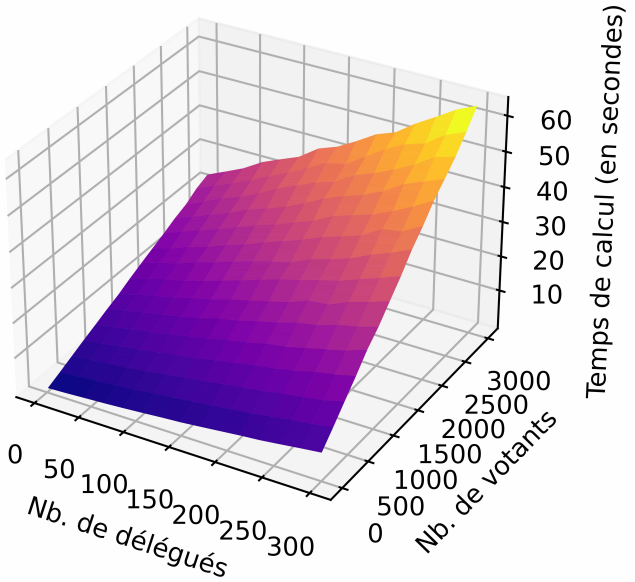

Figure 6.4: Graphique du temps de calcul (z) en fonction du nombre de délégués (x) et du nombre de votants (y)

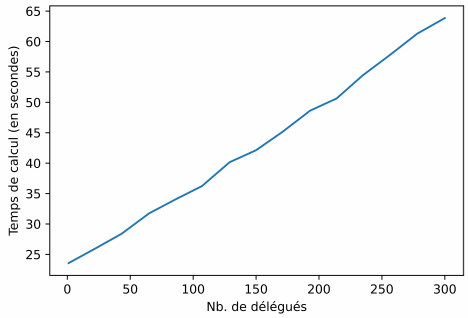

Figure 6.5: Projection en 2 dimensions sur la courbe de niveau où y=3000 votants

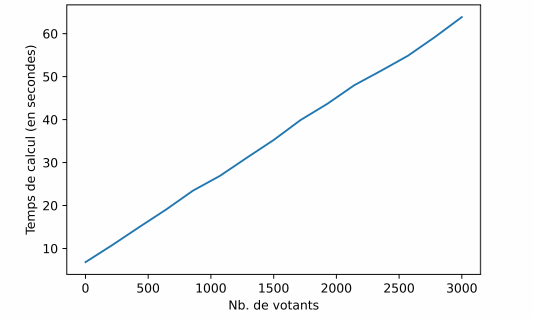

Figure 6.6: Projection en 2 dimensions sur la courbe de niveau où x=300 délégué

L'analyse des graphes permet de se rendre compte qu'à nouveau le nombre de votants impacte linéairement le temps de calcul lorsque le nombre de délégués est fixe. On remarque également que le nombre de délégué impacte linéairement le temps de calcul lorsque le nombre de votants classiques est fixe. Cependant l'impact du nombre de votants sur le temps de calcul est influencé par le nombre de délégué. Plus le nombre de délégué est élevé plus l'impact sur le temps de calcul du nombre de votants sera élevé lui aussi.

Lors d'une élection réelle en utilisant des systèmes de vote utilisant la méthode 2 de copie, il faudra donc faire attention à limiter le nombre de délégués si l'on veut garder le temps de calcul bas.

6.2 Seconde analyse: nombre de délégués et proportion de vote direct

Lors de ces simulations, le nombre de votants classiques est fixé à 3000, le nombre de réponses est fixé à 3 et le nombre de curateurs à 5. Les variables testées sont donc le nombre de délégués qui varie entre 1 et 300 et la proportion de vote direct au sein des votants et délégués varie entre 0.1 et 1. Nous commencerons par analyser les résultats des simulations d'élection utilisant la méthode 1 de copie de vote.

6.2.1 Méthode 1

Afin de se représenter l'influence des variables testées, observons ce graphique où le nombre de délégués est en abscisse, la proportion de vote direct en ordonnée et le temps de calcul en secondes en côte.

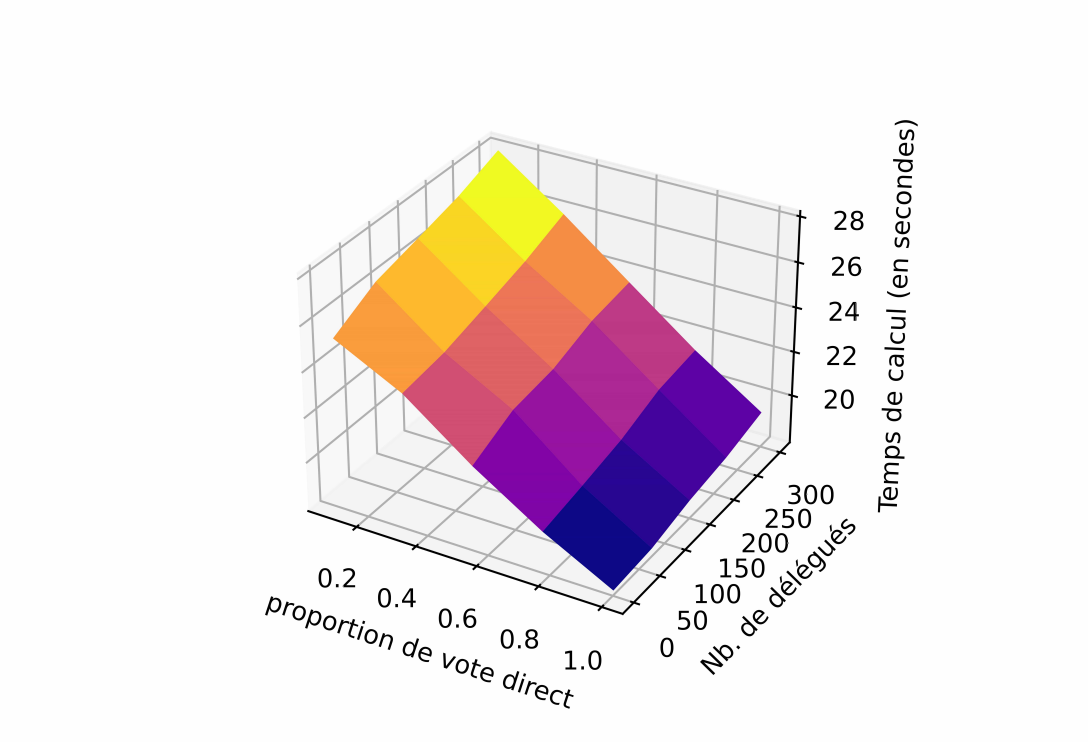

Figure 6.7: Graphique du temps de calcul(z) en fonction de la proportion de vote direct (x) et du nombre de délégués(y) (méthode 1)

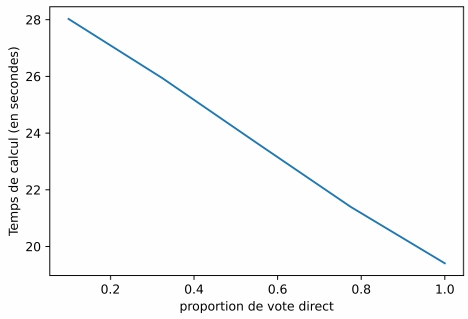

Figure 6.8: Projection en 2 dimensions sur la courbe de niveau où y=300 délégués

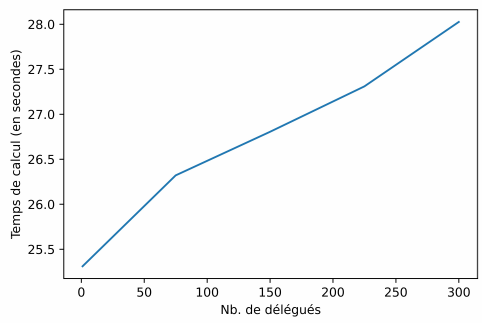

Figure 6.9: Projection en 2 dimensions sur la courbe de niveau où la proportion x=0.1

En observant les graphes, on remarque que le temps de calcul augmente linéairement avec la proportion de vote direct et le nombre de délégué. La proportion

de vote direct semble avoir un impact conséquent sur le temps de calcul, indépendamment du nombre de délégués. Dans le cadre d'élections organisées avec des systèmes de votes utilisant la méthode de copie 1, la proportion de votant direct peut grandement influer sur le temps de calcul de l'élection. La copie d'un vote prend donc davantage de temps qu'un vote direct.

6.2.2 Méthode 2

Afin de se représenter l'influence des variables testées, observons ce graphique où le nombre de délégués est en abscisse, la proportion de vote direct en ordonnée et le temps de calcul en secondes en côte.

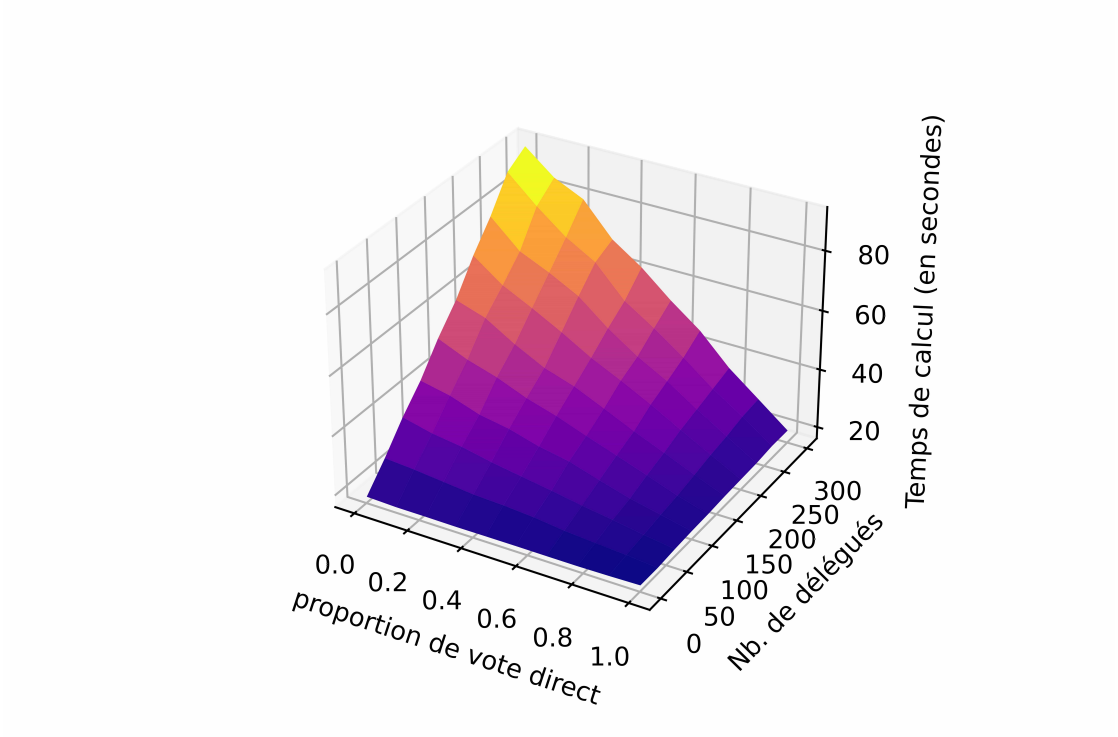

Figure 6.10: Graphique du temps de calcul (z) en fonction de la proportion de vote (x) et du nombre de délégués (y) (méthode 2)

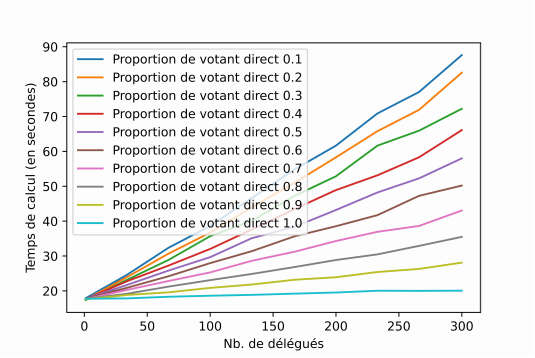

Figure 6.11: Projection en 2 dimensions sur un ensemble de courbe de niveau de nombre de délégués fixes entre 1 et 300

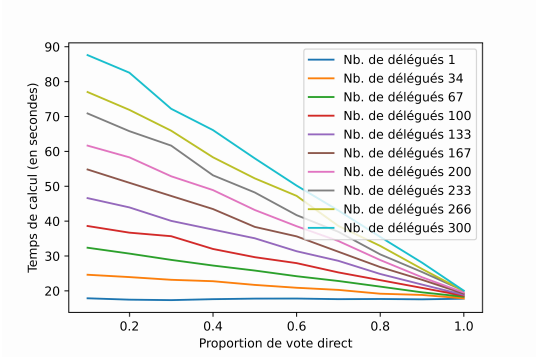

Figure 6.12: Projection en 2 dimensions sur un ensemble de courbe de niveau de proportions fixes entre 0 et 1

On remarque que lorsque le nombre de délégué est élevé, le temps de calcul augmente beaucoup plus rapidement lorsque la proportion de vote direct diminue et inversement.
Lors d'une utilisation des systèmes de vote comprenant la deuxième méthode de copie, si on veut maintenir un temps de calcul raisonnable, le nombre de délégués doit être particulièrement surveillé si le nombre de copie de vote est élevé.

6.3 Troisième analyse: le nombre de curateurs

Lors de ces simulations, la nombre de votants classiques est fixé à 3000, le nombre de délégué à 300, le nombre de réponses possibles à 3 et la proportion de vote direct à 0.5. La variables testée est le nombre de curateurs qui varie entre 1 et 100. Nous allons analyser les temps de calcul des simulations de l'élection avec ces paramètres.

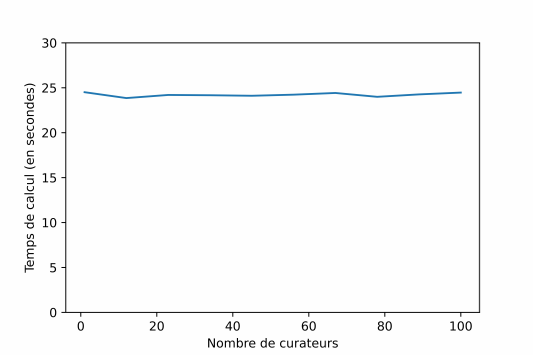

Figure 6.13: Graphique du temps de calcul (en s) en fonction du nombre de curateurs (méthode 1)

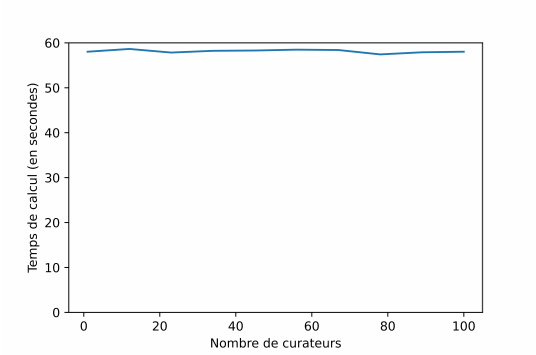

Figure 6.14: Graphique du temps de calcul (en s) en fonction du nombre de curateurs (méthode 2)

La modification du temps de calcul du au nombre de curateurs avec ces variables fixes est négligeable et ce pour les simulations des deux systèmes de vote.

Il n'est donc pas nécessaire de porter une attention particulière aux nombres de curateurs pour maintenir un temps de calcul bas.

6.4 Quatrième analyse: le nombre de réponses

Lors de ces simulations, la nombre de votants classiques est fixé à 3000, le nombre de délégué à 300 et le nombre de curateurs à 5. La variables testée est le nombre de réponses possibles qui varie entre 1 et 10. Nous allons analyser les temps de calcul des simulations de l'élection avec ces paramètres.

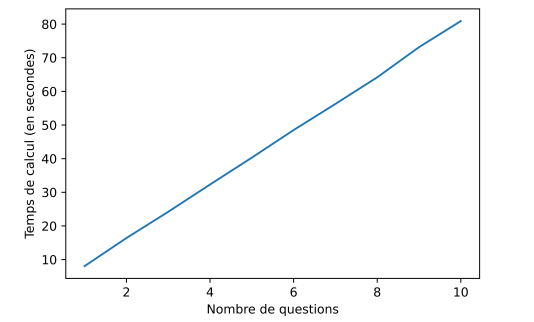

Figure 6.15: Graphique du temps de calcul (en s) en fonction du nombre de réponses possibles (méthode 1)

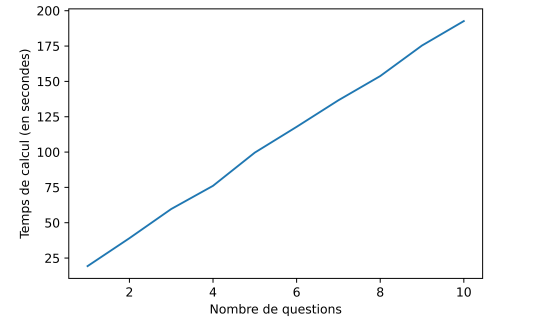

Figure 6.16: Graphique du temps de calcul (en s) en fonction du nombre de réponses possibles (méthode 2)

Les temps de calcul des simulations pour les deux méthodes de copie notent une augmentation linéaire conséquente dans le temps de calcul lorsque le nombre de questions augmentent. Il est donc judicieux de réfléchir aux nombres de réponses possibles, lors d'une élection utilisant l'un de ses deux systèmes de vote.

6.5 Résumé

Pour les systèmes d'élection utilisant la méthode 1 de copie, ces analyses nous suggèrent que le temps de calcul est augmenté grandement par la proportion de vote copié et par le nombre de questions. Lors d'une élection dans la cadre d'une démocratie liquide, le proportion de vote copié n'est pas une variable que l'on peut contrôler étant donné que le votant décide s'il vote directement ou s'il délègue son vote. Il faut donc maintenir un nombre de réponses raisonnable si l'on veut organiser des élections à grandes échelles avec un temps de calcul raisonnable.

Ensuite, pour les systèmes d'élection utilisant le méthode 2 de copie, nous avons vu que le nombre de délégués, la proportion de vote direct, et le nombre de réponses influent grandement le temps de calcul d'une élection. Comme dit précédemment, dans le cadre d'une démocratie liquide, la proportion de vote direct ne peut être contrôlé, il faut donc contrôler le nombre de délégués pour que le temps de calcul n'augmente pas de façon conséquente avec la proportion de vote copié. Il faut également maintenir un nombre de réponses raisonnable si l'on ne veut pas que le temps de calcul de l'élection augmente en conséquence.

Chapitre 7

Conclusion

Le but de ce mémoire était de créer une simulation de système d'élection permettant la mise en place d'élection dans le cadre d'une démocratie liquide.

Pour ce faire, nous avons commencé par définir et modéliser ce qu'est la démocratie liquide.

Nous avons ensuite spécifié les acteurs, les adversaires, les hypothèses de confiance et l'organisation de l'élection nécessaire à la création d'un système de vote dans ce cadre. Nous avons remarqué que le challenge de ce mémoire consistait à trouver une méthode de délégation de vote qui permettait de maintenir le secret de vote des votants en créant un nouvel acteur appelé délégué privé. En effet, les votants devaient pouvoir déléguer leurs votes à ce nouvel acteur sans que celui-ci puisse savoir qui avait voté pour lui.

Subséquemment, nous avons introduit les pré-requis nécessaire à la compréhension des outils mathématiques utilisés dans le protocole. Nous avons ensuite défini le protocole en nous inspirant du protocole du système de vote Helios. Nous avons commencé par expliquer comment l'administrateur crée l'élection et les curateurs génèrent la clef publique de l'élection dans la partie pré-élection. Nous avons ensuite expliqué la méthode de génération de bulletin avec sa preuve de validité utilisée par Helios. Ensuite nous avons utilisé la proposition de copie de vote sous Helios proposée par Desmedt et Chaidos [12] afin de proposer une méthode de copie de vote. Cependant leur méthode de copie de preuve de vote ne convenait pas pour une élection dans la cadre d'une démocratie liquide, car elle était interactive et nécessitait donc une interaction entre le délégué et le votant. En utilisant le méthode de copie de vote et en rendant le vote des délégués public avant la fin de l'élection en partageant l'aléatoire du chiffrement de leur vote, nous avons créé une méthode de copie de vote que nous avons appelé méthode 1 de copie de vote. Cette méthode de vote ne nous satisfaisait pas parce que le vote des délégués était révélé avant la fin de l'élection. C'est pour cette raison que nous avons créé une deuxième méthode de copie de bulletin de vote. Celle-ci utilise la méthode de copie de vote

précédemment introduite et modifie le type de preuve associé. Les votes copiés devront être associés à une preuve qu'ils sont une copie d'un des votes des délégués. Nous avons appelé cette méthode de copie, méthode 2 de copie de vote. Pour finir ce protocole, nous avons présenté les étapes nécessaires au décompte des voix.

Suite à la présentation du protocole, nous avons présenté les pseudo codes des fonctions nécessaires à l'implémentation de ce système de vote. Nous avons commencé par présenter les fonctions communes à une système de vote classique pour ensuite présenter les méthodes d'envoi et de copie de bulletin de vote spécifique aux méthodes 1 et 2 de copie.

Nous avons utilisé le code présenté précédemment pour procéder à des simulations d'élection utilisant la méthode 1 de copie de vote et des simulations d'élection utilisant la méthode 2 de copie de vote afin de les analyser. Nous avons examiné l'influence du nombre des votants classiques, de délégués, de la proportion de vote direct, du nombre de curateurs et du nombres de réponses possibles sur le temps de calcul de ces simulations d'élections. Nous avons pu tirer des conclusions sur les variables qu'il fallait surveiller pour maintenir un temps de calcul raisonnable. Pour la méthode 1 de copie, nous avons remarqué que le temps de calcul était grandement augmenté par le nombre de réponses possibles et par la proportion de vote copié. La proportion de vote copié étant non contrôlable dans le cadre d'une démocratie liquide, il faut donc maintenir un nombre de réponses possibles raisonnable pour garder une temps de calcul correct d'une élection utilisant une système de vote comprenant la méthode 1 de copie. Pour la méthode 2 de copie, nous avons remarqué que le temps de calcul était augmenté par le nombre de réponses possibles et qu'une combinaison de vote copié élevée et de nombre de délégués élevé entraînaient un augmentation conséquente du temps de calcul. Le proportion de vote copié étant non contrôlable, il faut donc maintenir le nombre de réponses possibles raisonnablement bas et le nombre de délégués bas en fonction de la proportion de votes copié.

Grâce à ces différentes étapes, nous avons donc créés deux différents systèmes de vote permettant la création d'élection satisfaisante pour une utilisation dans le cadre d'une démocratie liquide.

Chapitre 8

Discussion

Bien qu'une simulation de vote soit possible, cette simulation n'est pas encore suffisante afin d'instaurer de vraies élections dans le cadre d'une démocratie liquide. Dans cette partie, nous allons détailler des propositions d'amélioration et d'optimisation dans le but de donner des pistes de futures recherches possibles.

Autant pour les systèmes utilisant la méthode 1 de copie que la méthode 2 de copie, le nombre de réponses possibles augmentent grandement le temps de calcul des simulations d'élections. Il n'est donc pas recommandé d'implémenter l'existence de délégué publics en rajoutant leur noms en réponses aux questions posées lors de l'élection car cela ralentirait considérablement le déroulement de l'élection. Il faudrait donc au choix optimiser le système de vote afin que le temps de calcul ne soit pas trop long lorsqu'on introduit des délégués en guise de réponse, ou trouver une méthode alternative à leur implémentation.

Lors de ce mémoire, nous avons essayé de créer des systèmes de vote permettant la mise en place d'élection dans le cadre d'une démocratie liquide mais nous avons créé des systèmes de vote permettant de générer des simulations d'élections sur un ordinateur unique. Etant donné que la fin recherchée est de pouvoir implémenter ce système dans le cadre d'une démocratie liquide, il est nécessaire d'y ajouter une interface graphique. Elle devrait pouvoir être facilement maniable par les votants potentiels ce qui nécessiterait la prise en compte de son ergonomie, étant donné que, par exemple, des personnes âgées devraient pouvoir être à même de les utiliser. Il apparaît important qu'elle soit de plus capable de tourner sur n'importe quel ordinateur, et en nécessitant peu d'installation. De ce fait, une utilisation par navigateur web paraît optimale. De plus, il faudra prendre en compte l'interaction entre les différents appareils personnels et le serveur afin de pouvoir afficher les résultats.

Toujours dans le cadre de sa transformation en système de vote utilisable, il faudra implémenter les fonctionnalités nécessaires au maintien des propriétés "End-to-End" du système de vote. Il faudra donc créer une fonction qui permettra

aux votants d'effectuer un Benaloh challenge. Cette fonction doit pouvoir vérifier si son bulletin de vote est conforme à son intention de vote avant son envoi sur le tableau des bulletins. Il faudra aussi s'assurer que le votant aie un accès aux informations nécessaires sur le tableaux des bulletins afin qu'il puisse s'assurer que son vote est bien sur le tableau de vote et qu'il puisse constater de la justesse du calcul du résultat de l'élection et la validité des différentes preuves associées à la génération de la clef publique de l'élection, des votes des autres votants et du calcul des décomptes partiels.

Bien que d'autres points d'amélioration puissent être avancés, les trois points énoncés ci-dessus nous semblent être les principales pistes permettant d'avancer un peu plus vers une implémentation d'un système de vote de démocratie liquide viable et utilisable. Étant donné que les conditions imposées par le fait d'utiliser ce type de démocratie semblent nécessiter une plate-forme e-voting, notamment à cause de la condition de rappel instantané, cela apparaît comme étant d'autant plus important. Cela permettrait à, qui sait, un pays, d'adopter une démocratie liquide sans avoir à se soucier (ou moins) des considérations techniques qui y sont liées.

Bibliography

[1] LA CONSTITUTION BELGE, . URL https://www.senate.be/doc/const_fr.html#art65.

[2] Democracia en Red, . URL https://democraciaenred.org/.

[3] Open Verificatum Project, . URL https://verificatum.org.

[4] B. Adida, O. de Marneffe, O. Pereira, and J.-J. Quisquater. Electing a University President using Open-Audit Voting: Analysis of real-world use of Helios. 2009. URL https://dial.uclouvain.be/pr/boreal/object/boreal:92266.

[5] S. T. Ali and J. Murray. An Overview of End-to-End Verifiable Voting Systems. In *Real-World Electronic Voting*. Auerbach Publications, 2016. ISBN 978-1-315-37129-0. Num Pages: 46.

[6] J. Behrens, A. Kistner, A. Nitsche, and B. Swierczek. The Principles of LiquidFeedback, 2014. URL https://principles.liquidfeedback.org/.

[7] D. Bernhard and B. Warinschi. Cryptographic Voting — A Gentle Introduction. In A. Aldini, J. Lopez, and F. Martinelli, editors, *Foundations of Security Analysis and Design VII: FOSAD 2012/2013 Tutorial Lectures*, Lecture Notes in Computer Science, pages 167–211. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10082-1. doi: 10.1007/978-3-319-10082-1_7. URL https://doi.org/10.1007/978-3-319-10082-1_7.

[8] D. Bernhard, O. Pereira, and B. Warinschi. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *ASIACRYPT*, 2012. doi: 10.1007/978-3-642-34961-4_38.

[9] C. Blum and C. I. Zuber. Liquid Democracy: Potentials, Problems, and Perspectives: Liquid Democracy. *Journal of Political Philosophy*, 24(2): 162–182, June 2016. ISSN 09638016. doi: 10.1111/jopp.12065. URL http://doi.wiley.com/10.1111/jopp.12065.

[10] D. Boneh. The Decision Diffie-Hellman problem. In J. P. Buhler, editor, *Algorithmic Number Theory*, Lecture Notes in Computer Science, pages 48–63, Berlin, Heidelberg, 1998. Springer. ISBN 978-3-540-69113-6. doi: 10.1007/BFb0054851.

[11] V. Brengarth. Les gilets jaunes nous invitent à repenser la démocratie par Vincent Brengarth | Politis. URL https://www.politis.fr/articles/2019/01/les-gilets-jaunes-nous-invitent-a-repenser-la-democratie-39857/.

[12] Y. Desmedt and P. Chaidos. Applying Divertibility to Blind Ballot Copying in the Helios Internet Voting System. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, S. Foresti, M. Yung, and F. Martinelli, editors, *Computer Security – ESORICS 2012*, volume 7459, pages 433–450. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-33166-4 978-3-642-33167-1. doi: 10.1007/978-3-642-33167-1_25. URL http://link.springer.com/10.1007/978-3-642-33167-1_25. Series Title: Lecture Notes in Computer Science.

[13] I. o. C. S. a. A. G. B. A. Division. Voting at general meetings. *Keeping Good Companies*, Apr. 2004. URL https://search.informit.org/doi/abs/10.3316/ielapa.200404323.

[14] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology*, Lecture Notes in Computer Science, pages 10–18, Berlin, Heidelberg, 1985. Springer. ISBN 978-3-540-39568-3. doi: 10.1007/3-540-39568-7_2.

[15] H. Feng and R. Peter. Preface. In *Real-world Electronic Voting: Design, Analysis and Deployment*. Auerbach Publications, 2017.

[16] O. Godard. « Le Monde des lecteurs » - Gilets jaunes : une démocratie horizontale plébiscitée par l'opinion ? *Le Monde.fr*, Dec. 2018.

[17] J. Green-Armytage. Direct voting and proxy voting. *Constitutional Political Economy*, 26, June 2015. doi: 10.1007/s10602-014-9176-9.

[18] G. Grunberg. Les « gilets jaunes » et la crise de la démocratie représentative. *Le Debat*, n° 204(2):95–103, Apr. 2019. ISSN 0246-2346. Publisher: Gallimard.

[19] S. Hardt and L. Lopes. Google Votes: A Liquid Democracy Experiment on a Corporate Social Network. *Defensive Publications Series*, June 2015. URL https://www.tdcommons.org/dpubs_series/79.

[20] G. Kessler. An Overview of Cryptography (Updated Version, 3 March 2016). *Publications*, Mar. 2016. URL https://commons.erau.edu/publication/127.

[21] H. Landemore. *Open Democracy: Reinventing Popular Rule for the Twenty-First Century*. Princeton University Press, Oct. 2020. ISBN 978-0-691-20872-5. doi: 10.1515/9780691208725. URL https://www.degruyter.com/document/doi/10.1515/9780691208725/html. Publication Title: Open Democracy.

[22] H. Macq. *Militer pour et sur Internet. L'adhésion au Parti Pirate belge*. PhD thesis, Université de Liège, Liège, Belgique, Sept. 2015. URL https://orbi.uliege.be/handle/2268/198419.

[23] J. C. Miller. A Program for Direct and Proxy Voting in the Legislative Process. *Public Choice*, 7:107–113, 1969. ISSN 0048-5829. URL https://www.jstor.org/stable/30022612. Publisher: Springer.

[24] A. Paulin. An Overview of Ten Years of Liquid Democracy Research. In *The 21st Annual International Conference on Digital Government Research*, dg.o '20, pages 116–121, New York, NY, USA, June 2020. Association for Computing Machinery. ISBN 978-1-4503-8791-0. doi: 10.1145/3396956.3396963. URL https://doi.org/10.1145/3396956.3396963.

[25] O. Pereira. Internet Voting with Helios. 2016.

[26] B. Swierczek. 5 years of Liquid Democracy in Germany (The Liquid Democracy Journal, Issue 1), 2011. URL https://liquid-democracy-journal.org/issue/1/The_Liquid_Democracy_Journal-Issue001-02-Five_years_of_Liquid_Democracy_in_Germany.html.

[27] C. Valsangiacomo. Clarifying and Defining the Concept of Liquid Democracy. *Swiss Political Science Review*, n/a(n/a). ISSN 1662-6370. doi: 10.1111/spsr.12486. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/spsr.12486. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/spsr.12486.

[28] C. Valsangiacomo. Political Representation in Liquid Democracy. *Frontiers in Political Science*, 3, 2021. ISSN 2673-3145. URL https://www.frontiersin.org/article/10.3389/fpos.2021.591853.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**