MASTER EN SCIENCES MATHÉMATIQUES

Vote à voix unique transférable : développement d'un protocole cryptographique

DESLEE, Pierre

Award date:
2015

Awarding institution:
Universite de Namur

UNIVERSITE DE NAMUR
Faculté des Sciences

VOTE A VOIX UNIQUE TRANSFÉRABLE : DÉVELOPPEMENT D'UN PROTOCOLE CRYPTOGRAPHIQUE

Mémoire présenté pour l'obtention
du grade académique de master en sciences mathématiques

Pierre Deslee

Août 2015

UNIVERSITE DE NAMUR
Faculté des Sciences

VOTE A VOIX UNIQUE TRANSFÉRABLE : DÉVELOPPEMENT D'UN PROTOCOLE CRYPTOGRAPHIQUE

Mémoire

Pierre Deslee

Directeur : Olivier Pereira
Promoteur : Renaud Lambiotte
Année académique : 2014-2015

"Ce qui compte ce n'est pas le vote, c'est comment on compte les votes."

J. Staline

Remerciements

Je me suis lancé il y a cinq ans dans les études de sciences mathématiques avec un objectif : découvrir la cryptographie. Grâce à ce mémoire, j'ai pu réaliser ce vieux rêve. Pour cela, je tiens à remercier toutes les personnes qui m'ont apporté leur aide et qui m'ont suivi. Un merci tout particulier aux personnes suivantes pour leur aide inestimable.

Tout d'abord, merci à mes promoteurs Olivier Pereira et Renaud Lambiotte pour m'avoir guidé tout au long de ce mémoire.

Je remercie ensuite ma famille pour les nombreuses relectures et corrections orthographiques de ce manuscrit, ainsi que pour leur soutien tout au long de mes études. Je n'oublie pas Marie-Lyne Maus pour les relectures et corrections qu'elle a également effectuées.

Pour finir, un tout grand merci à tous mes amis mathématiciens pour tous ces moments passés ensemble ainsi qu'à à Valentin Vander Borght et Violette Bricout de m'avoir soutenu depuis tant d'années.

Résumé

Le single transferable voting ou en français vote à voix unique transférable est un système de vote qui fonctionne de la manière suivante. L'électeur trie les candidats selon ses préférences pour constituer son bulletin de vote. La voix unique de l'électeur va alors au premier candidat de son bulletin. Pour être élu, un candidat doit dépasser un certain seuil. Toutes les voix d'un candidat élu qui dépassent ce seuil sont transférées aux candidats en deuxième position sur les bulletins des électeurs où le candidat élu était en tête. Si aucun candidat n'est élu, le candidat ayant le moins de voix est éliminé et ses votes sont transférés aux candidats en deuxième position sur les bulletins des électeurs où le candidat éliminé était en tête. La particularité de ce type de vote est sa capacité à prendre en compte les minorités, son ouverture aux changements d'opinion, sa capacité à promouvoir la place de la femme dans les gouvernements, ect. Il est utilisé dans des pays comme l'Australie et l'Irlande. Son problème réside en la difficulté de mise en œuvre.

L'objectif de ce mémoire est de développer un protocole cryptographique pour rendre ce système utilisable de manière électronique tout en respectant les principes fondamentaux d'une élection, c'est-à-dire que le protocole doit respecter les principes de confidentialité, de robustesse, de vérifiabilité universelle et de résistance à la coercition. Dans un premier temps, nous allons présenter en détails le single transferable voting. Dans un deuxième temps, nous nous intéressons au problème du vote électronique dans sa généralité pour prendre en main les principes présentés ci-dessus. Dans un dernier temps, nous adapterons toutes ces notions pour les appliquer au cas du single transferable voting.

Mots clefs : vote à voix unique transférable, STV, chiffrement Paillier, vote électronique, cryptographie à clef publique.

Abstract

The single transferable voting is a system of vote that works in the following ways. The voter sorts out the candidates according to his preferences. The unique vote of the voter goes then to the first candidate of its list. To be elected, a candidate has to overtake certain threshold. Quite votes of an elected candidate which overtake threshold are transferred to the candidates in the second position on the lists of the voters where the elected candidate was in the lead. If no candidate is elected, the candidate having least vote is eliminated and his votes are to transfer to the candidates in the second position on the lists of the voters where the eliminated candidate was in the lead. The peculiarity of this type of vote is its capacity to take into account minorities, his openness in the changes of opinion, its capacity to promote the gender equality in government... It is used in countries such as Australia and Ireland. His problem is its difficulty being set up.

The main goal of this master's thesis is to develop a concrete cryptographic protocol which respects the fundamental principles of an election. That is the protocol has to respect the principles of confidentiality, robustness, universal verifiability and resistance in the pressure. At first, we present in details the transferable voting single. Secondly, we are interested in the problem of the electronic vote in its majority to take in hand the principles presented above. And in a last time, we adapt all these notions for them applied to the case of the single transferable voting.

Keywords : single transferable voting, STV, Encryption Paillier, electronic vote, cryptography with public key.

Table des matières

Chapitre 1

Introduction

Jusqu'à la fin du 20^e siècle, la cryptographie était considérée comme un art : « *l'art d'écrire en caractères secrets qui sont ou de convention ou le résultat d'une transposition des lettres de l'alphabet*[1] ». Cet art reposait sur l'imagination et les compétences personnelles, il n'y avait pas de notion théorique sur ce qui constitue un bon chiffrement[2]. Tous les codes inventés durant cette période sont d'ailleurs très faciles à casser. C'est après la seconde guerre mondiale que la cryptographie a radicalement changé. Une théorie est apparue transformant cet art en une véritable science qui ne s'intéresse plus seulement à garder secret le contenu de communications, mais s'intéresse également à des problèmes comme l'authentification des messages, l'échange de clefs secrètes, les élections électroniques, les transactions bancaires,... Alors qu'avant la cryptographie était essentiellement utilisée à des fins militaires, aujourd'hui elle est utilisée dans tous les ordinateurs. La cryptographie est passée d'un art utilisé par les militaires à une science aidant à sécuriser les opérations de la vie courante pour les gens ordinaires à travers le monde. Selon [1], une définition adéquate pour la cryptographie moderne serait : *« la cryptographie est l'étude scientifique des techniques pour sécuriser l'information numérique, les transactions et calculs distribués*[3] *»*.

Parmi les plus anciens chiffrements, nous pouvons citer celui de César (50 avant Jésus-Christ) qui consiste à faire une permutation des lettres de l'alphabet de trois caractères : a devient d, b devient e, ... Nous pouvons également mentionner la substitution mono-alphabétique qui est une amélioration du chiffrement de César. A la place de décaler chaque lettre de trois caractères, on substitue une lettre à une autre. Le problème de ces chiffrements est que chaque lettre est à chaque fois remplacée par la même ; une simple analyse fréquentielle[4] suffit à les casser. Supposons que le langage de base soit l'anglais, nous savons alors que la lettre la plus utilisée sera le e suivie du t et du a. Il suffit alors de comptabiliser les occurrences de chaque caractère chiffré et de les comparer avec le tableau de fréquences de la langue anglaise. Le caractère apparaissant le plus souvent dans le texte chiffré correspondra alors certainement au e dans le message clair. Ces informations sont amplement suffisantes pour casser ces codes. Notons quand même que pour des textes courts, la fréquence d'occurrence des lettres dans le chiffré sera bien différente de celle qui caractérise le langage de base. Pour des textes de moins

1. Dictionnaire français Littré.
2. Le chiffrement est un procédé cryptographique qui consiste à rendre impossible la compréhension d'un message à toute personne qui n'a pas la clé de (dé)chiffrement.
3. Le calcul distribué est l'action de répartir un calcul ou un traitement sur toute unité centrale informatique.
4. L'analyse fréquentielle est une méthode du 8^e siècle inventée par le savant arabe al-Kindi.

de cent caractères, l'analyse fréquentielle est rarement utile. Une amélioration des chiffrements à substitutions mono-alphabétiques pourrait être le chiffre de Vigenère (1586) qui consiste à choisir une clef qui servira à chiffrer le texte clair en ajoutant à chaque caractère clair le caractère correspondant de la clef. A une lettre donnée d'un message, il peut correspondre autant de lettres que de lettres différentes dans la clef. C'est ce qu'on appelle un chiffrement polyalphabétique. Par exemple :

Texte clair :	chiffrementvigenere
Clef :	codecodecodecodecod
Texte chiffré	fwmkigirhcxalvishgi

Mais, pour un texte long, comme la clef se répète, il peut être assez aisé de découvrir la taille de la clef en repérant une succession de lettres qui reviennent souvent. Le procédé est le suivant : on fait la liste de tous les caractères répétés et l'espace qu'il y a entre eux, puis on cherche les diviseurs entiers du nombre de caractères des espaces. La longueur de la clef est l'un des diviseurs communs. Une fois la taille de la clef découverte, une analyse fréquentielle sur les caractères placés à même fréquence que la longueur de clef suffit à déchiffrer le message. Pour découvrir cette méthode, il faut attendre le cryptanalyste Charles Babbage qui découvrit cet exercice en 1854. Cependant, Babbage ne publia pas ses résultats. En 1863, Friedrich Krasiski publia enfin une méthode similaire qui mit fin au règne des chiffrements polyalphabétiques. Au fur et à mesure du temps, grâce à l'apparition de la mécanisation, ces chiffrements "anciens" deviendront de plus en plus compliqués jusqu'à l'invention par les Nazis de la machine Enigma. C'est le cryptanalyste Alan Turing, aidé de ses collègues de Bletchley Park, qui réussit à déchiffrer les messages émis par la version la plus complexe d'Enigma à l'aide d'une analyse mathématique très poussée. Il conçut un appareil électrique qui permit de reproduire toutes les 1 054 650 combinaisons possibles des trois retors d'Enigma en moins de 5h. L'amélioration de cet appareil permit à l'équipe de Betchley Park de développer le premier prototype d'ordinateur de l'histoire, nommé *Colossus*, en $1943 - 44$.

Ces différents exemples nous montrent qu'il est impossible de construire intuitivement un chiffrement sécurisé. Le problème principal de ces anciens chiffrements est qu'aucun phénomène complètement aléatoire n'apparait dans leurs définitions. Une lettre sera régulièrement chiffrée de la même manière. De réelles règles mathématiques doivent être utilisées pour pouvoir construire des chiffrements sécurisés. Un premier chiffrement de ce type fut inventé en 1929, le chiffre de Hill, qui ne parvint pas à percer dans le milieu des communications secrètes. Il faudra attendre le cassage d'Enigma pour une réelle remise en question. Depuis 1977, des chiffrements sûrs ont vu le jour à la suite de travaux de scientifiques. Nous pouvons citer les chiffrements RSA, El Gamal, Paillier, et bien d'autres. Une explication générale sur le fonctionnement de ces chiffrements modernes est donnée à la section (3.1). Le point commun entre tous ces chiffrements modernes est l'utilisation de problèmes mathématiques impossibles à résoudre de nos jours, comme la factorisation d'un grand nombre en un produit de facteurs premiers.

Aujourd'hui, la cryptographie est présente partout : dans les cartes à puces, quand nous envoyons un e-mail, quand nous discutons avec quelqu'un en ligne, ... L'un des plus grands défis restant est le vote électronique. Organiser un scrutin électoral coûte très cher et le vote électronique pourrait être une source d'économie. De plus, il améliorerait le

ressenti du citoyen par rapport au vote. En effet, il n'y aurait plus de files interminables, le citoyen pourrait voter sur une période plus longue et à un moment qui lui semblerait adéquat, ... Cependant le fait de passer par un système informatique rend les fraudes plus faciles. C'est pourquoi, il est nécessaire de développer des protocoles permettant à chaque citoyen de vérifier que son vote a bien été pris en compte tout en le laissant anonyme, mais permettant aussi de s'assurer que personne ne tente d'introduire de faux votes, de modifier des votes existants, d'empêcher l'achat des votes des électeurs, ... Ce sont bien là les défis à relever dans un contexte aussi important que particulier qu'est le vote électronique. Dans ce travail, je me focaliserai sur un cas particulier de système de vote : le scrutin à vote unique transférable ou single transferable voting (STV).

Il est à noter que de nombreuses recherches sur le sujet sont à l'heure actuelle en court. Nous pouvons pointer le système vVote [2] déployé en Australie. C'est un système qui donne la possibilité à chaque votant de vérifier que son vote a été généré, enregistré et compté correctement sans qu'il n'ait besoin d'avoir confiance en un quelconque logiciel, observateur, procédure, autorité,... En 2011, l'Etat de Victoria (Australie) a demandé de développer spécialement pour eux le système vVote. Cet état utilise deux modes de scrutins : le vote préférentiel pour l'assemblée législative et le vote à voix unique transférable pour le conseil législatif. Chaque représentant de l'assemblée est élu par vote préférentiel à l'aide d'une liste complète de préférence qui dépasse rarement 10 candidats tandis que les membres du conseil sont élus par le STV à l'aide d'une liste de préférence d'au moins 5 candidats parmi 30. Une première approche a été publiée en septembre 2014 et l'Etat de Victoria a pu la tester lors de l'élection de novembre 2014. Les algorithmes de cette méthode sont basés sur des mixnets, solution très coûteuse qui pose certains soucis de confidentialité. Il reste donc de nombreuses améliorations à apporter à cette méthode.

Chapitre 2

Scrutin à vote unique transférable ou STV

Tous les résultats présentés dans ce chapitre sont issus de [3], [4] et [5] agrémentés d'un développement personnel.

Le sujet de ce travail est le scrutin à vote unique transférable ou STV. Dans ce chapitre, nous en expliquons le principe, donnons un exemple et présentons l'intérêt d'un tel mode de scrutin électoral.

2.1 Présentation

Il existe dans le monde toute une série de scrutins électoraux. Dans le cadre de ce travail, nous nous intéressons au STV. Il s'agit d'un système de vote où l'électeur, bien que ne votant que pour un seul candidat, a la possibilité de noter sur son bulletin un deuxième, troisième, ..., n^{e} candidat. Le vote de l'électeur sera transféré au j^{e} candidat si les $j-1$ précédents atteignent le quotient électoral leur permettant d'être élus ou s'ils sont éliminés.
Le quotient électoral est calculé comme suit :

$$\lfloor \frac{v}{n+1} \rfloor + 1 = \text{le nombre de votes nécessaires pour être élu}$$

où v est le nombre d'électeurs, n le nombre de sièges vacants et $\lfloor\ \rfloor$ la fonction à partie entière ; remarquons que pas plus de n candidats peuvent dépasser ce quotient.

Les votes sont comptabilisés une première fois et les candidats ayant atteint le quotient électoral sont élus. Si un candidat dépasse le quotient, ses voix en surplus (c'est-à-dire celles non nécessaires à son élection) sont transférées équitablement aux seconds des bulletins ayant pour tête de liste ce candidat. Équitablement signifie ici : « de manière proportionnelle aux nombres de votes de chaque liste où le candidat élu était en tête ».
Si aucun candidat n'atteint le quotient, le candidat comptabilisant le moins de voix est éliminé. Les voix de ce candidat sont alors transférées équitablement aux seconds des bulletins ayant pour tête de liste le candidat éliminé. Le processus se répète jusqu'à ce que les n sièges vacants soient occupés. Cela peut paraitre un peu complexe, illustrons tout ceci à l'aide d'un exemple.

2.2 Exemple

Supposons que cent électeurs doivent voter pour les candidats A, B, C, D et que seuls deux d'entre eux seront élus. Voici la répartition des votes :

nombres de votes	28	14	15	17	26
1er préférence	A	A	B	C	D
2e préférence	B	C	C	A	B
3e préférence	C	D	A	B	C
4e préférence	D	B	D	D	A

Commençons par calculer le quotient électoral :

$$\lfloor \frac{100}{2+1} \rfloor + 1 = 33 + 1 = 34.$$

Et procédons au dépouillement :

candidats	A	B	C	D	résultat
1er tour	42	15	17	26	Répartition : <table><tr><td>28</td><td>14</td><td>15</td><td>17</td><td>26</td></tr><tr><td>A</td><td>A</td><td>B</td><td>C</td><td>D</td></tr><tr><td>B</td><td>C</td><td>C</td><td>A</td><td>B</td></tr><tr><td>C</td><td>D</td><td>A</td><td>B</td><td>C</td></tr><tr><td>D</td><td>B</td><td>D</td><td>D</td><td>A</td></tr></table> Seul le candidat A atteint le quotient et est donc élu. Le surplus de ses voix (42-34=8) est transféré équitablement aux seconds de chaque bulletin où A était en tête.
2e tour	34	$15+\frac{28*8}{42}$ =15+5.3 =20.3	$17+\frac{14*8}{42}$ =17+2.7 =19.7	26	Répartition : <table><tr><td>5.3</td><td>2.7</td><td>15</td><td>17</td><td>26</td></tr><tr><td>B</td><td>C</td><td>B</td><td>C</td><td>D</td></tr><tr><td>C</td><td>D</td><td>C</td><td>B</td><td>B</td></tr><tr><td>D</td><td>B</td><td>D</td><td>D</td><td>C</td></tr></table> Aucun candidat n'est élu. Le candidat C, ayant le moins de suffrages, est éliminé et ses voix (19.7) sont transférées équitablement aux seconds de chaque liste où C était en tête.
3e tour	34	20.3+17 =37.3	0	26+2.7= 28.7	Répartition : <table><tr><td>5.3</td><td>2.7</td><td>15</td><td>17</td><td>26</td></tr><tr><td>B</td><td>D</td><td>B</td><td>B</td><td>D</td></tr><tr><td>D</td><td>B</td><td>D</td><td>D</td><td>B</td></tr></table> Le candidat B est alors élu.

Dans cet exemple, nous voyons que tous les électeurs ont placé tous les candidats sur leur bulletin de vote. Cela implique d'une part que tous les candidats ont été choisis et d'autre part que toutes les listes ont le même nombre de candidats. Que se passe-t-il

si un électeur n'a plus qu'un nom sur son bulletin de vote et que ce nom est élu ou éliminé ? Que se passe-t-il si seulement le nom de trois candidats apparait sur l'ensemble de bulletins alors que quatre doivent être élus ? Il n'existe pas de bonne solution. Dans le premier cas, les voix peuvent être soit perdues soit distribuées équitablement entre les autres listes. Remarquons que les votes ne peuvent être simplement supprimés, car le quotient électoral ne serait alors plus adapté au problème puisque le nombre de votes total diminuerait. Dans le second cas cela parait encore plus compliqué. Dans le cadre de ce travail, nous considérons un cas d'étude similaire à l'exemple présenté ci-dessus, où tous les électeurs classent tous les candidats par ordre de préférence.

2.3 Intérêts

Le mode de scrutin indique la méthode utilisée pour déterminer les gagnants d'une élection, il permet de passer du décompte des voix à la désignation des élus. Bien que le principe d'élection au suffrage universel (chaque individu possède une voix) soit appliqué dans toutes les démocraties représentatives, ce n'est pas le cas du mode de scrutin. Sachant qu'il existe trois grandes familles de scrutins (les scrutins majoritaires, proportionnels et mixtes), le choix de celui-ci dépend de toute une série de facteurs comme : l'histoire politique nationale, les besoins de représentativités, l'opinion des partis,...

Le scrutin majoritaire est celui appliqué en France ou en Grande-Bretagne. Il donne le ou les sièges vacants aux candidats ou à la liste électorale ayant obtenu la majorité des voix. Ce type de scrutin a tendance à rendre les victoires catégoriques, sans concession et permet ainsi de désigner une majorité stable en mesure de gouverner. Il est également facile à mettre en œuvre et facile à comprendre. Son principal défaut est de ne pas représenter correctement le corps électoral. Il a tendance à exclure les partis minoritaires, ce qui empêche de nombreux partis de se faire connaitre. C'est une véritable barrière à l'entrée dans l'arène politique. Il est également très peu sensible au changement de l'opinion publique. Un parti peut passer de 80% de taux de préférence à seulement 60% sans pour autant en ressentir les effets. Ce manque de sensibilité a généralement pour conséquence un faible entrain des électeurs à aller voter. Le système majoritaire pénalise également les femmes. Souvent, dans ce système, il n'y a qu'un siège par circonscription à pourvoir. La compétition est donc rude et les partis font généralement moins confiance aux femmes. Ils préfèrent choisir des candidats connus, ce qui défavorise généralement les femmes et les jeunes.
Le scrutin proportionnel reflète le plus équitablement possible les diverses opinions des électeurs. Les sièges vacants sont répartis en divisant le nombre de voix associés à chaque liste par le quotient électoral. Il est utilisé en Belgique, en Italie, aux Pays-Bas,... Ce type de scrutin est considéré comme plus juste et plus démocratique que le précédent mais entraine une instabilité gouvernementale. Pour obtenir une majorité au parlement, les partis doivent le plus souvent s'unir pour former des coalitions. Les disputes sont plus fréquentes et des compromis doivent être faits. Les petits partis ont plus de facilité à entrer au gouvernement, une large proportion de la population est donc représentée. Ce système est fort sensible aux changements de l'opinion publique, les électeurs sont donc plus enclin à aller voter. Le système proportionnel donne également plus de chances aux femmes d'être élues. Dans les pays qui disposent d'un tel système, les partis accueillent plus de femmes sur leurs listes dans le but d'attirer les électeurs. Les 10 pays qui ont les pourcentages les plus élevés de femmes au parlement ont opté pour une représentation proportionnelle.

Le scrutin mixte est un mélange des deux précédents. Il essaie de maximiser les avantages des scrutins majoritaires et proportionnels et de minimiser leurs inconvénients. Ce type de scrutin est par exemple utilisé pour élire les membres du sénat en France. Dans les départements fort peuplés, les membres sont élus au scrutin majoritaire tandis que dans les départements moins peuplés, ceux-ci sont élus au scrutin proportionnel. Il existe toute une série de formes de scrutins mixtes avec leurs avantages et inconvénients qui seraient trop longs à expliquer ici. Le lecteur est invité à consulter [5] pour de plus amples informations.

Notons encore que le système majoritaire est connu pour faciliter le vote pour des candidats particuliers alors que le système proportionnel est connu pour faciliter le vote pour un parti. Il est cependant intéressant de noter que, par exemple en Belgique, on tente de combler ce fossé. L'électeur peut voter de différentes manières sur son bulletin. Il ne peut certe pas voter pour des listes différentes et des candidats de différentes listes, mais il peut tout de même faire le choix de voter pour une liste dans son ensemble ou effectuer un vote de préférence, c'est-à-dire voter pour un candidat en particulier. Les sièges sont attribués à chaque liste en fonction du résultat global obtenu par la liste (votes de préférence et votes en tête de liste). La distribution des sièges se fait de façon proportionnelle, ce qui signifie que la liste qui bénéficie du plus de voix recevra le plus de sièges. Par contre, pour savoir qui sera élu au sein de chaque liste, on comptabilise en premier lieu uniquement les votes de préférence obtenus par chaque candidat. Ceux qui ont reçu un nombre de votes de préférence suffisant pour occuper un siège sont élus. A ce moment-là, s'il reste des sièges à pourvoir au sein de la liste, on fait intervenir les votes en tête de liste. Ceux-ci s'ajoutent alors aux votes de préférence des candidats dans l'ordre proposé sur le bulletin de vote. Concrètement, cela signifie que si le premier candidat dispose de 5000 voix de préférence et qu'il lui en faut 6000 pour siéger, 1000 votes provenant du groupe des votes en tête de liste lui seront attribués afin qu'il obtienne le chiffre d'éligibilité nécessaire pour occuper un siège. Le deuxième candidat, lui, a obtenu 4500 voix de préférence. Il lui manque donc encore 1500 voix pour être élu. 1500 votes issus du groupe des votes en tête de liste vont donc lui être attribués et lui permettre de siéger. Et ainsi de suite, suivant l'ordre de la liste jusqu'à ce que tous les sièges attribués à la liste soient occupés.

La théorie du choix met en évidence de nombreuses méthodes pour élire des candidats lors d'une élection. Nous pouvons citer la pluralité qui consiste à demander aux électeurs de voter pour un candidat unique et élire le candidat ayant le plus de voix. Dans un autre style, le "Borda count" est une méthode où chaque électeur classe les candidats par ordre de préférence. Les candidats en tête de préférence reçoivent plus de points que les candidats aux deuxièmes, troisièmes, ... places. Nous nous attardons sur le "Borda Count", car ce type de scrutin semble nettement plus facile à comprendre et à mettre en œuvre tout en étant fort semblable au STV. Le "Borda Count" ne possède pas certaines qualités du STV. En Irlande du Nord, un Institut de "Borda Count" veut faire adopter ce système pour les élections politiques, mais ses adversaires prétendent qu'il est trop facile de fausser le système en votant de manière stratégique, ce qui peut mener à des résultats catastrophiques. Par exemple, un électeur pourrait voter de la manière suivante :

Premier choix : Son candidat préféré ;

Choix suivants : Tous les idiots, les incompétents et les extrémistes dangereux pour lesquels personne ne devrait rationnellement voter ;

Dernier choix : Le candidat conciliant qui devrait logiquement être le deuxième choix

de tout le monde.
Avec le STV, on ne peut pas aider son candidat préféré en modifiant l'ordre des candidats qu'on place en-dessous. Selon le théorème de Gibbard–Satterthwaite, aucun système ne peut éliminer le vote stratégique, mais les seules stratégies efficaces avec le STV sont de placer en tête du bulletin de vote un candidat de second choix dans le but que le résultat final soit moins mauvais que le résultat anticipé. Le "Borda Count" fonctionne très bien dans le cas d'électeurs involontaires mais sincères comme par exemple : « Ma position m'oblige à participer à la sélection d'un premier prix pour tel concours. Les candidats me semblent tous méritants. Selon ma modeste opinion, voici l'ordre dans lesquels ils se rangent »mais peut être dangereux quand les électeurs sont trop impliqués.

Poursuivons avec le STV. Aucune des méthodes précédentes n'apporte les avantages du STV. Ce mode de scrutin a la prétention de pouvoir concilier les deux objectifs suivants :

— il doit être possible de voter pour un ou des candidat(s) particulier(s) et non pour un parti dans son ensemble ;
— le nombre d'élus doit correspondre à une répartition proportionnelle.

Le STV est le seul système à l'heure actuelle à répondre pleinement à ces deux objectifs, de proportionnalité et de choix des élus, qui sont généralement perçus comme incompatibles.
De plus, alors que les systèmes de Borda count, de la pluralité et autres possèdent déjà des implémentations cryptographiques (c'est-à-dire qu'ils peuvent déjà avoir lieu de manière électronique sans qu'aucune fraude ne vienne empêcher leurs bons déroulements), le STV ne possède aucune solution cryptographique valable bien que le système vVote soit en bonne voie. Une solution efficace doit encore être trouvée.

Présentons encore les idées principales décrites par le mathématicien Canadien Claude Tardif qui met en évidence les avantages et inconvénients du STV de manière plus explicite (voir [4]). Notons que nous avons pu discuter par e-mails du contenu des différents paragraphes suivants.

Le vote unique transférable est sans doute le plus sophistiqué des modes de scrutins. Il fait partie de la famille des scrutins proportionnels dans le sens où il permet un choix entre les partis et les candidats des partis. Il est particulièrement utile pour les élections communales, car il permet à n'importe qui d'être élu. Il n'est plus nécessaire d'appartenir à un parti politique. Actuellement, il est utilisé dans ce cadre en Écosse et en Nouvelle-Zélande. *« Pendant la première moitié du vingtième siècle, il fut aussi utilisé dans une vingtaine de villes des États-Unis, où il a permis à des minorités ethniques d'être représentées pour la première fois au conseil municipal. »*. En effet, alors qu'avec un type de scrutin ordinaire, le vote des minorités ethniques peut avoir un effet négligeable, le vote unique transférable permet à ces minorités de tirer profit d'affinités communes. *« A New York, Cincinnati, Hamilton et Toledo, les premiers conseillers noirs furent élus à l'époque où ces villes utilisaient le vote unique transférable. Ce fut le cas aussi pour les catholiques irlandais de la ville d'Ashtabula. »*

Comme on peut s'en douter, permettre aux minorités d'accéder à des niveaux de pouvoirs de haute importance n'a pas toujours été bien accueilli. Vers la moitié du 20)*e*

siècle, la peur d'avoir un jour un maire noir a gagné les Blancs et de nombreuses manifestations ont eu lieu pour revenir à un mode de scrutin pluraliste. En de nombreux endroits, le vote à voix unique transférable a été aboli. Seule, la ville de Cambridge Massachussets a conservé le vote à voix unique transférable jusqu'à nos jours. Le politologue Douglas J. Amy, expert en système de vote électoral, est catégorique, c'est le vote unique transférable plus que toute autre chose qui avait permis aux minorités de se faire représenter. La politologue Kathleen L. Barber, ancienne directrice du département de sciences politiques de l'université de John Carroll University (Ohio) met essentiellement en avant la capacité du STV à prendre en compte l'évolution des mentalités.

Le vote à voix unique unique transférable est également utilisé pour les élections nationales en Irlande et en Australie. *« En Australie, on utilise le vote par élimination pour la chambre des représentants, et le vote à voix unique transférable pour le sénat. »* Cette différence de mode de scrutin à la chambre et au sénat permet de comparer le nombre de femmes au gouvernement : 35% de femmes au sénat contre 25% à la chambre des représentants. Ce n'est pas surprenant car, comme nous l'enseigne Claude Tardif, les femmes sont généralement plus nombreuses dans les pays utilisant des scrutins proportionnels. Notons cependant que la proportion de femmes dans le gouvernement d'un pays n'est pas influencée uniquement par le type de scrutin ; la culture du pays joue également un rôle. C'est pourquoi le cas de l'Australie est intéressant : deux types de politiques sont utilisées. Le docteur en politique, Dennis Pilon, réfute, via cet exemple et bien d'autres, la critique selon laquelle le vote à voix unique transférable empêche les femmes d'être représentées.
En Irlande, l'élection générale du 25 février 2011 illustre bien l'aspect stratégique du vote à voix unique transférable. Le parti « Fianna Fáil » avait toujours formé la majeure partie des gouvernements, seul ou en coalition. Mais, peu avant l'élection, la mise en œuvre de certaines mesures pendant la récession le discrédita parmi les partisans des autres partis qui l'ont donc généralement classé très bas dans leur liste de préférence. *« La représentation du parti « Fianna Fáil » a alors chuté de 77 représentants sur 166 en 2007, à seulement 20 en 2011, avec 17% des votes de première préférence. »*

Ces différents exemples présentés par Mr. Tardif illustrent de manière flagrante l'importance des minorités dans le STV. Considérons encore la Belgique et plus particulièrement le cas de la Wallonie où est utilisé un système proportionnel. Deux grands partis fondamentalement opposés (le Mouvement Réformateur (MR) et le Parti Socialiste (PS)) font parties du paysage politique Wallon. Cependant, de manière générale, ceux-ci n'obtiennent pas la majorité des voix (50%) nécessaire pour gouverner. Il doivent alors se tourner vers des partis moins influents dans le but de former une coalition. Les partis n'étant pas repris dans la majorité forment l'opposition. Les conséquences d'une telle méthode sont que des partis ayant reçu très peu de voix peuvent se retrouver dans la majorité alors que des partis possédant un nombre de voix plus élevé ne s'y retrouvent pas. Les partis ont une grande marge de manœuvre quant à l'élaboration du gouvernement. L'opinion publique n'est donc pas représentée de manière optimale. Étant moi-même belge et côtoyant de nombreux belges, j'entends régulièrement des citoyens penser que voter pour des petits partis n'est pas intéressant, car leurs voix seraient ainsi « perdues ». Les petits partis seront quand même « repêchés »par les plus gros, à la guise de ceux-ci. Ils préfèrent donc voter pour un des partis dominants (MR ou PS) pour essayer d'avoir celui qui représente plus ou moins leurs opinions afin de ne pas se retrouver avec celui qui ne les représente pas du tout. Si la constitution belge optait pour le STV permettant

ainsi aux électeurs de classer les parti par ordre de préférence, nous pourrions probablement observer une remontée spectaculaire des plus petit partis. Il serait cependant encore plus difficile d'obtenir un gouvernement car, les coalisions devraient contenir un nombre de partis conséquent . Tout ceci montre bien les différents intérêts qu'apporte le vote à voix unique transférable.

2.4 Objectifs du travail

Jusqu'ici, nous avons montré le fonctionnement et l'intérêt du STV. Présentons maintenant les défis que ce travail va tenter de relever. Supposons que nous voulons utiliser le STV dans le cadre d'un vote en ligne et non plus sur papier. Plusieurs problèmes démocratiques surviennent alors. Premièrement, l'anonymat des votes ne va plus être respecté. N'importe qui pourra intercepter le vote d'un électeur et examiner celui-ci. Or, l'anonymat est un droit élémentaire et nécessaire car, comme le montre un certain nombre d'études, beaucoup d'individus ont tendance à adopter le même comportement électoral que leurs amis, leurs voisins, ... L'anonymat empêcherait les électeurs de savoir comment leur entourage a voté et leur permettrait donc de voter en âme et conscience. Une solution pour garder ce droit est de crypter le vote de l'électeur. Cependant, dans le cadre de STV, ce n'est pas évident, car pour pouvoir transférer les votes aux candidat de second choix, nous avons besoin de savoir quel est l'ordre de classement des candidats pour chaque électeur. Les votes cryptés deviennent dès lors un problème que nous tenterons de résoudre au décomptage des voix. Deuxièmement, si nous supposons que nous avons réussi à crypter les votes, il faut être sûr que personne n'émette de faux votes, c'est-à dire plusieurs votes au lieu d'un seul ou un seul vote contenant plusieurs voix. Troisièmement, remarquons que plus il y a de candidats en liste, plus le scrutin d'un électeur risque d'être unique et donc repérable parmi tous les autres, même crypté. Cela peut mener à des abus du type : « je te donne de l'argent et en échange, tu votes de telle façon », ce qui est totalement antidémocratique et inacceptable. Les votes pratiqués en Australie et en Irlande qui utilisent justement le STV sont fortement soumis à ce genre de problème. Finalement, il faut pouvoir être certain que tous les électeurs ont voté et ce, une seule et unique fois.

La plupart de ces problèmes sont inhérents à tous les votes électroniques. Dans un premier temps, nous nous approprierons donc les méthodes cryptographiques pré-établies dans ce contexte : nous développerons un moyen de chiffrer et décompter les votes de manière à garder un secret maximal sur le contenu des scrutins (chapitre 3), de développer un schéma de preuves permettant de certifier que les bulletins sont valides (le message caché appartient bien à l'ensemble des résultats attendus) et n'ont pas été modifiés entre le moment du vote de l'électeur et celui du dépouillement (chapitre 4), de développer un schéma de preuve permettant à tout le monde de vérifier que le résultat final est bien correct (chapitre 5) et enfin de développer un schéma de preuve permettant de rendre inutile l'achat de votes (chapitre 6). Dans un deuxième temps, nous développerons, à partir de ce qui aura été présenté jusque-là, notre propre méthode adaptée au STV permettant de transférer les votes valides à différents candidats selon leur ordre de préférence sur les listes des électeurs sans avoir besoin de déchiffrer ceux-ci (chapitre 7).

Chapitre 3

Mode de chiffrement

Tous les résultats, définitions et schémas algorithmiques liés à la cryptographie présents dans les sections (3.1) et (3.2) sont tirés de [1] et [6]. Les résultats de la section (3.3) sont tirés de [7] et [8].

Dans ce chapitre, nous commençons par décrire les principes fondamentaux de la cryptographie moderne pour ensuite nous attaquer à la construction du chiffrement Paillier que nous utiliserons pour chiffrer les votes. Nous expliquons également pourquoi nous avons choisi ce chiffrement.

3.1 Introduction à la cryptographie moderne

Avec l'apparition de l'informatique, la cryptographie a subi un véritable lifting. Dès le début des années 1970, des chiffrements de plus en plus complexes et sûrs, ne pouvant désormais plus être exécutés à la main, ont vu le jour. Malgré ces avancées, un problème subsiste : comment faire pour communiquer la clef du chiffrement ? Il est impensable d'exiger que deux personnes voulant communiquer secrètement ensemble doivent préalablement se rencontrer physiquement... C'est ce qu'on appelle *le problème de la distribution de clef*. Ce n'est qu'en 1976 qu'une vraie solution fut trouvée par Diffie et Hellman : la méthode de la clef publique.
Un individu publie de quelque manière que ce soit une clef dite publique notée pk, et garde pour lui une clef dite secrète notée sk. Quiconque voulant envoyer un message secret à l'individu, chiffre son message avec la clef pk et l envoie à l'individu. L'individu déchiffre alors le message à l'aide de la clef secrète sk. L'unique moyen théorique de déchiffrer ce message chiffré à l'aide de la clef publique pk est d'utiliser la clef secrète sk associée. Il existe donc un lien mathématique entre ces deux clefs : une fonction dite à sens unique, c'est-à-dire qu'il est aisé de calculer, à partir de la sk, la pk, mais le calcul inverse est relativement impossible. Les chiffrements à clefs publiques, aussi appelés chiffrements asymétriques, s'opposent aux chiffrements dits symétriques où l'émetteur et le récepteur du message possèdent la même clef. Les avantages d'avoir deux clefs différentes sont totalement essentiels dans le cadre du vote électronique où des milliers d'électeurs vont être amenés à chiffrer leur vote. Le fait d'avoir deux clefs distinctes pour chiffrer et déchiffrer va nous permettre de fournir une clef unique de chiffrement à tous les électeurs. Cette méthode enlève tous les problèmes de distributions de clefs : "comment faire parvenir à chaque électeur sa clef secrète ?" et enlève également tous les problèmes de stockage de ces clefs : "comment savoir quelle clef appartient à quel électeur ?".

Présentons à présent le schéma de chiffrement à clef publique de manière formelle.

Schéma 3.1.1 *Chiffrement à clef publique*

Soit un triplet <Gen,Enc,Dec> d'algorithmes probabilistes à temps polynomial (PPT) :

- ***Gen :*** *sélectionne (pk,sk)←Gen(1^n) (n paramètre de sécurité).*
- ***Enc :*** *fournit $c \leftarrow Enc_{pk}(m)$ (m message à chiffrer).*
- ***Dec :*** *fournit $m := Dec_{sk}(c)$.*

tel que, $\exists$ une fonction négligeable ϵ : $\forall n, (pk, sk) \leftarrow Gen(1^n)$, and $\forall m$:

$$Pr[Dec_{sk}(Enc_{pk}(m)) \neq m] < \epsilon(n).$$

Définition 3.1.1

Une fonction $\epsilon(.)$ est dite négligeable si pour tout polynôme $p(.)$ il existe un N tel que pour tout entier $n > N$, $\epsilon(n) < \frac{1}{p(n)}$.

Par exemple, les fonctions 2^{-n}, $2^{-\sqrt{n}}$ *et* $n^{-\log n}$ sont négligeables.

Nous avons **Gen** un algorithme qui va générer les clefs secrètes et publiques dont les longueurs seront fonctions du paramètre de sécurité n donné en entrée. La façon dont ces clefs sont générées sera donnée ultérieurement. Les algorithmes **Enc** et **Dec** vont respectivement permettre de chiffrer et déchiffrer le message m à l'aide des clefs générées par **Gen** ; ils seront également décrits ultérieurement. La probabilité que le déchiffrement à l'aide de la bonne clef secrète du chiffré c de m ne donne pas m doit être négligeable.

La science qui étudie les techniques permettant de déchiffrer un message sans posséder la clef de déchiffrement s'appelle la cryptanalyse. Elle consiste à tenter d'attaquer un système cryptographique afin d'étudier le niveau de sécurité effectif. Il existe différents types d'attaques. La plus intuitive est sans doute l'attaque de force brute qui consiste à essayer toutes les clefs secrètes admissibles jusqu'à trouver la bonne clef de déchiffrement. Plus la taille de la clef est grande, moins l'attaque de force brute a de chance de succès. Il faut donc une taille de clef qui permette aux schémas de chiffrements d'être facilement utilisables tout en résistant aux attaques de force brute. La cryptographie moderne a fait le compromis de permettre aux schémas de chiffrements d'être cassables avec une probabilité négligeable ; c'est-à-dire qu'un attaquant finira par trouver la clef secrète après un temps tellement grand (par exemple 30 ans) que le message chiffré sera devenu obsolète.

Il existe également des techniques nettement plus sophistiquées que l'attaque par force brute, qui tentent de prendre avantage des faiblesses mathématiques du chiffrement. Le chiffrement doit donc pouvoir résister à différents niveaux d'attaques. Nous pouvons dégager quatre scénarios :

1. **Ciphertext-only attack :** l'adversaire observe un ou plusieurs chiffrés et essaye de déterminer le message sous-jacent.

2. **Known-plaintext attack :** l'adversaire a en sa possession une ou plusieurs paires de textes clairs/chiffrés. En les analysant, il va essayer de déterminer le message sous-jacent à un chiffré n'apparaissant pas dans les paires qu'il possède.

3. **Chosen-plaintext attack (CPA) :** l'adversaire a la possibilité de demander le chiffrement de presque n'importe quel message clair et va essayer de déterminer le message sous-jacent à un chiffré pour lequel il n'a pas pu demander le chiffrement.

4. **Chosen-ciphertext attack (CCA) :** l'adversaire a la possibilité de demander le déchiffrement de n'importe quel message chiffré et va essayer de déterminer le message sous-jacent d'un autre chiffrement pour lequel il n'a pas pu demander le déchiffrement.

Le scénario 1 est le plus basique tandis que le 4 est le plus poussé. Ils sont tous réalistes et divers cas d'histoire peuvent en témoigner. L'anecdote suivante est un bon exemple de CPA. Durant la deuxième guerre mondiale, en mai 1942, les cryptanalystes de la Navy américaine ont découvert que les Japonais planifiaient d'attaquer les îles Midway situées à mi-chemin entre les Etats-Unis et le Japon. Ils ont découvert cette information en interceptant un message codé des Japonais où apparaissaient les initiales « AF ». La Navy américaine pensait que ces initiales correspondaient aux îles Midway, mais ils n'avaient pas de preuves et le gouvernement américain n'en croyait pas un mot. La Navy mit alors un plan pour convaincre ses supérieurs. Elle envoya un message non crypté disant que la flotte américaine était très faible dans les environs de l'île. Les Japonais ont intercepté le message et ont immédiatement prévenu leurs supérieurs dans un message crypté que les alentours de « AF »étaient sans protection. La preuve était faite ! Les Américains ont pu sauver les îles et la guerre a pris un tout autre tournant. Cette manœuvre est typiquement une CPA.

Dans le cadre du vote électronique, être « CCA-secure » n'est pas nécessaire, car seul le résultat final du vote sera rendu public, aucune information sur le déchiffrement d'un vote particulier ne sera disponible. Être « CPA-secure » est par contre essentiel, car tout un chacun connaissant la clef publique peut chiffrer le message de son choix. Définissons donc de manière formelle les notions de chiffrement « CPA-secure ».

Schéma 3.1.2 $PubK^{cpa}_{\mathscr{A},\Pi}(n)$

Soit un schéma de chiffrement à clef publique Π = <Gen,Enc,Dec> *et un adversaire à temps polynomial* $\mathscr{A}$ *:*

1. *Gen*(1^n) *fournit (pk,sk).*

2. *L'adversaire* $\mathscr{A}$ *reçoit pk et a accès à l'oracle* $Enc_{pk}(.)$ *autant de fois qu'il le souhaite. Il ressort alors une paire de messages* m_0, m_1 *de même longueur pour lesquels il ne peut demander le chiffrement.*

3. *Un bit aléatoire* $b \leftarrow \{0,1\}$ *est choisi et le chiffrement* $c \leftarrow Enc_{pk}(m_b)$ *est calculé et fourni à* $\mathscr{A}$*.*

4. $\mathscr{A}$ *continue d'avoir accès à l'oracle* $Enc_{pk}(.)$ *et ressort un bit b'*$\leftarrow \{0,1\}$*.*

5. $PubK^{cpa}_{\mathscr{A},\Pi}(n) = 1$ *si* $b' = b$ *et* 0 *sinon.*

Définition 3.1.2

Un schéma de chiffrement à clef publique Π = <Gen,Enc,Dec> *est « CPA-secure »si pour tout adversaire PPT* $\mathscr{A}$*, il existe une fonction négligeable* ϵ *telle que :*

$$Pr[PubK^{cpa}_{\mathscr{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n).$$

Ce schéma et cette définition considérés ensemble signifient qu'un adversaire $\mathscr{A}$ avec une puissance de calcul polynomial qui mène une attaque sur le chiffrement est incapable de choisir deux messages m_0 et m_1 de même longueur tels que, lorsqu'il reçoit un chiffré c qui correspond soit au chiffrement de m_0 soit au chiffrement de m_1, il soit capable de déterminer si ce chiffré c est celui de m_0 ou m_1. Et ce, même si l'adversaire $\mathscr{A}$ a un accès illimité à l'oracle $Enc_{pk}(.)$, c'est-à dire qu'il peut demander le chiffrement de n'importe quel message (excepté celui de m_0 et m_1) à un tiers individu ou machine. Dans un schéma à clef publique, l'adversaire a forcément connaissance de la clef pk et a donc dans tous les cas un accès illimité à l'oracle $Enc_{pk}(.)$. Notre chiffrement devra donc être « CPA-secure ».

3.2 Le chiffrement Paillier

Nous avons montrer à la section précédente que nous avons besoin d'un chiffrement à clef publique qui soit « CPA-secure ». C'est le cas du chiffrement Paillier. Ce chiffrement repose sur un problème mathématique difficile à résoudre : la factorisation d'un nombre composite N (i.e. non premier) en un produit de deux nombres premiers p, q. Les notations N, p et q seront utilisées tout au long de ce travail sauf avis contraire. Pour construire le chiffrement Paillier, nous aurons besoin de quelques résultats préliminaires. Nous verrons également les différents avantages que nous apporte ce chiffrement pour l'élaboration de protocoles cryptographiques pour le vote électronique.

3.2.1 Quelques résultats préliminaires

Nous nous intéressons ici au groupe produit $\mathbb{Z}^*_{N^2}$ où $N = pq$, un produit de deux nombres premiers distincts de même longueur. Nous noterons $\phi(N) = (p-1) * (q-1)$ l'ordre du groupe $\mathbb{Z}^*_N$. Le nombre N constituera la clef publique et $\phi(N)$ la clef secrète du chiffrement de Paillier. Ces deux clefs sont les sorties de l'algorithme probabiliste à temps polynomial **Gen** décrit dans le schéma du chiffrement à clef publique (3.1.1). Cet algorithme doit donc générer des nombres premiers longs de n bits. Pour cela, nous allons générer jusqu'à t nombres longs de n bits et tester leurs primalités, comme le décrit l'algorithme suivant.

Algorithme 3.2.1 *Génération d'un nombre premier de longueur n bits*

Entrée : *Le paramètre de sécurité n et le paramètre t*
Sortie : *Un nombre premier de n bits avec une probabilité* $1 - 2^{-t}$.

for $i = 1$ *to* t : {
  $p' \leftarrow \{0,1\}^{n-1}$
  $p \coloneqq 1||p'$
  if p *est premier* ***ret*** p
}
ret *erreur*

Comme nous pouvons le voir, nous forçons le nombre p à être long de n bits en posant le bit de poids le plus fort à 1. Nous testons alors sa primalité ; si le nombre p est premier, l'algorithme a réussi et s'arrête. Si au bout de t itérations l'algorithme n'a pas généré de nombre premier, il s'arrête et ressort une erreur. La probabilité que l'algorithme ne marche pas est de 2^{-t} comme nous allons le découvrir au bout de cette discussion.
Il nous reste à développer une méthode pour déterminer si un nombre p est premier ou non. C'est un problème mathématique très vieux. La première solution fut développée par Eratosthène vers 200 avant Jésus-Christ. Elle consiste à essayer de diviser p par tous les nombres plus petits que lui. C'est une méthode évidemment très coûteuse en temps de calcul surtout qu'une division peut prendre jusqu'à 50 fois plus de temps qu'une addition. Cette solution peut être améliorée pour donner un algorithme plus efficace : le crible d'Eratosthène.

Algorithme 3.2.2 *Crible d'Eratosthène*

Entrée : *Un nombre p*
Sortie : *p est premier ou non*

1. *Noter les numéros 1,...,p. Nous éliminerons les nombres composés (non-premiers) en les marquant. Au commencement, aucun nombre n'est marqué.*
2. *Marquer le numéro 1 comme spécial (il n'est ni premier ni composé).*
3. *Initialiser k à 1. Tant que k n'excède pas la racine carrée de p, itérer :*
   - *Prendre le plus petit nombre m qui soit plus grand que k et qui n'a pas été marqué (à la première itération, on trouve 2). Marquer tous ses multiples comme nombres composites (dans le premier passage nous marquons ainsi tous les nombres pairs plus grands que 2, dans le deuxième passage nous marquons tous les multiples de 3 plus grands que 3, etc.).*
   - *m est premier.*
   - *Affecter k à m et recommencer.*
4. *Tous les nombres restants non marqués sont aussi premiers et donc si p est non marqué, il est premier.*

Cet algorithme consiste en le calcul d'une racine carrée et d'une succession d'additions. Cependant, il reste inutilisable dans notre contexte. En effet, sachant que la taille de la clef d'un chiffrement à clef publique vaut généralement au moins 1024 bits et au vu de la table 3.1, le crible d'Eratosthène est inutilisable. Notons que l'univers existe depuis environ 10^{18} secondes...

# de bits de p	temps d'exécution de l'algorithme (s)	Mémoire utilisée
20	$\cong 10^0$	$\cong 10^1$ Mo
24	$\cong 10^1$	$\cong 10^2$ Mo
28	$\cong 10^2$	$\cong 10^3$ Mo
1024	$\cong 10^{300}$	$\cong 10^{301}$ Mo

TABLE 3.1 – Approximations tirées de [9]

Il nous faut donc une autre méthode. L'un des meilleurs tests de primalité est celui de Miller et Rabin publié en 1976. Il repose sur la proposition suivante :

Proposition 3.2.1

Soit $\mathbb{G}$ un groupe fini d'ordre m. Alors, pour tout élément $g \in \mathbb{G}$, $g^m = 1$.

Si p est un nombre premier alors le groupe $\mathbb{Z}_p^*$ sera d'ordre $p-1$ et donc,

$$\forall a \in \{1, ..., p-1\}, a^{p-1} = 1 \ mod \ p.$$

Par conséquent, si

$$\exists a \in \{1, ..., p-1\} : a^{p-1} \neq 1 \ mod \ p,$$

alors p est composite.
Nous considérons dès lors que tout nombre p est premier jusqu'à ce que nous trouvions un $a \in \mathbb{Z}_p^*$ tel que $a^{p-1} \neq 1 \ mod \ p$. En testant un nombre relativement important de a, nous pourrons déterminer avec une erreur négligeable si p est premier ou non. Nous pouvons résumer tout ceci par l'algorithme suivant.

Algorithme 3.2.3 *Test probabiliste de primalité*

Entrée : *Un nombre entier p et un paramètre t*
Sortie : *p est premier ou non*

for $i = 1 \ to \ t : \{$
$a \ \leftarrow \ \{1, ..., p-1\}$
if $pgcd(a, p) \neq 1$ ***ret*** *composite*
if $a^{p-1} \neq 1 \ mod \ p$ ***ret*** *composite*
$\}$
ret *erreur*

Le fait de poser $pgcd(a, p) \neq 1$ nous assure que $a \in \mathbb{Z}_p^*$. La proposition suivante nous donne une idée de l'efficacité d'un tel algorithme.

Proposition 3.2.2

Soit p un nombre composite. Supposons que :

$$\exists b \in \{a : a^{p-1} \neq 1 \ mod \ p\} \cap \mathbb{Z}_p^*$$

Alors

$$|\{a : a^{p-1} \neq 1 \ mod \ p\} \cap \mathbb{Z}_p^*| \geq \frac{|\mathbb{Z}_p^*|}{2}$$

Cela signifie que, pour un nombre composite p, la probabilité que, si $\exists b \in \{a : a^{p-1} \neq 1 \ mod \ p\} \cap \mathbb{Z}_p^*$, nous trouvions à chaque itération de l'algorithme ci-dessus, un élément $b \in \{a : a^{p-1} \neq 1 \ mod \ p\}$ ou un élément qui n'est pas dans $\mathbb{Z}_p^*$, vaut au moins :

$$\frac{\frac{|\mathbb{Z}_p^*|}{2} + ((p-1) - |\mathbb{Z}_p^*|)}{p-1} = 1 - \frac{|\mathbb{Z}_p^*|/2}{p-1} \geq 1 - \frac{|\mathbb{Z}_p^*|/2}{|\mathbb{Z}_p^*|} = \frac{1}{2}.$$

L'algorithme déclare donc un nombre premier comme étant premier avec une probabilité de 1 et déclare un nombre composite comme étant premier avec une probabilité d'au plus 2^{-t}, ce qui est négligeable.

Le problème est qu'il n'existe pas, pour tout nombre composite p, d'élément $b \in \{a : a^{p-1} \neq 1 \ mod \ p\} \cap \mathbb{Z}_p^*$. Ces nombres sont dis de Carmichael (exemple : 561, 1105, 1729,...). Ces nombres sont composites et vérifient la propriété :

$$\forall a \in \{1, ..., p-1\}, a^{p-1} = 1 \ mod \ p,$$

ce qui les rend indistinguables des nombres premiers par la méthode décrite ci-dessus. L'algorithme de Miller-Rabin va donc utiliser d'autres méthodes de détection en plus de

celle décrite précédemment. Ceci dépasse le cadre de ce travail, mais le lecteur intéressé est invité à consulter [1, pages 266 à 271].
Ajoutons simplement que l'algorithme de Miller-Rabin est très efficace comme le montre la figure (3.1). Le nombre de chiffres représente le nombre de chiffres dans la clef publique.

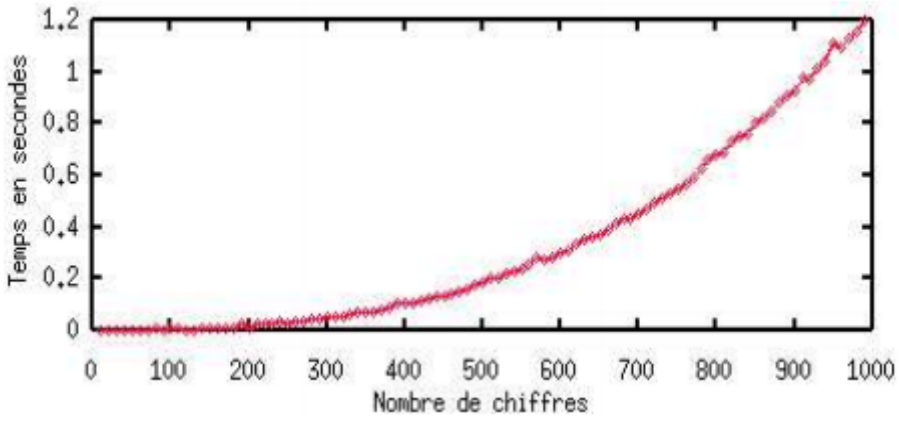

FIGURE 3.1 – : Test de Miller-Rabin sur un Pentium III à 735 MHz [9].

Nous avons donc donné une méthode efficace pour générer des nombres premiers de même longueur et donc un algorithme efficace pour calculer les clefs publique et secrète d'un chiffrement à clef publique. Nous appellerons cet algorithme **GenModulus**. Il prend en entrée une suite de bits de longueur n et sort un triplet (N, p, q) tel que $N = pq$. Il est évident qu'il ne doit pas être possible de retrouver la clef secrète à partir de la clef publique. C'est-à-dire qu'il doit être difficile de factoriser N en p et q, c'est ce qui s'appelle l'hypothèse de factorisation. Le simple fait de calculer p et q pour déduire N, comme le fait **GenModulus**, rend le problème de factorisation difficile à résoudre. En effet, ce problème a été étudié pendant des centaines d'années (bien avant la cryptographie moderne) et aucun algorithme efficace n'a pu être trouvé.

Dégageons à présent une propriété particulièrement intéressante du groupe $\mathbb{Z}^*_{N^2}$.

Théorème 3.2.1

Soit $N = pq$ où p, q deux nombres premiers impairs distincts de même longueur. Alors $f : \mathbb{Z}_N \times \mathbb{Z}^*_N \longrightarrow \mathbb{Z}^*_{N^2}$ tel que $f(a, b) = [(1+N)^a b^N \ mod\ N^2]\ \forall a \in \mathbb{Z}_N, \forall b \in \mathbb{Z}^*_N$ est un isomorphisme de groupe.

Preuve :

1) $f(a, b) = (1 + N)^a b^N \ mod\ N^2 \in \mathbb{Z}^*_{N^2}$ car tous les facteurs sont premiers avec N^2. En effet, $\mathbb{Z}^*_{N^2} \stackrel{def}{=} \{a \in \{1, ..., N^2 - 1\} |\ pgcd(a, N^2) = 1\}$ et $pgcd((1 + N), N^2) = 1$ ainsi que $pgcd(b, N^2) = 1$.

2) Montrons que f est un isomorphisme de groupe entre $\mathbb{Z}_N \times \mathbb{Z}^*_N$ et $\mathbb{Z}^*_{N^2}$. Autrement dit, montrons que f un morphisme bijectif ou encore que f est bijective et que $f(a_1, b_1) \cdot f(a_2, b_2) = f(a_1 + a_2, b_1 \cdot b_2)\ \forall a_1, a_2 \in \mathbb{Z}_N$ et $\forall b_1, b_2 \in \mathbb{Z}^*_N$.

a) f est une bijection. L'ordre du groupe de $\mathbb{Z}_{N^2}^*$ est :

$$\begin{aligned} |\mathbb{Z}_{N^2}^*| = \phi(N^2) \overset{Prop3.2.3}{=} p\cdot(p-1)\cdot q\cdot(q-1) &= pq\cdot(p-1)(q-1)(= N\cdot\phi(N)) \\ &= |\mathbb{Z}_N|\cdot|\mathbb{Z}_N^*| = |\mathbb{Z}_N\times\mathbb{Z}_N^*| \end{aligned}$$

Proposition 3.2.3

Soit $N=\prod_i p_i^{e_i}$ où $\{p_i\}$ un ensemble de nombres premiers et $e_i\geq 1$. Alors $\phi(N)=\prod_i p_i^{e_i-1}(p_i-1)$.

Il reste donc à montrer que f est injective.
Soit $a_1,a_2\in\mathbb{Z}_N$ *et* $b_1,b_2\in\mathbb{Z}_N^*$ tels que $f(a_1,b_1)=f(a_2,b_2)$. Alors,

$$\begin{aligned} (1+N)^{a_1}b_1^N &= (1+N)^{a_2}b_2^N \ mod\ N^2 \\ \Leftrightarrow (1+N)^{a_1-a_2}(\frac{b_1}{b_2})^N &= 1\ mod\ N^2 \qquad (\star) \end{aligned}$$

Remarque : Par la Proposition 3.2.4 et le fait que $pgcd(b_2,N^2)=1$, b_2^{-1} existe.

Proposition 3.2.4

Soit a,N deux entiers avec $N>1$. Alors, a est inversible modulo $N\Leftrightarrow pgcd(a,N)=1$.

Mettons en puissance $\phi(N)$ chaque côté de l'équation $(\star)$ en remarquant que $\phi(N^2)=N\phi(N)$. Nous obtenons alors :

$$\begin{aligned} (1+N)^{(a_1-a_2)\phi(N)}(\frac{b_1}{b_2})^{N\phi(N)} &= 1\ mod\ N^2 \\ \overset{Prop3.2.1}{\Rightarrow}(1+N)^{(a_1-a_2)\phi(N)} &= 1\ mod\ N^2\ (=(1+N)^0\ mod\ N^2) \end{aligned}$$

Notons à présent que l'ordre de l'élément $(1+N)$ dans $\mathbb{Z}_{N^2}^*$ est N.

En effet, par la proposition suivante :

Proposition 3.2.5

Soit $N=pq$ où p,q deux nombres premiers impairs distincts de même longueur. Alors $\forall a\in\mathbb{N},(1+N)^a=(1+aN)\ mod\ N^2$.

nous avons que, avec $a=N$, $(1+N)^N=1+N^2\ mod\ N^2=1\ mod\ N^2$ mais aussi que $(1+N)^a\neq 1\ mod\ N^2\ \forall 1\leq a<N$.

En nous servant de la propriété suivante :

Proposition 3.2.6

Soit $\mathbb{G}$ un groupe fini et $g\in\mathbb{G}$ un élément d'ordre i. Alors, $g^x=g^y\Leftrightarrow x=y\ mod\ i$.

nous pouvons écrire

$$(a_1-a_2)\phi(N)=0\ mod\ N.$$

Ce qui signifie que N divise $(a_1-a_2)\phi(N)$ mais comme $pgcd(N,\phi(N))=1$, N divise (a_1-a_2). Or comme $a_1,a_2\in\mathbb{Z}_N$, nous avons que

$$a_1=a_2. \qquad (\star\star)$$

En mettant ensemble ($\star$) et ($\star\star$), nous obtenons :

$$\begin{aligned}
(\frac{b_1}{b_2})^N &= 1 \ mod\ N^2 \\
\Rightarrow b_1^N &= b_2^N \ mod\ N^2 \\
\overset{N|N^2}{\Rightarrow} b_1^N &= b_2^N \ mod\ N \\
\overset{Prop3.2.7}{\underset{pgcd(N,\phi(N))=1}{\Rightarrow}} b_1 &= b_2 \ mod\ N \\
\overset{b_1,b_2\in\mathbb{Z}_N^*}{\Rightarrow} b_1 &= b_2.
\end{aligned}$$

Proposition 3.2.7

Soit $\mathbb{G}$ un groupe fini d'ordre $m > 1$, $e \in \mathbb{N}_0$ et $f_e : \mathbb{G} \longrightarrow \mathbb{G}$ tel que $f_e(g) = g^e$. Alors, si $pgcd(e, m) = 1$, f_e est une bijection.

Nous avons donc f surjective et injective et donc f est une bijection.

b) $f(a_1, b_1) \cdot f(a_2, b_2) = f(a_1 + a_2, b_1 \cdot b_2)$ $\forall a_1, a_2 \in \mathbb{Z}_N$ et $\forall b_1, b_2 \in \mathbb{Z}_N^*$.

Soit $a_1, a_2 \in \mathbb{Z}_N$ et $b_1, b_2 \in \mathbb{Z}_N^*$. Alors,

$$\begin{aligned}
f(a_1, b_1) \cdot f(a_2, b_2) &= ((1+N)^{a_1} b_1^N) \cdot ((1+N)^{a_2} b_2^N) \ mod\ N^2 \\
&= (1+N)^{a_1+a_2} (b_1 b_2)^N \ mod\ N^2 \\
&= (1+N)^{(a_1+a_2) \ mod\ N} (b_1 b_2)^N \ mod\ N^2
\end{aligned}$$

La dernière égalité s'obtient par la proposition ci-dessous et le fait que l'élément $1 + N$ est d'ordre N.

Proposition 3.2.8

Soit $\mathbb{G}$ un groupe fini et $g \in \mathbb{G}$ un élément d'ordre i. Alors, $\forall x \in \mathbb{Z}$, $g^x = g^{x \ mod\ i}$.

Remarquons que nous pouvons écrire $b_1 b_2 = r + \gamma N$ avec $\gamma, r \in \mathbb{Z}$ tel que $1 \leq r < N$ (nous avons bien $r \neq 0$ car comme $b_1, b_2 \in \mathbb{Z}_N^*$, $b_1 b_2 \neq 0 \ mod\ N$) et donc :

$$\begin{aligned}
(b_1 b_2)^N \ mod\ N^2 &= (r + \gamma N)^N \ mod\ N^2 \qquad \text{Théorème d'expansion binomial} \\
&= \sum_{k=0}^{N} \binom{N}{k} r^{N-k} (\gamma N)^k \ mod\ N^2 \\
&= r^N + N r^{N-1} \gamma N \ mod\ N^2 + ... + \gamma N^N \ mod\ N^2 \\
&= r^N \ mod\ N^2 \\
&= (b_1 b_2 \ mod\ N)^N \ mod\ N^2
\end{aligned}$$

Ce qui nous permet de conclure :
$f(a_1, b_1) \cdot f(a_2, b_2) = (1+N)^{(a_1+a_2) \ mod\ N} (b_1 b_2 \ mod\ N)^N \ modN^2 = f(a_1 + a_2, b_1 b_2)$.

∎

Cet isomorphisme va nous permettre de définir notre chiffrement comme $c = f(m, r)$ où m est le message à chiffrer et r un élément aléatoire. Comme nous l'avons vu à la section (3.1), un chiffrement doit pouvoir résister à différentes attaques. Dans le but de montrer que ce chiffrement a les qualités requises, présentons la notion de *résidu* N^{e} *mod* N^2 :

Définition 3.2.1

Soit $y \in \mathbb{Z}^*_{N^2}$. *On dit que* y *est un résidu* N^e *mod* N^2 *si* y *est une* N^e *puissance, i.e. si* $\exists x \in \mathbb{Z}^*_{N^2}$ *tel que* $y = x^N \ mod \ N^2$.
Nous noterons alors $y \in Res(N^2)$.

Caractérisons ce résidu :

- Soit $x \in \mathbb{Z}^*_{N^2}$ tel que $x \longleftrightarrow (a, b)$, $a \in \mathbb{Z}_N$, $b \in \mathbb{Z}^*_N$.
  Alors $x^N \ mod \ N^2 = f(a, b)^N \ mod \ N^2 = f(aN \ mod \ N, \ b^N \ mod \ N) = f(0, b^N \ mod \ N)$.

  $$\begin{aligned} \text{En effet, } \ f(a,b)^N \ mod \ N^2 &\overset{def}{=} [(1+N)^a b^N \ mod \ N^2]^N \ mod \ N^2 \\ &= (1+N)^{aN} b^{NN} \ mod \ N^2 \\ &\overset{(\star)}{=} (1+N)^{aN \ mod \ N} (b^N \ mod \ N)^N \ mod \ N^2 \\ &\overset{def}{=} f(aN \ mod \ N, b^N \ mod \ N). \end{aligned}$$

  ($\star$) Le passage de la ligne deux à la ligne trois est donné dans la démonstration du théorème (3.2.1) au point b.

- Soit $y \in \mathbb{Z}^*_{N^2}$ tel que $y \longleftrightarrow (0, b)$, $b \in \mathbb{Z}^*_N$,
  alors $y \in \ Res(N^2)$.

  En effet, nous devons trouver $x \in \mathbb{Z}^*_{N^2}$ tel que $y = x^N \ mod \ N^2$.
  Alors

  $$x = (a, b^d \ mod \ N) \text{ où } \begin{cases} a \in \mathbb{Z}_N \\ d = N^{-1} \ mod \ \phi(N) \end{cases}$$

  convient car

  $$x^N = (a, b^d \ mod \ N)^N = (Na \ mod \ N, b^{dN} \ mod \ N) = (0, b) \longleftrightarrow y.$$

  Remarque : Par la Proposition 3.2.4 et le fait que $pgcd(N, \phi(N)) = 1$, N^{-1} existe.

Nous avons donc que $Res(N^2) = \{(0, b) | b \in \mathbb{Z}^*_N\}$ et que l'ensemble des racines N^{e} d'un élément $y \longleftrightarrow (0, b) \in Res(N^2)$ est : $\{x = (a, b^d \ mod \ N), \ \forall a \in \mathbb{Z}_N, \ d = N^{-1} \ mod \ \phi(N)\}$. Un élément $y \in Res(N^2)$ a donc exactement N racines ($a = 0, ..., N - 1$).

Ce qui nous amène au « décisional composite residuosity problem »(DCR problem) qui est de distinguer un élément aléatoire de $\mathbb{Z}^*_{N^2}$ d'un élément aléatoire de $Res(N^2)$. De manière plus formelle, si nous considérons un algorithme à temps polynomial nommé « **GenModulus** »qui prend en entrée une suite de bits de longueur n et sort un triplet (N, p, q) tel que $N = pq$ alors :

Définition 3.2.2

Le DCR problem est difficile à résoudre relativement à l'algorithme ***GenModulus*** *si pour tout PPT algorithme $\mathscr{A}$, il existe une fonction négligeable ϵ telle que :*

$$|Pr[\mathscr{A}(N, [r^N \ mod \ N^2]) = 1] - Pr[\mathscr{A}(N, r) = 1]| \leq \epsilon(n)$$

où r un élément aléatoire de $\mathbb{Z}^*_{N^2}$ et donc, r^N un élément aléatoire de $Res(N^2)$.

En d'autres termes, quelle que soit l'action de l'algorithme $\mathscr{A}$, celui-ci ne fait pas la différence entre r et $[r^N \ mod \ N^2]$ où $r \in \mathbb{Z}^*_{N^2}$.

Le problème est qu'il n'y a ni preuve ni méthode de construction d'un tel **GenModulus**. Cependant, toute une série de tels algorithmes ont été testés et certains n'ont pu être mis en défaut. Nous ferons donc l'hypothèse qu'un tel algorithme existe et continuerons sur cette base.

3.2.2 Construction du chiffrement Paillier

Maintenant que nous sommes suffisamment armés, construisons le chiffrement Paillier. Définissons notre chiffrement comme :

$$c \underset{f}{\overset{f^{-1}}{\longleftrightarrow}} (m, 1) \cdot (0, r) = (m + 0, 1 \times r) = (m, r)$$

avec $m \in \mathbb{Z}_N$ le message à chiffrer et (0,r) un *résidu* N^{e} aléatoire.

Montrons que ce chiffrement est bien défini. Pour cela, reprenons l'isomorphisme de groupe entre $\mathbb{Z}^*_{N^2}$ et $\mathbb{Z}_N \times \mathbb{Z}^*_N$ (proposition 3.2.1) et montrons qu'il permet de décomposer notre chiffré $c \longleftrightarrow (m, r)$ en $(m, 1) \cdot (0, r)$:

$$\begin{aligned} c &\overset{def c}{=} f(m, r) \\ &\overset{def f}{=} (1 + N)^m r^N \ mod \ N^2 \\ &= ((1 + N)^m 1^N)((1 + N)^0 r^N) \ mod \ N^2 \\ &\longleftrightarrow (m, 1) \cdot (0, r). \end{aligned}$$

Montrons également que notre chiffrement est bien sûr. Supposons que nous travaillions avec un algorithme **GenModulus** qui rend le « DCR problem »difficile à résoudre. Il est dès lors difficile de distinguer un élément aléatoire $\mathbb{Z}^*_{N^2} \ni a \longleftrightarrow (r'', r')$ d'un élément aléatoire $Res(N^2) \ni b \longleftrightarrow (0, r)$ où $r'' \in \mathbb{Z}_N$ *et* $r', r \in \mathbb{Z}^*_N$ sont aléatoires. Ce chiffrement est donc bien sûr, car comme un élément aléatoire $(0, r) \in Res(N^2)$ ne peut être distingué d'un élément aléatoire $(r', r) \in \mathbb{Z}^*_{N^2}$, le chiffré c défini ci-dessus ne peut être distingué d'un autre chiffré :

$$c' \longleftrightarrow (m, 1) \cdot (r', r) = ([m + r' \ mod \ N], r) \ \forall r' \in \mathbb{Z}_N.$$

De plus, comme $[m + r' \ mod \ N]$ est distribué uniformément dans $\mathbb{Z}_N$, c' est indépendant de m et comme $r \in \mathbb{Z}^*_N$ est choisi aléatoirement, le chiffrement est sûr sous une CPA,

comme nous le prouve plus formellement le théorème suivant.

Théorème 3.2.2

Si ***GenModulus*** *rend le « DCR problem »difficile à résoudre, alors le chiffrement de Paillier est « CPA-Secure ».*

Preuve :

Soit $\Pi = < Gen, Enc, Dec >$, le schéma de chiffrement de Paillier.
Supposons $\mathscr{A}$, un adversaire PPT, et définissons

$$\epsilon(n) \stackrel{def}{=} Pr[PubK^{cpa}_{\mathscr{A},\Pi}(n) = 1],$$

la probabilité que $\mathscr{A}$ réussisse une CPA sur la schéma Π (schéma 3.1.2).

Considérons un adversaire PPT $\mathscr{D}$ qui tente de résoudre le « DCR problem ». L'algorithme utilisé par $\mathscr{D}$ peut être vu comme l'environnement extérieur du schéma (3.1.2). Comme ce problème est difficile à résoudre par hypothèse, nous avons que les probabilités de succès sont négligeables.

Algorithme 3.2.4 *Algorithme de* $\mathscr{D}$

Entrée : *La clef publique* N *et un entier* $y \in \mathbb{Z}^*_{N^2}$
Sortie : 0 *ou* 1 *en fonction de la réussite de l'algorithme.*

- $\mathscr{D}$ *exécute* $\mathscr{A}(N)$ *et obtient deux messages* m_0 *et* m_1.
- $\mathscr{D}$ *choisi aléatoirement un bit* $b \leftarrow \{0,1\}$ *et calculer*

$$c = [(1+N)^{m_b} y \ mod \ N^2]$$

- $\mathscr{D}$ *donne le chiffrement* c *à* $\mathscr{A}$ *et obtient un bit* b'.
- *Si* $b' = b, \mathscr{D}$ *ressort* 1 *et si* $b' \neq b, \mathscr{D}$ *ressort* 0.

Analysons le comportement de $\mathscr{D}$.

Cas 1 : Supposons que r ait été choisi aléatoirement dans $\mathbb{Z}^*_{N^2}$ et posons $y = [r^N \ mod \ N^2]$.
Normalement r devrait être pris dans $\mathbb{Z}^*_N$ et non dans $\mathbb{Z}^*_{N^2}$ pour respecter l'isomorphisme, mais comme la distribution de $r^N \ mod \ N^2$ est exactement la même que r soit pris dans $\mathbb{Z}^*_N$ que dans $\mathbb{Z}^*_{N^2}$, nous pouvons prendre r dans $\mathbb{Z}^*_{N^2}$.

Nous avons donc que $y \in \ Res(N^2)$ et $c = [(1+N)^{m_b} r^N \ mod \ N^2]$.

Or comme $\mathscr{D}$ ressort 1 si et seulement si la sortie b' de l'algorithme $\mathscr{A}$ est égale à b, nous avons

$$Pr[\mathscr{D}(N, [r^N \ modN^2] = 1] = Pr[PubK^{cpa}_{\mathscr{A},\Pi}(n) = 1] = \epsilon(n).$$

Cas 2 : Supposons que y soit directement pris aléatoirement dans $\mathbb{Z}^*_{N^2}$. Il est donc équivalent au r dans le cas 1.

Nous avons alors que notre chiffrement c est totalement indépendant du bit b. En effet, comme y est un élément totalement aléatoire de $\mathbb{Z}^*_{N^2}$, c est distribué aléatoirement dans $\mathbb{Z}^*_{N^2}$ par la proposition (3.2.9) ci-dessous et est donc indépendant de m_b.
Cela signifie que la probabilité que $\mathscr{A}$ ressorte $b^{'}$ tel que $b^{'} = b$ est exactement $\frac{1}{2}$ et donc

$$Pr[\mathscr{D}(N, r) = 1] = \frac{1}{2}.$$

En combinant les **cas 1** et **cas 2**, nous obtenons

$$|Pr[\mathscr{D}(N, [r^N \ mod N^2] = 1] - Pr[\mathscr{D}(N, r) = 1]| = |\epsilon(n) - \frac{1}{2}|.$$

Or comme par hypothèse, le « DCR problem »est difficile à résoudre pour notre schéma, nous avons par la définition (3.2.2),

$$|\epsilon(n) - \frac{1}{2}| \leq negl(n)$$

où *negl* est une fonction négligeable.
Et donc, $\epsilon(n) \leq \frac{1}{2} + negl(n)$.

∎

Proposition 3.2.9

Soit $\mathbb{G}$ un groupe fini et $m \in \mathbb{G}$ un élément arbitraire. Alors, si on choisit aléatoirement $g \in \mathbb{G}$ et si on pose $g^{'} := m \cdot g$, on obtient la même distribution pour $g^{'}$ que si on l'avait choisi aléatoirement. C'est-à-dire que pour tout $g^{'} \in \mathbb{G}$,

$$Pr[m \cdot g = g^{'}] = \frac{1}{|\mathbb{G}|}.$$

Preuve :

Prenons $g^{'} \in \mathbb{G}$ aléatoirement. Alors,

$$Pr[m \cdot g = g^{'}] = Pr[g = m^{-1} \cdot g^{'}].$$

Comme g est pris aléatoirement, la probabilité que g soit égal à l'élément fixé $m^{-1} \cdot g^{'}$ est de $\frac{1}{|\mathbb{G}|}$.

∎

Voyons maintenant comment décoder le chiffrement c pour obtenir le message m sous-jacent. Rappelons que nous notons $\phi(N) = (p-1)(q-1)$ l'ordre du groupe $\mathbb{Z}^*_N$. Le message m peut alors être retrouvé comme suit :

1. $\check{c} = c^{\phi(n)} \ mod \ N^2$.

2. $\check{m} = \frac{\check{c}-1}{N} \in \mathbb{Z}$.

3. $m = [\check{m}\phi(N)^{-1} \ mod \ N]$.
   Remarque : Par la Proposition (3.2.4) et le fait que $pgcd(N, \phi(N)) = 1$, $\phi(N)^{-1}$ existe.

En effet, soit $c = f(m, r)$ où $r \in \mathbb{Z}_N^*$ arbitraire. Alors,

$$
\begin{array}{lll}
1)\ \check{c} & \overset{def}{=} & c^{\phi(N)} \ mod\ N^2 \\
& \overset{defc}{=} & f(m,r)^{\phi(N)} \ mod\ N^2 \\
& = & f(m\phi(N) \ mod\ N, r^{\phi(N)} \ mod\ N) \\
& \overset{Prop3.2.1}{=} & f(m\phi(N) \ mod\ N, 1) \\
& \overset{Thm3.2.1}{=} & (1+N)^{[m\phi(N) \ mod\ N]} \ mod\ N^2 \\
& \overset{Prop3.2.5}{=} & \underbrace{[1 + [m\phi(N) \ mod\ N]N]}_{<N^2} \ mod\ N^2 \\
& = & 1 + [m\phi(N) \ mod\ N]N \\
\\
2)\ \check{m} & \overset{def}{=} & \dfrac{\check{c} - 1}{N} \\
& \overset{1)}{=} & \dfrac{1 + [m\phi(N) \ mod\ N]N - 1}{N} \\
& = & m\phi(N) \ mod\ N \\
\\
3)\ m & \overset{def}{=} & \check{m}\phi(N)^{-1} \ mod\ N \\
& \overset{2)}{=} & m\phi(N)\phi(N)^{-1} \ mod\ N \\
& = & m \ mod\ N \\
& = & m \qquad\qquad (m \in \mathbb{Z}_N).
\end{array}
$$

∎

Résumons tout ceci par le schéma de chiffrement suivant :

Schéma 3.2.1 *Chiffrement Paillier*

Soit GenModulus un algorithme PPT qui prend en entrée une suite de n bits et ressort (N, p, q) où $N = pq$ et p, q des nombres premiers longs de n bits (excepté avec une probabilité négligeable en n). Supposons également que GenModulus rend le « DCR problem »difficile à résoudre. Le chiffrement Paillier se définit alors comme suit :

- ***Gen :*** *reçoit en entrée une suite de n bits et exécute GenModulus(1^n) pour obtenir (N, p, q). La clef publique pk est alors N et la clef privée sk est $\langle N, \phi(N)\rangle$ avec $\phi(N) = (p-1)\cdot(q-1)$.*

- ***Enc :*** *reçoit en entrée la clef publique $pk = \langle N\rangle$ et un message $m \in \mathbb{Z}_N$, choisit un élément aléatoire $r \longleftarrow \mathbb{Z}_N^*$ et ressort le chiffrement*

$$c := [(1+N)^m \cdot r^N \ mod\ N^2].$$

- ***Dec :*** *reçoit en entrée la clef secrète $sk = \langle N, \phi(N)\rangle$ ainsi qu'un chiffrement c et ressort*

$$m := [\frac{[c^{\phi(N)}\ mod\ N^2] - 1}{N} \cdot \phi(N)^{-1} \mod N].$$

Illustrons ce schéma à l'aide d'un exemple : prenons pour clef publique $N = 15 = 5*3$. Il s'en suit donc que la clef privée est $\langle N, \phi(N)\rangle = \langle 15, (5-1)*(3-1)\rangle = \langle 15, 8\rangle$. Procédons au chiffrement et déchiffrement du message $\boxed{\text{m=6}} \in \mathbb{Z}_{15}$ à l'aide de l'élément aléatoire $r = 7 \in \mathbb{Z}_{15}^*$ en notant préalablement que $N^2 = 225$ et $\phi(N)^{-1} = 2$ car $8*2 = 16 \mod 15 = 1$:

$$\begin{aligned} c &= (1+15)^6 7^{15}\ mod\ 225 \\ &= (16^3)(16^3)(7^7)(7^8)\ mod\ 225 \\ &= 46*46*43*76\ mod\ 225 \\ &= 91*43*76\ mod\ 225 \\ &= 88*76\ mod\ 225 \\ &= \boxed{163} \end{aligned}$$

$$\begin{aligned} \check{c} &= 163^8\ mod\ 225 \\ &= 136*136\ mod\ 225 \\ &= 46 \end{aligned}$$

$$\begin{aligned} \check{m} &= \frac{46-1}{15} \\ &= 3 \end{aligned}$$

$$\begin{aligned} m &= 3*\phi(N)^{-1}\ mod\ 15 \\ &= 3*2 \\ &= \boxed{6} \end{aligned}$$

Nous avons donc construit un chiffrement « CPA-Secure ». Voyons les avantages que nous apporte ce chiffrement pour la réalisation d'une élection.

3.2.3 Avantages pour le vote électronique

Une question légitime est de se demander pourquoi avoir choisi ce chiffrement et pas un autre. Par son aspect homomorphe, ce schéma de chiffrement se révèle très utile dans beaucoup d'applications et en particulier dans les systèmes de vote électronique.

Définition 3.2.3 *Chiffrement homomorphe*

Un schéma à clef publique $\Pi =< Gen, Enc, Dec >$ *est dit homomorphique si pour toute paire* (pk, sk) *ressorti par* $Gen(1^n)$*, il est possible de définir les groupes* $\mathbb{C}, \mathbb{M}$ *tels que :*

- *l'espace des messages clairs est* $\mathbb{M}$*, et tous les chiffrements sortis par* Enc_{pk} *sont des éléments de* $\mathbb{C}$*.*
- $\forall m_1, m_2 \in \mathbb{M}$ *et* c_1, c_2 $\mathbb{C}$ *tels que* $m_1 = Dec_{sk}(c_1)$ *et* $m_2 = Dec_{sk}(c_2)$*, on a :*

$$Dec_{sk}(c_1 \cdot c_2) = m_1 \cdot m_2,$$

avec les opérations appropriées aux groupes $\mathbb{C}, \mathbb{M}$*.*

Montrons que le chiffrement Paillier est bien homomorphe. Soit $Enc_N(m)$ le chiffrement Paillier du message $m \in \mathbb{Z}_N$ avec respect de la clef publique N. Nous avons alors :

$$\begin{aligned} Dec(c_1 * c_2) &= Dec(Enc_N(m_1) * Enc_N(m_2)) \\ &= Dec(((1+N)^{m_1} r_1^N) * ((1+N)^{m_2} r_2^N) \ mod \ N^2) \\ &= Dec((1+N)^{m_1+m_2 \ mod \ N} * (r_1 r_2)^N \ mod \ N^2) \\ &= Dec(Enc_N([m_1 + m_2 \ mod \ N])) \\ &= [m_1 + m_2 \ mod \ N]. \end{aligned}$$

∎

Remarquons que l'opération de groupe associée à l'espace des messages clairs $\mathbb{M}$ est l'addition. Cette propriété est très utile dans le cadre de votes comme le montre l'illustration suivante.

Supposons un vote où l électeurs doivent voter « oui »ou « non »à l'adoption d'une loi. Faisons, pour cela, correspondre les messages « oui »et « non »aux messages 1 et 0 respectivement. Une clef publique N est générée et le votant i chiffre alors son choix ($v_i = 0$ ou $v_i = 1$), en prenant aléatoirement un $r \in \mathbb{Z}_N^*$, de la manière suivante :

$$c_i := [(1+N)^{v_i} r^N \ mod \ N^2].$$

Une fois que chaque électeur a voté, une autorité considérée par tous comme honnête multiplie tous les bulletins :

$$c^* := [\prod_{i=1}^{l} c_i \ mod \ N^2]$$

et déchiffre le résultat obtenu à l'aide de la clef secrète. Personne ne peut donc prendre connaissance de la valeur de chaque bulletin. A la fin du déchiffrement, nous obtiendrons $\sum_{i=1}^{l} v_i$, c'est-à-dire le nombre de votes approuvant la loi, tout en n'ayant aucune idée du contenu des votes individuels.
Cependant, qui nous dit qu'une personne mal intentionné ne va pas chiffrer le vote $v = 100$ au lieu de $v = 1$ et truquer le vote ? Il faut donc développer une preuve de vérification des votes individuels sans pour autant révéler leurs contenus. Pour qu'une élection soit considérée comme valable, elle doit pouvoir vérifier quelques propriétés :

1. **Confidentialité :** Seul le résultat final est rendu public et rien d'autre.

2. **Robustesse :** le résultat doit refléter correctement tous les votes soumis et valides même si certains électeurs essayent de tricher.

3. **Vérifiabilité universelle :** Après les élections, le résultat peut être vérifié par tout le monde.

4. **Résistance à la coercition :** Les votants sont incapables de prouver qu'ils ont voté de telle ou telle façon, et ce à qui que ce soit.

De plus, notons que dans notre exemple nous permettons seulement deux choix : « oui »et « non ». Mais qu'en est-il lorsque plus de deux candidats sont en jeu ? Tout ceci sera l'objet des prochains chapitres. Avant cela, généralisons notre chiffrement pour le rendre plus performant et efficace, choses essentielles si l'on veut pouvoir organiser des scrutins où un nombre conséquent d'électeurs élit un nombre conséquent de candidats.

3.3 Généralisation du chiffrement Paillier

Le chiffrement Paillier classique repose sur un isomorphisme de groupe du type :

$$f : \mathbb{Z}_N \times \mathbb{Z}_N^* \longrightarrow \mathbb{Z}_{N^2}^*$$

tel que $f(a, b) = [(1 + N)^a b^N \ mod\ N^2]\ \forall a \in \mathbb{Z}_N, \forall b \in \mathbb{Z}_N^*$. Cet isomorphisme nous a permis de dégager une méthode de chiffrement pour des messages appartenant à $\mathbb{Z}_N$. Ce qui implique que plus nous voudrons chiffrer de grands messages, plus N devra être grand. Le problème est que les temps de chiffrements et de déchiffrements sont proportionnels à la taille de la clef. Plus N est grand, plus le chiffrement prend du temps comme nous allons le montrer ; du moins pour le chiffrement de Paillier classique. Nous pouvons éviter ce problème en généralisant le chiffrement Paillier à l'aide d'un isomorphisme plus adéquat.

Théorème 3.3.1

Soit $N = pq$ où p, q deux nombres premiers impaires distincts de même longueur et $k < p, q$ un nombre naturel.
Alors $f_k : \mathbb{Z}_{N^k} \times \mathbb{Z}_N^* \longrightarrow \mathbb{Z}_{N^{k+1}}^*$ tel que $f_k(a, b) = [(1 + N)^a b^{N^k}\ mod\ N^{k+1}]\ \forall a \in \mathbb{Z}_{N^k}, \forall b \in \mathbb{Z}_N^*$ est un isomorphisme de groupe.

La preuve de ce théorème est semblable à celle du théorème (3.2.1), nous ne la referons donc pas.

Remarquons que nous avons toujours la propriété très pratique d'homomorphisme :

$$f_k(m_1, r_1) f_k(m_2, r_2) = f_k(m_1 + m_2, r_1 r_2).$$

En effet :

$$\begin{aligned} f_k(m_1, r_1) f_k(m_2, r_2) &= ((1+N)^{m_1} r_1^{N^k}) * ((1+N)^{m_2} r_2^{N^k}) \ mod\ N^{k+1} \\ &= (1+N)^{m_1+m_2} * (r_1 r_2)^{N^k} \ mod\ N^{k+1} \\ &= f_k(m_1 + m_2, r_1 r_2). \end{aligned}$$

∎

L'espace des messages au clair est bien devenu plus grand : $\mathbb{Z}_{N^k}$ tandis que la taille de la clef n'a pas augmenté. Notre nouveau chiffrement sera donc plus efficace, comme nous allons le montrer à la fin de cette section. Rappelons que la taille de la clef doit rester relativement grande pour résister suffisamment longtemps aux attaques de forces brutes.

Notre nouveau schéma de chiffrement peut alors s'écrire :

Schéma 3.3.1 *Généralisation du chiffrement Paillier*

Soit ***GenModulus*** *un algorithme à temps polynomial qui prend en entrée une suite de n bits et ressort le triplet (N, p, q) où $N = pq$ et p, q des nombres premiers longs de n bits (excepté avec une probabilité négligeable en n). Supposons également que* ***GenModulus*** *rend le « DCR problem »difficile à résoudre. Soit $k \in \mathbb{N}$ tel que $k < p, q$. La généralisation du chiffrement Paillier se définit alors comme suit :*

- ***Gen :*** *reçoit en entrée une suite de n bits et exécute* ***GenModulus***(1^n) *pour obtenir (N, p, q). La clef publique pk est alors $\langle N, k\rangle$ et la clef secrète sk est $\langle N, \phi(N), k\rangle$.*

- ***Enc :*** *reçoit en entrée la clef publique $pk = \langle N, k\rangle$ et un message $m \in \mathbb{Z}_{N^k}$, choisit un élément aléatoire $r \longleftarrow \mathbb{Z}_N^*$ et ressort le chiffrement*

$$c := [(1+N)^m \cdot r^{N^k} \ mod\ N^{k+1}].$$

- ***Dec :*** *reçoit en entrée la clef secrète $sk = \langle N, \phi(N), k\rangle$ ainsi qu'un chiffrement c et ressort*

$$m := [\frac{[c^{\phi(N)} \ mod\ N^{k+1}] - 1}{N} \cdot \phi(N)^{-1} \mod N^k].$$

De la même façon que précédemment, nous pouvons montrer que $m = Dec_{sk}(Enc_{pk}(m,r))$:

$$
\begin{aligned}
1)\ \check{c} &\overset{def}{=} c^{\phi(N)} \ mod\ N^{k+1} \\
&\overset{defc}{=} f_k(m,r)^{\phi(N)} \\
&= f_k(m\phi(N) \ mod\ N^k, r^{\phi(N)} \ mod\ N) \\
&\overset{Prop3.2.1}{=} f_k(m\phi(N) \ mod\ N^k, 1) \\
&\overset{Thm3.3.1}{=} (1+N)^{[m\phi(N) \ mod\ N^k]} \ mod\ N^{k+1} \\
&\overset{Prop3.2.5}{=} \underbrace{[1+[m\phi(N) \ mod\ N^k]N]}_{<N^{k+1}} \ mod\ N^{k+1} \\
&= 1+[m\phi(N) \ mod\ N^k]N \\
\\
2)\ \check{m} &\overset{def}{=} \frac{\check{c}-1}{N} \\
&\overset{1)}{=} \frac{1+[m\phi(N) \ mod\ N^k]N-1}{N} \\
&= m\phi(N) \ mod\ N^k \\
\\
3)\ m &\overset{def}{=} \check{m}\phi(N)^{-1} \ mod\ N^k \\
&\overset{2)}{=} m\phi(N)\phi(N)^{-1} \ mod\ N^k \\
&= m \ mod\ N^k \\
&= m \qquad\qquad (m \in \mathbb{Z}_{N^k}).
\end{aligned}
$$

∎

Le *DCR problem* (définition 3.2.2) nous fournit encore un certificat de sécurité. Le schéma (3.3.1) est « CPA-secure »pour tout $k < p, q$ si et seulement si le *DRC problem* est difficile à résoudre relativement à l'algorithme **GenModulus** utilisé dans le schéma pour les mêmes raisons que celles décrites à la section précédente.

Je ne montrerai pas comment ce dernier schéma peut encore être optimisé en vue d'être implémenté (voir [7] pp. 17-20) et me contenterai de vous exposer les améliorations de performances qu'il nous fournit. Notons que comme la taille du groupe contenant les messages clairs est maintenant indépendante de N (en effet, il suffit de choisir k de manière adaptée au problème), nous ne sommes plus obligés de choisir une taille de clef N extrêmement grande. Cette optimisation permet de réduire considérablement le temps de chiffrement et déchiffrement comme le montrent les tables (3.2) et (3.3).

Expliquons brièvement comment se lisent ces tableaux. Pour chaque valeur de k, nous voulons que la taille de l'ensemble des textes clairs $\mathbb{Z}_{N^k}$ soit constante (2048 ou 4096 bits). Autrement dit, lorsque k augmente, le nombre de bits pour encoder N diminue. Si nous avons besoin de 4096 bits pour encoder N^1, nous aurons besoin que N fasse respectivement 4096/2, 4096/3 et 4096/4 bits pour encoder N^2, N^3 et N^4. La taille maximum de $\mathbb{Z}^*_{N^{k+1}}$ représente la taille de l'ensemble des chiffrés, elle vaut $n * (k+1)$. Le facteur d'expansion représente le rapport entre la taille de l'ensemble des textes clairs (2048 ou 4096 bits) et celui des textes chiffrés ($\mathbb{Z}^*_{N^{k+1}}$). Les lignes chiffrements/déchiffrements représentent le temps en millisecondes (ms) pour chiffrer/déchiffrer un message. On peut

	k=1	k=2
taille clef N	2048	1024
taille max $\mathbb{Z}^*_{N^{k+1}}$	4096	3072
facteur d'expansion	2	1.5
chiffrement (ms)	1969	578
déchiffrement (ms)	1030	312

TABLE 3.2 – Comparaison avec l'ensemble des textes chiffrables de taille 2048 bits, utilisation d'une implémentation Java. Source : [7] p. 21.

	k=1	k=2	k=3	k=4
taille clef N	4096	2048	1366	1024
taille max $\mathbb{Z}^*_{N^{k+1}}$	8192	6144	5462	5120
facteur d'expansion	2	1.5	1.33	1.25
chiffrement (ms)	15264	4397	2370	1591
déchiffrement (ms)	7779	2290	1280	873

TABLE 3.3 – Comparaison avec l'ensemble des textes chiffrables de taille 4096 bits, utilisation d'une implémentation Java. Source : [7] p. 21.

remarquer un facteur deux entre elles.
Nous pouvons également voir que pour une taille fixée de textes clairs, il y a un gain en terme de temps de calcul entre l'utilisation des différents k. Plus k devient grand, plus l'algorithme est efficace. Le facteur entre le temps de calcul avec $k = 1$ et $k = 2$ est environ 3.5, avec $k = 1$ et $k = 3$ est environ 6 et avec $k = 1$ et $k = 4$ est environ 9. Ces résultats sont très intéressants dans le contexte du vote électronique, car dans ce genre d'application, nous connaissons à l'avance la taille de l'ensemble des messages clairs.

Nous avons donc construit un chiffrement « CPA-Secure »et efficace qui est utile au vote électronique de par ses propriétés homomorphiques. Il nous reste à développer un schéma de preuves permettant de certifier que les bulletins, chiffrés par le cryptosystème de Paillier, sont valides (le message caché appartient bien à l'ensemble des résultats attendus) et n'ont pas été modifiés entre le moment du vote de l'électeur et celui du dépouillement (chapitre 4), à développer un schéma de preuves permettant à tout le monde de vérifier que le résultat final est bien correct (chapitre 5), à développer un schéma de preuves permettant de rendre inutile l'achat de votes (chapitre 6), et enfin, à partir de ce qui aura été présenté jusque-là, à développer une méthode de chiffrement adaptée au STV permettant de transférer les votes valides à différents candidats selon leur ordre de préférence sur les bulletins des électeurs sans avoir besoin de déchiffrer ceux-ci (chapitre 7).

Chapitre 4

Validité des chiffrés

Tous les protocoles et définitions présents dans la section (4.1) sont tirés de [10]. Ceux des sections (4.2) et (4.3) sont tirés de [7].

Dans ce chapitre, nous allons décrire une procédure qui permettra de nous assurer que lorsque un électeur envoie un vote chiffré, ce vote est valide et non truqué afin que le résultat final reflète correctement tous les votes soumis et valides même si certains électeurs essayent de tricher. C'est ce que décrit la notion de « Robustesse » (section 3.2.3).

4.1 Protocoles à divulgation nulle

Le but des protocoles à divulgation nulle est de permettre à une personne P de prouver qu'une déclaration publique v, où est caché un secret s, fait partie d'un certain langage $\mathscr{L}$. Par exemple, si P publie $v = Enc_{pk}(0, s)$, un chiffré de Paillier du message $m = 0$, elle va pouvoir prouver que v fait partie du langage des chiffrés de Paillier du message $m = 0$. P va prouver cela à une personne V, appelée vérifieur, en montrant qu'elle connait le secret s caché au sein de la déclaration publique v. V ne doit pas spécialement connaitre s, v étant suffisant. Bien sûr P pourrait très bien dire : « voici s », mais n'importe qui se faisant passer pour V pourrait alors apprendre s et se faire passer pour P auprès de V. Les protocoles à divulgation nulle vont permettre à P de prouver à V que v fait partie du langage $\mathscr{L}$ sans révéler aucune information sur s à V au cas où celui-ci serait un imposteur. Le vérifieur ne doit en aucun cas être capable de rejouer le protocole en se faisant passer pour P. Nous verrons dans la section suivante comment ces protocoles peuvent être utiles pour vérifier qu'un électeur émet un vote valide.

Dans ce type de protocoles, le vérifieur V va poser une série de questions qui demandent la connaissance de s pour pouvoir répondre correctement. Il est évident que ces questions ne doivent en aucun cas contenir d'informations sur un quelconque s valide, sinon P pourrait deviner s au travers de la conversation. Pour qu'un protocole soit à divulgation nulle, il doit être complet, solide et satisfaire la propriété de divulgation nulle.

Définition 4.1.1 *Protocole complet*

Un protocole est dit complet si, lorsque le prouveur P et le vérificateur V sont honnêtes, il valide les dires de P avec une probabilité de $1 - negl$ où $negl$ est une fonction négligeable.

Définition 4.1.2 *Protocole solide*

Un protocole est dit solide si aucun prouveur P' malhonnête ne peut tromper un vérifieur honnête V excepté avec une probabilité négligeable.

Définition 4.1.3 *Propriété de divulgation nulle*

Un protocole complet et solide vérifie la propriété de divulgation nulle s'il existe un algorithme à temps polynomial (appelé simulateur) qui peut produire une conversation non distinguable d'une conversation entre de vrais prouveurs et vérifieurs, et ce sans la connaissance de s.

Donnons ci-dessous un exemple de protocole pour nous aider à y voir plus clair : le protocole de Fiat-Shamir. Ce protocole a pour but de prouver que la déclaration publique $v = s^2 \ mod \ N$ fait partie des résidus quadratiques. Si nous connaissons $\phi(N)$, il est alors facile de retrouver s. En effet, il suffit de calculer

$$d = 2^{-1} \ mod \ \phi(N)$$

et

$$v^d = s^{2d} = s \ mod \ N.$$

Cependant, pour calculer $\phi(N)$, il faut pouvoir factoriser N en un produit de deux nombres premiers de même longueur, chose difficile avec l'algorithme **GenModulus** décrit en section (3.2.1). C'est pourquoi il est difficile de montrer que v est un résidu quadratique sans la connaissance de s.
Notons, pour l'anecdote, que ce principe de chiffrement ($c = m^e \ mod \ N$) est utilisé dans le chiffrement bien connu RSA. C'est pourquoi les algorithmes **GenModulus** ne permettant pas de calculer facilement $d = 2^{-1} \ mod \ \phi(N)$ sont appelés **RSAModulus**.

Décrivons de manière formelle le protocole Fiat-Shamir.

Protocole 4.1.1 *Fiat-Shamir*

1. ***Initialisation*** *Un centre considéré comme honnête utilise un algorithme **RSAModulus** qui ressort $N = pq$ et garde secret p et q. Chaque participant P sélectionne un secret s relativement premier à N et enregistre $v = s^2 \ mod \ N$, sa clef publique, dans le centre.*
2. ***Protocole*** *Les quatre étapes suivantes sont exécutées t fois et le vérificateur V accepte la preuve de connaissance s'il accepte la réponse de P à chaque fois.*

   i. P choisit aléatoirement $0 \neq r \in \mathbb{Z}_N$ et envoie $x = r^2 \ mod \ N$ à V.
   ii. V envoie à P un challenge $e \in \{0, 1\}$ pris aléatoirement.
   iii. P envoie à V la réponse $y = r$ si $e = 0$ ou $y = rs \ mod \ N$ si $e = 1$.
   iiii. V rejette la preuve si $y = 0$ ou si $y^2 \neq x \cdot v^e$ et l'accepte sinon.

Remarquons qu'il est obligatoire de renouveler le choix de r à chaque nouvelle question. Supposons qu'à la première itération, V envoie le challenge $e = 0$. Nous avons donc

$y_1 = r$. Supposons qu'à la deuxième itération, V envoie le challenge $e = 1$. Nous avons donc $y_2 = rs$. Un observateur n'aura alors plus qu'à calculer $\frac{y_2}{y_1} = \frac{rs}{r}$ pour retrouver s.

Montrons à présent que le protocole de Fiat-Shamir est à divulgation nulle.

- Ce protocole est complet, car si P connait r et s alors il peut répondre au challenge de V. En effet, le prouveur envoie

  $$y \equiv rs^e \ mod \ N$$

  et le vérificateur, qui connait x, v et e, vérifie si

  $$y^2 \equiv r^2 s^{2e} \ mod \ N \equiv xv^e \ mod \ N.$$

- Il est également solide dans le sens où aucun P' malhonnête ne peut tromper V excepté avec une probabilité négligeable. D'une part, un observateur qui analyse les échanges entre P et V est incapable de trouver s à moins de connaitre le r utilisé quand le challenge e est égal à 1. En effet, il suffirait alors à un adversaire de calculer $\frac{y}{r}$ pour retrouver s. Cependant, connaitre r revient à calculer la racine carrée de $x = r^2 \ mod \ N$ et c'est impossible en un temps polynomial du fait de l'utilisation du module RSA résistant. Remarquons que déduire s de y est impossible du fait que celui-ci est uniformément distribué dans $\mathbb{Z}_N$ de par l'utilisation de r pris aléatoirement dans $\mathbb{Z}_N$. D'autre part, un P' malhonnête qui essaye de tricher (c'est-à-dire prouver à V qu'il connait s alors que ce n'est pas vrai) peut le faire de deux manières différentes. Il peut envoyer à V $x = r^2 \ mod \ N$ (il connait donc r) et espérer que le challenge sera $e = 0$ pour répondre $y = r$. Il ne pourra pas répondre au challenge $e = 1$ vu qu'il ne connait pas s. Ou il peut envoyer $x = \frac{r^2}{v} \ mod \ N$ et espérer que $e = 1$ pour répondre $y = r$. En effet, dans ce deuxième cas, V va vérifier que $y^2 = xv = \frac{r^2}{v} * v$ et trouvera bien $y^2 = r^2$. Par contre, il ne pourra pas répondre au challenge $e = 0$ vu qu'il ne connait pas le r sous-jacent au x qu'il a envoyé. Notons que nous devons interdire le cas $y = 0$, car sinon r pourrait être égal à zéro 0 et la preuve serait alors toujours vérifiée. L'individu malhonnête a donc une chance sur deux de pouvoir répondre correctement et donc une probabilité de 2^{-t} au bout de t itérations de pouvoir passer le protocole, ce qui est négligeable pour un t suffisamment grand. La seule information que laisse transparaître le protocole sur s est que s est une racine carrée $mod \ N$ de v.

- Il vérifie également la propriété de divulgation nulle car il existe un simulateur à temps polynomial qui peut produire une conversation non distinguable d'une conversation entre de vrais prouveurs et vérificateurs. Ce simulateur agit de la manière suivante. Il prend en entrée la clef publique N, la déclaration publique v, choisit aléatoirement un $y \in \mathbb{Z}_N$ et calcule :

  — $x = y^2 \ mod \ N$ pour répondre au challenge $e = 0$
  — $x = y^2 v^{-1} \ mod \ N$ pour répondre au challenge $e = 1$

  Nous avons ainsi une simulation basée sur une connaissance préalable des challenges.

4.2 Application au vote « oui », « non »

Comme précédemment (section 3.2.3), supposons que nous avons 2 candidats en liste et que pour l'un d'eux on vote 0 et pour l'autre 1. Supposons également qu'il y ait E électeurs, une autorité A qui calcule le résultat et un tableau T où les votants enregistrent leurs paires de votes et de preuves.

Présentons un protocole à divulgation nulle adapté à ce cas de figure. Comme dans le protocole de Fiat-Shamir, le prouveur (l'électeur ici) doit prouver qu'il connait un secret s. Ce s doit lui avoir servi à chiffrer soit le message 0, soit le message 1 et rien d'autre. Pour prouver que son vote chiffré c par le chiffrement de Paillier généralisé contient soit le message 0, soit le message 1 et rien d'autre, l'électeur va utiliser le protocole 4.2.1 « 1er *ou* 2e *puissance* N^k ». Ce protocole permet de vérifier si le message m sous-jacent à un chiffré de Paillier généralisé $c = Enc_{pk}(m, s)$ $(s \in \mathbb{Z}_N^*)$ est bien une des deux valeurs attendues par le vérificateur (m_1 *ou* m_2) ; 0 et 1 dans notre cas. Pour cela, il prend en entrée trois arguments : N, c_1, c_2 ; N représente la clef publique du chiffrement de Paillier généralisé, c_1 représente le chiffré c divisé par $(1+N)^{m_1}$ et c_2 représente le chiffré c divisé par $(1+N)^{m_2}$. Nous avons donc que soit c_1, soit c_2 est un chiffré de 0 par construction du chiffrement de Paillier généralisé ($f_k(m, r) = (1+N)^m r^{N^k} \ mod\ N^{k+1}$) si c est effectivement un chiffrement de m_1 ou de m_2. Ce protocole va donc supposer que soit c_1, soit c_2 est un chiffrement de 0 et va vérifier que c'est le cas. Supposons sans perdre de généralité que le message m sous-jacent à c est m_1 (c_1 chiffré de 0).

Protocole 4.2.1 *1^{er} ou 2^e puissance N^k*

Entrée : *$pk = \langle N, k\rangle, c_1, c_2$.*
Entrée secrète de *P : $s_1 \in \mathbb{Z}_N^*$ tel que $c_1 = Enc_{pk}(0, s_1)$*

1. *P invoque un simulateur M (définition 4.1.3) sur le protocole à divulgation nulle 4.2.2 « puissance N^k »qui prend en entrée N, c_2 et ressort la conversation valide x_2, e_2, y_2, c'est-à-dire dire qui vérifie $Enc_{pk}(0, y_2) = x_2 c_2^{e_2} \ mod\ N^{k+1}$.*

2. *P choisit aléatoirement $r \in \mathbb{Z}_N^*$ et envoie $x_1 = Enc_{pk}(0, r)$ et x_2 à V.*

3. *V choisit aléatoirement un challenge e long d'un nombre aléatoire de k_1 bits et l'envoie à P.*

4. *P calcule $e_1 = e - e_2 \ mod\ 2^{k_1}$ et $y_1 = r s_1^{e_1} \ mod\ N$. Il envoie alors e_1, y_1, x_2, e_2, y_2 à V.*

5. *V vérifie alors que :*
   - *$e = e_1 + e_2 \ mod\ 2^{k_1}$,*
   - *$Enc_{pk}(0, y_1) = x_1 c_1^{e_1} \ mod\ N^{k+1}$,*
   - *$Enc_{pk}(0, y_2) = x_2 c_2^{e_2} \ mod\ N^{k+1}$,*
   - *$c_1, c_2, x_1, x_2, y_1, y_2$ sont relativement premiers à N.*

Décrivons plus en détails la première étape de ce protocole.

1. Le protocole (4.2.2) *Puissance* N^k, sur lequel le prouveur P invoque un simulateur M dans le protocole (4.2.1), prend en entrée deux arguments : N et c; N représente la clef publique du chiffrement de Paillier généralisé et c un chiffré. Il a pour but de vérifier que P connait bien le secret s caché dans le chiffré de Paillier c du message $m = 0$.

Protocole 4.2.2 *Puissance* N^k

Entrée : $pk = \langle N, k \rangle, c$.
Entrée secrète de P ***:*** $s \in \mathbb{Z}_N^*$ *tel que* $c = Enc_{pk}(0, s)$

1. P *choisit aléatoirement* $r \in \mathbb{Z}_N^*$ *et envoie* $x = Enc_{pk}(0, r)$ *à* V.
2. V *choisit aléatoirement un challenge* e *long d'un nombre aléatoire de* k_1 *bits et l'envoie à* P.
3. P *envoie* $y = rs^e \ mod \ N$ *à* V.
4. V *vérifie que* c, x, y *sont relativement premiers à* N *et que* $Enc_{pk}(0, y) = xc^e \ mod \ N^{k+1}$.

Ce protocole (4.2.2) est bien à divulgation nulle.

- Il est complet car si P connait s et r alors il peut répondre au challenge e de V. En effet, le prouveur envoie

$$y \equiv rs^e \ mod \ N$$

et le vérificateur, qui connait x, c et e, vérifie si

$$\begin{aligned} Enc_{pk}(0, y) &= Enc_{pk}(0, rs^e \ mod \ N) \\ &= Enc_{pk}(0, r) Enc_{pk}(0, s^e \ mod \ N) \\ &= Enc_{pk}(0, r) Enc_{pk}(0, s \ mod \ N)^e \\ &= xc^e \ mod \ N^{k+1} \end{aligned}$$

par la propriété d'homomorphisme du chiffrement de Paillier.

- Il est également solide dans le sens où aucun P' malhonnête ne peut tromper V excepté avec une probabilité négligeable. D'une part, un observateur qui analyse les échanges entre P et V est incapable de trouver s à moins de connaitre le r utilisé et de calculer $d = e^{-1} \ mod \ \phi(N)$. En effet, il suffirait alors à un adversaire de calculer y/r pour retrouver $s^e \ mod \ N$ et de multiplier ce résultat par d. Cependant, connaitre r revient à casser le chiffrement de Paillier pour x ce qui est impossible comme on l'a montré au chapitre précédent et calculer $d = e^{-1} \ mod \ \phi(N)$ est également impossible, car c'est un problème équivalent à la factorisation (voir section précédente). D'autre part, un P' malhonnête peut essayer de tricher (c'est-à-dire prouver à V qu'il connait s alors que ce n'est vrai) en envoyant à V : $x = Enc_{pk}(0, r)/c^{e'} \ mod \ N$ et en espérant que le challenge sera $e = e'$ pour répondre $y = r$. En effet, V va

vérifier que $Enc_{pk}(0,y) = xc^e \ mod \ N^{k+1}$ et trouvera bien

$$Enc_{pk}(0,y) = Enc_{pk}(0,r) = \frac{Enc_{pk}(0,r)}{c^{e'}} c^e \ mod \ N^{k+1} = xc^e \ mod \ N^{k+1}.$$

Cependant, l'individu malhonnête aura une probabilité $2^{-(k_1-1)}$ de pouvoir répondre correctement (k_1 étant la longueur de bit du challenge e), ce qui est négligeable pour k_1 suffisamment grand.

- Il vérifie également la propriété de divulgation nulle, car il existe un simulateur à temps polynomial qui peut produire une conversation non distinguable d'une conversation entre de vrais prouveurs et vérificateurs. Ce simulateur agit de la manière suivante. Il prend en entrée la clé publique N et le chiffré de Paillier c, choisit aléatoirement un $y \in \mathbb{Z}_N$ et un challenge e puis calcule :
  $x = Enc_{pk}(0,y)c^{-e} \ mod \ N^{k+1}$.
  Il ressort alors la conversation parfaitement valide (x,e,y). En effet,
  $xc^e \ mod \ N^{k+1} = Enc_{pk}(0,y)c^{-e}c^e \ mod \ N^{k+1} = Enc_{pk}(0,y)$.
  Remarquons que c n'a plus besoin d'être un chiffrement de 0, mais peut être quelconque.

C'est ce simulateur M qu'utilise P dans le protocole (4.2.1). Comme P sait que c'est le chiffré c_1 qui est le chiffrement de 0 (par hypothèse), il n'a qu'à utiliser le simulateur sur le chiffré c_2 qui ne l'est pas, vu que le simulateur n'a pas besoin d'utiliser un chiffré qui soit un chiffrement de 0.

Montrons à présent que le protocole (4.2.1) est bien à divulgation nulle.

- Il est complet, car si P connait s et r alors il peut répondre au challenge e de V. En effet, le prouveur envoie x_2, e_2, y_2 et

$$\begin{aligned} x_1 &= Enc_{pk}(0,r) \\ e_1 &= e - e_2 \ mod \ 2^{k_1} \\ y_1 &= rs_1^{e_1} \ mod \ N \end{aligned}$$

  et le vérificateur, qui connait x, c et e, vérifie si

$$\begin{aligned} e_1 + e_2 \ mod \ 2^{k_1} &= e - e_2 + e_2 \ mod \ 2^{k_1} \\ &= e, \\ x_1 c_1^{e_1} \ mod \ N^{k+1} &= Enc_{pk}(0,r)Enc_{pk}(0,s_1)^{e_1} \ mod \ N^{k+1} \\ &= Enc_{pk}(0, rs_1^{e_1}) \ (\star) \\ &= Enc_{pk}(0,y_1), \\ Enc_{pk}(0,y_2) &= x_2 c_2^{e_2} \ mod \ N^{k+1}. \ (\star\star) \end{aligned}$$

  Le résultat $(\star)$ s'obtient par la propriété d'homomorphisme du chiffrement de Paillier.
  Le résultat $(\star\star)$ s'obtient par la propriété du simulateur M.

- Il est également solide pour les mêmes raisons que le protocole précédent. Aucun P' malhonnête ne peut tromper V excepté avec une probabilité négligeable. D'une part, un observateur qui analyse les échanges entre P et V est incapable de

trouver s_1 à moins de connaitre le r utilisé et de calculer $d = e_1^{-1}\ mod\ \phi(N)$. En effet, il suffirait alors à un adversaire de calculer y_1/r pour retrouver $s_1^{e_1}\ mod\ N$ et de multiplier ce résultat par d. Cependant, connaitre r revient à casser le chiffrement de Paillier pour x_1 ce qui est impossible comme nous l'avons montré au chapitre précédent, et calculer $d = e_1^{-1}\ mod\ \phi(N)$ est également impossible, car c'est un problème équivalent à la factorisation (voir section précédente). Notons également que le prouveur P ne peut faire appel au simulateur pour construire une discussion valide pour c_1 du fait que V exige de connaitre x_1 et x_2 avant de fournir le challenge e. Le fait que V connaissent x_1 et x_2 avant de fournir le challenge e empêche P de choisir a posteriori les challenges e_1 et e_2. D'autre part, un P' malhonnête peut essayer de tricher (c'est-à-dire prouver à V qu'il connait s_1 alors que ce n'est vrai) en envoyant à V : $x_1 = Enc_{pk}(0,r)/c_1^{e_1}\ mod\ N$ et en espérant que le challenge sera $e = e_1 + e_2\ mod\ 2^{k_1}$ pour répondre $y_1 = r$. En effet, V va vérifier que $Enc_{pk}(0,y_1) = x_1 c_1^{e_1}\ mod\ N^{k+1}$ et trouvera bien

$$Enc_{pk}(0,y_1) = Enc_{pk}(0,r) = \frac{Enc_{pk}(0,r)}{c_1^{e_1}} c_1^{e_1}\ mod\ N^{k+1} = x_1 c_1^{e_1}\ mod\ N^{k+1}.$$

Cependant, l'individu malhonnête aura une probabilité de $2^{-(k_1-1)}$ de pouvoir répondre correctement (k_1 étant la longueur de bit du challenge e), ce qui est négligeable pour k_1 suffisamment grand. Notons que n'importe qui peut générer une conversation valide pour c_2.
Rappelons également que le chiffrement de Paillier généralisé repose sur l'isomorphisme de groupe $f_k : \mathbb{Z}_{N^k} \times \mathbb{Z}_N^* \longrightarrow \mathbb{Z}_{N^{k+1}}^*$ tel que $f_k(a,b) = [(1+N)^a b^N\ mod\ N^{k+1}]\ \forall a \in \mathbb{Z}_N, \forall b \in \mathbb{Z}_N^*$. Cela signifie qu'il n'existe pas pour un même chiffré c deux pairs $(a,b) \neq (v,w)$ tel que $c = f(a,b) = f(v,w)$. Un prouveur malhonnête ne peut donc pas chiffrer un message m qui soit différent de 1 et de 0 à l'aide du nombre aléatoire r ($c = f(m,r)$ et trouver une paire $(0,r')$ tel que ce même c vérifie $c = f(0,r')$ pour pouvoir passer la validité du vote.

- Il vérifie également la propriété de divulgation nulle, car il existe un simulateur qui agit de la manière suivante. Il prend en entrée la clef publique N et le chiffré de Paillier c_1 et c_2, choisit aléatoirement un $y \in \mathbb{Z}_N$ et un challenge e_1 puis calcule :
  $x_1 = Enc_{pk}(0,y_1) c_1^{-e_1}\ mod\ N^{k+1}$.
  Il ressort alors la conversation parfaitement valide (x,e,y). En effet,
  $x_1 c_1^{e_1}\ mod\ N^{k+1} = Enc_{pk}(0,y_1) c_1^{-e_1} c_1^{e_1}\ mod\ N^{k+1} = Enc_{pk}(0,y_1)$.
  Il fait également appel à un simulateur sur le protocole (4.2.2) pour N et c_2 qui ressort la conversation valide

$$Enc_{pk}(0,y_2) = x_2 c_2^{e_2}\ mod\ N^{k+1}.$$

Le protocole (4.2.2) présente encore un inconvénient pour pouvoir être utilisé dans un système de vote : il utilise une preuve interactive. Expliquons : le prouveur P envoie un message au vérificateur V et celui-ci lui répond. Vous imaginez bien que c'est quelque peu embarrassant dans le cadre du vote électronique avec des centaines d'électeurs car il faudrait retenir quel votant a envoyé quel x et à quel votant on a envoyé tel challenge, ce qui demande une quantité de stockage dont on pourrait se passer. Nous pouvons rendre cette preuve à divulgation nulle non interactive en utilisant une fonction de hachage $\mathscr{H}$ pour remplacer le challenge e de V. Les résultats utilisés pour faire cela sont tirés de [11].
Cette transformation doit se faire de manière rigoureuse pour ne pas entamer la sécurité

du protocole. Pour cela, nous allons repartir du protocole (4.2.2) et le transformer en preuve non-interactive. Cette transformation peut s'appliquer à tous les protocoles à divulgation nulle présentés jusqu'ici.

Protocole 4.2.3 *Puissance N^k non − interactif*

Entrée : *$pk = \langle N, k\rangle$.*
Entrée secrète de *P :* *c et $s \in \mathbb{Z}_N^*$ tel que $c = Enc_{pk}(0, s)$*

1. *P choisit aléatoirement $r \in \mathbb{Z}_N^*$ et calcule $x = Enc_{pk}(0, r)$.*
2. *P évalue la fonction de hachage $\mathscr{H}(c, x)$ et obtient le challenge e.*
3. *P calcule $y = rs^e \ mod \ N$.*
4. *P envoie la conversation c, x, y à V qui évalue la fonction de hachage $\mathscr{H}(c, x) = e$ et qui vérifie que*

$$\mathscr{H}(c, \frac{Enc_{pk}(0, y)}{c^e}) = e.$$

Ce protocole est complet car si P connait s et r alors il peut répondre au challenge e. En effet, le prouveur envoie la conversation c, x, y. V retrouve alors le challenge $e = \mathscr{H}(c, x)$ utilisé par P et vérifie si

$$\begin{aligned}
\mathscr{H}(c, \frac{Enc_{pk}(0, y)}{c^e}) &= \mathscr{H}(c, \frac{Enc_{pk}(0, rs^e \ mod \ N)}{Enc_{pk}(0, s)^e}) \\
&= \mathscr{H}(c, Enc_{pk}(0, rs^e s^{-e} \ mod \ N)) \\
&= \mathscr{H}(c, Enc_{pk}(0, r)) \\
&= \mathscr{H}(c, x) \\
&= e
\end{aligned}$$

par la propriété d'homomorphisme du chiffrement de Paillier.

Il est tout à fait essentiel pour la sécurité du protocole que la fonction $\mathscr{H}$ prenne en entrée à la fois c et x, car lorsque $\mathscr{H}$ ne prend que x en entrée, le protocole n'est plus sûr comme le montre la démarche suivante.
Si $\mathscr{H}$ prend uniquement x en entrée, V vérifie que

$$\mathscr{H}(\frac{Enc_{pk}(0, y)}{c^e}) = e.$$

Dés lors, un prouveur malhonnête P' peut tricher de la manière suivante :

- P' choisit aléatoirement $y \in \mathbb{Z}_N^*$.
- P' choisit aléatoirement $x \in \mathbb{Z}_N^{k+1}$ et calcule la challenge $e = \mathscr{H}(x)$.
- P' calcule alors $c = (\frac{Enc(0,y)}{x})^{\frac{1}{e}}$ et envoie la conversation c, x, y à V.

V calcule $\mathscr{H}(x) = e$ vérifie alors que

$$\begin{aligned}
\mathscr{H}(\frac{Enc_{pk}(0, y)}{c^e}) &= \mathscr{H}(\frac{Enc_{pk}(0, y)}{((\frac{Enc(0,y)}{x})^{\frac{1}{e}})^e}) \\
&= \mathscr{H}(\frac{Enc_{pk}(0, y)}{(\frac{Enc(0,y)}{x})}) \\
&= \mathscr{H}(x) \\
&= e.
\end{aligned}$$

Par cette méthode, l'adversaire P' parvient à prouver que le chiffré $c = (\frac{Enc(0,y)}{x})^{\frac{1}{e}}$ est un chiffré de 0 alors que ce n'est pas forcément le cas.
Le fait que $\mathscr{H}$ dépende de c et x empêche P' de choisir le chiffré c en fonction du challenge e et le protocole devient sécurisé. Cela permet également d'obtenir la propriété de non-malléabilité qui est essentielle lors d'un vote. La malléabilité désigne la possibilité de modifier de quelque manière que ce soit un chiffré c et sa preuve de validité associée pour obtenir un nouveau chiffré c' avec un preuve associée valide.

Montrons que si $\mathscr{H}$ prend uniquement x en entrée, la propriété de non-malléabilité n'est pas obtenue. Supposons que la conversation c, x, y soit valide. C'est-à-dire que $\mathscr{H}(x) = e$ et que $\mathscr{H}(\frac{Enc_{pk}(0,y)}{c^e}) = e$. Dés lors, un prouveur malhonnête P' peut imiter ce vote en appliquant la méthode suivante :

- P' calcule $y' = yr'^e$.
- P' calcule $c' = cEnc_{pk}(0, r')$.
- P' pose $x' = x$ et envoie la conversation c', x', y'.

V calcule $\mathscr{H}(x') = \mathscr{H}(x) = e$ vérifie alors que

$$\begin{aligned}
\mathscr{H}(\frac{Enc_{pk}(0,y')}{c'^e}) &= \mathscr{H}(\frac{Enc_{pk}(0,yr'^e)}{(cEnc_{pk}(0,r'))^e}) \\
&= \mathscr{H}(\frac{Enc_{pk}(0,y)Enc_{pk}(0,r'^e)}{c^e Enc_{pk}(0,r')^e}) \\
&= \mathscr{H}(\frac{Enc_{pk}(0,y)}{c^e}) \\
&= e.
\end{aligned}$$

Le prouveur P' peut donc chiffrer le même message contenu dans c, mais à l'aide d'un autre chiffré c' et fournir une preuve valide sur ce chiffré. La propriété de non-malléabilité n'est pas obtenue.

Par contre, le fait que $\mathscr{H}$ dépende de c et x empêche P' d'obtenir le même challenge e pour deux chiffrés différents c et c'. Nous obtenons alors la propriété de non-malléabilité. En combinant l'interdiction de paires de votes et de preuves identiques dans le tableau T où les électeurs enregistrent leurs paires de vote et de preuve avec la propriété de non-malléabilité, nous empêchons les électeurs d'imiter le vote d'un autre électeur.

Grâce à ces différentes analyses, nous pouvons enfin décrire un protocole de vote complet pour un cas où deux candidats se présentent et où E électeurs votent. Rappelons que nous voulons montrer que le message sous-jacent au chiffré c_i de l'électeur i est soit 0, soit 1 ; c'est-à-dire que soit c_i, soit $c_i/(1+N)$ est un chiffrement de 0.

Protocole 4.2.4 *Vote « oui », « non »*

1. *Chaque électeur E_i $i = 1, ..., E$, choisit son vote $v_i \in \{0, 1\}$ et le chiffre à l'aide du chiffrement Paillier généralisé (schéma 3.3.1) :*

$$c_i = Enc_{pk}(v_i, r_i)$$

où r_i est aléatoire dans $\mathbb{Z}_N^*$ et $pk = \langle N, k \rangle$. Il calcule aussi une preuve de validité pour son chiffrement :

$$Preuve_{E_i}(c_i \text{ ou } c_i/(N+1) \text{ est un } chiffrement \text{ de } 0)$$

à l'aide du protocole (4.2.1) « 1^{er} ou 2^e puissance N^k »rendu non-interactif.

2. *L'autorité A agit alors de la manière suivante :*

   a. *Pose $c = 1$.*
   b. *Pour chaque votant E_i :*
      - *Vérifie si la preuve de E_i pour son chiffré c_i est valide.*
      - *Si elle est valide, calcule $c * c_i \ mod \ N^{k+1}$*
   c. *Déchiffre le chiffré c comme expliqué dans le schéma (3.3.1).*

Nous obtenons le résultat correct du vote à condition que $N^k > E + 1$, car si l'on suppose que tous les votes sont valides, nous obtenons

$$c = \prod_{i=1}^{E} Enc_{pk}(v_i, r_i) = Enc_{pk}(\sum_{i=1}^{E} v_i \ mod \ N^k, \prod_{i=1}^{E} r_i \ mod \ N)$$

et donc

$$Dec_{sk}(c) = \sum_{i=1}^{E} v_i \ mod \ N^k = \sum_{i=1}^{E} v_i \text{ si } N^k > E + 1.$$

Cette condition peut toujours être obtenue en prenant un N ou un k suffisamment grand.

Nous avons donc défini un moyen de voter pour deux candidats à raison d'un vote par électeurs qui les empêche de tricher.

4.3 Vote pour l candidats parmi L

Supposons à présent qu'il y ait L candidats en liste et que chaque votant doit voter pour exactement l d'entre eux. Nous pouvons utiliser le même protocole qu'au-dessus, mais à la place d'un unique vote, chaque électeur E_i chiffre L votes et leurs preuves associées :

$$c_{ij} = Enc_{pk}(v_{ij}, r_{ij}) \quad j = 0, ..., L-1$$

$$Preuve_{ij}(c_{ij} \; ou \; \frac{c_{ij}}{N+1} \; est \; un \; chiffrement \; de \; 0) \quad j = 0, ..., L-1.$$

En supposant que toutes les preuves soient valides, le nombre de voix pour le candidat j sera calculé par :

$$c_{*j} = \prod_{i=1}^{E} c_{ij} = Enc_{pk}(\sum_{i=1}^{E} v_{ij} \; mod \; N^k, \prod_{i=1}^{E} r_{ij} \; mod \; N)$$

$$Dec_{sk}(c_{*j}) = \sum_{i=1}^{E} v_{ij} \; mod \; N^k = \sum_{i=1}^{E} v_{ij} \; si \; N^k > E+1.$$

Il faut donc L déchiffrements pour obtenir tous les résultats. L'autorité A peut prouver que chaque votant i a voté pour exactement l candidats en vérifiant que :

$$l = Dec_{sk}(\prod_{j=0}^{L-1} Enc_{pk}(v_{ij}, r_{ij})) \quad i = 1, ..., E.$$

Ce système peut encore être généralisé à un cas de vote pour au plus l candidats parmi L. Il suffit de rajouter l candidats factices aux L réels et de procéder de la même façon que précédemment mais avec $L+l$ candidats. Quand un électeur ne souhaite pas utiliser toutes ses voix, il peut donner le surplus aux candidats factices.

Nous avons donc construit jusqu'ici un chiffrement « CPA-Secure »et efficace qui est utile au vote électronique de par ses propriétés homomorphiques : le chiffrement de Paillier. Nous venons de développer un schéma de preuves à divulgation nulle permettant de certifier que les bulletins, chiffrés par le cryptosystème de Paillier, sont valides et ce, pour un système de vote où l'on peut voter jusqu'à l parmi L. Il nous reste à développer un schéma de preuves permettant à tout le monde de vérifier que le résultat final est bien correct (chapitre 5), de rendre inutile l'achat de votes (chapitre 6) et enfin, à partir de ce qui aura été présenté jusque-là, à développer une méthode de chiffrement adaptée au STV permettant de transférer les votes valides à différents candidats selon leur ordre de préférence sur les bulletins des électeurs sans avoir besoin de déchiffrer ceux-ci (chapitre 7).

Chapitre 5

Contrôle de l'autorité

Tous les protocoles, définitions et algorithmes de la section (5.1) sont tirés de [12]. Les résultats de la section (5.2) viennent de l'auteur en s'inspirant de ce qui a été développé précédemment.

Dans ce chapitre, nous allons décrire une procédure qui permettra de s'assurer que l'autorité qui dépouille les votes et publie le résultat final ne triche pas. En effet, l'autorité pourrait être corrompue et révéler le contenu des votes à certaine(s) personne(s) ou encore mal compter les votes. D'une part, cela signifie que l'autorité ne peut être dans la capacité de déchiffrer n'importe quel chiffré. Pour cela nous utiliserons la méthode de partage de la clef secrète. D'autre part, cela signifie que tout un chacun doit être dans la capacité de vérifier que tous les votes valides ont bien été pris en compte dans le décompte final sans pour autant connaitre autre chose que le résultat final. C'est ce que décrit la notion de « Vérifiabilité universelle » (section 3.2.3).

5.1 Partage de la clef secrète

Jusqu'ici nous avons considéré que une seule personne possédait l'entièreté de la clef secrète sk du chiffrement de Paillier. Dans cette section, nous allons partager sk en l parties en donnant à chaque participant sa propre clef partagée, de manière à ce qu'un sous-ensemble de m parties soient nécessaire afin de reconstruire sk. Ce système, en augmentant le nombre d'autorités nécessaires au déchiffrage, va nous permettre de limiter les risques de corruption. Pour ce faire, nous utilisons une interpolation de Lagrange du polynôme $f(X) = \sum_{i=0}^{m-1} a_i X^i \ mod \ N^{k+1}$ où les coefficients $a_i \ \ i \in \{1, ..., k-1\}$ sont pris aléatoirement dans l'ensemble $\{0, ..., N^{k+1} - 1\}$ et $a_0 = \phi(N)$. Nous évaluons ce polynôme aux points $X = 1, ...l$ où $m \leq l$ et donnons chaque valeur calculée à une des l parties. Chaque valeur constituera la clef secrète de la partie. Nous obtenons alors l points (i, s_i) où $s_i = f(i)$ est la clef secrète de la partie i. Nous allons alors prendre un sous-ensemble S de m points parmi les l possibles et les interpoler par un polynôme de Lagrange. Nous savons qu'il existe un unique polynôme P de degré maximum $m-1$ qui passe par l'ensemble de ces m points et qui s'écrit

$$L(X) = \sum_{i \in S} s_i \prod_{i' \in S \setminus i} \frac{X - i'}{i - i'}.$$

Comme il n'existe qu'un unique polynôme passant par les points $(i, s_i) \ i \in S$ de degré $m-1$, ce polynôme n'est autre que $f(X) = \sum_{i=0}^{m-1} a_i X^i \ mod \ N^{k+1}$. En évaluant

$L(X)$ en $X = 0$, c'est-à-dire en calculant $L(0) = \sum_{i \in S} s_i \prod_{i' \in S \setminus i} \frac{0-i'}{i-i'}$, nous calculons $f(0) = \sum_{i=0}^{m-1} a_i 0^i \ mod\ N^{k+1} = a_0 = \phi(N)$ qui n'est autre que la clef secrète du chiffrement de Paillier classique. Décrivons ceci formellement par le schéma suivant :

Schéma 5.1.1 *Chiffrement Paillier généralisé avec partage de la clef secrète.*

Soit ***GenModulus*** *un algorithme à temps polynomial qui prend en entrée une suite de n bits et génère deux nombres premiers p et q longs de n bits (excepté avec une probabilité négligeable en n). Il pose alors $N = pq$ la clef publique du chiffrement. Nous choisissons $k \in \mathbb{N}$ tel que $k < p, q$ pour travailler dans l'espace des textes clairs $\mathbb{Z}_{N^k}$.* ***GenModulus*** *va alors construire le polynôme $f(X) = \sum_{i=0}^{m-1} a_i X^i \ mod\ N^{k+1}$ en prenant les coefficients $a_i \ \ i \in \{1, ..., m-1\}$ aléatoirement dans l'ensemble $\{0, ..., N^{k+1} - 1\}$ et $a_0 = \phi(N)$. Chaque partie i possédera le secret $s_i = f(i)$, $i \in \{1, ..., l\}$, tel que $m \leq l$. Pour chacune des l parties, une clef de vérification pour le déchiffrage va également être calculée : $v_i = v^{s_i} \ mod\ N^{k+1}$ où $v \in \mathbb{Z}^*_{N^{k+1}}$. Supposons également que* ***GenModulus*** *rend le « DCR problem »difficile à résoudre. Le chiffrement Paillier généralisé avec partage de clef secrète se définit alors comme suit :*

- ***Gen :*** *reçoit en entrée une suite de n bits et exécute* ***GenModulus***(1^n) *pour obtenir (N, p, q). La clef publique pk est alors $\langle N, k \rangle$ et les clefs secrètes s_i des l parties sont $f(i),\ i \in \{1, ..., l\}$.*

- ***Enc :*** *reçoit en entrée la clef publique $pk = \langle N, k \rangle$ et un message $m \in \mathbb{Z}_{N^k}$, choisit un élément aléatoire $r \longleftarrow \mathbb{Z}^*_N$ et ressort le chiffrement*

$$c := [(1+N)^m \cdot r^{N^k} \ mod\ N^{k+1}].$$

- ***Dec :*** *chaque partie i calcule $c_i = c^{s_i}$ $i \in \{1, ..., l\}$ et prouve à l'aide d'une preuve à divulgation nulle que $log_c(c_i) = log_v(v_i)$, ce qui montre que les c_i ont été calculés avec une clef secrète partagée valide. Supposons que nous ayons au moins m c_i valides. Choisissons un sous-ensemble S d'exactement m c_i et calculons :*

$$c' = \prod_{i \in S} c_i^{\lambda^S_{0,i}} \ mod\ N^{k+1}$$

où

$$\lambda^S_{0,i} = \prod_{i' \in S \setminus i} \frac{-i'}{i - i'}$$

un polynôme de Lagrange. Nous obtenons alors $c' = c^{a_0} = c^{\phi(N)} \ mod\ N^{k+1}$ et il nous reste à calculer

$$\frac{c' - 1}{N} \phi(N)^{-1} \mod N^k = m.$$

Nous avons bien $c' = c^{a_0}$ car

$$c' = \prod_{i \in S} c_i^{\lambda^S_{0,i}} \ mod\ N^{k+1} = c^{\sum_{i \in S} s_i \lambda^S_{0,i}} \ mod\ N^{k+1} = c^{\sum_{i \in S} s_i \prod_{i' \in S \setminus i} \frac{-i'}{i-i'}} \ mod\ N^{k+1} = c^{a_0}$$

et le calcul

$$\frac{c^{'}-1}{N}\phi(N)^{-1} \mod N^k = m.$$

a déjà été vérifié à la section (3.3).

Nous avons donc réussi à partager la clef secrète sk en l parties. Dorénavant, lorsque nous parlerons d'une autorité A qui possède la clef secrète du chiffrement, il faudra sous-entendre un ensemble de l autorités qui se partagent sk.

5.2 Vérifiabilité universelle

La vérifiabilité permet de détecter, sans compromettre le secret du vote, une éventuelle défaillance du système qui se produirait pendant la procédure de vote à cause d'une erreur du logiciel, d'une erreur humaine ou d'une tentative de triche. Elle prévoit que l'électeur obtienne la preuve que sa voix a été comptabilisée correctement et qu'elle n'a été modifiée d'aucune manière. La vérifiabilité constitue un outil solide pour assurer le bon déroulement d'un scrutin et rendre l'électeur confiant.

Reprenons le cas de figure d'un vote pour l candidats parmi L présenté à la section (4.3). Nous avions E électeurs, une autorité A qui calcule le résultat et un tableau T où les votants enregistrent leurs paires de votes et de preuves. Une fois que chaque électeur a enregistré ses L paires (une pour chaque candidat), la phase de vote se termine et le décompte commence. Nous avons déjà présenté à la section (4.3) comment l'autorité A agit. Elle vérifie que tous les votes sont valides, c'est-à-dire que la preuve associée est valide, puis calcule

$$c_{*j} = \prod_{i=1}^{E-k_j} c_{ij} \quad j \in \{0, ..., L-1\}$$

où k_j est le nombre de votes non valides pour le candidat j.

En réalité, ce processus peut être réalisé pour un quidam, vu que tous les votes et leurs preuves associées sont enregistrés dans le tableau public T.

Une fois que tous les c_{*j} sont calculés, l'autorité les déchiffre pour obtenir les résultats à l'aide de la clef secrète sk que seule elle-même connait :

$$result_j = Dec_{sk}(c_{*j}) = \sum_{i=1}^{E-k_j} v_{ij} \quad j \in \{0, ..., L-1\}.$$

Ce déchiffrement ne peut être fait que par l'autorité A et l'électeur doit donc lui faire confiance. En réalité, ce n'est pas le cas. Pour chaque $result_j \quad j \in \{0, ..., L-1\}$ publié, l'autorité va devoir prouver qu'elle connait le secret s_j sous-jacent à c_{*j}. Pour cela, elle va calculer

$$\frac{c_{*j}}{(1+N)^{result_j}} = \frac{(1+N)^{result_j} s_j^N \ mod \ N^{k+1}}{(1+N)^{result_j}} = s_j^N \ mod \ N^{k+1}$$

puis calculer

$$d = N^{-1} \ mod \ \phi(N)$$

pour enfin trouver

$$s_j^{Nd} \ mod \ N^{k+1} = s_j.$$

Elle connait effectivement $\phi(N)$ vu que c'est la clef secrète.

N'importe qui voulant alors vérifier que le résultat pour le candidat j est correct pourra alors calculer

1. c_{*j} via le tableau T ;
2. c'_{*j} via le chiffrement de Pailler de $result_j$ et s_j

et vérifier que $c_{*j} = c'_{*j}$.

Comme chaque c_{*j} possède une unique décomposition $(result_j, s_j)$ par la propriété d'isomorphisme de groupe, l'autorité ne peut pas tricher. Il est évident que le fait de connaitre la décomposition $(result_j, s_j)$ de chaque c_{*j} n'apporte aucune information sur la clef secrète à un potentiel adversaire, car le chiffrement de Paillier est « CPA-secure ».

Nous avons donc construit jusqu'ici un chiffrement « CPA-Secure »efficace qui est utile au vote électronique de par ses propriétés homomorphiques : le chiffrement de Paillier. Nous avons développé un schéma de preuves à divulgation nulle permettant de certifier que les bulletins, chiffrés par le cryptosystème de Paillier, sont valides et ce, pour un système de vote où l'on peut voter jusqu'à l parmi L. Nous venons de développer un schéma de preuves empêchant la corruption de l'autorité et permettant à tout le monde de vérifier que le résultat final est bien correct. Il nous reste à développer un schéma de preuves permettant de rendre inutile l'achat de votes (chapitre 6) et enfin, à partir de ce qui aura été présenté jusque-là, à développer une méthode de chiffrement adaptée au STV permettant de transférer les votes valides à différents candidats selon leur ordre de préférence sur les listes des électeurs sans avoir besoin de déchiffrer ceux-ci (chapitre 7).

Chapitre 6

Résistance à la coercition

Tous les protocoles, définitions et algorithmes des sections (6.1) et (6.2) sont tirés de [13]. Les protocoles, définitions et algorithmes de la section (6.3) sont tirés de [14] et [15].

Lors d'élections, il arrive que des candidats désirent être élus à tout prix et aillent acheter les électeurs, ce qui est totalement antidémocratique et inacceptable. Ce problème est des plus importants dans le cadre du STV où les électeurs peuvent classer les candidats par ordre de leurs préférence et donc créer des bulletins de vote unique en leur genre. C'est ce que décrit la notion de « Résistance à la coercition » (section 3.2.3).

Dans le cas du vote pour l candidats parmi L, où l'on doit juste voter « oui »ou « non », ce problème de coercition ne se pose pas lorsque l'on dévoile uniquement les résultats finaux de l'élection. Mais ce travail traite bien du STV et dans ce cas, de gros problèmes peuvent survenir. Dans ce chapitre, nous allons illustrer ces problèmes en décrivant les méthodes utilisées par des pays utilisant le STV, puis nous allons dégager une solution réelle.

6.1 Exemple concret

En Irlande et en Australie, le STV est utilisé, mais sans cryptographie. Ces pays sont donc amenés à faire des compromis entre la vérifiabilité universelle et la résistance à la coercition, qui sont des notions quelque peu opposées. En effet, nous pourrions soit rendre tous les votes publiques et ainsi obtenir la vérifiabilité universelle, ce qui laisserait les élections ouvertes à la coercition, soit garder tous les votes secrets pour empêcher la coercition, ce qui empêcherait quiconque de vérifier que le résultat annoncé est correct. Dans les deux cas, nous obtenons des solutions qui vont à l'encontre de toute démocratie.

L'Australie et l'Irlande ont décidé de publier tous les votes des électeurs rendant ainsi les élections fortement soumises au problème de coercition. Je dis bien fortement, car à la différence d'un scrutin classique où les électeurs expriment leurs préférences pour un candidat ou un ensemble de candidats sans les classer par ordre de préférence, le STV possède un nombre considérable de possibilités de votes ; ce nombre revient à la factorielle du nombre de candidats pouvant être élus. Par exemple, en Australie, une élection au Sénat se fait parmi environ 70 candidats. Il est dès lors facile d'imaginer un sénateur demander à une personne lambda de le mettre en tête de son bulletin et d'utiliser les 69! autres possibilités de votes restantes pour identifier cet être lambda parmi tous les bulletins de vote. En effet, parmi les 70 candidats en liste, près de 50 ont peu ou pas du

tout de chances d'obtenir un siège. Il est dès lors très facile d'utiliser ces candidats pour signer son bulletin de vote. Un Coercer [1] pourrait très bien arriver à manipuler près de 1000 bulletins à son avantage.

Certaines études se sont donc attelées à ce problème et ont proposé différentes solutions.

Solutions

Publication des parties de bulletins consommées

Dans ce type de solution, seule la partie du bulletin de l'électeur qui a déjà été consommée est rendue publique. Cependant, cette solution n'en est pas vraiment une, comme nous le montre la démarche suivante. Supposons qu'un Coercer demande à un quidam de voter pour lui en première position, puis de « signer », c'est-à dire d'identifier son bulletin à l'aide d'une permutation de candidats ayant une très faible probabilité d'être élus (nous appellerons ce type de candidats « Identifiants »). Le déroulement le plus probable serait alors que le Coercer soit élu (vu qu'il triche) et qu'il possède de surcroit un certain nombre de voix en surplus qui seront transférées aux candidats ayant peu de chances d'être élus voire aucune. Ce transfert de voix implique que de nombreux candidats qui identifient le bulletin du quidam seront éliminés et que donc une bonne partie du bulletin de vote du quidam sera divulguée permettant ainsi au Coercer de vérifier que le quidam a bien respecté ses instructions.

Publication partielle des résultats

Dans ce type de solution, après chaque étape du dépouillement, on divulgue le résultat de chaque candidat, ce qui entraine que tout le monde peut voir l'évolution des candidats dans la course à l'élection. Un Coercer peut donc demander à un quidam de mettre un candidat « Identifiant »en tête de liste, suivi de lui-même. Lorsque le candidat « Identifiant »est éliminé, il n'a plus qu'à vérifier que son nombre de voix a augmenté. Plus il y a des candidats « Identifiants », plus il y a de possibilité de coercition. Cette solution n'en est donc également pas une.

6.2 Solution cryptographique

La cryptographie va nous permettre de trouver une solution viable assurant de concilier la vérifiabilité universelle et la coercition. Le gros problème du STV est qu'à chaque étape du dépouillement, les votes doivent être transférés entre candidats selon l'ordre demandé par les électeurs, mais nous ne pouvons pas savoir quel est cet ordre ni ce que valent les votes.

Une telle chose parait très difficile à obtenir. Après réflexion, nous avons opté pour la solution suivante. Supposons qu'il y ait L candidats en liste. Nous allons alors générer les $L!$ listes de bulletins possibles et demander à chaque électeur de voter pour une seule liste. Nous nous retrouvons alors dans le protocole de vote de la section (4.3) avec $l = 1$. Le gros problème de cette solution est que s'il y a beaucoup de candidats en jeu, chaque

1. Personne qui voudrait obliger quelqu'un à voter de telle ou telle façon par la menace, l'argent, ...

électeur devra émettre un nombre considérable de votes ($L!$). Remarquons que le but de ce chapitre est de développer une méthode qui permette à l'élection de résister à la coercition. Nous resterons très évasif sur le mode de décompte des votes, etc. pour nous y attarder plus en détails dans le prochain chapitre.

Supposons que nous ayons $L = 3$ candidats en liste. Nous obtenons donc, après multiplication de tous les bulletins valides pour chaque liste, le tableau suivant contenant le chiffré des résultats de chaque liste.

1	2	3	4	5	6
c_{*1}	c_{*2}	c_{*3}	c_{*4}	c_{*5}	c_{*6}
A	A	B	B	C	C
B	C	A	C	A	B
C	B	C	A	B	A

Ce tableau peut être recalculé par tout un chacun et est donc public. Une solution serait de déchiffrer chaque $c_{*j} \quad j \in \{1, ..., L!\}$, mais alors notre méthode ne serait absolument pas résistante à la coercition. Le vote d'une personne ayant voté de manière inattendue serait directement repérable. Pour éviter la coercition tout en maintenant la vérifiabilité universelle, nous devons donc révéler uniquement l'information nécessaire au dépouillement. Lors de chaque étape du dépouillement, nous allons répondre à différentes questions dans un ordre précis.
La première question à se poser est : « Existe-t-il un ou plusieurs candidats qui dépassent le quotient électoral ? ». Nous répondrons à cette question à l'aide de la méthode de comparaison de Paillier qui consiste à déterminer si le résultat obtenu par une liste j est plus grand ou plus petit que le quotient électoral et ce, sans avoir besoin de connaitre le nombre exact des voix que cumule une liste (c_{*j} - quota $\overset{?}{\geqslant} 0$). La méthode de comparaison de Pailler est décrite dans la prochaine section. Si un candidat (ou plusieurs) dépasse(nt) le quotient, nous pouvons dévoiler publiquement combien de voix ce(s) candidat(s) a(ont) obtenu et transférer proportionnellement les voix en surplus aux candidats en deuxième position là où le(s) candidat(s) élu(s) étai(n)t en première place. Remarquons que si aucun candidat ne dépasse le quotient, nous ne possédons aucune information sur l'ordre de préférence des candidats.
Nous pouvons alors passer au deuxième cas du dépouillement. Si aucun candidat n'est élu, nous devons répondre à une deuxième question : « Quel candidat possède le moins de voix ? ». De la même manière que précédemment, nous utiliserons la méthode de comparaison de Paillier pour déterminer le candidat possédant le moins de voix. Dans ce cas-ci, la méthode pourrait, dans un cas très peu probable où l'on comparerait chaque candidat par ordre décroissant de nombre de voix, connaitre l'ordre exact de préférence des candidats à l'étape considérée. Nous obtiendrions alors une information non nécessaire au dépouillement. Nous devrons donc proposer une méthode permettant d'éviter ce cas particulier. La référence [13] nous propose une idée d'algorithme qui sera utile pour résoudre notre problème. Présentons donc cette idée.

Supposons qu'il y ait L candidats et que les élections aient eu lieu. A l'aide de la méthode de comparaison de Paillier, nous pouvons déterminer à chaque étape la relation d'ordre de chaque candidat par rapport à un autre sans pour autant connaitre le nombre de voix que chaque candidat possède. Créons une matrice de taille $L \times L$ notée LL de

comparaison des candidats définie de la manière suivante :

$$Dec_{sk}(LL_{ij}) = \begin{cases} -1 & \text{si le candidat } i \text{ est préféré au candidat } j, \\ 0 & \text{sinon.} \end{cases}$$

Les éléments diagonaux de la matrice n'ont pas d'importance. Les éléments de la matrice sont cryptés à l'aide du chiffrement Paillier. Pour connaitre la position des candidats, il faut créer un vecteur de comparaison vc à c compartiments définis par :

$$vc_j = -\prod_{i=0}^{L-1} LL_{ij} \qquad\qquad j \in \{0, ..., L-1\}.$$

Lélément j du vecteur vc contient de manière chiffrée la position du candidat j dans les préférences. Le candidat possédant la valeur 0 est le candidat préféré tandis que le candidat possédant la valeur $L-1$ est celui qui a reçu le moins de voix.
Ce n'est pas facile à comprendre, donc illustrons ceci par un exemple. Supposons que nous ayons 5 candidats A, B, C, D et E et qu'au premier tour nous obtenions les résultats suivants :

A	B	C	D	E
$Enc(20)$	$Enc(5)$	$Enc(50)$	$Enc(30)$	$Enc(40)$

où, pour faciliter la notation, nous avons noté $Enc(20)$ au lieu de $Enc_{pk}(20, r),\ r \in \mathbb{Z}_N^*$. Nous maintiendrons cet abus de notation dans les cas où aucune confusion ne peut avoir lieu. La matrice de comparaison des candidats sera alors de la forme :

LL =

A	B	C	D	E
×	Enc(-1)	Enc(0)	Enc(0)	Enc(0)
Enc(0)	×	Enc(0)	Enc(0)	Enc(0)
Enc(-1)	Enc(-1)	×	Enc(-1)	Enc(-1)
Enc(-1)	Enc(-1)	Enc(0)	×	Enc(0)
Enc(-1)	Enc(-1)	Enc(0)	Enc(-1)	×

vc =

Enc(3)	Enc(4)	Enc(0)	Enc(2)	Enc(1)

Rappelons que nous voulons déterminer les candidats ayant le moins de préférence, c'est-à-dire les candidats qui possèdent dans le vecteur vc le plus grand numéro en clair. Supposons pour l'instant qu'il n'y a pas d'égalité dans le nombre de voix des candidats. Comme nous savons combien de candidats restent en jeu (ici 5), nous connaissons le plus grand nombre gn dans vc (ici $gn = 5 - 1 = 4$). Il ne reste plus qu'à déterminer quel candidat possède ce numéro précis (4) sans pour autant décrypter quoi que ce soit. Cela peut se faire grâce à la méthode de comparaison de Paillier en comparant le contenu de chaque chiffré avec $gn - 1$. Si le contenu d'un chiffré est plus grand que $gn - 1 (= 3\ ici)$, c'est qu'il représente le plus grand nombre en clair et nous avons trouvé notre candidat à éliminer (B). Supposons à présent que nous ayons des égalités entre les candidats. Par exemple, nous obtenons les résultats suivants :

A	B	C	D	E
$Enc(5)$	$Enc(5)$	$Enc(50)$	$Enc(40)$	$Enc(40)$

et donc le vecteur :

vc =	Enc(3)	Enc(3)	Enc(0)	Enc(1)	Enc(1)

Nous allons procéder comme précédemment. Nous allons comparer chaque chiffré à $gn - 1 (= 3\ ici)$, mais cette fois nous n'allons trouver aucun chiffrement qui soit plus grand que $gn - 1$. Cela signifie que nous avons une ou plusieurs égalités. Nous allons alors comparer chaque chiffré à $gn - 1 - 1 (= 2\ ici)$ et ainsi de suite jusqu'à trouver les chiffrés possédant les messages en clair les plus grands. Dans notre exemple nous trouvons au deuxième essai les deux chiffrés possédant le contenu le plus grand (3) et donc les candidats à éliminer (A et B). Notre méthode va donc éliminer l'ensemble des candidats ayant le moins de vote de préférence. Cette remarque est importante, car dans la pratique cela ne se passe pas comme ça. Par exemple, l'Australie qui pratique le STV procède de deux manières différentes dans ses différents états lorsqu'elle est face à une égalité. Soit elle élimine un candidat aléatoirement parmi ceux à éliminer, soit elle classe de manière préliminaire au vote l'ensemble des candidats et élimine le candidat le plus bas dans cette liste parmi les candidats à éliminer.

La méthode décrite ci-dessus nous permet donc de décompter les votes du STV sans avoir à déchiffrer ceux-ci, ce qui rend notre protocole de vote résistant à la coercition. Il nous reste à présenter le méthode de comparaison de Paillier afin de rendre notre méthode utilisable.

6.3 Comparaison de Paillier

Nous développons ici une méthode de comparaison de Paillier. Elle a pour but de déterminer lequel des deux messages x et y cachés dans les chiffrés de Paillier $c_1 = Enc_{pk}(x)$ et $c_2 = Enc_{pk}(y)$ est le plus grand, sans pour autant donner une quelconque information supplémentaire sur ces messages.
Plus formellement, nous devons trouver un protocole permettant d'évaluer de manière sécurisée la fonction

$$f(x, y) = [x > y]$$

où la notation $[B]$ est définie, pour une condition B,comme $[B] = 1$ si B est vraie et $[B] = 0$ si B est fausse et où les messages x et y sont chiffrés.

Pour cela, nous allons utiliser la méthode décrite dans [14]. Supposons que x et y soient donnés sous leurs formes binaires (k bits) et chiffrés bits à bits. Nous avons donc pour le message x, la séquence $Enc_{pk}(x_{k-1}), ..., Enc_{pk}(x_0)$ tel que $x = \sum_{i=0}^{k-1} x_i 2^i$. Nous verrons un peu plus tard une méthode pour passer de $c_1 = Enc_{pk}(x)$ à cette séquence.
Pour bien comprendre la méthode, nous allons l'expliquer en considérant que nous travaillons en clair et non en chiffré, puis nous l'adapterons en chiffré. Nous pouvons déterminer si $x > y$ en les comparant bits à bits à l'aide de la formule de récurrence :

$$\begin{aligned} t_0 &= 0, \\ t_{i+1} &= (1 - (x_i - y_i)^2)t_i + x_i(1 - y_i) \end{aligned}$$

où t_k est la valeur de la fonction $f(x, y) = [x > y]$.
Nous commençons au bit de poids le plus faible pour arriver au bit de poids le plus fort dans le but de ne donner aucune information sur le moment où x devient plus grand ou

égal à y. Expliquons plus en détails cette formule.
Le terme $x_i(1-y_i)$ vaut 1 si le bit x_i vaut 1 et le bit y_i vaut 0. Il permet donc de déterminer si x est plus grand que y pour les i^e premiers bits. Si c'est le cas, t_{i+1} vaut 1 sinon 0.
Le terme $(1-(x_i-y_i)^2)t_i$ permet soit de transférer la valeur de t_i à t_{i+1} lorsque x_i et y_i sont égaux, soit de remettre à 0 t_i et transmettre à t_{i+1} la valeur du terme $x_i(1-y_i)$ lorsque x_i et y_i sont différents.
La valeur de t_k sera alors bien 1 si $x > y$ et 0 sinon.

Adaptons maintenant cette formule pour travailler en chiffré et non en clair, c'est à dire que nous connaissons $Enc_{pk}(x_{k-1}), ..., Enc_{pk}(x_0)$ et $Enc_{pk}(y_{k-1}), ..., Enc_{pk}(y_0)$ et non plus $x_{k-1}, ..., x_0$ et $y_{k-1}, ..., y_0$. Nous voulons également qu'à chaque étape de la récurrence, seul soit connu $Enc_{pk}(t_i)$ et non plus t_i. Rappelons la propriété d'homomorphisme du chiffrement de Paillier :

$$\begin{aligned} a+b &= Enc_{pk}(a)Enc_{pk}(b) \\ a-b &= Enc_{pk}(a)/Enc_{pk}(b) \end{aligned}$$

Remarquons également que :

$$Enc_{pk}(a)^b = Enc_{pk}(ab).$$

La formule de récurrence devient alors :

$$\begin{aligned} Enc_{pk}(t_0) &= Enc_{pk}(0), \\ Enc_{pk}(t_{i+1}) &= Enc_{pk}((1-(x_i-y_i)^2)t_i)Enc_{pk}(x_i(1-y_i)). \end{aligned}$$

Nous pouvons calculer $Enc_{pk}((1-(x_i-y_i)^2)t_i)$ par

$$Enc_{pk}((1-(x_i-y_i)^2)t_i) = (Enc_{pk}(1)/Enc_{pk}((x_i-y_i)^2)))^{t_i}.$$

Nous avons donc un problème : comment calculer la formule suivante

$$Enc_{pk}(ab) = Enc_{pk}(a)^b$$

alors que nous ne pouvons pas connaitre b, mais seulement son chiffrement ?
Le protocole (6.3.3) développé en fin de section va nous donner cet outil.

Nous avons aussi

$$\begin{aligned} Enc_{pk}((x_i-y_i)^2) &= Enc_{pk}(x_i^2+y_i^2-2x_iy_i) \\ &= Enc_{pk}(x_i+y_i-2x_iy_i)\ (\star) \\ &= \frac{Enc_{pk}(x_i)Enc_{pk}(y_i)}{Enc_{pk}(2x_iy_i)}. \end{aligned}$$

Le résultat $(\star)$ s'obtient, car x_i et y_i valent soit 0 soit 1. Le chiffré $Enc_{pk}(2x_iy_i)$ peut se calculer à l'aide du protocole (6.3.3).
Nous pouvons calculer $Enc_{pk}(x_i(1-y_i))$ par

$$Enc_{pk}(x_i(1-y_i)) = Enc_{pk}(x_i)/Enc_{pk}(x_iy_i)$$

où $Enc_{pk}(x_i y_i)$ peut se calculer à l'aide du protocole (6.3.3).

La formule complète se calcule donc par

$$\begin{aligned} Enc_{pk}(t_0) &= Enc_{pk}(0), \\ Enc_{pk}(t_{i+1}) &= (\frac{Enc_{pk}(1)}{\frac{Enc_{pk}(x_i)Enc_{pk}(y_i)}{Enc_{pk}(2x_i y_i)}})^{t_i}(Enc_{pk}(x_i)/Enc_{pk}(x_i y_i)). \end{aligned}$$

La valeur de t_k nous dit si nous avons $x > y$ ou non.

Transformer un chiffré en sa séquence de bits

Nous allons développer un protocole pour transformer un chiffré $Enc_{pk}(x)$ en m chiffrés $Enc_{pk}(x_{m-1}), ..., Enc_{pk}(x_0)$ tels que $x = \sum_{i=0}^{m-1} x_i 2^i$ où m est la longueur de bit de N^k, où $\mathbb{Z}_{N^k}$ est l'espace des messages en clair du chiffrement de Paillier généralisé. Pour cela, nous allons utiliser le protocole *BITREP* décrit dans [15]. Ce protocole nécessite le partage de la clef secrète en l parties qui a déjà été décrit en section (5.1).

Protocole 6.3.1 *BITREP*

Entrée : *le chiffré $Enc_{pk}(x)$ où $x \in \mathbb{Z}_{N^k}$ long de m bits.*
Sortie : *m chiffrés $Enc_{pk}(x_{m-1}), ..., Enc_{pk}(x_0)$ tels que $x = \sum_{i=0}^{-1} x_i 2^i$.*

1. *m parties génèrent aléatoirement m bits chiffrés $Enc_{pk}(r_{m-1}), ..., Enc_{pk}(r_0)$ tels que $r = \sum_{i=0}^{m-1} r_i 2^i \in \{0, ..., N^k - 1\}$ où N^k est long de m bits. Ces m bits chiffrés sont rendus publics.*

2. *Les parties calculent $c = Enc_{pk}(x) \prod_{i=0}^{m-1} Enc_{pk}(r_i)^{2^i}$, déchiffrent c et obtiennent $y = x + r \bmod N^k$. Les parties montrent qu'elles ont déchiffré correctement c en publiant y et s, le secret sous-jacent au chiffré du message y par le chiffrement de Paillier généralisé.*

3. *Les parties chiffrent les m bits de y et montrent à l'aide du protocoles (4.2.2) développé au chapitre 4 qu'ils sont valides.*

4. *Les parties déterminent les chiffrés $Enc_{pk}(z_m), ..., Enc_{pk}(z_0)$ tels que $z = x$ ou $z = x - N^k$, avec z_m un bit signé à l'aide du protocole de soustraction (6.3.2) qui calcule $y - r = x \bmod N^k$.*

5. *Les parties réduisent la valeur de z modulo N^k en ajoutant à z la valeur $N^k z_m$ à l'aide d'un protocole d'addition.*

Présentons ce protocole en détails.
A l'étape 1., m parties génèrent de leur côté 1 bit chiffré de telle sorte que la valeur $r = \sum_{i=0}^{m-1} r_i 2^i \in \{0, ..., N^k - 1\}$ ne soit connue de personne.
A l'étape 2., tout le monde peut calculer c, car tous les facteurs sont publics. Les parties montrent qu'elles ont déchiffré correctement c en publiant y et s, le secret sous-jacent au chiffré du message y par le chiffrement de Paillier généralisé. Pour retrouver y, elles

utilisent le schéma (5.1.1). Pour retrouver s, elles utilisent la formule :

$$\begin{aligned} c^{N^{-1} \ mod \ \phi(N)} &= Enc_{pk}(m,s)^{N^{-1} \ mod \ \phi(N)} \\ &= Enc_{pk}(m(N^{-1} \ mod \ \phi(N)) \ mod \ N^k, s^{N^{-1} \ mod \ \phi(N)} \ mod \ N) \\ &= (1+N)^{m(N^{-1} \ mod \ \phi(N)) \ mod \ N^k}(s^{N(N^{-1} \ mod \ \phi(N))} \ mod \ N) \\ &= (1+mN(N^{-1} \ mod \ \phi(N) \ mod \ N^k))s \\ &= (1+0)s \\ &= s. \end{aligned}$$

Tout le monde connait y et donc les bits $y_{m-1}, ..., y_0$ qui le composent. Personne ne peut déduire x de y, car r est uniformément distribué dans $\mathbb{Z}_{N^k}$.
A l'étape 3., les parties peuvent montrer qu'elles ont bien chiffré les bits de y en montrant pour tout $i \in \{1, ..., m\}$, que $Enc(y_i)$ est un chiffré de 0 ou que $Enc(y_i)/((1+N)$ est un chiffré de 0 en fonction de la valeur connue de y_i (0 ou 1) à l'aide du protocole (4.2.2).
Les étapes 4. et 5. s'expliquent ensemble avec l'utilisation du protocole de soustraction suivant.

Protocole 6.3.2 *Soustraction de bits*

Entrée : *les chiffrés* $Enc_{pk}(y_{m-1}), ..., Enc_{pk}(y_0)$ *et* $Enc_{pk}(r_{m-1}), ..., Enc_{pk}(r_0)$.
Sortie : $m+1$ *chiffrés* $Enc_{pk}(z_m), ..., Enc_{pk}(z_0)$ *tels que* $z = \sum_{i=0}^{m-1} z_i 2^i = (y-r) - z_m N^k$.

1. *Prendre le complément des bits de* r, *noté* $\bar{r}$ *à l'aide de la formule :*

$$\bar{r_i} = r_i + 1 - 2r_i \ i \in \{0, ..., m-1\}$$

2. *Calculer* $\bar{r}+1$ *sur les bits* $\bar{r_i}$ *à l'aide de la formule de récurrence pour* $i \in \{0, ..., m-1\}$ *:*

$$\begin{aligned} c_{-1} &= 0, \\ c_i &= \bar{r_i}h_i + \bar{r_i}c_{i-1} + h_i c_{i-1} - 2\bar{r_i}h_i c_{i-1}, \\ w_i &= \bar{r_i} + h_i + c_{i-1} - 2c_i \end{aligned}$$

où $h = \sum_{i=0}^{m-1} h_i 2^i = 1$.

3. *Calculer* $z = (\bar{r}+1) + y$ *à l'aide de la formule de récurrence pour* $i \in \{0, ..., m-1\}$ *:*

$$\begin{aligned} c_{-1} &= 0, \\ c_i &= w_i y_i + w_i c_{i-1} + y_i c_{i-1} - 2w_i y_i c_{i-1}, \\ z_i &= w_+ y_i + c_{i-1} - 2c_i. \end{aligned}$$

Puis calculer z_m *comme le complément à bit de* c_{m-1} *par :*

$$z_m = c_{m-1} + 1 - 2c_{m-1}.$$

Le but de ce protocole est de calculer la valeur $z = y - r$. Pour cela, nous allons utiliser la méthode de complément à 2 qui dit que $(y-r)_{10} = (y+(-r))_{10} = (y+\bar{r}+1)_2$ où $\bar{r}$ est le complément des bits de r, c'est-à-dire que là où les bits de r valent 1, les bits de $\bar{r}$ valent 0 et là où les bits de r valent 0, les bits de $\bar{r}$ valent 1. Les formules dans le

protocole ci-dessus sont exprimées en clair pour plus de facilité, mais sont évidement à utiliser avec les chiffrés associés. Elles sont constituées d'additions et de soustractions qui sont faciles à transformer en chiffrés à l'aide de la propriété d'homomorphisme du chiffrement de Paillier. Il y a également des multiplications qui peuvent être effectuées en chiffré à l'aide du protocole (6.3.3) présenté ci-dessous. Nous obtenons alors la valeur $z = y - r$ que nous devons encore réduire modulo N^k. Pour cela, nous convertissons N^k en sa séquence de bits que nous chiffrons. Nous multiplions alors le contenu des chiffrés des bits de N^k avec le contenu du chiffré de z_m à l'aide du protocole (6.3.3). Nous possédons alors une séquence $Enc(g_i) = Enc(N_i^k)^{z_m} \quad i \in \{0, ..., m-1\}$. Nous utilisons ensuite la formule de récurrence pour $i \in \{0, ..., m-1\}$ suivante, mais sous sa forme chiffrée :

$$\begin{aligned} c_{-1} &= 0, \\ c_i &= z_i g_i + z_i c_{i-1} + g_i c_{i-1} - 2 z_i g_i c_{i-1}, \\ x_i &= z_i + g_i + c_{i-1} - 2 c_i \end{aligned}$$

C'est ce que décrit l'étape 5. du protocole (6.3.1).

Nous avons donc réussi à ressortir les bits chiffrés du chiffré d'un massage x, et ce, sans révéler quoi que ce soit sur ce dernier.

Multiplication du contenu de deux chiffrés

Étant donné les chiffrés $Enc_{pk}(a)$ et $Enc_{pk}(b)$, nous allons développer un protocole pour calculer de manière sécurisée le chiffré $Enc_{pk}(ab)$. Ce protocole nécessite le partage de la clef secrète en l parties comme cela a déjà été décrit en section (5.1). Il se décrit formellement comme ceci :

Protocole 6.3.3 *Mutliplication du contenu de deux chiffrés*

1. *Les l parties vont se partager le secret a de la manière suivante :*
   (i) Chaque partie i choisit une valeur aléatoire d_i et la chiffre comme $Enc_{pk}(d_i)$. Nous notons $d = \sum_{i=1}^{l} d_i$.
   (ii) Les l parties utilisent le protocole (5.1) pour déchiffrer $Enc_{pk}(a)Enc_{pk}(d_1)...Enc_{pk}(d_l)$ et obtiennent (a+d)
   (iii) La partie 1 pose $a_1 = (a + d) - d_1$ et les autres posent $a_i = -d_i \; i \in \{2, ..., l\}$. Nous avons donc que $a = \sum_{i=1}^{l} a_i$. Chaque partie i chiffre son a_i, le publie et tout le monde peut alors vérifier que $Enc_{pk}(a_1)...Enc_{pk}(a_l) = Enc_{pk}(a)$.

2. *Chaque partie i va calculer le chiffré $Enc_{pk}(b)^{a_i} = Enc_{pk}(ba_i)$ et prouver à l'aide du protocole (6.3.4)que son chiffré est correct.*

3. *Les parties calculent alors $Enc_{pk}(ba_1)...Enc_{pk}(ba_l) = Enc_{pk}(a_1+...+a_l = a)Enc_{pk}(b) = Enc_{pk}(ab)$.*

Décrivons ce protocole en détails.
Le but de la première étape est de partager le secret a en clair entre les l parties. Au point (i), chaque partie i va choisir aléatoirement un secret d_i. Au point (ii), les parties vont combiner leurs secrets pour former la valeur chiffrée $Enc(a + d)$ pour ensuite la déchiffrer ensemble à l'aide du protocole de partage de la clef secrète (5.1). Toutes les

parties obtiennent alors la valeur en clair $(a+d)$ qui ne contient aucune information sur a, car d est pris aléatoirement. Au point (iii), chaque partie i utilise sont secret d_i pour se partager le secret a. La première partie calcule $a_1 = a + d - d_1$ et les autres calculent $a_i = -d_i$, de tel sorte que $\sum_{i=1}^{l} a_i = a$. Bien sûr, ce dernier calcul ne peut jamais avoir lieu, car nous voulons que a reste secret. Nous supposons en effet que parmi les l parties, au moins la moitié est honnête et que donc il est impossible pour quiconque de retrouver a. Par contre, chaque partie i chiffre son a_i et publie le chiffré de telle sorte que tout le monde puisse vérifier que le produit des chiffrés des a_i donne bien le chiffré a. Nous avons donc au bout de la première étape que chaque chiffré $Enc(a_i)$ est connu de tous, que $Enc_{pk}(a_1)...Enc_{pk}(a_l) = Enc_{pk}(a)$ et que chaque a_i est connu en clair uniquement par la partie i.
A la deuxième étape, chaque partie i va calculer $Enc_{pk}(b)^{a_i} = Enc_{pk}(ba_i)$. En effet, le chiffré $Enc_{pk}(b)$ est public et la valeur de chaque a_i est connue en clair par la partie i. Il faut juste s'assurer que toutes les parties calculent correctement les chiffrés $Enc_{pk}(ba_i)$, $i \in \{1, ...l\}$. Pour cela, nous allons utiliser le protocole à divulgation nulle suivant, qui permet à un prouveur P de convaincre un vérifieur V que trois chiffrés c_a, c_b, c_c contenant les valeurs a, b, c sont tels que $ab = c \ mod \ N^k$.

Protocole 6.3.4 *Mutliplication*

Entrée : $N, c_a = Enc_{pk}(a, s_a), c_b = Enc_{pk}(b, s_b), c_c = Enc_{pk}(c, s_c)$.
Entrée secrète de P ***:*** $\phi(N)$ *la clef secrète .*

1. *P choisit aléatoirement une valeur $d \in \mathbb{Z}_{N^k}$ et envoie à V les chiffrés $c_d = Enc_{pk}(d, s_d), c_{db} = Enc_{pk}(b, s_b)^d = Enc_{pk}(db, s_{db})$.*

2. *V envoie un challenge e, long de t bits et l'envoie à P.*

3. *P calcule le chiffré $c_a^e c_d = Enc_{pk}(ea + d, s_a^e s_d)$, le déchiffre et envoie à V les valeurs $f = ea + d \ mod \ N^k$ et $y_1 = s_a^e s_d \ mod \ N^{k+1}$. P calcule ensuite le chiffré $c_b^f (c_{db} c_c^e)^{-1} = Enc_{pk}(0, s_b^f (s_{db} s_c^e)^{-1} \ mod \ N^{k+1})$, le déchiffre et envoie $y_2 = s_b^f (s_{db} s_c^e)^{-1}$.*

4. *V vérifie si*
   - *$c_a^e c_d = Enc_{pk}(f, y_1) \ mod \ N^{k+1}$*
   - *$c_b^f (c_{db} c_c^e)^{-1} = Enc_{pk}(0, y_2) \ mod \ N^{k+1}$*

   et accepte la preuve si c'est le cas.

Ce protocole est complet, car :

$$\bullet c_a^e c_d = Enc_{pk}(ea + d, s_a^e s_d) = Enc_{pk}(f, y_1) \ mod \ N^{k+1}$$

si P calcule correctement f et y_1. Il peut retrouver f en déchiffrant le chiffrement de Paillier et il peut retrouver y_1 à l'aide de la formule

$$c^{N^{-1} \ mod \ \phi(N)} = s.$$

$$\begin{aligned} \bullet c_b^f (c_{db} c_c^e)^{-1} &= Enc(b, s_b)^{ea+d} (Enc(db, s_{db}) Enc(c, s_c)^e)^{-1} \\ &= Enc(eab + db, s_b^f)(Enc(-db, s_{db}^{-1}) Enc(-ce, s_c^{-e}) \\ &= Enc(eab + db - db - ce, s_b^f s_{db}^{-1} s_c^{-e}) \\ &= Enc(0, y_2)\ mod\ N^{k+1} \end{aligned}$$

si P calcule correctement y_2 et si $ab = c\ mod\ N^k$.
A la troisième étape, nous calculons le chiffré $Enc(ab)$ en multipliant l'ensemble des chiffrés produits par les l parties à l'étape 2.

Nous avons donc construit jusqu'ici un chiffrement « CPA-Secure »efficace qui est utile au vote électronique de par ses propriétés homomorphiques : le chiffrement de Paillier. Nous avons développé un schéma de preuves à divulgation nulle permettant de certifier que les bulletins, chiffrés par le cryptosystème de Paillier, sont valides et ce, pour un système de vote où l'on peut voter jusqu'à l parmi L. Nous avons également développé des protocoles permettant à tout le monde de vérifier que le résultat final est bien correct tout en rendant inutile l'achat de votes. Il nous reste enfin à combiner ce qui a été présenté jusqu'ici pour présenter le protocole de vote complet du STV qui respecte les propriétés fondamentales d'un vote démocratique : la confidentialité, la robustesse, la vérifiabilité universelle et la résistance à la coercition (chapitre 7).

Chapitre 7

Déroulement du vote de a à z

Tout ce qui est développé dans ce chapitre vient de l'auteur en s'inspirant de ce qui a été développé précédemment.

Dans ce chapitre, nous développons une méthode de vote complète pour le STV à l'aide des différents points et informations que nous avons décrits jusqu'ici. Ce chapitre clôture ce travail.

7.1 Émission des votes

Nous avons développé dans le chapitre 4 une méthode pour voter pour au plus l candidats parmi L sans ordre de préférence, les l candidats reçoivent le même nombre de voix.

Partons de la méthode de vote pour 1 candidat parmi L et généralisons-la. La difficulté est de permettre aux électeurs d'émettre un vote dans lequel les candidats sont classés par ordre de préférence et ce, sans augmenter de manière exorbitante le temps de calcul. La solution que nous appliquons est de générer toutes les permutations de candidats possibles pour former toutes les listes possibles et d'ensuite numéroter chacune de celles-ci. S'il existe L candidats, il y aura $L!$ listes à envisager. Au vu du grand nombre de listes à générer, nous ne pouvons employer notre méthode que pour un nombre limité de candidats. Cette méthode devient alors identique à celle : « Vote pour l candidats parmi L », de la section 4.3 avec $l = 1$. Illustrons ceci avec un exemple ; prenons A,B et C comme candidats. Comme $L = 3$, il existe $3! = 6$ permutations différentes.

1	2	3	4	5	6
A	A	B	B	C	C
B	C	A	C	A	B
C	B	C	A	B	A

L'électeur i, $i \in \{1, ..., E\}$ se retrouve devant ce tableau et applique le protocole (4.2.4) du chapitre 4 de la manière suivante. Il décide de donner son **unique** voix à une des 6 listes et chiffre 6 votes, un pour chaque liste, avec leurs preuves associées. Ces votes et leurs preuves sont placés dans un tableau visible par tous. Nous notons c_{ij} le chiffré de Paillier du vote de l'électeur i associé à la liste j et v_{ij} la valeur de ce vote (0 ou 1).

$$c_{ij} = Enc_{pk}(v_{ij}, r_{ij}) \quad j = 0, ..., L! - 1$$

$$Preuve_{ij}(c_{ij} \text{ ou } \frac{c_{ij}}{N+1} \text{ est un } chiffrement \text{ de } 0).$$

L'autorité A vérifie que chaque votant a bien voté 0 ou 1 pour chaque vote qu'il a émis et vérifie aussi que chaque votant a voté pour exactement une liste parmi les $L!$ possibles en vérifiant que :

$$Dec_{sk}(\prod_{j=0}^{L!-1} Enc_{pk}(v_{ij}, r_{ij})) = 1.$$

Tout un chacun est capable de vérifier si les preuves $Preuve_{ij}$ sont valides. L'autorité calcule également pour tout électeur i

$$v_{i*} = \sum_{j=0}^{L!-1} v_{ij},$$

et publie cette somme et le secret

$$r_{i*} = \prod_{j=0}^{L!-1} r_{ij}$$

associé pour que tout le monde puisse vérifier que les votes de l'électeur i sont valides ou non en vérifiant que $c_{i*} = Enc_{pk}(v_{i*}, r_{i*})$. Une fois que l'autorité a trié les différents votes émis et n'a gardé que ceux valides, nous obtenons à l'aide de la formule :

$$c_{*j} = \prod_{i=1}^{E-f} c_{ij} = Enc_{pk}(\sum_{i=1}^{E-f} v_{ij} \ mod \ N^k, \prod_{i=1}^{E-f} r_{ij} \ mod \ N)$$

où f est le nombre de votants frauduleux, un tableau contenant le chiffré des résultats de chaque liste (c_{*j}) :

1	2	3	4	5	6
c_{*1}	c_{*2}	c_{*3}	c_{*4}	c_{*5}	c_{*6}
A	A	B	B	C	C
B	C	A	C	A	B
C	B	C	A	B	A

Ce tableau peut être recalculé par tous et est donc public.

7.2 Dépouillement des votes

Vient alors la phase décisive et technique du dépouillement. Celle-ci se décompose en différentes étapes que nous pouvons exprimer à l'aide d'un algorithme (figure 7.1).

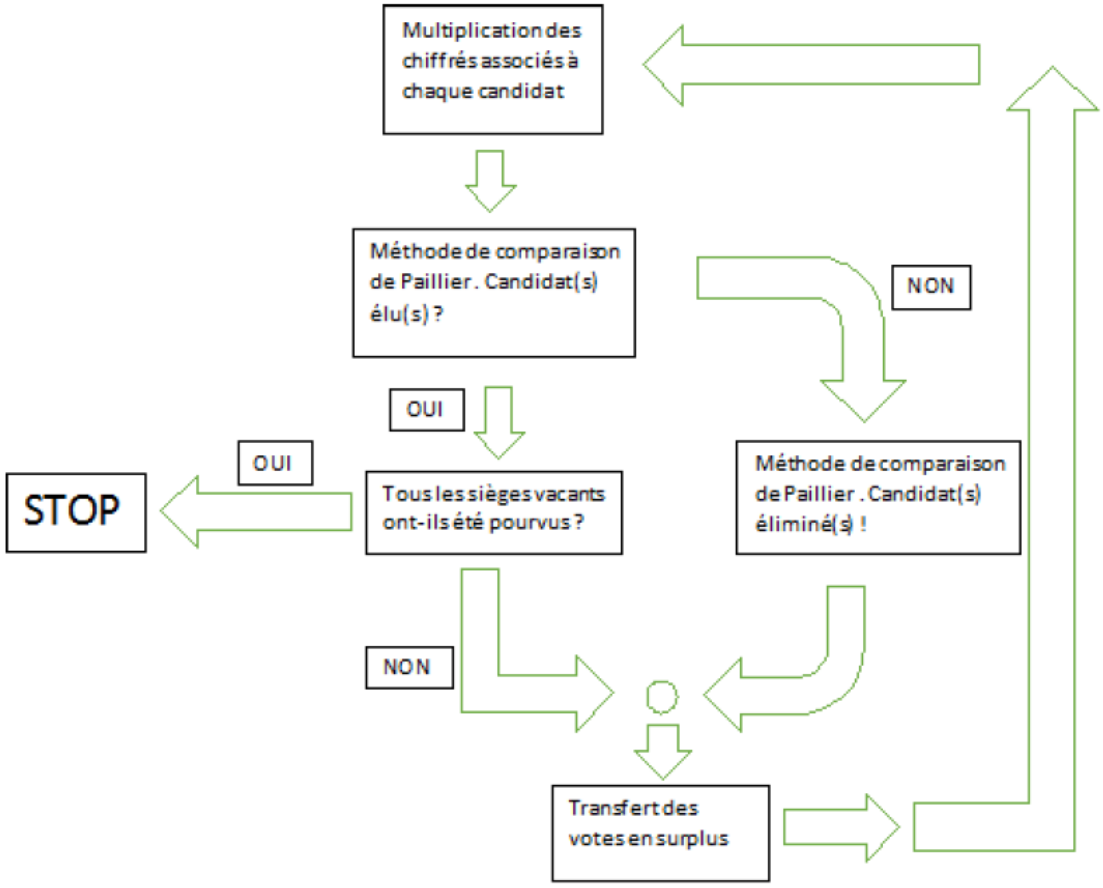

FIGURE 7.1 – Algorithme de dépouillement

La première étape « multiplication des chiffrés associés à chaque candidat »consiste à multiplier les chiffrés associés aux listes possédant le même candidat en première position. Ceci est totalement faisable vu que nous connaissons quel chiffré est associé à quelle liste. Dans l'exemple de la section précédente, cela consiste à calculer pour la première étape

- pour le candidat A : $c_A = c_{*1}c_{*2}$
- pour le candidat B : $c_B = c_{*3}c_{*4}$
- pour le candidat C : $c_C = c_{*5}c_{*6}$

La deuxième étape « Méthode de comparaison de Paillier. Candidat(s) élu(s) ? »consiste à déterminer si certains candidats atteignent le quotient électoral pour être élus. Pour cela, nous utiliserons la méthode de comparaison de Paillier (section 6.3) en comparant le chiffré total de chaque candidat avec le quotient électoral.
Si un chiffré dépasse le quotient, le candidat associé est élu. Nous déchiffrons alors le(s) chiffré(s) associé(s) à ce(s) candidat(s) et publions le(s) résultat(s). Nous connaissons alors les résultats des candidats élus ainsi que la valeur du quotient donc nous connaissons les votes en surplus à transférer. Nous passons alors à l'étape trois.
Si aucun chiffré ne dépasse le quotient électoral, nous devons passer à l'étape trois bis.

La troisième étape « Tous les sièges vacants ont-ils été pourvus ? »consiste à se demander s'il reste encore des candidats à élire. Si tous les sièges ont été pourvus, nous nous arrêtons et publions les résultats de chaque candidat, l'élection est finie. Sinon nous continuons en passant à l'étape quatre.

L'étape trois bis « Méthode de comparaison de Paillier. Candidat(s) éliminé(s) ! »se déroule lorsqu'aucun candidat ne dépasse le quotient. Nous devons déterminer quel(s) candidat(s) possède(nt)le moins de votes. Nous utilisons de nouveau la méthode de comparaison de Paillier (section 6.3) ainsi que la méthode décrite en section (6.2). Nous connaissons alors le nom des candidats à éliminer et pouvons déchiffrer leurs résultats. Notons que nous avons décidé d'éliminer l'ensemble des candidats se trouvant en position d'élimination. Toutes leurs voix doivent être transférées. Nous continuons le décompte des voix en passant à l'étape quatre.

L'étape quatre « Transfert des votes en surplus »consiste à transférer les votes en surplus des candidats élus ou éliminés. Pour le cas où nous éliminons des candidats, c'est très facile. Il suffit de supprimer dans chaque liste les candidats en question. Les chiffrés iront naturellement aux candidats se trouvant en tête de liste. La partie réellement délicate est celle pour les candidats élus. Lorsque un candidat est élu, nous devons transférer les votes en surplus de manière proportionnelle à chaque liste j où le candidat élu était en tête en utilisant la formule :

$$\frac{\text{voix du candidat élu à la liste } j \text{ où il était en tête} \times \text{voix en surplus du candidat élu}}{\text{voix totales du candidat élu}}$$

$$= \frac{v_{*j} \cdot vs}{vt}$$

$$= \text{nombre de voix restantes pour la liste } j \text{ où le candidat élu était en tête}$$

où nous notons

— voix du candidat élu à la liste j où il était en tête $\equiv v_{*j}$

— voix en surplus du candidat élu $\equiv vs$

— voix totales du candidat élu $\equiv vt$

pour plus de facilités.

Le nombre de voix que possède le candidat élu pour une liste où il était en tête n'est connu que sous la forme d'un chiffré de Paillier ($c_{*j} = (1+N)^{v_{*j}} \cdot r^{N^k} \ mod\ N^{k+1}$) tandis que les deux autres paramètres sont connus en texte clair. Alors comment effectuer ce transfert ? Il est évident que nous voulons obtenir le résultat de cette formule sous la forme d'un chiffré de Paillier pour obtenir un minimum d'informations sur la structure du vote. Nous pouvons appliquer la formule suivante :

$$\begin{aligned} c_{*j}^{\frac{vs}{vt}} &= [(1+N)^{v_{*j}} \cdot r^{N^k} \ mod\ N^{k+1}]^{\frac{vs}{vt}} \\ &= (1+N)^{\frac{v_{*j} \cdot vs}{vt}} \cdot r^{N^k \cdot \frac{vs}{vt}} \ mod\ N^{k+1}. \end{aligned}$$

Un problème survient alors. La valeur $\frac{v*j \cdot vs}{vt}$ a de grandes chances de ne pas être entière alors que la propriété utilisée dans le chiffrement Paillier est un isomorphisme de groupe de la forme

$$f_k : \mathbb{Z}_{N^k} \times \mathbb{Z}_N^* \longrightarrow \mathbb{Z}_{N^{k+1}}^*$$

tel que

$$f_k(a,b) = [(1+N)^a b^{N^k} \ mod\ N^{k+1}]\ \forall a \in \mathbb{Z}_{N^k}, \forall b \in \mathbb{Z}_N^*.$$

La valeur de a et donc de $\frac{v*j \cdot vs}{vt}$ doit être entière, ce qui est impossible à obtenir dans la plupart des cas. Pour résoudre ce problème, observons d'abord que pour chaque liste où le candidat élu est en tête, le rapport $\frac{vs}{vt}$ est le même. Une solution est donc de multiplier chaque chiffré lié à une liste où le candidat élu était en tête (noté c_{*j}) par vs et chaque chiffré des permutations où il ne l'était pas (noté c_{*i}) par vt. Nous obtenons alors les formules suivantes :

$$\begin{aligned} c_{*j}^{vs} &= [(1+N)^{v_{*j}} \cdot r^{N^k} \ mod\ N^{k+1}]^{vs} \\ &= (1+N)^{v_{*j} \cdot vs} \cdot r^{N^k \cdot vs} \ mod\ N^{k+1} \\ c_{*i}^{vt} &= [(1+N)^{v_{*i}} \cdot r^{N^k} \ mod\ N^{k+1}]^{vt} \\ &= (1+N)^{v_{*i} \cdot vt} \cdot r^{N^k \cdot vt} \ mod\ N^{k+1\ 1} \end{aligned}$$

qui respectent les conditions d'utilisations du chiffrement Paillier. Cette solution garde également le bon rapport entre les chiffrés de chaque liste. Notons encore qu'à chaque fois qu'un candidat est élu, v_{*j} est multiplié par vs et v_{*i} est multiplié par vt tout en remarquant que cette suite de multiplications doit rester dans $\mathbb{Z}_{N^k}$. Ceci est évidemment possible en prenant un k suffisamment grand (voir section 3.3). De plus, toutes ces manipulations peuvent être calculées par tout un chacun, ce qui entraine que la notion de vérifiabilité universelle est encore respectée. Ce petit tour de passe passe implique quelques subtilités lors de la proclamation des résultats. Après avoir déchiffré le chiffré associé au candidat élu, nous devons diviser le résultat obtenu par

$$\begin{cases} vt(1) \cdots vt(m) & \text{si } m \neq\ 0 \\ 1 & \text{sinon} \end{cases}$$

où m est le nombre de candidats ayant déjà été élus, pour obtenir le score électoral du candidat. Une fois le transfert des voix effectué, nous revenons à l'étape une de notre algorithme.

Tout ceci est fort abstrait et complexe, illustrons donc par un exemple.

7.3 Exemple

Nous allons illustrer le procédé de dépouillement pour un cas où l'on doit élire 2 candidats parmi 4. Supposons que nous ayons 100 électeurs.

Émission des votes

Commençons par calculer les clefs publique et secrète du chiffrement de Paillier. Normalement, les clefs sont de l'ordre de 1024 bits pour rendre le problème de factorisation difficile à résoudre, mais pour des facilités de calculs nous utiliserons les clefs suivantes : $pk = 15 = 3 * 5$ et $sk = (3-1) * (5-1) = 8$. Normalement, nous utilisons également le chiffrement de Paillier généralisé pour avoir un grand espace de texte en clair, mais ici nous considérerons le Paillier classique pour les exemples de calculs de chiffrés. Les $4! = 24$ listes sont générées et chaque électeur envoie ses votes avec les preuves associées. Prenons un électeur i au hasard et supposons qu'il vote pour la liste 1. Il envoie alors :

$$\begin{cases} c_{i1} = Enc_{pk}(1, r_{i1}) = (1+N)^1 r_{i1}^N \ mod\ N^{k+1} & (j=1) \\ c_{ij} = Enc_{pk}(0, r_{ij}) = (1+N)^0 r_{ij}^N \ mod\ N^{k+1} & j \in \{2, ..., 24\}. \end{cases}$$

Par exemple pour c_{i1}, il choisit $r_{i1} = 7 \in \mathbb{Z}_N^*$ et calcule

$$\begin{aligned} c_{i1} &= (1+15)7^{225} \ mod\ 225 \\ &= 16 * 118 \ mod\ 225 \\ &= 88 \ mod\ 225. \end{aligned}$$

Et pour chaque c_{ij} $j \in \{1, ..., 24\}$, il calcule

$$Preuve_{ij}(c_{ij} \ ou \ \frac{c_{ij}}{N+1} \ est\ un\ chiffrement\ de\ 0).$$

Prenons le cas du chiffré c_{i1} et montrons en détails comment il s'y prend (section 4.2). Il fait appel à un simulateur pour sortir une conversation valide pour le chiffré c_{i1}. C'est-à-dire qu'il choisit aléatoirement un challenge e_2 et un secret $y_2 \in \mathbb{Z}_N$, puis calcule $x_2 = Enc_{pk}(0, y_2)c_{i1}^{-e_2} \ mod\ N^{k+1}$ et ressort la conversation (x_2, e_2, y_2). Il prouve ensuite que $c_{i1}/(1+N)$ est un chiffrement de 0, c'est-à-dire qu'il choisit aléatoirement $r \in \mathbb{Z}_N^*$, calcule $x_1 = Enc_{pk}(0, r)$ et fait appel à la fonction de hachage $\mathscr{H}(c_{i1}, c_{i1}/(1+N), x_1, x_2)$ qui est publique pour obtenir un challenge e. L'électeur i calcule alors $e_1 = e - e_2 \ mod\ 2^{k_1}$ et $y_1 = rs_1^{e_1} \ mod\ N$. La preuve totale que c_{ij} *ou* $\frac{c_{ij}}{N+1}$ *est un chiffrement de* 0 est donc la conversation $(x_2, e_2, y_2, x_1, e_1, y_1)$. L'électeur i fait donc cette preuve pour les 24 votes. L'autorité, mais également n'importe qui, peut alors, après avoir évalué $\mathscr{H}(c_{i1}, c_{i1}/(1+N), x_1, x_2) = e$, vérifier que ces preuves sont valides en vérifiant que :

- $e = e_1 + e_2 \ mod\ 2^{k_1}$,
- $Enc_{pk}(0, y_1) = x_1 c_1^{e_1} \ mod\ N^{k+1}$,
- $Enc_{pk}(0, y_2) = x_2 c_2^{e_2} \ mod\ N^{k+1}$,
- $c_1, c_2, x_1, x_2, y_1, y_2$ sont relativement premiers à N.

L'autorité vérifie alors que la somme des votes de l'électeur i vaut exactement 1 en calculant :

$$v_{i*} = Dec_{sk}(\prod_{j=1}^{24} c_{ij}).$$

Elle publie alors le résultat v_{i*} et le secret r_{i*} associé pour que tout un chacun puisse valider ou non le vote de l'électeur i.

Une fois que tous les votes de tous les électeurs ont été validés ou non, par l'autorité, et par qui veut, l'autorité calcule le résultat de chaque liste. Supposons que les 100 électeurs de notre exemple aient émis des votes valides. Pour chaque liste j, elle calcule

$$c_{*j} = \prod_{i=1}^{100} c_{ij}.$$

Nous pouvons alors passer à la phase de dépouillement.

Dépouillement des votes

Les tableaux suivants reprennent la répartition des résultats qui viennent d'être calculés par l'autorité ainsi que le procédé de dépouillement dans son intégralité.

Les résultats du tableaux ci-dessous peuvent être calculés par tout un chacun par la méthode décrite ci-dessus.

1	2	3	4	5	6	7	8	9	10	11	12
A	A	A	A	A	A	B	B	B	B	B	B
B	B	C	C	D	D	A	A	C	C	D	D
C	D	B	D	B	C	C	D	A	D	A	C
D	C	D	B	C	B	D	C	D	A	C	A
$Enc_1(4)$	$Enc_2(0)$	$Enc_3(8)$	$Enc_4(2)$	$Enc_5(4)$	$Enc_6(0)$	$Enc_7(10)$	$Enc_8(0)$	$Enc_9(3)$	$Enc_{10}(7)$	$Enc_{11}(9)$	$Enc_{12}(0)$
13	14	15	16	17	18	19	20	21	22	23	24
C	C	C	C	C	C	D	D	D	D	D	D
A	A	B	B	D	D	A	A	B	B	C	C
B	D	A	D	A	B	B	C	A	C	A	B
D	B	D	A	B	A	C	B	C	A	B	A
$Enc_{13}(0)$	$Enc_{14}(17)$	$Enc_{15}(12)$	$Enc_{16}(0)$	$Enc_{17}(0)$	$Enc_{18}(7)$	$Enc_{19}(6)$	$Enc_{20}(2)$	$Enc_{21}(1)$	$Enc_{22}(8)$	$Enc_{23}(0)$	$Enc_{24}(0)$

1) Multiplication des chiffrés associés à chaque candidat :

$Enc_A(4+8+2+4)$
$Enc_B(10+3+7+9)$
$Enc_C(17+12+7)$
$Enc_D(6+2+1+8)$

2) Déterminer si des candidats dépassent le quotient électoral qui est de 34. Le candidat C dépasse le quotient, donc on déchiffre le résultat et on le publie : $Dec(Enc_C(17+12+7)) = 36$.

3)Est-ce que tous les sièges ont été pourvus ? Non, nous devons encore élire un candidat.

4) Transférer les voix en surplus. Nous avons $36 - 34 = 2 = vs$ voix en surplus et $vt = 36$. Nous allons donc élever les chiffrés 13 à 18 à la puissance 2 et les autres à la puissance 36. Nous obtenons alors un nouveau tableau de résultats.

Développons en détails chaque étape.

L'étape 1) consiste simplement en une multiplication de chiffrés. Par exemple pour le candidat A, nous avons calculé :

$$Enc_A(18) = Enc_1(4)Enc_2(0)Enc_3(8)Enc_4(2)Enc_5(4)Enc_6(0)$$

par la propriété d'homomorphisme du chiffrement de Paillier.

Lors de l'étape 2), nous comparons chaque chiffré avec le quotient électoral. Pour cela, nous utilisons la méthode de comparaison de Paillier présentée théoriquement en section (6.3). Prenons par exemple le chiffré $Enc_C(36)$ et supposons, pour l'exemple de l'utilisation de la comparaison de Paillier, que nous travaillons dans l'espace des textes en clairs $\mathbb{Z}_{N^k}$ avec $N = 15$ et $k = 2$. Nous avons donc $N^2 = 225 = 11100001_2$, qui est long de 8 bits. Nous appliquons alors la méthode suivante :

1) Conversion du quotient électoral en binaire : $34 = 00100010_2 = q$ tel que $q0 = 0, q1 = 1, q2 = 0, ...$ Nous chiffrons ces bits et obtenons la séquence $Enc_{q0}(0), Enc_{q1}(1), Enc_{q2}(0), Enc_{q3}(0), Enc_{q4}(0), Enc_{q5}(1), Enc_{q6}(0), Enc_{q7}(0)$.
2) Conversion de $Enc_C(36)$ en binaire ($36 = 100100_2$). Les parties génèrent aléatoirement 8 bits, par exemple $200 = 11001000 = r \in \mathbb{Z}_{N^2}$ et les chiffres. Nous avons alors la séquence publique $Enc_{r0}(0), Enc_{r1}(0), Enc_{r2}(0), Enc_{r3}(1), Enc_{r4}(0), Enc_{r5}(0), Enc_{r6}(1), Enc_{r7}(1)$. Nous pouvons donc calculer le chiffré $c = Enc(y = 36 + r \ mod \ N^2)$:

$$\begin{aligned}
c &= Enc_C(36)Enc_{r0}(0)^{2^0}Enc_{r1}(0)^{2^1}Enc_{r2}(0)^{2^2}Enc_{r3}(1)^{2^3}Enc_{r4}(0)^{2^4}Enc_{r5}(0)^{2^5}Enc_{r6}(1)^{2^6}Enc_{r7}(1)^{2^7} \\
&= Enc_C(36)Enc_{r0}(0)Enc_{r1}(0)Enc_{r2}(0)Enc_{r3}(1*2^3)Enc_{r4}(0)Enc_{r5}(0)Enc_{r6}(1*2^6)Enc_{r7}(1*2^7) \\
&= Enc(36 + 8 + 64 + 128 \ mod \ 225) \\
&= Enc(236 \ mod \ 225) \\
&= Enc(11)
\end{aligned}$$

Les parties déchiffrent alors c et obtiennent le message $y = 11$. Elles publient également le secret s utilisé pour chiffrer c pour que tout le monde puisse vérifier que c a été déchiffré correctement. En binaire, $y = 11$ s'exprime comme 00001011. Nous pouvons alors chiffrer chaque bit comme $Enc_{y0}(1), Enc_{y1}(1), Enc_{y2}(0), Enc_{y3}(1), Enc_{y4}(0), Enc_{y5}(0), Enc_{y6}(0), Enc_{y7}(0)$ et calculer les chiffrés des bits de $x = y - 200 \ mod \ 225$ à l'aide du protocole (6.3.2). Nous commençons par calculer $\bar{r}$:

$$\begin{aligned}
Enc_{r0}(0)Enc(1)/Enc_{r0}(0)^2 &= Enc_{\bar{r0}}(0 + 1 - 2 * 0 = 1) \\
Enc_{r1}(0)Enc(1)/Enc_{r1}(0)^2 &= Enc_{\bar{r1}}(0 + 1 - 2 * 0 = 1) \\
Enc_{r2}(0)Enc(1)/Enc_{r2}(0)^2 &= Enc_{\bar{r2}}(0 + 1 - 2 * 0 = 1) \\
Enc_{r3}(1)Enc(1)/Enc_{r3}(1)^2 &= Enc_{\bar{r3}}(1 + 1 - 2 * 1 = 0) \\
Enc_{r4}(0)Enc(1)/Enc_{r4}(0)^2 &= Enc_{\bar{r4}}(0 + 1 - 2 * 0 = 1) \\
Enc_{r5}(0)Enc(1)/Enc_{r5}(0)^2 &= Enc_{\bar{r5}}(0 + 1 - 2 * 0 = 1) \\
Enc_{r6}(1)Enc(1)/Enc_{r6}(1)^2 &= Enc_{\bar{r6}}(1 + 1 - 2 * 1 = 0) \\
Enc_{r7}(1)Enc(1)/Enc_{r7}(1)^2 &= Enc_{\bar{r7}}(1 + 1 - 2 * 1 = 0)
\end{aligned}$$

Nous continuons en calculant $w = \bar{r} + 1$. La multiplication du contenu de deux chiffrés doit se faire à l'aide du protocole (6.3.1).

$$\begin{aligned}
Enc(c_0) &= Enc_{\bar{r0}}(1)^{h_0=1} Enc_{\bar{r0}}(1)^{c_{-1}=0} Enc_{h_0}(1)^{c_{-1}=0} / Enc_{\bar{r0}}(1)^{2*(h_0=1)*(c_{-1}=0)} \\
&= Enc_{c_0}(1+0+0-0=1) \\
Enc(w_0) &= Enc_{\bar{r0}}(1) Enc_{h_0}(1) Enc_{c_{-1}}(0) / Enc_{c_0}(1)^2 \\
&= Enc(w_0)(1+1+0-2*1=0) \\
Enc(c_1) &= Enc_{\bar{r1}}(1)^{h_1=0} Enc_{\bar{r1}}(1)^{c_0=1} Enc_{h_1}(0)^{c_0=1} / Enc_{\bar{r1}}(1)^{2*(h_1=0)*(c_0=1)} \\
&= Enc_{c_1}(1*0+1*1+0*1-1*2*0*1=1) \\
Enc(w_1) &= Enc_{\bar{r1}}(1) Enc_{h_1}(0) Enc_{c_0}(1) / Enc_{c_1}(1)^2 \\
&= Enc(w_1)(1+0+1-2*1=0) \\
Enc(c_2) &= Enc_{\bar{r2}}(1)^{h_2=0} Enc_{\bar{r2}}(1)^{c_1=1} Enc_{h_2}(0)^{c_1=1} / Enc_{\bar{r2}}(1)^{2*(h_2=0)*(c_1=1)} \\
&= Enc_{c_2}(1*0+1*1+0*1-1*2*0*1=1) \\
Enc(w_2) &= Enc_{\bar{r2}}(1) Enc_{h_2}(0) Enc_{c_1}(1) / Enc_{c_2}(1)^2 \\
&= Enc(w_2)(1+0+1-2*1=0) \\
Enc(c_3) &= Enc_{\bar{r3}}(0)^{h_3=0} Enc_{\bar{r3}}(0)^{c_2=1} Enc_{h_3}(0)^{c_2=1} / Enc_{\bar{r3}}(0)^{2*(h_3=0)*(c_2=1)} \\
&= Enc_{c_3}(0*0+0*1+0*1-0*2*0*1=0) \\
Enc(w_3) &= Enc_{\bar{r3}}(0) Enc_{h_3}(0) Enc_{c_2}(1) / Enc_{c_3}(0)^2 \\
&= Enc(w_3)(0+0+1-2*0=1) \\
&\dots
\end{aligned}$$

Nous obtenons alors la séquence $Enc_{w_0}(0), Enc_{w_1}(0), Enc_{w_2}(0), Enc_{w_3}(1), Enc_{w_4}(1), Enc_{w_5}(1), Enc_{w_6}(0), Enc_{w_7}(0)$. Nous calculons ensuite $z = w+y$ par la même méthode que ci-dessus. Nous obtenons la séquence $Enc_{z_0}(1), Enc_{z_1}(1), Enc_{z_2}(0), Enc_{z_3}(0), Enc_{z_4}(0), Enc_{z_5}(0), Enc_{z_6}(1), Enc_{z_7}(0)$ et un dernier report $Enc_{c_7} = 0$. Nous prenons le complément à bit de ce chiffré en calculant :

$$Enc(z_8) = Enc_{c_7}(0) Enc(1) / Enc_{c_7}(0)^2 = Enc_{z_8}(1).$$

Il nous reste alors à réduire la valeur de z modulo N^2 en ajoutant à z la valeur $N^2 z_8$. Pour cela, nous commençons par transformer $N^2 = 225$ en binaire (10000111_2) et chiffrons tous les bits en prouvant qu'ils sont chiffrés correctement à l'aide d'une preuve à divulgation nulle. Nous obtenons alors la séquence $Enc_{N_0^2}(1), Enc_{N_1^2}(1), Enc_{N_2^2}(1), Enc_{N_3^2}(0), Enc_{N_4^2}(0), Enc_{N_5^2}(0), Enc_{N_6^2}(0), Enc_{N_7^2}(1)$ que

nous multiplions par le contenu du chiffré $Enc_{z_8}(1)$ à l'aide du protocole (6.3.1) pour obtenir les bits chiffrés de $N^2 z_8$. Dans notre cas, la séquence ne change pas. Nous terminons par calculer $36 = z + N^2 z_8$ par la même méthode que précédemment.

Nous retrouvons bien la séquence $Enc(0), Enc(0), Enc(1), Enc(0), Enc(0), Enc(1), Enc(0), Enc(0)$ qui est la conversion binaire de $Enc_C(36)$.

3) Il nous reste alors à évaluer la formule de récurrence pour $i = 0, ...7$:

$$\begin{aligned} Enc_{pk}(t_0) &= Enc_{pk}(0), \\ Enc_{pk}(t_{i+1}) &= \left(\frac{Enc_{pk}(1)}{\frac{Enc_{pk}(q_i)Enc_{pk}(z_i)}{Enc_{pk}(2q_i z_i)}}\right)^{t_i} (Enc_{pk}(1)/Enc_{pk}(z_i))^{q_i}. \end{aligned}$$

Nous obtenons alors $c = Enc_{pk}(t_8 = 1)$. L'autorité déchiffre c et publie le résultat ainsi que le secret sous-jacent. Tout le monde peut donc vérifier qu'elle a déchiffré correctement. Nous obtenons le résultat 1 qui signifie que le candidat C possède plus de voix que le quotient électoral, ce qui entraine que C est élu.

Lors de l'étape 4), nous devons transférer les votes en surplus. Nous avons $36 - 34 = 2 = vs$ voix en surplus et $vt = 36$. Nous allons donc élever les chiffrés 13 à 18 à la puissance 2 et les autres à la puissance 36. Par exemple pour la liste 1, nous calculons $Enc_1(4)^{36} = Enc_1(144)$ et pour la liste 13, nous calculons $Enc_{13}(0)^2 = Enc_{13}(0)$. Nous retirons également le candidat C élu des différentes listes. Nous obtenons donc le nouveau tableau de résultats ci-dessous.

Mais avant, nous devons recalculer le quotient électoral car un candidat a été élu :

$$(\lfloor \frac{100}{2+1} \rfloor + 1) \cdot 36 = 34 * 36 = 1224.$$

Remarque : Nous avons expliqué en détails comment une étape de dépouillement se déroule, nous nous contenterons donc d'une explication succincte pour les autres.

1	2	3	4	5	6	7	8	9	10	11	12
A	A	A	A	A	A	B	B	B	B	B	B
B	B	B	D	D	D	A	A	A	D	D	D
D	D	D	B	B	B	D	D	D	A	A	A
$Enc_1(144)$	$Enc_2(0)$	$Enc_3(288)$	$Enc_4(72)$	$Enc_5(144)$	$Enc_6(0)$	$Enc_7(360)$	$Enc_8(0)$	$Enc_9(108)$	$Enc_{10}(252)$	$Enc_{11}(324)$	$Enc_{12}(0)$
13	14	15	16	17	18	19	20	21	22	23	24
A	A	B	B	D	D	D	D	D	D	D	D
B	D	A	D	A	B	A	A	B	B	A	B
D	B	D	A	B	A	B	B	A	A	B	A
$Enc_{13}(0)$	$Enc_{14}(34)$	$Enc_{15}(24)$	$Enc_{16}(0)$	$Enc_{17}(0)$	$Enc_{18}(14)$	$Enc_{19}(216)$	$Enc_{20}(72)$	$Enc_{21}(36)$	$Enc_{22}(288)$	$Enc_{23}(0)$	$Enc_{24}(0)$

1) Multiplication des chiffrés associés à chaque candidat :

$Enc_A(144 + 288 + 72 + 144 + 34) = Enc_A(682)$
$Enc_B(360 + 108 + 252 + 324 + 24) = Enc_B(1068)$
$Enc_D(14 + 216 + 72 + 36 + 288) = Enc_D(626)$

Remarquons que $\frac{610+1068+626}{36} = 66$ ce qui fait 100, le nombre d'électeurs, si on ajoute les 34 voix du candidat C.

2) Déterminer si des candidats dépassent le quotient électoral qui est maintenant de $34 * 36 = 1224$. Aucun candidat ne dépasse le quotient.

3) Quel candidat possède le moins de voix ? Le candidat D est éliminé.

4) Transférer les voix en surplus. Nous pouvons transférer les votes du candidat D au second de chaque permutation où D était en tête. Nous n'avons aucun calcul à faire, car vu que nous avons éliminé un candidat, le transfert va se faire naturellement. Nous obtenons alors un nouveau tableau de résultats.

Remarquons que nous n'avons pas besoin de recalculer le quotient électoral, car nous n'avons pas eu besoin de mettre en puissance les chiffrés.

1	2	3	4	5	6	7	8	9	10	11	12
A	A	A	A	A	A	B	B	B	B	B	B
B	B	B	B	B	B	A	A	A	A	A	A
$Enc_1(144)$	$Enc_2(0)$	$Enc_3(288)$	$Enc_4(72)$	$Enc_5(144)$	$Enc_6(0)$	$Enc_7(360)$	$Enc_8(0)$	$Enc_9(108)$	$Enc_{10}(252)$	$Enc_{11}(324)$	$Enc_{12}(0)$
13	14	15	16	17	18	19	20	21	22	23	24
A	A	B	B	A	B	A	A	B	B	A	B
B	B	A	A	B	A	B	B	A	A	B	A
$Enc_{13}(0)$	$Enc_{14}(34)$	$Enc_{15}(24)$	$Enc_{16}(0)$	$Enc_{17}(0)$	$Enc_{18}(14)$	$Enc_{19}(216)$	$Enc_{20}(72)$	$Enc_{21}(36)$	$Enc_{22}(288)$	$Enc_{23}(0)$	$Enc_{24}(0)$

1) Multiplication des chiffrés associés à chaque candidat :

$Enc_A(144 + 288 + 72 + 144 + 34 + 216 + 72) = Enc_A(970)$
$Enc_B(360 + 108 + 252 + 324 + 24 + 14 + 36 + 288) = Enc_B(1406)$

Remarquons que $\frac{970+1406}{36} = 66$ ce qui fait 100, le nombre d'électeurs, si on ajoute les 34 voix du candidat C.

2) Déterminer si des candidats dépassent le quotient électoral qui est maintenant de $34 * 36 = 1224$. Le candidat B dépasse le quotient donc on déchiffre le résultat et on le publie : $Dec(Enc_B(1406)) = 1406$. Comme un candidat a déjà été élu avec un total de voix de 36, nous devons diviser 1406 par 36. Ce qui fait $39,05$ voix en faveur de B.

3)Est-ce que tous les sièges ont été pourvus ? Oui, l'élection est donc finie et nous pouvons publier le reste des résultats. Le candidat restant A possède $Dec(Enc_A(970)) = 970$ voix, ce qui représente un score de $26,95$ voix.

Chapitre 8

Conclusion

Le but de ce travail était de développer une méthode cryptographique pour rendre le vote à voix unique transférable utilisable de manière électronique tout en gardant les propriétés de confidentialité, de robustesse, de vérifiabilité universelle et de résistance à la coercition.

Le développement d'une telle méthode cryptographique a nécessité la construction d'un système de chiffrement basé sur le cryptosystème de Paillier généralisé, étant donné qu'il est « CPA-secure »et qu'il nous fourni la propriété d'homomorphisme. Autour de ce chiffrement, nous avons développé un schéma de preuve à divulgation nulle permettant de certifier que les bulletins, chiffrés par le cryptosystème de Paillier, sont valides. Nous avons obtenu la propriété de robustesse. Nous avons également développé des protocoles permettant à tout le monde de vérifier que le résultat final est bien correct en contrôlant les faits et gestes des autorités responsables du vote. Pour cela, nous utilisons des protocoles à divulgations nulles ainsi qu'un protocole de partage de la clef secrète entre les différentes autorités. Ceci nous fournit la propriété de vérifiabilité universelle. Nous avons enfin rendu inutile l'achat de votes pour un adversaire qui tenterait de vérifier, après que les votes aient été émis, que quelqu'un ait comme il le désirait tout en conservant la propriété de vérifiabilité universelle. Pour ce faire, nous avons développé une méthode qui permet de dévoiler uniquement les résultats des candidats élus. Cette méthode consiste en une comparaison de deux chiffrés : la méthode de comparaison de Paillier. Une amélioration pourrait cependant être apportée. Nous dévoilons les résultats des candidats élus au tour où ceux-ci sont élus. Nous fournissons donc une information sur le tour où les candidats sont élus. Ceci pourrait être évité en dévoilant les candidats élus tout à la fin du dépouillement.

Tout au long de ce travail, nous nous sommes basés sur des articles que s'intéressent à la question du vote électronique sécurisé. Nous avons fait un travail de synthèse sur ces différents articles pour aller chercher l'information pertinente qui nous était nécessaire. Les chapitres $1, 2, 3$ et 4 sont essentiellement le fruit de cette synthèse. Les chapitres 5 et 6 contiennent quant à eux une réelle réflexion de l'application de toutes ces notions de votes électroniques sécurisés au cas du vote à voix unique transférable. L'originalité de ce travail se situe dans la méthode suivante. Nous générons toutes les permutations possible des candidats et demandons aux électeurs de voter pour la permutation qui représente ses préférences. C'est vraiment cette méthode qui nous a permis de concilier les notions de vérifiabilité universelle et de résistance à la coercition.

Cependant, notre méthode connait aussi certaines limites. Pour un vote à L candidats, chaque électeur doit chiffrer $L!$ votes. Ceci entraine que pour des élections d'assez forte ampleur, notre méthode souffre de problèmes de performance. Nous n'avons d'ailleurs pas implémenter notre méthode, ce qui pourrait faire l'objet d'une future étude ainsi qu'une recherche détaillée de ses performances. La recherche d'une autre méthode pour des élections à très grande ampleur peut donc venir enrichir ce mémoire.

Bibliographie

[1] J. Katz and Y. Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007.

[2] C. Culnane, P. Y. A. Ryan, S. Schneider, and V. Teague, "vvote : a verifiable voting system (DRAFT)," *CoRR*, vol. abs/1404.6822, 2014.

[3] Wikipedia, "Single transferable vote," `http://en.wikipedia.org/wiki/Single_transferable_vote` [En ligne ; Page disponible le 21-novembre-2013].

[4] C. Tardif, "Les systèmes électoraux : Le vote unique transférable," *Envol*, vol. no 155, p. pp. 19 à 23, avril-mai-juin 2011.

[5] A. Reynolds, B. Reilly, A. Ellis, I. I. for Democracy, and E. Assistance, *Electoral System Design : The New International IDEA Handbook*. Handbook series, International Institute for Democracy and Electoral Assistance, 2005.

[6] F. Koeune and O. Peraira, "Cours d'introduction à la cryptographie," *MA1 sciences mathématiques, Université catholique de Louvain*, 2013.

[7] M. J. Jurik, *Extensions to the Paillier Cryptosystem with Applications to Cryptological Protocols*. PhD thesis, University of Aarhus, 2003.

[8] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pp. 467–482, 2005.

[9] J.-G. Dumas, "Factorisation d'entiers, cryptographie," `http://ljk.imag.fr/membres/Jean-Guillaume.Dumas/Enseignements/Polys/RSA_FACTO.pdf` [En ligne ; Page disponible le 7-juillet-2015].

[10] A. Menezes, P. C. van Oorschot, S. A. Vanstone, and S. A. Vanstone, "Handbook of applied cryptography," 1996.

[11] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself : Pitfalls of the fiat-shamir heuristic and applications to helios," in *ASIACRYPT*, vol. 7658, pp. 626–643, Springer, 2012.

[12] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *In proceedings of PKC '01, LNCS series*, pp. 119–136, Springer-Verlag, 2001.

[13] V. Teague, K. Ramchen, and L. Naish, "Coercion-resistant tallying for STV voting," in *2008 USENIX/ACCURATE Electronic Voting Workshop, EVT 2008, July 28-29, 2008, San Jose, CA, USA, Proceedings*, 2008.

[14] J. A. Garay, B. Schoenmakers, and J. Villegas, "Practical and secure solutions for integer comparison.," in *Public Key Cryptography* (T. Okamoto and X. Wang, eds.), vol. 4450 of *Lecture Notes in Computer Science*, pp. 330–342, Springer, 2007.

[15] B. Schoenmakers and P. Tuyls, "Efficient binary conversion for paillier encrypted values.," in *EUROCRYPT* (S. Vaudenay, ed.), vol. 4004 of *Lecture Notes in Computer Science*, pp. 522–537, Springer, 2006.

[16] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques : Advances in Cryptology*, EUROCRYPT '01, (London, UK, UK), pp. 280–299, Springer-Verlag, 2001.