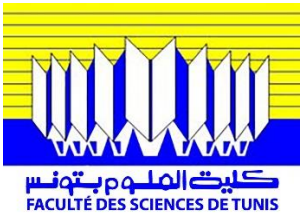
École Doctorale de Mathématiques, Informatique,
Science et Technologie de la Matière



THÈSE

En vue de l'obtention du

DOCTORAT EN INFORMATIQUE

Préparée au sein de l'unité de recherche
Algorithmique parallèle et analyse des données

Présentée par
Marwa CHAIEB

Utilisation de la technologie Blockchain pour sécuriser le vote électronique en ligne

Soutenue le

Devant le jury composé de

		Président
Hela KAFFEL-BEN AYED	Maître de conférences à la FST	Rapporteur
Abderrazak JEMAI	Professeur à l'INSAT	Rapporteur
		Examinateur
Riadh ROBBANA	Professeur à l'INSAT	Directeur de thèse
		Invité

Année Universitaire : 2019/2020

Remerciements

A l'issue de la rédaction de ce rapport de thèse de doctorat, je suis convaincue que la thèse est loin d'être un travail solitaire. En effet, La réalisation de ce travail doctoral a été possible grâce au concours de plusieurs personnes, à qui je voudrais témoigner toute ma gratitude.

Je tiens à remercier tout d'abord mon directeur de thèse, *monsieur Riadh ROBBANA professeur à l'INSAT*, pour toute son aide et pour la confiance qu'il m'a témoignée en acceptant la direction de mes travaux. Je lui suis reconnaissante de m'avoir fait bénéficier tout au long de ce travail de sa grande compétence, de sa rigueur intellectuelle, de son dynamisme, et de son efficacité.

Je voudrais également adresser toute ma reconnaissance à *monsieur Souheib YOUSFI, maitre-assistant à l'INSAT*, qui m'a co-encadré tout au long de cette thèse et qui m'a fait partager ses brillantes intuitions. Je le remercie pour sa patience, sa disponibilité permanente et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion. Je suis ravie d'avoir travaillé en sa compagnie car outre son appui scientifique, il a toujours été là pour me soutenir et me conseiller au cours de l'élaboration de cette thèse.

Je tiens à exprimer mes sincères remerciement à *monsieur Pascal LAFOURCADE*, *maître de conférences HDR à l'Université Clermont Auvergne France*, qui m'a accueilli pendant deux mois au sein de son laboratoire. Cette thèse est le fruit d'une collaboration de trois années avec lui. C'est à ses côtés que j'ai compris ce que rigueur et précision voulaient dire. Je le remercie pour les nombreux encouragements qu'il m'a prodiguée et pour l'intérêt manifesté à l'égard de ma recherche.

Je souhaiterais aussi adresser ma gratitude à Madame Hela KAFFEL-BEN AYED, Maître de conférences à la FST, ainsi qu'à Monsieur Abderrazak JEMAI, Professeur à l'INSAT, de l'honneur qu'ils m'ont fait en acceptant d'être rapporteurs de cette thèse.

Table des matières

Liste des figures

Liste des tableaux

Liste des publications

— Revue internationale

1. **Marwa Chaieb**, Souheib Yousfi, Pascal Lafourcade, Riadh Robbana. Design and Practical Implementation of Verify-Your-Vote Protocol. *Concurrency and Computation Practice and Experience Journal 2020, vol.32.* (Impact Factor : 1.167).

— Conférences internationales

1. **Marwa Chaieb**, Mirko Koscina , Souheib Yousfi, Pascal Lafourcade, Riadh Robbana. DABSTERS : A Privacy Preserving e-Voting Protocol for Permissioned Blockchain. *In : Hierons RM, Mosbah M., eds. Theoretical Aspects of Computing -ICTAC 2019 - 16th International Colloquium, Hammamet, Tunisia, October 31 - November 4, 2019, Proceedings. 11884 of Lecture Notes in Computer Science. Springer ; 2019 : 292–312.* (Classe B).
2. **Marwa Chaieb**, Mirko Koscina , Souheib Yousfi, Pascal Lafourcade, Riadh Robbana. DABSTERS : Distributed Authorities using Blind Signature to Effect Robust Security in e-Voting. *In : Obaidat MS, Samarati P., eds. Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2 : SECRYPT, Prague, Czech Republic, July 26-28, 2019SciTePress ; 2019 : 228–235.* (Classe B).
3. **Marwa Chaieb**, Souheib Yousfi, Pascal Lafourcade, Riadh Robbana. Verify-Your-Vote : A Verifiable Blockchain-Based Online Voting Protocol. *In : Themistocleous M, Cunha dPR., eds. Information Systems - 15th European, Mediterranean, and Middle Eastern Conference, EMCIS 2018, Limassol, Cyprus, October 4-5, 2018, Proceedings. 341 of Lecture Notes in Business Information Processing. Springer ; 2018 : 16–30.* (Classe B).

Introduction générale

La démocratie est un système où le peuple exerce le pouvoir par le vote. La force d'un tel système dépend fortement de la confiance accordée au processus électoral. Les électeurs sont présents lors du dépôt de leurs bulletins de vote, mais que se passe-t-il avec un bulletin de vote après que le vote a été effectué? Dans la plupart des cas, les électeurs supposent, tout simplement, que le processus de vote est sûr et que tout fonctionne correctement lors de dépouillement.

Avec l'apparition du vote électronique en ligne, de nombreux problèmes liés au vote traditionnel ont été résolus. En effet, le dépouillement des bulletins de vote devient automatisé ce qui offre une rapidité accrue et une marge d'erreur beaucoup plus réduite par rapport au vote traditionnel. Aussi, les votants n'ont plus besoin de se déplacer pour déposer leurs votes, puisqu'ils peuvent voter de chez eux. Par conséquent, le taux d'abstentionnistes devient beaucoup plus réduit. Enfin, le vote électronique en ligne réduit les coûts d'organisation d'élection puisqu'il élimine les dépenses associées à l'établissement des bureaux de vote et du personnel requis.

Cependant, les protocoles de vote électronique en ligne qui ont été proposés durant ces dernières décennies n'offrent pas un niveau de sécurité et de transparence satisfaisant les attentes des électeurs et des organisateurs des élections. La problématique du vote électronique apparaît alors très complexe malgré son attrait de par sa simplicité d'utilisation. Plusieurs systèmes de vote électronique ont été proposés, et même expérimentés, mais la plupart ne sont pas fiables car ils ne satisfont pas certaines exigences fondamentales de sécurité. En effet, ces systèmes souffrent d'un défaut de conception qui est la centralisation de leurs architectures.

L'un des moyens pour résoudre ces problèmes de sécurité est la technologie Blockchain. Il s'agit d'une forme de base de données distribuée où les enregistrements prennent la forme de transactions, un bloc étant un ensemble de ces transactions. Cette technologie a apparu avec l'introduction de Bitcoin en 2009. Il y a maintenant une nouvelle innovation à explorer. Une innovation qui retire le pouvoir aux autorités centrales et le partage entre plusieurs parties.

Un intérêt majeur de cette thèse est la découverte et l'amélioration de multiples protocoles de vote avec cette nouvelle technologie, de sorte qu'on peut faire de nouvelles hypothèses sur la faisabilité, l'utilité et la pertinence des Blockchains dans le vote électronique.

Notre rapport s'articule autour de quatre chapitres. Le premier, intitulé étude théorique, fournit une présentation détaillée du concept du vote électronique en ligne, ses exigences de sécurité, les différentes primitives cryptographiques utilisées dans nos approches et la technologie de pointe, Blockchain, ainsi que ses différentes caractéristiques et composantes. Le deuxième chapitre est un état de l'art sur les protocoles de vote électronique en ligne basés sur la technologie Blockchain existants. Le but de ce chapitre est d'évaluer l'existant et de cerner les problématiques actuelles liées à ce domaine. Dans le troisième chapitre, nous proposons un nouveau protocole de vote électronique en ligne,

vérifiable et basé sur la Blockchain *Ethereum*. Nous donnons les détails de la conception et de l'implémentation de ce protocole ainsi que l'évaluation de ses performances et du niveau de sécurité qu'il offre. Le chapitre suivant présente une autre contribution : un protocole de vote électronique en ligne assurant une vérifiabilité de bout en bout avec un niveau élevé d'anonymat et de confidentialité. Ce protocole est basé sur la Blockchain privée *Hyperledger Fabric*. Dans le cinquième chapitre, nous présentons une autre contribution : un protocole de vote électronique en ligne résistant à la coercition tout en respectant les autres exigences de sécurité. Ce protocole est aussi basé sur une Blockchain, appelée *Loki*. Le dernier chapitre est une conclusion et un ensemble de perspectives.

Chapitre 1

Étude théorique

Sommaire

1.1 Introduction

Le vote est un aspect crucial de la démocratie. Traditionnellement, lors d'une élection, le votant se déplace vers un bureau de vote et fait son choix d'une manière anonyme, sans aucune influence. Pour faire le dépouillement, nous devons faire confiance à une autorité centrale. De là vient le risque de fraude électorale. L'autorité de dépouillement a la possibilité de falsifier des votes et donc d'élire un candidat qui ne devrait pas être élu. Il est également possible pour l'autorité d'enregistrement de permettre aux électeurs inéligibles de voter. Ainsi, le vote devient inutile et on constate une diminution dans le taux de participation aux élections.

Le vote électronique en ligne est apparu comme une alternative au vote traditionnel puisque nous avons besoin d'un système de vote sécurisé, vérifiable et transparent pour nos élections. Cependant, le vote électronique en ligne fait toujours l'objet d'un débat actif. En effet, un nombre important de personnes pensent que le vote électronique en ligne n'est pas suffisamment confiant pour être utilisé dans un contexte à grande échelle. Le passage au vote électronique est donc réticent en raison de certains soupçons concernant l'intégrité des élections. Pour réussir, le vote électronique en ligne doit garantir une liste exhaustive des exigences de sécurité. Il doit assurer un niveau élevé de confiden-

tialité, d'anonymat, et de vérifiabilité. Ainsi, le vote électronique en ligne nécessite une approche plus vérifiable et plus sûre que celles proposées par les protocoles actuels.

La technologie Blockchain, bien qu'étant un concept relativement nouveau, a gagné assez de popularité dans divers domaines. En effet, le nombre de systèmes à base de Blockchain est en constante augmentation.

Un système de vote électronique en ligne doit donc être sécurisé, tout en permettant une vérification individuelle et universelle. Les Blockchains permettent d'atteindre ce niveau de sécurité et de vérifiabilité, tout en préservant la confidentialité et la non-malléabilité des transactions.

Dans ce chapitre, nous introduisons, dans un premier temps, le vote électronique, ses exigences de sécurité ainsi que la vérification formelle de la sécurité des protocoles de vote électronique. Ensuite, nous donnons les fondements de la technologie Blockchain et ses types. Nous présentons aussi quelques exemples des Blockchains les plus connues.

1.2 Vote électronique

Les systèmes de vote ont été utilisés depuis l'antiquité et constituaient un élément fondamental pour la démocratie dans différentes sociétés. Les premiers systèmes de vote ne garantissaient pas un niveau de sécurité comme celui qui existe actuellement. En effet, avant que les bulletins de vote imprimés deviennent une norme, les votes ont été émis en utilisant des tickets fournis par les parties politiques ou même par voix [1], compromettant l'anonymat de l'électeur. La nécessité de systèmes sécurisés est, ainsi, devenue évidente et a motivé la conception de nouveaux systèmes dès que les fraudes sont apparues. Plus récemment, le vote électronique est apparu comme une alternative au vote sur papier, afin de résoudre les problèmes de sécurité existants. Les systèmes de vote électronique peuvent être divisés en 2 grandes catégories : (1) les systèmes de vote électronique hors ligne et (2) les systèmes de vote électronique en ligne. La première catégorie consiste à conserver les outils du vote traditionnel, qui sont les bureaux de vote et les isoloirs, et à ajouter une machine à voter. L'avantage de cette méthode est de permettre une comptabilisation rapide et efficace des bulletins et de permettre aux votants de voter dans n'importe quel bureau de vote. Les bulletins seront ensuite acheminés par le réseau informatique vers leur destination. Le souci majeur du vote électronique hors ligne est la défaillance des machines à voter qui peut être extrêmement dangereuse dans un processus de vote. L'électeur doit utiliser certains appareils électroniques et il n'y a aucun argument convaincant pour qu'il fasse aveuglément confiance à ce dispositif. Le vote électronique en ligne diffère du vote électronique hors ligne. Le votant peut voter de chez lui en utilisant une simple connexion internet. Les inconvénients de cette méthode sont le manque de confidentialité et les vulnérabilités des logiciels utilisés lors du vote qui peuvent compromettre l'intégrité de l'élection ainsi que l'anonymat de l'électeur. Il faut alors concevoir un système de vote qui satisfait toutes les propriétés usuelles de sécurité.

Au cours des dernières décennies, les chercheurs ont fait de grands progrès dans la capacité des ordinateurs personnels à exécuter des protocoles cryptographiques d'une manière qui était auparavant présumée irréalisable. Les systèmes de vote à distance peuvent exploiter cette capacité pour rendre les élections fiables, vérifiables et accessibles. En conséquence, le vote à distance et par internet a été adopté pour les élections fédérales dans des pays comme l'Estonie [1] et l'Afrique du Sud [2].

1. http://homepage.divms.uiowa.edu/~jones/voting/pictures/

Ainsi, afin d'assurer la sécurité et la vérifiabilité des votes, les systèmes de vote modernes reposent sur la cryptographie. Cette technologie rend la sécurité de ces systèmes comparable, voire meilleure, que le vote traditionnel sur papier.

Dans ce qui suit, nous présentons les primitives cryptographiques que nous utilisons dans nos contributions, présentées dans les chapitres 3, 4 et 5.

1.2.1 Primitives cryptographiques

La cryptographie est la science qui consiste à utiliser les mathématiques pour stocker des informations sensibles ou les transmettre sur des réseaux non sécurisés (comme internet), sans qu'elles soient lues par une personne autre que le destinataire prévu. Elle offre des techniques de communication sécurisée entre deux parties *Alice* et *Bob* en présence d'une tierce personne *Oscar*. Les quatre principes de base de la cryptographie sont :

— *Confidentialité :* Définit un ensemble de règles qui limitent l'accès ou ajoutent des restrictions à certaines informations.
— *Intégrité des données :* Veille à la cohérence et à l'exactitude des données échangées.
— *Authentification :* Confirme la véracité des paramètres d'identification d'une personne.
— *Non-répudiation :* Garantit l'incapacité de l'émetteur d'un message ou d'une information à la nier.

Il existe deux grandes catégories de cryptographie :

1. *Cryptographie à clé publique ou asymétrique :* Dans ces cryptosystèmes, Alice et Bob ont, chacun, une clé privée et une clé publique. La clé publique peut être partagée avec tout le monde, de sorte que Bob peut utiliser la clé publique de Alice pour chiffrer un message destiné à Alice. Mais seule Alice, avec la clé privée correspondante, peut déchiffrer le message qu'elle a reçu de Bob (Figure 1.1). Exemple de cryptosystème à clé publique : le cryptosystème d'El-Gamal [3].

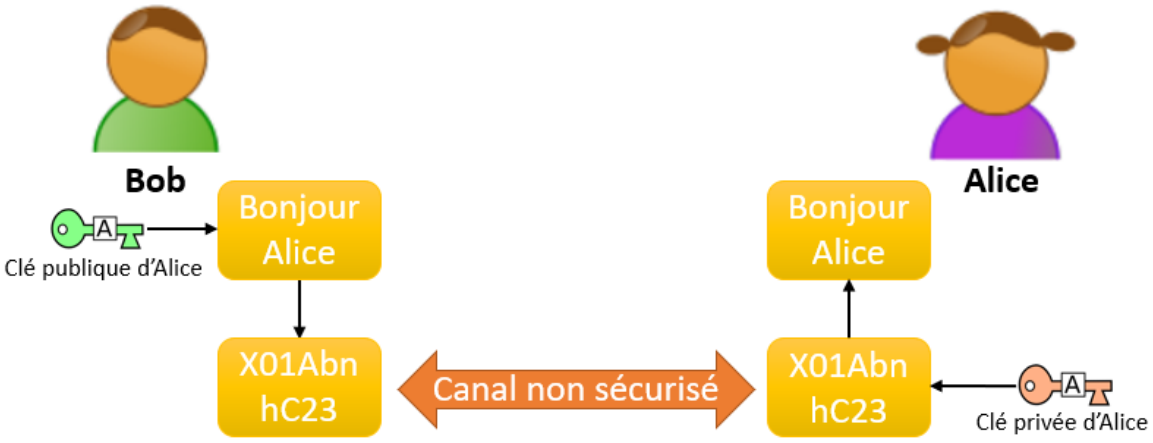


FIGURE 1.1 – Cryptosystème à clé publique.

2. *Cryptographie à clé secrète ou symétrique :* Dans ces cryptosytèmes, Alice et Bob doivent tous deux avoir la même clé secrète pour chiffrer et déchiffrer leur communication (Figure 1.2). Pour cela, ils doivent échanger la clé de manière sécurisée dans un premier temps. Exemples de cryptosystème à clé secrète : *Data Encryption Standard (DES)* [4] et *Advanced Encryption Standard (AES)*[2].

2. https://tools.ietf.org/pdf/rfc3602.pdf

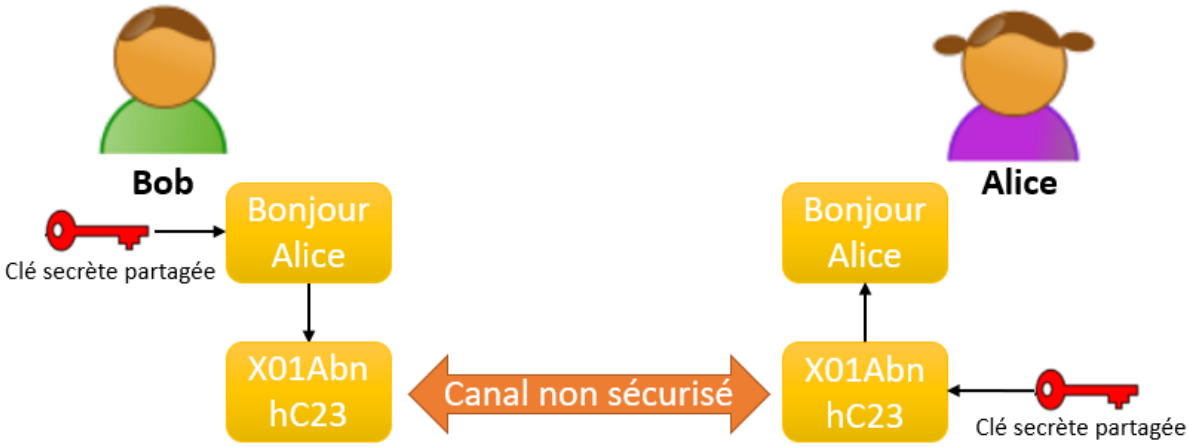


FIGURE 1.2 – Cryptosystème à clé secrète.

Dans ce qui suit, nous présentons les hypothèses de sécurité sur lesquelles se basent nos approches, les cryptosystèmes, les fonctions de hachage et les preuves à divulgation nulle de connaissance desquelles nous tirons profit lors de la conception de nos approches.

Hypothèses de sécurité

La sécurité de nos protocoles de vote se base sur les hypothèses suivantes :
Soient $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q, g un générateur de $\mathbb{G}$ et a, b, et c des nombres aléatoires de $\mathbb{Z}_q^*$

- **Le problème du logarithme discret (*Discrete Logarithm Problem DLP)* [5] :** Ce problème consiste à déterminer la valeur de a à partir de g et g^a.
- **Le problème décisionnel de Diffie-Hellman (*Decisional Diffie–Hellman DDH) [6]* :** Étant donné g, g^a, g^b et g^c, ce problème consiste à déterminer si $ab = c \mod q$ est vraie ou pas.
- **Le problème de *q-Strong Diffie-Hellman (q-SDH) [7]* :** Ce problème consiste à calculer $g^{1/(a+b)}$ à partir de $(g, g^a, g^{a^2}, ..., g^{a^q})$.

Cryptosystèmes

1. *Cryptographie à base de courbes elliptiques [8] :* Les courbes elliptiques sont des courbes géométriques ayant des propriétés particulièrement intéressantes pour le monde de la cryptographie. Pour ajouter deux points P et Q d'une courbe elliptique, il suffit de remarquer que dans certains cas, la ligne L passant par ces deux points, passe également par un troisième point de la courbe. Le résultat de l'addition sera représenté par le point R (voir Figure 1.3).
   En cryptographie, ces courbes sont utilisées pour des opérations asymétriques telles que les échanges de clés dans un canal non sécurisé. La cryptographie elliptique s'est développée à la fin des années 80, après les travaux de Miller [9] et de Koblitz [10]. L'un de ses principaux avantages est son efficacité par rapport à la cryptographie traditionnelle, puisqu'elle procure un niveau de sécurité équivalent ou supérieur aux autres méthodes, tout en ayant des tailles de clés beaucoup plus petites.

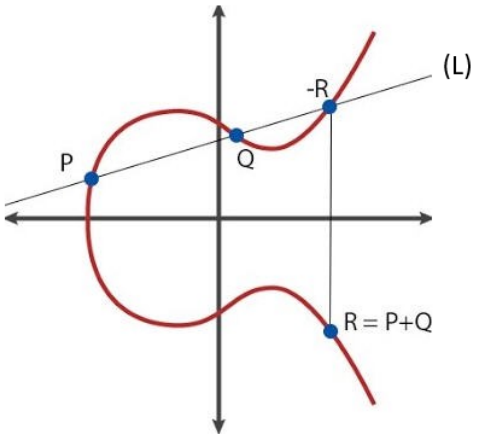


FIGURE 1.3 – Addition de deux points sur une courbe elliptique.

2. *Cryptographie à base de couplage [11] :* Un autre avantage de la cryptographie à base de courbes elliptiques est qu'un opérateur bilinéaire peut être défini entre les groupes. Soit $\mathbb{G}_1$ un groupe cyclique additif d'ordre un nombre premier q et $\mathbb{G}_2$ un groupe multiplicatif du même ordre q. Une fonction $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ est appelée couplage cryptographique bilinéaire (également appelé pairing) et notée $e(.,.)$, si elle satisfait les propriétés suivantes :
   (a) Bilinéarité : Pour tout $P, Q \in \mathbb{G}_1$ et $a, b \in \mathbb{Z}$; $e(aP, bQ) = e(P, Q)^{ab}$,
   (b) Non-dégénérescence : $e(P, P)$ est un générateur de $\mathbb{G}_2$ et ainsi $e(P, P) \neq 1$,
   (c) Calculabilité : Il existe un algorithme efficace pour calculer $e(P, Q)$ pour tout $P, Q \in \mathbb{G}_1$.
3. *Cryptographie à base d'identité [12, 13] :* Les pairings ont été introduites dans de nombreuses primitives cryptographiques telles que les signatures et le chiffrement. Mais l'application qui reste la plus importante est le chiffrement à base d'identité (IBE).
   L'un des gros problèmes des cryptosystèmes asymétriques est la gestion des clés. Il existe des autorités de certification mais celles-ci ne sont pas reconnues par tout le monde et la mise en place d'une infrastructure à clé publique (PKI) coûte très chère.
   Le chiffrement à base d'identité est un cryptosystème asymétrique où la clé publique d'un utilisateur est une chaîne arbitraire liée à son identité, par exemple : une adresse e-mail, un numéro de téléphone ou une adresse IP, et la clé privée est donnée par une autorité de confiance. Le chiffrement basé sur l'identité a été initialement proposé par Shamir [13] en 1984, et le premier protocole a été proposé en 2001 par Boneh et Franklin [12]. Dans ce protocole, le générateur de clé (PKG) connaît la clé privée. Ce problème peut être corrigé en utilisant un PKG distribué comme celui proposé par Pedersen [3] ou Gennaro [14]. Dans ces protocoles, une clé maîtresse est générée de manière à ce que chacun des m PKG construit aléatoirement un fragment. Un tel schéma se déroule en quatre étapes que nous représentons par les fonctions suivantes :
   (a) *Setup() :* Le centre de génération de clés (PKG) commence par générer les différents paramètres publics du schéma à savoir un groupe $\mathbf{G}_1$ additif et un autre $\mathbf{G}_2$ multiplicatif d'ordre un nombre premier q, un générateur $P \in \mathbf{G}_1$, une fonc-

3. https://www.cryptoworkshop.com/ximix/lib/exe/fetch.php?media=pedersen.pdf

tion de pairing $e(.,.)$, deux fonctions de hachage H_1 et H_2 ainsi que la clé maîtresse privée msk et sa clé maîtresse publique correspondante $mpk = msk \cdot P$.

(b) *Extract(msk, id) :* Le PKG calcule la clé privée relative à l'identité id en utilisant la formule suivante : $sk = msk \cdot H_1(id)$.

(c) *Encrypt(mpk, id, Msg) :* Afin de chiffrer un message Msg, on utilise la clé maîtresse publique générée dans la première étape mpk et l'identité du destinataire id. Le chiffré est donné par la formule suivante : $EncMsg = (r \cdot P, Msg \oplus H_2(g_{id}^r))$; avec $r \in \mathbf{Z}_q^*$ un nombre aléatoire et $g_{id} = e(H_1(id), mpk)$.

(d) *Decrypt(sk, EncMsg) :* Le destinataire utilise sa clé privée pour déchiffrer $EncMsg = (u, v)$ avec la formule suivante : $Msg = v \oplus H_2(e(msk, u))$.

4. *Cryptosystème de Paillier [15] :* Il s'agit d'un cryptosystème à clé publique et non déterministe, proposé par Pascal Paillier en 1999. Il est basé sur des calculs dans le groupe $\mathbb{Z}_{n^2}^*$. Ce schéma a la propriété d'homomorphisme additif, qui permet le chiffrement de nombreux bits en une seule opération avec un facteur d'expansion constant, et permet un déchiffrement efficace. Ce facteur le rend potentiellement intéressant pour de nombreux protocoles cryptographiques tels que le vote électronique et les réseaux de mélangeurs.
5. *Cryptosystème d'El-Gamal [3] :* En 1985, El-Gamal invente une technique de chiffrement asymétrique probabiliste, c'est à dire qu'un message m peut avoir plusieurs chiffrés différents. Ce cryptosystème est basé sur la difficulté du problème des logarithmes discrets (*Discret Logarithm Problem DLP*). Ce cryptosystème se déroule en trois étapes :

   (a) **Génération de clés :** Soit $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q. La clé publique est notée y et est représenté par la formule suivante : $y = g^x \mod p$; avec $g \in \mathbb{G}$ est un générateur, $x \in \mathbb{Z}_q$ est la clé secrète et $p = 2q + 1$.

   (b) **Chiffrement :** Le chiffré d'un message $m \in \mathbb{G}$ est représenté par le couple suivant : $E_y[m] = (u, v) = (g^r \mod p, m \cdot y^r \mod p)$; avec r un nombre aléatoire appartenant à $\mathbb{Z}_q$.

   (c) **Déchiffrement :** m est obtenu à partir de (u, v) en utilisant la formule suivante : $m = v / u^x$.

Les preuves à divulgation nulle de connaissance

Les preuves à divulgation nulle de connaissance (*Zero Knowledge proofs ZKP*) [16] sont des primitives cryptographiques qui permettent à une partie, appelée "prouveur", de prouver à une autre partie, appelée "vérificateur", qu'elle connaît un secret sans révéler le secret lui-même ni aucun autre secret. La preuve à divulgation nulle de connaissance non-interactive (*Non-Interactive Zero Knowledge Proofs NI-ZKP*) [17, 18] est une variante de la ZKP dans laquelle aucune interaction bidirectionnelle entre le vérificateur et le prouveur n'est nécessaire.

Fonction de hachage

Notée $H(m)$, il s'agit d'une fonction qui convertit un message $m \in \{0,1\}^*$, ayant une longueur quelconque, en un message $a \in \{0,1\}^n$ d'une longueur fixe donnée. Ces fonctions ont les propriétés utiles suivantes : elles sont difficiles à inverser, ce qui signifie qu'il est difficile de trouver un message m à partir de la valeur de sortie a. En outre, il est difficile de trouver deux entrées qui ont la même sortie, c'est-à-dire qu'elles sont résistantes aux collisions. La fonction de hachage est d'une grande utilité, notamment dans le contexte de la signature numérique, où elle évite la tâche fastidieuse de signer de longs messages.

1.2.2 Exigences de sécurité d'un protocole de vote électronique

Un protocole de vote électronique sécurisé doit satisfaire plusieurs propriétés et exigences de sécurité. Les chercheurs ont identifié de nombreuses propriétés de sécurité. Ces propriétés peuvent être classées en trois catégories [19] :

1. *Propriétés de correction et de robustesse :* Cette catégorie inclue les propriétés suivantes :
   - *Éligibilité :* Signifie que seuls les votants enregistrés à l'élection peuvent voter et qu'un seul vote par votant est comptabilisé. Si le votant a la possibilité de voter plus qu'une fois, le vote le plus récent est compté et tous les autres sont rejetés.
   - *Pas de résultat partiel :* Signifie qu'aucun résultat partiel, susceptible d'influencer les choix des électeurs, n'est publié avant le décompte officiel.
   - *Robustesse :* Signifie que le protocole peut tolérer un certain nombre de mauvais comportements de la part des électeurs.
   - *Intégrité :* Signifie que les bulletins de vote ne sont ni altérés ni supprimés tout au long du processus de vote.
2. *Propriétés de vérifiabilité :* Cette catégorie est généralement divisée en deux propriétés :
   - *Vérifiabilité individuelle :* Signifie que l'électeur lui-même doit être en mesure de vérifier que son bulletin de vote a été déposé comme prévu et comptabilisé comme déposé.
   - *Vérifiabilité universelle :* Tout le monde peut vérifier que le résultat annoncé correspond à la somme de tous les votes éligibles.
3. *Propriétés de confidentialité :* Elles sont :
   - *Anonymat :* Signifie qu'aucune personne, à part le votant lui même, ne peut établir un lien entre le votant et son vote.
   - *Sans-reçu :* Signifie que le votant ne peut pas construire un reçu lui permettant de prouver à un tiers qu'il a voté pour un candidat donné. Cela vise à empêcher l'achat de votes.
   - *Résistance à la coercition :* Signifie que même lorsqu'un électeur interagit avec un attaquant pendant le processus de vote, ce dernier ne peut pas être sûr si le votant a suivi ses instructions ou a voté pour un autre candidat.

Afin de prouver la sécurité de nos approches, présentées dans les chapitres 3, 4 et 5, nous les évaluons contre la liste des propriétés de sécurité définies ci-dessus. Nous ajoutons à cette liste deux autres propriétés qui sont :

- *Voter et quitter :* Signifie que le système n'oblige pas le votant à attendre la fin de la phase de vote ni de participer à la phase de comptage. Il peut tout simplement voter et quitter le système de vote.
- *Politique du vote :* Spécifie si le système offre aux votants la possibilité de modifier leur vote avant la fin de la phase de vote ou bien, une fois qu'ils votent, ils n'ont plus droit au changement d'avis.

1.2.3 Vérification formelle de la sécurité des protocoles de vote électronique

Les protocoles cryptographiques sont généralement difficiles à concevoir et à analyser. Nombreux sont les protocoles que l'on croyait corrects et sécurisés, depuis plusieurs années, et qui ont été découverts, au moyen des techniques de vérification formelle, dé-

faillants et vulnérables. De là vient l'importance des méthodes de vérification formelle de la sécurité des protocoles cryptographiques.

Dans ce contexte, et afin de vérifier la sécurité de nos protocoles, que nous proposons dans les chapitres 3, 4 et 5, nous élaborons des analyses formelles en utilisant le langage de modélisation formelle *Applied Pi-Calculus [20, 21]* ainsi que l'outil de vérification automatique *ProVerif [22]*. Nous considérons le modèle formel de Dolev-Yao [23] : nous supposons que les primitives cryptographiques fonctionnent parfaitement, et que l'attaquant contrôle les canaux publics. L'attaquant peut voir, intercepter et insérer des messages sur les canaux publics, mais il ne peut chiffrer, déchiffrer, signer des messages ou effectuer d'autres opérations cryptographiques que s'il dispose de la clé correspondante.

Nous vérifions formellement, pour chaque protocole, trois propriétés de sécurité qui sont : **Le secret de vote, l'authentification des votants et la confidentialité des votes**.

Applied Pi-Calculus [20, 21] C'est un langage de modélisation formelle permettant de décrire et d'analyser les protocoles de sécurité. Il fournit une syntaxe de processus intuitive pour détailler les actions des participants d'un protocole, en mettant l'accent sur leur communication. La syntaxe est couplée à une sémantique formelle pour pouvoir évaluer les protocoles. Le langage est basé sur *Pi-Calculus* tout en ajoutant un ensemble de termes algébriques pour permettre la modélisation des opérations cryptographiques utilisées par les protocoles de sécurité. Une grande variété de primitives cryptographiques peut être modélisée de façon abstraite au moyen d'une théorie équationnelle. Applied Pi-Calculus permet d'exprimer plusieurs types d'objectifs de sécurité, et d'analyser si le protocole atteint ou non son objectif.

ProVerif [22] Il s'agit d'un outil de vérification formelle automatique pour les protocoles cryptographiques, décrits dans le modèle formel de Dolev-Yao. Cet outil est basé sur une représentation du protocole par des clauses de Horn. Il peut traiter de nombreuses primitives cryptographiques différentes, y compris la cryptographie à clé partagée et à clé publique (chiffrement et signatures) et les fonctions de hachage. Il peut gérer un nombre illimité de sessions du protocole (même en parallèle) et un espace de message illimité. Ce résultat a été obtenu grâce à quelques approximations bien choisies. Cela signifie que le vérificateur peut donner de fausses attaques, mais s'il prétend que le protocole satisfait à une certaine propriété, alors cette propriété est effectivement satisfaite. Lorsque l'outil ne peut pas prouver une propriété, il tente de reconstruire une attaque, c'est-à-dire une trace d'exécution du protocole qui falsifie la propriété souhaitée.

ProVerif peut prouver les propriétés suivantes :

— *Le secret :* l'adversaire ne peut pas obtenir le secret,
— *L'authentification et plus généralement les propriétés de la correspondance,*
— *Les équivalences entre des processus qui ne diffèrent que par les termes.*

1.3 Technologie Blockchain

Comme son nom l'indique, la Blockchain est une chaîne de blocs qui contient des informations. Cette technologie permet de stocker et de transmettre des données d'une manière transparente et sûre, sans qu'il soit nécessaire de faire appel à un tiers de confiance. Elle peut être considérée comme un grand registre public qui contient l'historique de tous les échanges effectués entre ses utilisateurs depuis sa création. La technologie Blockchain

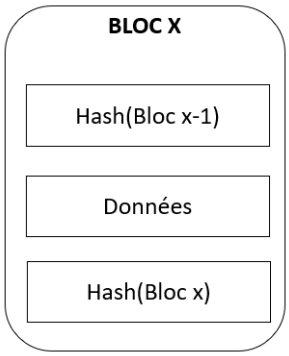


FIGURE 1.4 – Structure d'un bloc.

a une propriété intéressante : une fois que les données ont été enregistrées dans une Blockchain, il devient très difficile de les modifier. Elle a été introduite pour la première fois par un certain pseudonyme Satoshi Nakamoto, en 2009 [4], qui a créé la crypto-monnaie *Bitcoin*. Ensuite, plusieurs plateformes de Blockchain ont été proposées. Dans ce qui suit, nous détaillons le principe de fonctionnement de cette technologie.

1.3.1 Structure d'un bloc

La Figure 1.4 modélise la structure d'un bloc d'une manière simple. En effet, chaque bloc contient :

— Un ensemble de données qui dépendent du type de la Blockchain. La Blockchain Bitcoin, par exemple, stocke les détails de chaque transaction qui sont l'émetteur, le destinataire et le montant échangé en Bitcoin.
— Un bloc comporte également un hash. On peut comparer un hash à une empreinte digitale. Il identifie un bloc et tout son contenu. Il est toujours unique, tout comme une empreinte digitale. Une fois qu'un bloc est créé, son hash est calculé. Si on modifie quelque chose à l'intérieur du bloc, cela entraîne la modification de son hash. En d'autres termes, les hashs sont très utiles pour détecter les modifications d'un bloc.
— Le troisième élément qui constitue un bloc est le hash du bloc précédent. C'est cet hash qui crée la chaîne de blocs et c'est la technique qui permet de sécuriser la technologie Blockchain.

Le bloc initial d'une Blockchain est connu sous le nom de "bloc Genesis" ou "bloc 0". Sa particularité est qu'il ne contient pas de référence à un bloc précédent. Une fois que le bloc Genesis a été initialisé, le "Block 1" est créé et, lorsqu'il est complet, il est attaché au bloc Genesis. Comme nous l'avons mentionné, chaque bloc contient des données ou *des transactions*. Ces transactions sont hachées, puis les hashs sont rassemblés et hachés à nouveau, cela continue jusqu'à ce qu'il ne reste qu'un seul hash, appelé racine de Merkle. L'entête du bloc est l'endroit où est stockée la racine de Merkle. Pour s'assurer qu'une transaction ne peut pas être modifiée, chaque bloc conserve également un enregistrement de l'entête du bloc précédent, ce qui signifie que pour modifier des données, on doit modifier le bloc qui enregistre la transaction ainsi que tous les blocs suivants, comme le montre la Figure 1.5.

4. https://bitcoin.org/bitcoin.pdf

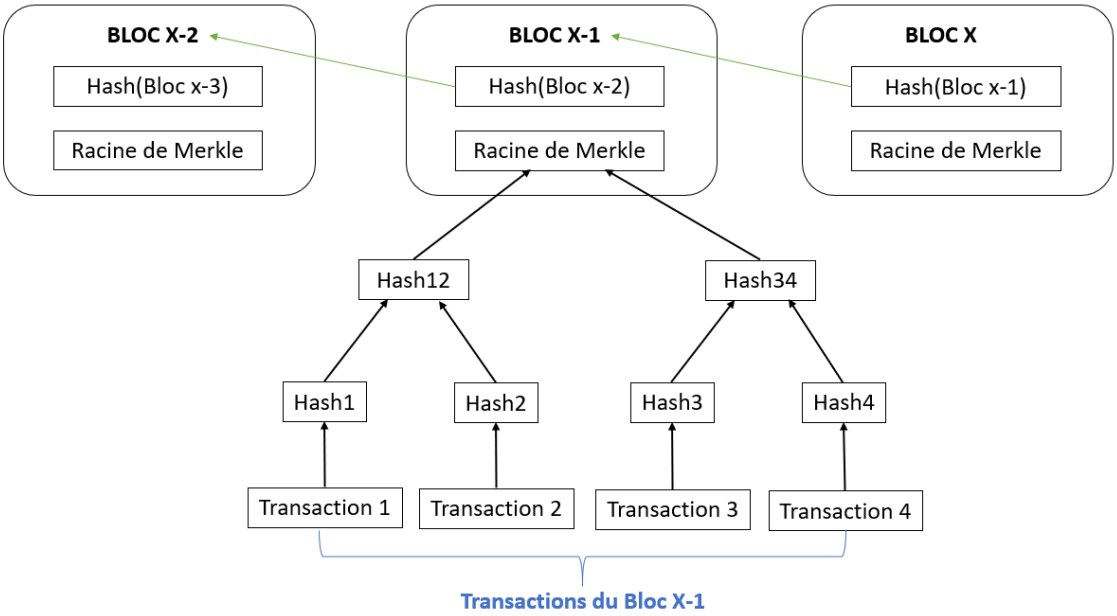


FIGURE 1.5 – Structure d'une Blockchain.

1.3.2 Consensus

Pour éviter les altérations, la technologie Blockchain ne se limite pas à l'utilisation des fonctions de hachage. De nos jours, les ordinateurs sont très rapides et peuvent calculer des centaines de milliers de hashs par seconde. On pourra effectivement altérer un bloc et recalculer tous les hashs des blocs suivants pour rendre une Blockchain valide de nouveau. Pour pallier ce problème, les Blockchains ont donc mis en place un mécanisme de consensus. C'est un mécanisme qui ralentit la création du bloc. Dans le cas de Bitcoin, il faut environ 10 minutes pour calculer l'algorithme de consensus requis, appelé "preuve de travail", et ajouter un nouveau bloc à la chaîne. Ce mécanisme rend la falsification des blocs très difficile car si on falsifie un bloc, on doit recalculer la preuve de travail pour tous les blocs suivants. La sécurité de la Blockchain est donc assurée grâce à l'utilisation des fonctions de hachage et au mécanisme de consensus.

Mais il y a une autre façon pour les Blockchains de se sécuriser, c'est en étant distribuées. Au lieu d'utiliser une entité centrale pour gérer la chaîne, la Blockchain utilise un réseau de pair à pair et tout le monde est autorisé à y participer. Lorsqu'un nœud rejoint ce réseau, il obtient une copie complète du réseau. Le nœud utilise cette copie pour vérifier la validité de la Blockchain. En effet, lorsqu'un nœud crée un nouveau bloc, ce dernier est diffusé à tous les nœuds du réseau. Ensuite, chaque nœud vérifie la validité du bloc et l'ajoute à sa copie locale de la Blockchain. Les blocs invalides ou altérés seront rejetés par les nœuds honnêtes du réseau. Par conséquent, pour falsifier un bloc dans une Blockchain, l'attaquant doit régénérer tous les blocs suivants, exécuter l'algorithme de consensus pour obtenir une preuve pour chaque bloc et prendre le contrôle de plus de 50% des nœuds dans la Blockchain, ce qui est presque difficile voire impossible à faire.

Allons maintenant un peu plus loin dans les détails techniques et essayons de comprendre comment les algorithmes de consensus fonctionnent. Chaque Blockchain a son propre algorithme de consensus. Il existe donc plusieurs algorithmes de consensus. Nous allons voir trois parmi les consensus les plus populaires.

— *Preuve de travail :* Pour qu'un mineur soit élu en tant que leader et choisisse le bloc suivant à ajouter à la Blockchain, il doit trouver une solution à un problème mathé-

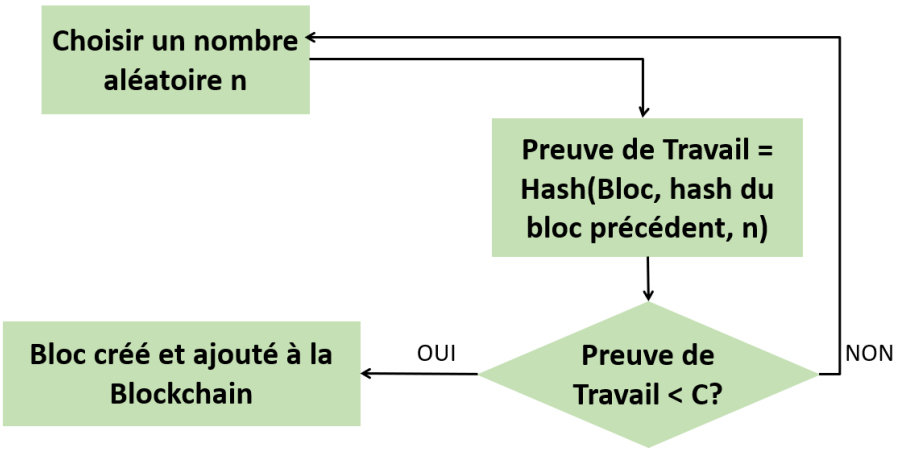


FIGURE 1.6 – Preuve de travail.

matique particulier. Que ce problème mathématique soit :
Étant donné un message "X" et un paramètre C *fixé au préalable, trouvez un nombre n tel que le hash de n* + X *est un nombre inférieur à* C.

$$
\begin{aligned}
&\mathrm{C} = 10, \mathrm{X} = {}' test' \\
&hash(\mathrm{X}) = hash('test') = 0x0f = 15 > 10 \\
&hash(\mathrm{X}+1) = hash('test1') = 0xff = 255 > 10 \\
&hash(\mathrm{X}+5) = hash('test5') = 0x\mathrm{A}2 = 162 > 10 \\
&\quad . \\
&\quad . \\
&\quad . \\
&hash(\mathrm{X}+n) = hash('testn') = 0x09 = 9 < 10 \ \textit{OK, résolu.}
\end{aligned}
$$

Étant donné que la fonction de hachage utilisée est sécurisée, la seule façon de trouver une solution à ce problème est de recourir à la force brute (en essayant toutes les combinaisons possibles). En d'autres termes, d'un point de vue probabiliste, l'acteur qui résoudra en premier le problème mentionné ci-dessus est la plupart du temps celui qui a accès à la plus grande puissance de calcul. Ce consensus a les propriétés suivantes :

- Il est difficile de trouver une solution pour ce problème donné,
- Lorsqu'une solution à ce problème est donnée, il est facile de vérifier qu'elle est correcte.

Ce consensus peut ainsi être modélisé par la Figure 1.6.

— *Preuve d'enjeu :* Dans une loterie, de manière probabiliste, si Bob a plus de billets qu'Alice, il a plus de chances de gagner. D'une manière très similaire, dans la preuve de l'enjeu, si Bob a plus d'enjeu qu'Alice, il a plus de chances de gagner (miner le bloc suivant). Le consensus de la preuve d'enjeu supprime les besoins en énergie et en puissance de calcul de la preuve de travail et les remplace par un enjeu. L'enjeu est une somme d'argent qu'un acteur est prêt à bloquer pendant un certain temps. En retour, il a une chance proportionnelle à sa mise, d'être le prochain mineur et de sélectionner le bloc suivant. Ce consensus est modélisé par la Figure 1.7.

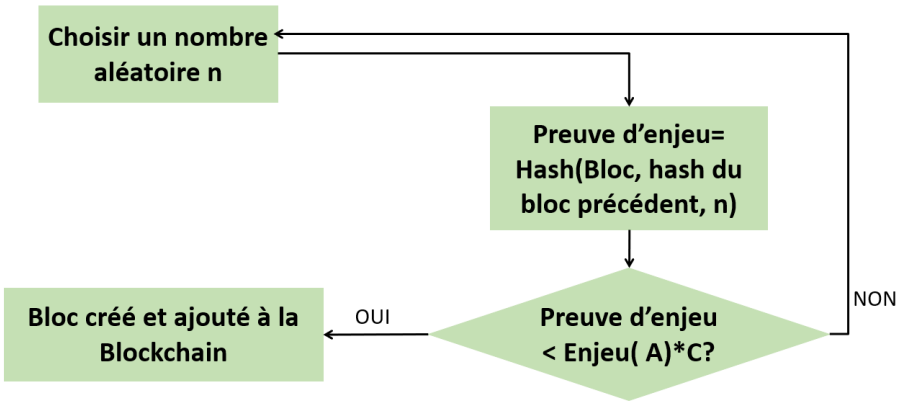


FIGURE 1.7 – Preuve d'enjeu.

— *Consensus basé sur la tolérance aux fautes byzantines :* L'algorithme de consensus basé sur la tolérance aux fautes byzantines se déroule en quatre étapes :

1. **Proposition de la transaction :** Un client génère un message pour exécuter des opérations spécifiques à résoudre par le réseau. Ce message correspond à un message de proposition de transaction qui doit être envoyé au pair endosseur. La proposition de transaction consiste à invoquer les opérations et à calculer les paramètres *stateUpdate* et *verDep*. Le paramètre *stateUpdate* correspond à l'état de la Blockchain après avoir simulé, localement, les opérations. Le paramètre *verDep* met en relation les variables impliquées dans la transaction avec les versions locales de la variable dans le nœud du client et leurs opérations respectives. Une fois l'exécution des opérations déclenchées par le client est terminée, ce dernier génère le message de proposition de transaction et le soumet ensuite aux pairs endosseurs.
2. **Endossement de la transaction :** Chaque pair endosseur vérifie la signature du client venant dans le message de proposition de transaction. Si la signature du client est valide, l'endosseur simule la proposition de transaction en exécutant l'opération correspondante, puis vérifie si les paramètres stateUpdate et verDep sont corrects. Si le processus de validation est réussi, l'endosseur génère un message de transaction valide à envoyer au client. En cas d'échec du processus de simulation, l'endosseur génère un message de transaction invalide.
3. **Diffusion au consensus :** Le client attend la réponse des pairs endosseurs. Lorsqu'il reçoit suffisamment de messages de transaction valide dûment signés, le client stocke les signatures d'endossement dans un paquet appelé endossement. Une fois que la transaction est considérée comme approuvée, le client invoque les services de consensus. Le nombre de réponses valides nécessaires pour considérer que la proposition de transaction est correctement endossée dépendra de la configuration de la Blockchain. Si la transaction n'a pas recueilli suffisamment d'endossements, le client abandonne cette proposition de transaction.
4. **Validation :** Une fois que le client a diffusé la transaction dûment approuvée au consensus, les services de commande collectent cette transaction et l'organisent en un bloc. Le processus de consensus pour le nouveau bloc est effectué par les donneurs d'ordre sur la base d'un processus de tolérance aux fautes byzantines, où les donneurs d'ordre actifs voteront pour la validité du bloc et

conviendront du nouveau bloc sur la base d'une majorité de votes. Le bloc construit par les services de commande a la forme suivante : B = ([tx_1, tx_2,..., tx_k]; h), où h correspond à la valeur de hachage du bloc précédent. Les pairs attendront de recevoir 50% + 1 des réponses valides du service de commande pour considérer que le bloc est prêt à être validé. Une fois que les pairs ont la confirmation que le nouveau bloc est correcte, ils vérifient si l'endossement de chaque transaction est valide selon la politique conguë dans le réseau. Les pairs vérifient ensuite le paramètre verDep afin de s'assurer qu'il n'y a pas de conflit entre l'opération, les variables impliquées dans la transaction et l'état actuel de la Blockchain. Si ce processus se termine avec succès, les transactions sont validées. En cas d'échec de l'une des validations, les pairs considèrent la transaction comme invalide et la suppriment. Enfin, les transactions invalides sont communiquées au client.

1.3.3 Propriétés des Blockchains

Les principales caractéristiques de la technologie Blockchain sont les suivantes :

— *Décentralisation :* Dans une Blockchain, les données ne sont pas stockées par une seule entité. En fait, tout le monde dans le réseau est propriétaire de l'information. Si un nœud souhaite interagir avec un autre nœud dans la même Blockchain, elle peut le faire directement sans passer par une tierce partie.
— *Transparence :* Toutes les informations stockées dans une Blockchain sont accessibles par tous les nœuds du réseaux.
— *Immutabilité :* Une fois qu'une donnée est ajoutée à une Blockchain, il devient impossible de la modifier ou de la supprimer.

1.3.4 Types des Blockchains

Les Blockchains peuvent être classées, généralement, en trois catégories :

— *Blockchains publiques :* Dans une Blockchain publique, n'importe qui dans le monde peut lire, envoyer des transactions et s'attendre à les voir incluses si elles sont valables, et peut participer au processus de consensus (c'est le processus qui permet de déterminer quels blocs sont ajoutés à la chaîne et quel est l'état actuel). Les Blockchains publiques sont sécurisées par la combinaison d'incitations économiques et de vérification cryptographique utilisant des mécanismes tels que la preuve du travail ou la preuve de l'enjeu, suivant un principe général selon lequel le degré d'influence d'une personne dans le processus de consensus est proportionnel à la quantité de ressources économiques qu'elle peut apporter. Ces Blockchains sont généralement considérées comme étant "totalement décentralisées".
— *Blockchain de consortium ou hybrides :* Dans une Blockchain hybride, le processus de consensus est contrôlé par un ensemble de nœuds pré-sélectionnés. Le droit de lecture à partir de ce type de Blockchain peut être publique, ou limité aux participants. Ces Blockchains peuvent être considérées comme "partiellement décentralisées".
— *Blockchain privées :* Dans une Blockchain privée, les droits d'écriture sont centralisées dans une organisation. Les autorisations de lecture peuvent être publiques ou limitées de façon arbitraire.

Nous élaborons un tableau comparatif entre ces différents types de Blockchain (Table 1.1).

	Blockchains publiques	**Blockchains hybrides et privées**
Accès	Sans autorisation	Avec autorisation
Sécurité	Algorithmes de consensus	Participants pré-approuvés
Identité	Pseudonyme	Connue
Confiance entre les nœuds	Non-nécessaire	Nécessaire
Performance	Lent	Rapide
Consommation d'énergie	Large	Faible
Coût des transactions	Élevé	Faible

TABLEAU 1.1 – Tableau comparatif des types de Blockchain

1.3.5 Blokchains les plus populaires

Dans cette partie, nous présentons quatre Blockchains parmi les Blockchains les plus connues et à partir desquelles nous allons tirer profit dans nos contributions. Ces Blockchains sont : *Bicoin, Ethereum, Hyperledger Fabric et Monero.*

Bitcoin

La célèbre crypto-monnaie pour laquelle la technologie Blockchain a été inventée est le Bitcoin, proposé par un certain pseudonyme "Satoshi Nakamoto"[5] . Ce dernier combine les technologies de sécurité précédentes : *Hashcash*[6]*, chiffrement asymétrique, consensus [24] et Merkle tree [25]* pour inventer Bitcoin. Officiellement, le premier bloc a été initié en 2009.

Bitcoin répond à la grande question qui a été posée il y a longtemps : comment pouvons-nous éliminer la banque et forcer les transactions P2P? La réponse de Nakamoto était simple : elle est analogue à "tout le monde est la banque", où la plupart des participants conservent une copie des données qui seraient de la responsabilité de la banque. Ainsi, l'expéditeur et le destinataire sont totalement indépendants de tout contrôle par un tiers. D'autre part, ils sont soumis à un nouveau contrôle de réseau, qui est le consensus. Grâce à ce consensus, la majorité des utilisateurs du réseau votent pour que les transactions soient acceptées ou bloquées. Bitcoin est un système peer-to-peer entièrement distribué. Ainsi, il n'y a pas de serveur central ou de système de point de contrôle unique. Les Bitcoins sont créés par un processus appelé *"mining"*. Tout participant au réseau Bitcoin peut agir comme un mineur, en utilisant la puissance de calcul dont il dispose pour résoudre le problème. Toutes les 10 minutes en moyenne, une nouvelle solution est trouvée par quelqu'un qui est alors capable de valider les transactions du bloc en cours. En résumé, l'extraction de Bitcoin décentralise l'émission de monnaie et concilie les différentes procédures, rendant inutile qu'un organisme similaire agisse comme une banque centrale. Il existe deux types de participants ou nœuds : les mineurs qui sont les utilisateurs participant à la création des blocs et sont incités par un montant de Bitcoin à garantir leur présence dans le réseau et les non mineurs qui bénéficient du système de Bitcoin sans participer à la création des blocs. Le deuxième rôle principal de cette incitation est la création de la crypto-monnaie Bitcoins, qui commence par 50BTC pour chaque bloc (toutes les 10 minutes) lors de son lancement en 2009 et qui est réduite de moitié tous les 4 ans pour atteindre environ 21 milliards de BTC en 2140.

5. https://bitcoin.org/bitcoin.pdf
6. http://www.hashcash.org/papers/hashcash.pdf

Ethereum

Ethereum [7] a été créée par un programmeur russo-canadien, appelé "Vitalik Buterin". Il a co-fondé Ethereum à l'âge de 19 ans, en septembre 2014. La première version de la plate-forme a été présentée en juin 2015 et la version officielle en mars 2016.

Cette plate-forme peut être considérée comme un ordinateur mondial, que chacun peut programmer et utiliser comme il le souhaite. Cet ordinateur est toujours disponible, sécurisé, et tout ce qui est fait à l'aide de cet ordinateur est public. Il permet de développer une nouvelle catégorie d'applications appelées applications décentralisées (DApps). Ces applications fonctionnent sur le réseau Ethereum, qui est composé de plusieurs milliers d'ordinateurs qui communiquent en permanence. Elles partagent les mêmes données qui sont stockées sur la Blockchain. Ethereum inclut dans ses blocs des programmes exécutables qui déclenchent des actions en fonction des informations reçues ou des conditions atteintes. On parle de contrats intelligents ou *smart contracts*. Un contrat intelligent est un programme autonome qui, une fois lancé et déployé dans la Blockchain, exécute des conditions prédéfinies. Ethereum peut être considérée comme une machine d'état basée sur des transactions, où les transactions peuvent changer l'état. Elle garde une trace des interactions et est composée de deux types de comptes :

— *Externally Owned Account (EOA) :* il s'agit d'un compte contrôlé par l'utilisateur. Il est caractérisé par une paire de clés publique/privée. Un EOA peut envoyer des transactions pour transférer de l'éther ou déclencher un code de contrat. Il est contrôlé par sa clé privée et n'a pas de code associé. Cette clé privée est utilisée pour signer des transactions et prouver l'identité de l'expéditeur.
— *Contract Account :* c'est un ensemble de codes et de données qui se trouve à une adresse définie. Il ne possède qu'une clé publique. Son exécution est activée par les transactions reçues des autres comptes.

Les comptes Ethereum sont identifiés par leurs adresses qui sont construites à partir de la clé publique en prenant les 20 derniers octets.

Chaque transaction dans la Blockchain Ethereum contient les information suivantes :

— *From :* Une signature d'un EOA pour autoriser la transaction.
— *To :* Le destinataire de la transaction et peut être une adresse d'un EOA ou une adresse de contrat.
— *Data :* Contient le code du contrat pour créer un nouveau contrat ou les instructions d'exécution du contrat.
— *Gas Price :* Le taux de conversion de l'achat de gas (sous unité de l'éther) en utilisant la crypto-monnaie éther.
— *Total Gas :* La quantité maximale de gas pouvant être consommée par la transaction.
— *Nonce :* Un compteur incrémenté pour chaque nouvelle transaction à partir d'un compte.

Hyperledger Fabric

La Fondation Linux a fondé le projet Hyperledger [8] en 2015 pour faire progresser la technologie Blockchain. Plutôt que de déclarer une norme unique pour les Blockchains, elle encourage une approche collaborative du développement de cette technologie via un processus communautaire, avec des droits de propriété intellectuelle.

7. https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf
8. https://www.hyperledger.org/

Hyperledger Fabric [26] est l'un des projets de Blockchain au sein d'Hyperledger. Comme d'autres technologies de Blockchain, il possède un registre, utilise des contrats intelligents et constitue un système permettant aux participants de gérer leurs transactions. Là où Hyperledger Fabric se distingue des autres Blockchains, c'est qu'il est privé et avec permission. Plutôt qu'un système ouvert sans autorisation qui permet à des identités inconnues de participer au réseau, les membres d'un réseau Hyperledger Fabric s'inscrivent par l'intermédiaire d'un *Membership Service Provider (MSP)* de confiance.

La plate-forme possède une architecture modulaire qui offre des degrés élevés de confidentialité, de flexibilité, de résilience et d'évolutivité. Cela permet aux solutions développées avec Fabric d'être adaptées à n'importe quel secteur d'activité. Fabric exploite la technologie des conteneurs pour héberger des contrats intelligents appelés "chaincode" qui contiennent les paramètres de configuration du système. Il est conçu pour prendre en charge divers composants enfichables et pour s'adapter à la complexité qui existe dans l'ensemble de l'économie. Fabric est une plate-forme extensible, prenant en charge divers protocoles de consensus, de sorte qu'elle peut être adaptée à différents cas d'utilisation et modèles de confiance. Fabric exécute des applications distribuées écrites dans des langages de programmation d'usage général sans dépendre d'une quelconque crytpo-monnaie. Fabric peut également créer des canaux, qui permettent à un groupe de participants de créer un registre séparé des transactions. Cela est particulièrement important pour les réseaux où certains participants peuvent être des concurrents qui ne veulent pas que chaque transaction soit connue de tous les participants du réseau. Si un groupe de participants forme un canal, seuls ces participants et aucun autre disposent de copies du registre pour ce canal.

Le rôle d'une transaction dans Hyperledger Fabric est de mémoriser l'état d'un objet. Chaque transaction comporte 5 champs :

— *Header :* L'entête d'une transaction contient les métadonnées importantes concernant la transaction, telles que le nom du chaincode et sa version.
— *Signature :* Le client signe la transaction avec une clé privée afin de vérifier l'intégrité des enregistrements de la transaction.
— *Proposal :* Pour chaque mise à jour de l'état du registre, le chaincode nécessite une proposition de l'application contenant les paramètres d'entrée pour changer l'état actuel en un nouvel état.
— *Response :* Chaque fois qu'un chaincode exécute une proposition de changement de l'état actuel, la réponse de la transaction saisit l'état avant et après, en tant que jeu de lecture/écriture. Si la sortie du chaincode est correcte, la transaction est validée avec succès et le registre met à jour son état.
— *Endorsement :* Le résultat d'une transaction doit être accepté par chaque organisation du réseau avant d'être mis à jour dans le registre. Ainsi, une liste de réponses de transactions signées doit être acceptée par l'organisation. Le développeur peut définir des politiques d'endossement pour l'organisation et peut être exécuté à l'aide de Chaincode.

Monero

Lancé en avril 2014, Monero est une crypto-monnaie open-source qui a été bifurquée du protocole CryptoNote[9]. Tout comme Bitcoin, Monero peut être utilisé pour acheter divers biens et services sur Internet. Monero se distingue par un niveau plus élevé de

9. https://www.allcryptowhitepapers.com/wp-content/uploads/2018/05/monero-whitepaper.pdf

confidentialité des transactions que ses homologues. Le protocole atteint un niveau plus élevé de confidentialité des transactions grâce à la mise en œuvre de technologies cryptographiques, à savoir :

— Transactions confidentielles en anneau (*Ring Confidential Transactions RingCT*) [27] : Elles fonctionnent d'une manière différente. Pour illustrer cela, considérons la situation suivante : A possède 10 monero, et voudrait envoyer 5 monero à B. Comme une sortie sur la Blockchain monero ne peut pas être dépensée deux fois, A est tenu de dépenser la sortie dans sa totalité, et de rendre la monnaie à lui-même. Ainsi, la transaction de A serait la suivante : une entrée de 10 monero, et 2 sorties. Une sortie de 5 monero est destinée à B, et l'autre de 5 monero est renvoyée à A comme monnaie.
L'objectif des transactions confidentielles en anneau est de permettre aux seuls participants à la transaction de voir le montant qui est transféré, et de dissimuler ce montant aux parties extérieures. Toutefois, il est également nécessaire que le réseau soit en mesure de confirmer la validité de la transaction qui a été initiée. Afin de prouver que la transaction entre A et B n'est pas frauduleuse, la somme des entrées de la transaction doit être égale à la somme de ses sorties. Dans ce cas, dans la transaction entre A et B, l'entrée de 10 monero, doit également être égale à la sortie de 10 monero.

— Adresses furtives (*stealth addresses*) [28] : C'est une méthode qui permet d'offrir une sécurité supplémentaire au destinataire d'une monnaie numérique en demandant à l'expéditeur de créer une adresse aléatoire unique pour une transaction donnée. Lorsque plusieurs transactions envoyant des fonds à une adresse furtive sont effectuées, au lieu que ces transactions apparaissent sur la Blockchain comme des paiements multiples à la même adresse, ce qui sera enregistré sera en fait de multiples paiements sortants à des adresses différentes. Il est donc impossible de relier les transactions à l'adresse publiée du bénéficiaire ou à des adresses générées une seule fois. Le propriétaire de l'adresse furtive peut alors utiliser sa clé privée pour voir toutes ses transactions entrantes.

— Signatures en anneau (*ring signatures*) [29] : Monero utilise la technologie de la signature en anneau pour protéger la vie privée de l'utilisateur lors de la saisie d'une transaction. Une signature en anneau est un type de signature numérique dans laquelle un groupe de signataires possibles sont fusionnés pour produire une signature distinctive qui peut autoriser une transaction. Elle est composée du signataire réel, qui est ensuite combiné avec des non-signataires pour former un anneau. Monero utilise la technologie de la signature en anneau pour aider l'expéditeur à masquer l'origine d'une transaction.

Monero utilise la preuve de travail en tant que mécanisme de consensus. Contrairement à d'autres Blockchain qui ont une taille de bloc fixe (tel que Bitcoin), Monero utilise une taille de bloc dynamique qui change en fonction des 100 blocs précédents. Les périodes de fort volume de transactions entraînent une augmentation de la taille du bloc, à l'inverse, les périodes de faible volume de transactions entraînent une diminution de la taille du bloc.

1.4 Conclusion

Dans ce chapitre, nous avons présenté les notions de bases de nos travaux de recherches à savoir le vote électronique en ligne, les primitives cryptographiques que nous allons utiliser dans nos approches ainsi que la technologie Blockchain. Le chapitre sui-

vant est un état de l'art sur les différents protocoles de vote électronique en ligne basés sur la technologie Blockchain existants.

Chapitre 2

État de l'art

Sommaire

2.1 Introduction

Avec l'apparition de la technologie Blockchain, il est devenu possible d'atteindre un niveau de sécurité et de vérifiabilité accru, tout en préservant la confidentialité et la non malléabilité des transactions. En effet, cette technologie fonctionne sans aucune autorité centrale de confiance et garantit l'intégrité des données puisque chaque transaction est vérifiée et stockée par chacun des nœuds de la Blockchain. Toutes les transactions sont rassemblées en blocs par les mineurs, qui doivent fournir une preuve telle qu'une preuve de travail ou une preuve d'enjeu. Ainsi, la Blockchain est considérée comme une structure de données immuable et sécurisée.

Dans ce contexte, plusieurs protocoles de vote électronique basés sur la technologie Blockchain ont été proposés durant ces dernières années. Dans ce chapitre, nous étudions quelques protocoles de vote existants et dressons un tableau comparatif entre ces différents systèmes afin de bien étudier l'avancement des recherches dans ce domaine et cerner les problématiques actuelles. Nous divisons les systèmes de vote électronique existants en deux catégories : (1) les systèmes de vote académiques qui ont été proposés par des chercheurs et (2) les systèmes de vote commerciaux qui ont été proposés par des entreprises et qui ont été commercialisés.

2.2 Protocoles de vote académiques

Dans cette partie, nous donnons un aperçu sur quelques protocoles de vote académique qui ont été proposés, récemment, par des chercheurs. Nous dressons aussi des tableaux qui comparent ces différents systèmes de vote et qui récapitulent les principales caractéristiques de chaque protocole de vote. Nous utilisons les abréviation suivantes :

- "✓" : Propriété vérifiée ,
- "X" : Propriété non-vérifiée ,
- "-" : Aucune information disponible,
- "NA" : Non-Applicable,
- "AC" : Autorité de Confiance,
- "MC" : Mineurs de Confiance,
- "GPL Vx" : Licence Publique Générale GNU Version x,
- "MIT" : Licence de logiciels libres, provenant de l'Institut de Technologie du Massachusetts (MIT).

2.2.1 BroncoVote

Description du protocole BroncoVote [30] est un système de vote dédié pour les élections à l'échelle universitaire qui utilise la Blockchain Ethereum et les contrats intelligents. Il utilise le cryptosystème Paillier pour chiffrer et déchiffrer les votes d'une manière homomorphique. Les utilisateurs ont la possibilité de choisir le type de l'élection et d'établir la liste des électeurs éligibles pour chaque élection. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 5.1 et les détails de son implémentation sont décrits dans le Tableau 2.2. C'est un projet open source et son code est disponible en ligne [1]. Le code est composé de :

- **Trois contrats intelligents :** Écrits en langage *solidity*, ces contrats sont :
  - Le contrat d'enregistrement : Créé par l'administrateur et permet de conserver les informations concernant les électeurs, les organisateurs, les numéros d'identification des bulletins de vote et les domaines de courrier électronique sur liste blanche afin de vérifier l'éligibilité des électeurs, de modifier les autorisations et de récupérer le contrat "Voting.sol".
  - Le contrat de création des élections : Possédé par l'administrateur, il définit les détails du contrat de vote en remplissant les informations requises dans "VoteUI.html".
  - Le contrat de vote : Possédé par son créateur, il contient des informations concernant le bulletin de vote, à savoir : l'identifiant du bulletin, le titre, le type, la limite de vote et le délai. Il fait également office d'urne dans laquelle sont stockés les votes chiffrés.
- **Deux scripts :** Écrits en JavaScript, ils sont :
  - Crypto.js : Implémente le cryptosystème Paillier (les fonctions de chiffrement et de déchiffrement).
  - VotingApp.js : Contient la totalité du processus de vote, rassemble les informations de VoteUI.html et interagit avec Crypto.js et la Blockchain Ethereum.
- **Un fichier HTML :** VoteUI.html : C'est l'interface du votant qui lui permet d'interagir avec le système de vote en invoquant les fonctions de VotingApp.js.

Entités BroncoVote fait appel aux entités suivantes :

1. https://goo.gl/nqBpzM

— *Administrateur :* Il déploie les contrats intelligents d'enregistrement et de création des élections lors de la configuration initiale du système et a la possibilité d'accorder ou de révoquer l'autorisation de création de bulletins de vote pour les électeurs/-créateurs inscrits.
— *Votants :* Chaque votant peut s'inscrire à une élection en utilisant son numéro d'étudiant ou d'employé et son adresse électronique.
— *Créateurs :* Ce sont des électeurs ayant la permission de créer des bulletins de vote.

Phases BroncoVote comporte six phases :
— *Configuration initiale :* Au cours de cette phase, l'administrateur active le système de vote en établissant une liste blanche de l'ensemble de domaines de messagerie électronique admissibles et en déployant les contrats intelligents d'enregistrement et de création pour permettre aux électeurs de commencer le processus de vote.
— *Enregistrement des votants :* Tout électeur/créateur qui souhaite s'inscrire à l'élection, accède au système de vote et saisie son identifiant d'étudiant/d'employé, son adresse électronique et une demande facultative de création de bulletin de vote dans l'interface utilisateur fournie par "VoteUI.html" qui envoie ces informations à "VotingApp.js". Le fichier JS interagit avec le contrat d'enregistrement pour vérifier les informations de l'électeur/le créateur et pour l'enregistrer en cas de vérification réussie.
— *Création de bulletin de vote :* Durant cette phase, le créateur a la possibilité de créer et de configurer une élection. Il a également la possibilité d'établir la liste des adresses électroniques ayant le droit de participer à l'élection.
— *Chargement de bulletin de vote :* En utilisant le numéro d'identification d'une élection, un électeur peut voter sur le bulletin de vote si la période de vote n'est pas écoulée ou vérifier le résultat final dans le cas contraire.
— *Vote :* Après avoir chargé le bulletin de vote, l'électeur choisit un candidat et vote pour lui avec son adresse électronique enregistrée. Le contrat d'enregistrement vérifie l'éligibilité de l'électeur, le choix de vote et le nombre de tentatives de vote faites par l'électeur. Si la vérification est réussie, le vote est envoyé à "Crypto.js" pour être chiffré, puis envoyé au contrat de vote qui stocke tous les votes chiffrés.
— *Résultat :* Pour obtenir le résultat final de l'élection, les électeurs invoquent la méthode "getVotes" du "VotingApp.js", qui envoie une transaction à la Blockchain pour récupérer la somme de tous les votes chiffrés. Enfin, "getVotes" envoie le décompte des votes chiffrés à "Crypto.js" pour être déchiffrés à l'aide de la clé privée du cryptosystème Paillier.

2.2.2 Blockchain-based E-Voting System (BEVS)

Description du protocole Blockchain-based E-Voting System [31] est un système de vote électronique par district conçu pour être mis en œuvre sur des Blockchains avec autorisations. Il utilise *Go-Ethereum* comme infrastructure de la Blockchain avec la preuve d'autorité en tant que mécanisme de consensus. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 5.1 et les détails de son implémentation sont décrits dans le Tableau 2.2.

Entités Le protocole de vote électronique en ligne BEVS fait appel aux entités suivantes :
— *Administrateur de l'élection :* Son rôle est de créer et de gérer l'élection, d'enregistrer les électeurs, de définir la durée de l'élection et d'attribuer les nœuds autorisés.

— *Votants :* Pour voter, chaque électeur s'authentifie, charge les bulletins de vote et fait son choix. Les électeurs ont également la possibilité de vérifier leur vote après la fin de l'élection.
— *Nœuds de district :* Lorsque l'administrateur crée l'élection et déploie les contrats intelligents de vote électronique, chaque nœud de district interagit avec son contrat de vote correspondant et vérifie les données de vote envoyées par un électeur. Lorsque les données de vote sont vérifiées par tous les nœuds de district correspondants, chaque vote approuvé est ajouté à la Blockchain.
— *Nœuds de démarrage :* Chaque institution, avec un accès autorisé au réseau, héberge un nœud de démarrage ou *bootnode*. Un bootnode aide les nœuds de district à se découvrir et à se communiquer.

Phases BEVS se déroule en cinq étapes :
— *Configuration de la Blockchain :* Cette phase consiste à définir les deux types de nœuds de Blockchain, qui sont :
  — Nœuds de district : Ce sont les districts électoraux qui déploient les contrats intelligents de vote, vérifient les données de vote et les ajoutent à la Blockchain.
  — Nœuds de démarrage : Ils constituent le service de découverte et de coordination qui permet aux nœuds de district de communiquer et de découvrir les uns les autres.
— *Création de l'élection :* L'administrateur crée le contrat intelligent de l'élection. Il définit la liste des candidats pour chaque district, déploie le contrat intelligent sur la Blockchain et donne accès aux nœuds de district pour interagir avec leurs contrats intelligents correspondants.
— *Enregistrement des votants :* Après avoir créé une élection, l'administrateur définit la liste des électeurs qui peuvent participer à cette élection. Cette phase est réalisée en utilisant un service gouvernemental de vérification de l'identité pour enregistrer et authentifier en toute sécurité les électeurs éligibles. À la fin de cette phase, l'administrateur génère un porte-monnaie ou *wallet* unique pour chaque électeur inscrit.
— *Comptage :* Le décompte est effectué par des contrats intelligents. Chaque contrat intelligent fait son propre décompte pour son district correspondant dans son propre espace de stockage.
— *Vérification des votes :* Après la phase de vote, chaque électeur reçoit l'identifiant de la transaction contenant son vote. Cet identifiant est utilisé durant cette phase pour vérifier que son vote est comptabilisé correctement.

	BroncoVote	BEVS
Primitives Cryptographiques	Paillier	Preuve à divulgation nulle de connaissance non interactive
Éligibilité	AC	AC
Pas de résultat partiel	AC	AC
Robustesse	✓	✓
Intégrité	MC	MC
Vérifiabilité Individuelle	✓	✓
Vérifiabilité Universelle	✓	-
Anonymat	AC	✓
Sans-Reçu	X	✓
Résistance à la coercition	X	X
Voter et quitter	✓	✓
Politique de vote	Configurable	Un seul vote
Empêcher le double vote	✓	✓
Empêcher la reproduction de vote	-	-

TABLEAU 2.1 – Évaluation de la sécurité des protocoles BroncoVote et BEVS

2.2.3 A Smart Contract for Boardroom Voting with Maximum Voter Privacy (SCBVMVP)

Description du protocole Il s'agit de la première mise en œuvre d'un protocole de vote électronique en ligne distribué et à auto-décompte, basé sur la technologie Blockchain [32]. Il s'agit d'un système de vote à l'échelle d'une salle de réunion (un maximum de 40 électeurs), implémenté par un contrat intelligent dans la Blockchain Ethereum. Il implique un petit groupe d'électeurs dont leurs identités sont publiquement connues avant le début du vote. Il exige que tous les électeurs inscrits votent pour permettre le calcul du décompte des voix et n'implémente que des élections avec uniquement deux options "oui" ou "non". L'inconvénient majeur d'un protocole de vote par auto-comptage est le fait que les derniers électeurs peuvent calculer le décompte avant tout le monde et donc changer d'avis ou abandonner sans voter s'ils ne sont pas satisfaits du décompte. Pour pallier cet inconvénient, le protocole SCBVMVP propose deux solutions. La première consiste à ajouter *une phase d'engagement* au cours de laquelle tous les électeurs hachent leurs votes chiffrés et les stockent dans la Blockchain. Ainsi, les derniers électeurs peuvent toujours calculer le décompte avant le décompte officiel mais ne peuvent pas modifier leurs votes. La deuxième solution vise à mettre en place un incitant financier à la participation des électeurs : tous les électeurs déposent de l'argent dans le contrat d'enregistrement à l'élection et sont remboursés lorsqu'ils votent. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 2.3 et les détails de son implémentation sont décrits dans le Tableau 2.4.

Le code de ce protocole comprend :

— *Deux contrats intelligents :* Écrits en Solidity et possédés par l'administrateur de l'élection :
  — *Contrat de vote* : Implémente et contrôle le processus de vote,
  — *Contrat de cryptographie* : Implémente les preuves à divulgation nulles de connaissance.

— *Trois pages HTML5/JavaScript :* Fournissent les interfaces graphiques des utilisa-

	BroncoVote	**BEVS**
Type de Blockchain	Avec autorisation	Avec autorisation
Plateforme	Ethereum	Ethereum
Implémentation	✓	✓
Framework	Ropsten	Geth
Langage de programmation	Solidity JavaScript HTML	Go
Contrat intelligent	✓	✓
Nombre de Contrats intelligents	3	= nombre de districts de vote
Nombre des phases	6	5
Noms des phases	Configuration initiale Enregistrement des votants Création de bulletin de vote Chargement de bulletin de vote Vote Résultat	Configuration de la Blockchain Création de l'élection Enregistrement des votants Comptage Vérification des votes
Nombre des entités	3	4
Noms des entités	Administrateur Votants Créateurs	Administrateur Votants Nœuds de district Nœuds de démarrage
Scalabilité	Élection à l'échelle de l'université	Élection à l'échelle du district
Date	2018	2018
Dernière mise à jour	22 Jan 2019	-
Licence	MIT Gratuit	-

TABLEAU 2.2 – Détails de mise en place des protocoles BroncoVote et BEVS

teurs :

- *Admin.html :* Fournit l'interface de l'administrateur de l'élection qui lui permet d'interagir avec la Blockchain et de gérer l'élection,
- *Vote.html :* Fournit l'interface web des électeurs qui leur permet de participer à l'élection,
- *Livefeed.html :* Offre une interface web pour les observateurs afin de suivre le déroulement des élections.

Entités Les entités impliquées dans ce système de vote sont :

- *Administrateur de l'élection :* Son rôle consiste à établir la liste des électeurs éligibles et une liste de dates limites. Il est également chargé de notifier à Ethereum le passage d'une phase à l'autre.
- *Votants :* Chaque électeur éligible peut s'inscrire à l'élection et, une fois inscrit, il doit voter pendant la phase de vote.
- *Observateurs :* Ils supervisent le déroulement des élections en temps réel.

Phases Le protocole se déroule en cinq étapes :

— *Configuration :* Au cours de cette phase, l'administrateur de l'élection inclut une liste de comptes d'électeurs éligibles dans le contrat de vote, définit une liste de minuteurs, fixe le dépôt d'inscription "d", la question de vote, active la phase d'engagement et notifie à Ethereum le passage à la phase suivante.

— *Enregistrement des votants :* Après avoir examiné la question de vote et les autres paramètres, chaque électeur qui souhaite s'inscrire choisit au hasard sa clé secrète x_i, calcule g^{x_i} et une preuve à divulgation nulle de connaissance ZKP(x_i) et les envoie avec un dépôt d'éther "d" au réseau Ethereum. À la fin de cette étape, Ethereum calcule une liste de clés reconstruites : $Y_i = \prod_{j=1}^{i-1} g^{x_j} / \prod_{j=i+1}^{n} g^{x_j}$. Le calcul ci-dessus garantit que $\sum_{i=1}^{n} x_i y_i = 0$.

— *Engagement* (optionnel) : Afin de s'engager à voter, chaque votant publie un hash de son vote chiffré $H(g^{x_i y_i} g^{v_i})$.

— *Vote* : Tous les électeurs publient leurs votes chiffrés $g^{x_i y_i} g^{v_i}$ avec une preuve à divulgation nulle de connaissance pour prouver que v_i est soit zéro soit un. Le dépôt "d" est remboursé à l'électeur lorsque son vote est accepté par Ethereum.

— *Comptage :* Afin de calculer le résultat final de l'élection, Ethereum calcule le produit de tous les votes reçus : $\prod_{i=1}^{n} g^{x_i y_i} g^{v_i} = g^{(\sum_{i=1}^{n} v_i)}$ puisque $g^{(\sum_{i=1}^{n} x_i y_i)} = g^0 = 1$ et force le logarithme discret du résultat afin de calculer le nombre de vote "oui".

2.2.4 Platform-independent Secure Blockchain-based Voting System (PSBVS)

Description du protocole Ce protocole [33] est implémenté en utilisant la Blockchain Hyperledger Fabric. Il utilise le cryptosystème Paillier pour chiffrer les votes avant qu'ils soient envoyés, des preuves à divulgation nulle de connaissance pour assurer l'exactitude et la cohérence des votes, et "Short Linkable Ring Signature (SLRS)" pour garantir la confidentialité des électeurs et prouver que le bulletin de vote provient d'un électeur éligible. Il met en œuvre le processus de vote comme un contrat intelligent et supporte des élections à grande échelle. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 2.3 et les détails de son implémentation sont décrits dans le Tableau 2.4.

Entités Ce système de vote électronique en ligne fait appel aux entités suivantes :

— **Administrateur du contrat intelligent :** Il met en place et déploie le contrat intelligent de vote.

— **Votants :** Ils s'inscrivent, téléchargent leurs clés publiques à partir de la Blockchain, déposent leurs bulletins de vote et ont la possibilité de vérifier l'exactitude du décompte final des élections.

— **Administrateur de vote :** Il charge les paramètres de vote, déclenche la phase de vote et de comptage et publie le résultat final ainsi que la preuve dans Blockchain.

Phases PSBVS comporte les phases suivantes :

— **Initiation du contrat intelligent :** Il est initié par la génération d'une paire de clés RSA pour signer et vérifier chaque transaction entre le contrat intelligent et les utilisateurs finaux.

— **Configuration du système de vote :** Au cours de cette phase, l'administrateur du vote télécharge, dans la Blockchain, la clé publique du cryptosystème Paillier et les

paramètres du schéma SLRS.

— **Enregistrement des votants :** Chaque électeur éligible doit s'inscrire à l'élection en fournissant les informations requises, à savoir une adresse électronique avec un mot de passe, un numéro d'identification avec un mot de passe ou une URL avec un mot de passe. Ensuite, le contrat intelligent vérifie l'éligibilité de l'électeur. Si la vérification est réussie, le votant télécharge les paramètres de schéma SLRS et la clé publique Paillier et génère sa paire de clés SLRS. Enfin, le votant charge sa clé publique du schéma SLRS dans le contrat intelligent qui sera publié sur la Blockchain de l'élection.

— **Vote :** Chaque électeur éligible encode et chiffre son choix avec la clé publique du cryptosystème Paillier, calcule une preuve à divulgation nulle de connaissance pour prouver l'exactitude de son vote, signe son choix chiffré avec sa clé secrète du schéma SLRS et l'envoie au contrat de l'élection pour être vérifié et conservé.

— **Comptage :** Le contrat intelligent calcule la somme de tous les votes chiffrés, noté "Esum", et l'envoie à l'administrateur pour la déchiffrer à l'aide de la clé secrète du cryptosystème Paillier. L'administrateur déchiffre "Esum" et obtient le résultat final de l'élection grâce à la propriété homomorphisme du cryptosystème Paillier. Il calcule également le nombre aléatoire "r" qui construit "Esum" et envoie (sum, r) au contrat intelligent. Ce dernier vérifie cette égalité : E$sum = g^{sum} \cdot r^n$ et accepte le décompte final si la vérification est réussie.

— **Vérification :** Les électeurs ont la possibilité de vérifier la validité du décompte officiel en téléchargeant et en faisant la somme de tous les bulletins chiffrés. Ils vérifient la correspondance entre la somme obtenue et le résultat publié sur la Blockchain. Chaque électeur éligible peut également vérifier que son bulletin de vote est enregistré dans la Blockchain de l'élection en vérifiant l'existence d'un bulletin portant sa signature.

	SCBVMVP	**PSBVS**
Primitives Cryptographiques	Cryptographie à base de courbes elliptiques Preuve à divulgation nulle de connaissance de Schnorr	Paillier SLRS Preuve à divulgation nulle de connaissance
Éligibilité	AC	AC
Pas de résultat partiel	X	AC
Robustesse	X	✓
Intégrité	MC	✓
Vérifiabilité Individuelle	✓	✓
Vérifiabilité Universelle	✓	✓
Anonymat	✓	AC
Sans-Reçu	X	X
Résistance à la coercition	X	X
Voter et quitter	X	✓
Politique de vote	Un seul vote	Un seul vote
Empêcher le double vote	✓	✓
Empêcher la reproduction de vote	✓	✓

TABLEAU 2.3 – Évaluation de la sécurité des protocoles SCBVMVP et PSBVS

	SCBVMVP	PSBVS
Type de Blockchain	Avec autorisation	Avec autorisation
Plateforme	Ethereum	Hyperledger Fabric
Implémentation	✓	✓
Framework	Réseau de test officiel de Ethereum	Hyperledger Fabric
Langage de programmation	Solidity JavaScript HTML5	-
Contrat intelligent	✓	✓
Nombre de Contrats intelligents	2	1
Nombre des phases	5	6
Noms des phases	Configuration Enregistrement des votants Engagement (optionnel) Vote Comptage	Initiation du contrat intelligent Configuration du système de vote Enregistrement des votants Vote Comptage Vérification
Nombre des entités	3	3
Noms des entités	Administrateur Votants Observateurs	Administrateur du contrat Votants Administrateur du vote
Scalabilité	Élection à l'échelle de la salle de réunion	Élection à large échelle
Date	2017	2018
Dernière mise à jour	23 Août 2017	-
Licence	MIT Gratuit	-

TABLEAU 2.4 – Détails de mise en place des protocoles SCBVMVP et PSBVS

2.3 Protocoles de vote commerciaux

Dans cette partie, nous étudions quelques protocoles de votes électroniques en ligne basés sur la technologie Blockchain parmi les plus connus dans le monde et qui ont été proposés par des entreprises. Nous dressons, ensuite, des tableaux qui évaluent la sécurité de chaque protocole et qui présentent les détails de sa mise en place. Nous utilisons les mêmes abréviations que les tableaux de la Section 2.2.

2.3.1 Agora

Description du protocole Agora[2] est une plateforme de vote vérifiable de bout en bout développée par une entreprise de technologie de vote basée en Suisse, en 2015. Cette plate-forme offre une vérifiabilité publique sur l'ensemble du processus de vote puisqu'elle est basée sur une Blockchain publique, considérée comme un tableau d'affichage

2. https://static1.squarespace.com/static/5b0be2f4e2ccd12e7e8a9be9/t/5b6c38550e2e725e9cad3f18/1533818/968655/Agora_Whitepaper.pdf

public. Ce tableau d'affichage stocke toutes les données publiques tout au long du processus électoral et est accessible par toute partie intéressée par la vérification de la validité de toute étape intermédiaire de l'élection. Afin de garantir la confidentialité, les bulletins de vote sont chiffrés avant d'être envoyés, en utilisant le cryptosystème d'ElGamal [3, 34], et anonymisés en faisant appel à Neff Shuffling [35]. Agora utilise son propre jeton sur la Blockchain pour les élections. Ainsi, les gouvernements et les institutions achètent ces jetons pour chaque électeur. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 2.5 et les détails de son implémentation sont décrits dans le Tableau 2.6. La plateforme de vote Agora est composée de cinq couches technologiques dont quatre sont inventées et développées par l'équipe d'Agora. Elles communiquent pendant le processus de vote et fournissent des preuves vérifiables. Ces couches sont :

- *La Blockchain du tableau d'affichage :* Il s'agit d'une Blockchain avec autorisation basée sur l'architecture Skipchain [36]. La structure des données de la Skipchain a été proposée par l'équipe d'Agora et est basée sur un mécanisme de consensus de tolérance aux fautes byzantines. Ce tableau d'affichage fournit un enregistrement immuable et permanent de toutes les données tout au long du processus électoral, sur lequel seules les parties autorisées peuvent écrire des messages signés et à partir duquel toutes les parties peuvent lire les données. Cette couche est sécurisée et gérée par une autorité, appelée "Cothority", qui valide et confirme les transactions.
- *Cotena :* Basée sur Catena [37], cette couche interagit directement avec la Blokchain du tableau d'affichage et la relie à la Blockchain Bitcoin. Il s'agit d'un mécanisme d'enregistrement inviolable implémenté au-dessus de Bicoin.
- *La Blockchain Bitcoin :* Cette couche permet à quiconque de vérifier que les journaux Cotena et le tableau d'affichage restent intacts. En effet, Agora utilise la Blockchain Bitcoin pour stocker périodiquement le hash des blocs récents de la Blockchain du tableau d'affichage (appelée Skipblock).
- *Le réseau Valeda :* Il est composé de nœuds d'auditeurs citoyens qui valident les résultats des élections et calculent les différentes preuves cryptographiques du système. Cette couche prouve l'honnêteté du Cothority et la validité des résultats des élections.
- *Votapp :* Il s'agit de la couche d'application d'Agora. Elle permet aux électeurs, aux auditeurs et à toute autre partie d'interagir avec le tableau d'affichage par le biais de trois applications : "Voting Booth" qui permet aux électeurs autorisés de participer à l'élection, "Audit" qui fournit un ensemble d'outils pour une vérifiabilité de l'élection de bout-en-bout et "Node" qui donne accès à la totalité de l'historique du tableau d'affichage et des journaux Cotena.

Entités Agora fait appel aux entités suivantes :

- *Administrateurs de l'élection :* Ils créent l'élection, génèrent ses paramètres, désignent les auditeurs officiels et surveillent le déroulement du vote.
- *Votants :* Chaque votant éligible chiffre et dépose un bulletin de vote pendant la phase de dépôt. Ils ont le droit de vérifier que leurs votes sont exprimés comme prévu et de contrôler la validité de chaque étape du processus électoral en inspectant le tableau d'affichage.
- *Nœuds d'auditeurs citoyens :* Ils vérifient toutes les preuves produites à chaque étape du processus électoral, y compris le résultat final de l'élection. Ces nœuds peuvent être gérés par les administrateurs électoraux, les électeurs ou tout tiers intéressé.
- *Nœuds :* Ils comptent tous les votes valides et publient le résultat final de l'élection sur le tableau d'affichage pendant la phase de décompte.

— *Cothority :* Il s'agit d'un réseau distribué de serveurs témoins indépendants qui maintiennent la Blockchain d'Agora, en conservant une copie de toutes les transactions et en approuvant les nouvelles transactions en blocs dans le cadre du mécanisme de consensus du réseau. Pendant la phase de déchiffrement, les membres de Cothority coopèrent pour déchiffrer les votes et les publier avec la preuve de l'exactitude du déchiffrement.

Phases Cette plate-forme de vote électronique en ligne comprend les phases suivantes :

— *Configuration :* Au cours de cette phase, les administrateurs électoraux établissent un fichier de configuration pour l'élection qui contient les paramètres suivants : la liste des agents électoraux, le type d'élection (vote majoritaire ou vote unique transférable), la liste des électeurs admissibles, la liste des candidats et les heures de début et de fin de l'élection.

— *Vote :* Au cours de cette phase, chaque électeur choisit son candidat préféré, chiffre son vote à l'aide du cryptosystème Threshold ElGamal et soumet un bulletin de vote. Pour assurer la vérifiabilité individuelle, l'électeur a la possibilité d'effectuer une validation "Cast-as-Intended".

— *Anonymisation :* À la fin de la période de vote, tous les bulletins chiffrés passent par un réseau de mixage pour couper le lien entre un votant et son vote. Cette étape est réalisée en utilisant un réseau de mélangeurs Neff-Shuffling.

— *Déchiffrement :* Durant cette phase, les nœuds de Cothority vérifient la validité des preuves à divulgation nulle de connaissance générées par le mixnet pour anonymiser les bulletins de vote puis ils coopèrent et les déchiffrent. Ils publient les votes déchiffrés sur le tableau d'affichage en même temps que les preuves de déchiffrement. Ces preuves sont vérifiées par les administrateurs électoraux à la fin de cette étape.

— *Comptage :* Les nœuds d'Agora, l'administrateur des élections ou tout autre tiers peuvent effectuer le décompte à partir des votes valides déchiffrés et publiés sur le tableau d'affichage. Le résultat final de l'élection est ensuite publié sur le tableau d'affichage et peut être vérifié par des auditeurs.

— *Audit :* Grâce au nœud d'audit citoyen, tout le monde peut vérifier la validité de toutes les preuves cryptographiques générées durant chaque étape du processus électoral, ce qui garantit une vérification de bout en bout.

2.3.2 TIVI

Description du protocole La plate-forme de vote en ligne TIVI [3] est le résultat d'un partenariat stratégique entre deux entreprises : Smartmatic et Cybernetica. Il s'agit d'une solution de vote en ligne basée sur l'authentification biométrique. Elle permet de vérifier l'identité de l'électeur via un selfie. Il suffit de télécharger une photo du visage du votant dans le système avant le vote, puis la technologie de reconnaissance faciale compare la biométrie de son visage à l'image téléchargée pendant la phase d'enregistrement. Elle propose également de différentes techniques d'authentification. Après avoir été authentifié avec succès, chaque électeur éligible choisit son candidat préféré. Son vote est automatiquement chiffré et signé sur l'appareil du votant. Le vote est ensuite envoyé sur un canal chiffré à un serveur de vote sécurisé et le votant reçoit son reçu, qui peut être utilisé ultérieurement pour vérifier le vote via une application pour smartphone et un tableau d'affichage public, qui est la Blockchain de l'élection. À la fin de la phase de vote, les votes

3. http://www.smartmatic.com/voting/online-voting-tivi/

chiffrés sont passés à travers un réseau de mélangeurs afin de garantir la confidentialité des votes. Enfin, *m* des *n* membres du conseil d'administration des élections coopèrent et reconstruisent la clé de déchiffrement pour déchiffrer les votes. Ensuite, les votes déchiffrés sont envoyés au serveur de comptage pour être comptés. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 2.5 et les détails de son implémentation sont décrits dans le Tableau 2.6.

Entités TIVI fait appel aux entités suivantes :

— *Administrateur de l'élection :* Il met en place et gère l'élection, crée le bulletin de vote et authentifie les électeurs.
— *Autorité d'enregistrement :* Vérifie l'éligibilité de chaque électeur et lui fournit ses paramètres d'authentification.
— *Votants :* Pour participer au processus de vote, chaque électeur s'inscrit, se connecte au système à l'aide de ses paramètres d'authentification, choisit un candidat, vote pour lui et enfin vérifie que son vote a été exprimé comme prévu via une application mobile et le tableau d'affichage public.
— *Autorité de mixage :* Pour garantir l'anonymat des votes, l'autorité de mixage mixe tous les votes chiffrés et publie sur la Blockchain de l'élection la preuve mathématique de mixage.
— *Comité électoral :* Les membres du comité coopèrent pour construire la clé de déchiffrement, déchiffrent les votes et publient une preuve mathématique du déchiffrement.
— *Auditeurs :* Ils observent le déroulement des élections en temps réel et vérifient les preuves mathématiques de mixage et de déchiffrement des votes affichées sur le tableau d'affichage public.

Phases Le protocole de vote se déroule en sept étapes :

— *Authentification :* Au cours de cette phase, l'électeur se connecte au système de vote et s'authentifie au moyen de différentes techniques d'authentification, notamment les technologies biométriques.
— *Vote :* Après avoir été authentifié avec succès, chaque électeur a accès à un bulletin de vote contenant les noms des candidats, il choisit son candidat préféré, confirme son choix et émet son vote. Les votes sont chiffrés et signés sur le dispositif de vote.
— *Validation :* Au cours de cette phase, une autorité de confiance vérifie la validité et l'éligibilité des votes.
— *Stockage :* Les votes éligibles sont stockés sur la Blockchain de vote. À ce moment, les électeurs peuvent vérifier le contenu du vote reçu sur le serveur de vote au moyen d'une application pour smartphone.
— *Mixage :* À la fin de l'élection, une autorité de confiance procède au mélange des votes chiffrés pour garantir la confidentialité des votes.
— *Déchiffrement :* Les votes sont déchiffrés uniquement en présence du comité électoral qui doit se réunir pour créer la clé de déchiffrement.
— *Comptage :* Tous les votes déchiffrés sont envoyés à un serveur pour être comptés et le résultat final de l'élection sera publié et vérifié via le tableau d'affichage public.

	Agora	TIVI
Primitives cryptographiques	ElGamal Neff-Shuffling	Chiffrement asymétrique Signature digitale
Éligibilité	AC	AC
Pas de résultat partiel	AC	(N-M+1)/N AC
Robustesse	✓	✓
Intégrité	AC	-
Vérifiabilité individuelle	✓	✓
Vérifiabilité universelle	✓	✓
Anonymat	AC	AC
Sans-reçu	✓	✓
Résistance à la coercition	X	X
Voter et quitter	✓	✓
Politique de vote	Configurable	Votes multiples
Empêcher le double vote	-	✓
Empêcher la reproduction de vote	✓	✓

TABLEAU 2.5 – Évaluation de la sécurité des protocoles Agora et TIVI

2.3.3 Follow My Vote

Description du vote Follow My Vote[4] est un système de vote commercial qui utilise la technologie Blockchain comme urne. Pour voter, chaque électeur doit télécharger et installer le système de vote "Follow My Vote" sur son appareil personnel, se connecter avec ses informations d'identité qui seront vérifiées par une autorité et procéder au vote si la vérification est réussie. Une autorité de confiance est nécessaire pour garantir la confidentialité des électeurs et dissimuler la correspondance entre leurs identités réelles et leurs votes. Chaque électeur a la possibilité d'imprimer un reçu comme preuve de son vote. Grâce à ce reçu, il a la possibilité de modifier son vote avant la fin de la phase de vote et seul le dernier vote soumis par chaque électeur sera considéré comme un vote officiel. En outre, les électeurs peuvent localiser leurs votes dans l'urne pour s'assurer qu'il a été émis comme prévu et comptabilisé correctement. Les électeurs peuvent également suivre le déroulement de l'élection en temps réel au fur et à mesure que les votes sont envoyés et sont autorisés à vérifier chaque bulletin dans l'urne pour confirmer l'exactitude du décompte final. L'évaluation de la sécurité de ce protocole est donnée par le Tableau 2.7 et les détails de son implémentation sont décrits dans le Tableau 2.8.

Entités Ce système de vote électronique en ligne fait appel aux entités suivantes :

— *Votants :* Chaque électeur éligible a le droit de s'enregistrer à l'élection, de s'authentifier, de demander un bulletin de vote au bureau d'enregistrement, d'exprimer un ou plusieurs votes jusqu'à la fin de la phase de vote, de vérifier que son dernier vote est émis comme prévu et comptabilisé convenablement et de vérifier l'exactitude du résultat final de l'élection.

— *Vérifieur d'identité :* Son rôle est de vérifier la validité des paramètres d'authentification des électeurs, de permettre aux seuls électeurs éligibles de voter et de leurs

4. https://followmyvote.com/

	Agora	**TIVI**
Type de Blockchain	Hybride	Avec autorisation
Plateforme	Skipchain Bitcoin	-
Implémentation	✓	✓
Framework	-	-
Langage de programmation	Python JavaScript Scala Shell Ruby	-
Contrat intelligent	X	-
Nombre de contrats intelligents	NA	-
Nombre des phases	6	7
Noms des phases	Configuration Vote Anonymisation Déchiffrement Comptage Audit	Authentification Vote Validation Stockage Mixage Déchiffrement Comptage
Nombre des entités	5	6
Nom des entités	Administrateurs Votants Auditeurs citoyens Nœuds Cothority	Administrateur Autorité d'enregistrement Votants Autorité de mixage Comité électoral Auditeurs
Scalabilité	Large échelle	Large échelle
Date	2015	2016
Dernière mise à jour	Continue	-
Licence	Affero GPL V3	Payant

TABLEAU 2.6 – Détails de mise en place des protocoles Agora et TIVI

fournir les paramètres nécessaires en cas de changement de leurs votes pendant la phase de vote.

— *Autorité d'enregistrement :* Son rôle est d'enregistrer les électeurs et de leur fournir leurs bulletins de vote.

	Follow My Vote
Primitives cryptographiques	Cryptographie à base de courbes elliptiques
Éligibilité	AC
Pas de résultat partiel	X
Robustesse	✓
Intégrité	MC & AC
Vérifiabilité individuelle	✓
Vérifiabilité universelle	✓
Anonymat	AC
Sans-reçu	✓
Résistance à la coercition	X
Voter et quitter	✓
Politique de vote	Votes multiples
Empêcher le double vote	✓
Empêcher la reproduction de vote	✓

TABLEAU 2.7 – Évaluation de la sécurité du protocole Follow My Vote

	Follow My Vote
Type de Blockchain	Avec autorisation
Plateforme	BitShares
Implémentation	✓
Framework	BitShares
Langage de programmation	JavaScript
Contrat intelligent	-
Nombre de contrats intelligents	-
Nombre des phases	-
Noms des phases	-
Nombre des entités	3
Nom des entités	Votants Vérifieur d'identité Autorité d'enregistrement
Scalabilité	Large échelle
Date	2012
Dernière mise à jour	23 Mars 2019
Licence	MIT Payant

TABLEAU 2.8 – Détails de mise en place du protocole Follow My Vote

2.4 Conclusion

Dans ce chapitre, nous avons étudié quelques protocoles de vote électronique en ligne basés sur la technologie Blockchain, qui ont été proposés durant ces dernières années. Nous avons évalué les niveaux de sécurité de ces protocoles contre la liste des exigences de sécurité définie dans le premier chapitre de ce rapport. Nous avons aussi présenté leurs détails de mise en place. Le prochain chapitre décrit un nouveau protocole de vote électronique en ligne, que nous proposons. Ce protocole est vérifiable de bout-en-bout et est basé sur la Blockchain Ethereum.

Chapitre 3

"Verify-Your-Vote" : Un protocole de vote électronique en ligne vérifiable et basé sur la Blockchain Ethereum

Sommaire

3.1 Introduction

La technologie Blockchain offre la possibilité de concevoir de nouveaux types d'applications et de systèmes qui permettent à leurs utilisateurs de stocker des données de manière sécurisée et transparente. L'une des propriétés de sécurité les plus importantes à garantir dans un protocole de vote électronique en ligne est la vérifiabilité de bout en bout. En effet, les différentes parties politiques, les observateurs et en particulier les électeurs veulent pouvoir vérifier que tous les votes éligibles sont collectés et comptés sans avoir

été altérés ou supprimés, tout en préservant le secret des votes et l'anonymat des électeurs. Au cours des dernières décennies, un nombre important de protocoles de vote électronique tentent de résoudre ce problème en utilisant des techniques cryptographiques et/ou un bulletin de bord public. Récemment, certains systèmes de vote électronique basés sur la technologie Blockchain ont été proposés, mais qui n'ont pas été utilisés dans le monde réel, car ils ne supportent pas un grand nombre de candidats et d'électeurs. Dans ce chapitre, nous concevons et implémentons un protocole de vote électronique en ligne basé sur une Blockchain et vérifiable de bout en bout, appelé "Verify-Your-Vote" (VYV). Notre protocole garantit plusieurs propriétés de sécurité grâce à la technologie Blockchain et à l'utilisation d'une variété de primitives cryptographiques à savoir le cryptosystème de Paillier [15], la cryptographie à base de courbes elliptique [8], de couplage [11] et d'identité [12, 13]. Nous évaluons également ses performance en termes de temps, de coût et de nombre d'électeurs et de candidats pouvant être soutenus. Finalement, nous évaluons, informellement puis formellement, la sécurité de notre approche. Cette contribution faisait l'objet de deux publications internationales [38, 39].

3.2 Description du protocole proposé : Verify-Your-Vote

Dans cette partie, nous présentons, d'une manière détaillée, le protocole Verify-Your-Vote. Il s'agit d'un protocole de vote électronique en ligne qui utilise la Blockchain Ethereum comme *un tableau d'affichage public*[1] pour afficher les paramètres publics de l'élection et offrir une vue persistante à tous les votants. Dans VYV, nous tirons parti de la technologie Blockchain et explorons la possibilité d'utiliser cette technologie comme un tableau d'affichage public.

Nous commençons par présenter la structure d'un bulletin de vote et la signification de chacun de ses paramètres, nous présentons ensuite les différentes entités impliquées dans le protocole ainsi que leurs rôles au cours du processus de vote et enfin nous donnons les différentes phases du protocole avec leurs détails d'implémentation.

3.2.1 Entités du protocole

Notre protocole fait appel à cinq entités qui sont les suivantes :

— *Autorité d'enregistrement (AE) :* Elle vérifie l'éligibilité des votant en leurs demandant une preuve d'identité. Seuls les votants qui ont droit de participer à l'élection auront accès au serveur d'enregistrement.

— *Serveur d'enregistrement (SE) :* Son rôle est d'enregistrer les électeurs éligibles et de leurs fournir leurs paramètres d'authentification.

— *Administrateur (Admin) :* Son rôle est de gérer l'élection. Cela comprend la définition d'une liste de minuteries pour garantir que l'élection se déroule en temps opportun, la définition des paramètres de l'élection, l'authentification des électeurs et la construction des bulletins de vote en coopération avec l'autorité de comptage.

— *n Votants éligibles (V_i) :* Chaque votant possède un compte sur la Blockchain, lui permettant de participer à l'élection. Il a le droit de voter plusieurs fois mais seulement son dernier vote valide sera comptabilisé dans le résultat final. Il a également la possibilité de vérifier que son vote a été comptabilisé correctement et que le résultat final de l'élection correspond bien à la somme de tous les votes éligibles.

1. C'est un tableau public où tout le monde peut lire et écrire des informations, mais personne ne peut modifier et/ou supprimer les données écrites.

Numéro du bulletin BN			
Pseudo ID "C_j"	**Nom du candidat** "nom_j"	**Choix**	**Contre-valeurs** "$CV_{BN,nom_j,k}$"
0	Paul	☐	$CV_{BN,nom_0,0}$
1	Nico	☐	$CV_{BN,nom_1,1}$
2	Joel	☐	$CV_{BN,nom_2,2}$

TABLEAU 3.1 – Structure du bulletin de vote.

— *m Autorités de comptage* (AC_k) *:* Elles participent à la construction des bulletins de vote, déchiffrent les votes, calculent le résultat final de l'élection et publient les différentes valeurs qui permettent aux électeurs de vérifier l'exactitude de ce résultat.

3.2.2 Structure du bulletin de vote

L'idée principale de la structure des bulletins de vote est inspirée de la proposition de David Chaum et al. en 2005 [40]. Cette structure est décrite dans le tableau 4.1.

Chaque bulletin vierge contient quatre informations :

— *Le numéro de bulletin "*BN*" :* C'est un numéro unique. Il est calculé en utilisant la formule suivante : $BN = E_{PK_A}(g, D)$, avec g est un générateur d'un groupe cyclique additif $\mathbb{G}$, D est un nombre aléatoire et $E_{PK_A}(x)$ désigne le chiffrement d'un message x par la clé publique de l'administrateur PK_A, en utilisant le cryptosystème de Paillier.

— *Les pseudo identités des candidats "*C_j*" :* C'est la position de chaque candidat dans le bulletin de vote. Il est calculé à partir d'un ordre initial et d'une valeur de décalage. Cette dernière est calculée, pour chaque bulletin de vote, en utilisant la formule suivante : $Decalage = H(g) \mod m$; avec H est une fonction de hachage, g est le même générateur utilisé pour calculer le numéro du bulletin et m est le nombre de candidats participants à l'élection.

— *Les noms des candidats "*nom_j*".*

— *Les contre-valeurs "*$CV_{BN,nom_j,k}$*" :* Chaque bulletin de vote inclue un ensemble de m contre-valeurs qui sont utilisées par les votants afin de vérifier que leurs votes ont été comptés correctement. Elles sont calculées pour chaque candidat et dépendent du numéro de bulletin de vote et du nom du candidat. Elles sont obtenues en utilisant la formule suivante : $CV_{BN,nom_j,k} = e(Q_{nom_j}, S_k \cdot Q_{BN})$; avec S_k est la clé secrète de l'autorité de comptage AC_k, $Q_{nom_j} = H_1(nom_j)$ et $Q_{BN} = H_1(BN)$ sont deux points d'une courbe elliptique E et H_1 est une fonction de hachage.

3.2.3 Phases du protocole et leur implémentation

Le protocole VYV se déroule en six étapes. Dans cette partie, nous décrivons chaque phase et donnons les détails de son implémentation. Nous commençons par présenter les environnements logiciel et matériel de l'implémentation du protocole.

— *Environnement matériel :* Nous déployons le protocole sur un ordinateur personnel avec un processeur Intel (R) Core (TM) i5-3210M avec 4 Go de RAM et Ubuntu 16.04 en tant que système d'exploitation.

— *Environnement logiciel :* Pour l'implémentation de notre système, nous choisissons un langage de programmation open source, interprété et orienté objet qui est *Py-*

thon[2] . Nous utilisons également un ensemble de bibliothèques cryptographiques, écrites en Python, pour développer les primitives cryptographiques de notre protocole. Ces bibliothèques sont :

- *Charm Crypto [41]* : un framework de prototypage rapide pour les cryptosystèmes avancés, livré avec une bibliothèque de cryptosystèmes implémentés. Nous importons de ce framework le paquet de la cryptographie à base d'identité, qui va être utilisé pour chiffrer et déchiffrer les votes,
- *Cryptography*[3] : met en œuvre une variété de primitives cryptographiques dont la cryptographie à base de courbes elliptiques fait partie. Nous choisissons la courbe elliptique SECP256K1, définie dans les normes pour une cryptographie efficace (Standards for Efficient Cryptography) [4],
- *Tate_bilinear_pairing0.6*[5] *:* permet de calculer les fonctions de couplage bilinéaires. Nous utilisons ce paquet pour calculer les contre-valeurs ($CV_{BN,nom_j,k} = e(Q_{nom_j}, S_k \cdot Q_{BN})$),
- *PHE*[6] : une bibliothèque Python pour le chiffrement partiellement homomorphe. Nous l'utilisons pour générer les paires de clés des votants et de l'administrateur ainsi que pour chiffrer et déchiffrer les paramètres d'authentification des votants éligibles,
- *Hashlib :* implémente de nombreux algorithmes de hachage de messages sécurisés. À partir de ce paquet, nous utilisons l'algorithme de hachage SHA256 pour calculer la valeur de décalage de chaque bulletin de vote ($Décalage = H(g) \mod m$).

Nous présentons maintenant les différentes phases de notre approche ainsi que l'ensemble des fonctions Python développées pour mettre en œuvre chaque phase.

- *Phase de configuration :* Cette phase est illustrée par la Figure 3.1. L'administrateur commence par générer les paramètres de l'élection et les publie sur la Blockchain. Ces paramètres sont :
  - $\mathbb{G}_1$ un groupe cyclique additif d'ordre un nombre premier q,
  - $\mathbb{G}_2$ un groupe multiplicatif du même ordre q,
  - Une fonction de hachage $H : \{0,1\}^* \rightarrow G_1$,
  - La clé publique de l'administrateur PK_A, générée avec le cryptosystème de Paillier.

  Ensuite, l'administrateur calcule le numéro de chaque bulletin et sa valeur de décalage correspondante (BN = $\{g, D\}_{PK_A}$, Décalage= H(g) mod m) et les envoie aux autorités de comptage ACs via la Blockchain pour obtenir, pour une autorité de comptage donnée AC_k, la contre-valeur correspondante ($CV_{BN,nom_j,k} = e(Q_{nomj}, S_k \cdot Q_{BN})$).

  Pour mettre en œuvre cette phase, nous développons la fonction décrite par l'algorithme 1.
  - ConstruireBulletin (Algo1) : Cette fonction prend en paramètre le nombre de votants l, le nombre de candidats m, ainsi que la liste de leurs noms $ListeNoms[]$, la clé secrète de l'autorité de comptage S_k, le groupe $\mathbb{G}1$ et son ordre n et retourne une liste de l bulletins de vote.

2. https://www.python.org/
3. https://pypi.org/project/cryptography/
4. http://www.secg.org/sec2v2.pdf
5. https://pypi.org/project/tate_bilinear_pairing/
6. https://pypi.org/project/phe/

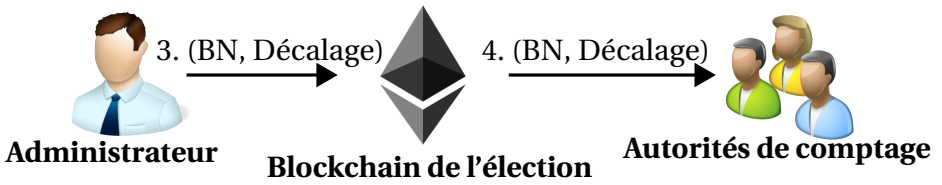


FIGURE 3.1 – Phase de configuration.

Algorithme 1 *ConstruireBulletin*

Paramètres : l, m, $ListeNoms[\,]$, S_k, $\mathbb{G}_{\mathbb{1}}$, n
Résultat : $ListeBulletins[\,]$

```
1 : Pour i ← 1 à l :
2 :     D ← nombre aléatoire ;
3 :     g ← générateur de (G_1, n) ;
4 :     BN ← ChiffrementPaillier(g||D, PK_A) ;
5 :     Décalage ← H(g) mod m ;
6 :     Pour j ← 1 à m :
7 :         CV_{BN,nom_j,k} ← pairing(H(ListeNoms[j]), S_k · H(BN)) ;
8 :         ListeCV ← ajout(CV_{BN,nom_j,k})
9 :     Fin Pour
10 :    Bulletin = [BN, Décalage, ListeNoms[ ], ListeCV[ ] ]
11 :    ListeBulletins[ ] ← ajout(Bulletin)
12 : Fin Pour
13 : retourner(ListeBulletins[ ]) ;
```

— *Phase d'enregistrement :* Afin de s'enregistrer et obtenir ses paramètres d'authentification, chaque personne ayant le droit de voter et souhaite participer à l'élection, se déplace vers un bureau de vote. Pour vérifier son éligibilité, chaque électeur doit fournir sa carte d'identité à l'autorité d'enregistrement (AE), qui vérifie si l'électeur est éligible pour participer à l'élection. Ensuite, seuls les électeurs éligibles auront accès à un serveur d'enregistrement (SE) pour obtenir leurs paramètres d'authentification. Les paramètres d'authentification de chaque électeur ont la forme suivante : $(S_{PW_i} = S_{SE\_A}.H_1(PW_i), P_{PW_i} = H_1(PW_i))$, où PW_i est un mot de passe saisi par le votant V_i et S_{SE_A} est une valeur secrète partagée entre le SE et l'Admin. Cette phase est illustrée par la figure 3.2.

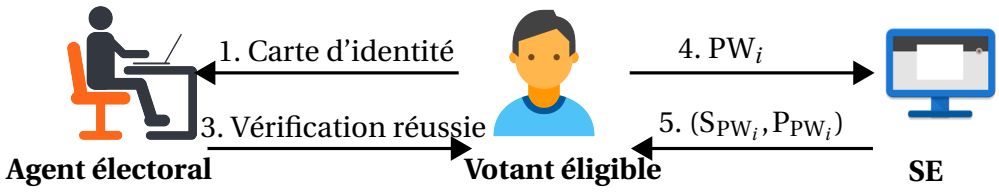


FIGURE 3.2 – Phase d'enregistrement.

— *Phase d'authentification :* La Figure 3.3 illustre les interactions entre les votants et l'administrateur durant cette phase.

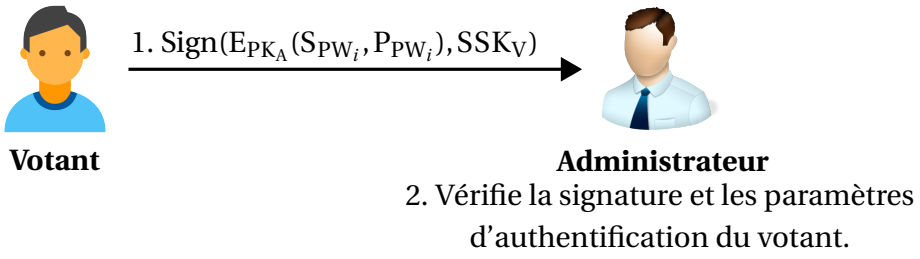


FIGURE 3.3 – Phase d'authentification.

Tous les votants inscrits dans l'élection signent et chiffrent leurs paramètres d'authentification en utilisant la clé publique de l'administrateur PK_A et les envoient à ce dernier qui, à son tour, déchiffre et vérifie la validité de chacun de ces paramètres. Si le votant est éligible pour voter, il a le droit de créer son propre compte sur la Blockchain de l'élection, qui va lui permettre de participer à l'élection.
Pour mettre en œuvre cette phase, nous développons deux fonctions décrites par les algorithmes 2 et 3.

— ChiffrerParamAuth (Algo2) : Cette fonction permet de chiffrer le login et le mot de passe d'un votant. Elle prend en entrée la clé publique de l'administrateur PK_A, le login $Login$ et le mot de passe PWD du votant et retourne leurs chiffrés ($LoginChiffré$, $PWDChiffré$).

Algorithme 2 $ChiffrerParamAuth$
Paramètres : PK_A, $Login$, PWD
Résultat : $LoginChiffré$, $PWDChiffré$
1 : $LoginChiffré \leftarrow ChiffrementPaillier(Login, PK_A)$;
2 : $PWDChiffré \leftarrow ChiffrementPaillier(PWD, PK_A)$;
3 : retourner($LoginChiffré$, $PWDChiffré$) ;

— VerifierParamAuth (Algo3) : Cette fonction met en œuvre le rôle de l'administrateur pendant la phase d'authentification. Elle prend en entrée un login chiffré $LoginChiffré$, un mot de passe chiffré $PWDChiffré$, la clé privée de l'administrateur SK_A et la valeur secrète partagée entre l'administrateur et le serveur d'enregistrement S_{SE_A}. Elle retourne $Vrai$ si les paramètres d'authentification sont valides et $Faux$ sinon.

Algorithme 3 $VerifierParamAuth$
Paramètres : $LoginChiffré$, $PWDChiffré$, SK_A, S_{SE_A}
Résultat : Valeur booléenne
1 : $LoginDéchiffré \leftarrow DéchiffrementPaillier(LoginChiffré, SK_A)$;
2 : $PWDDéchiffré \leftarrow DéchiffrementPaillier(PWDChiffré, SK_A)$;
3 : Si ($LoginDéchiffré = S_{SE\_A} \cdot PWDDéchiffré$)
4 : retourner($Vrai$) ;
5 : Sinon
6 : retourner($Faux$) ;
7 : Fin Si

— *Phase de vote :* Comme le montre la Figure 3.4, deux entités participent à cette phase :
  — Autorités de comptage ACs : Elles choisissent au hasard un bulletin de vote pour chaque votant, le chiffre avec la clé publique du votant et l'envoient au votant correspondant via la Blockchain de l'élection.
  — Votants éligibles : Lorsqu'il reçoit son bulletin de vote, chaque votant procède au déchiffrement de son bulletin, choisit son candidat préféré, chiffre son vote, le transmet aux ACs via la Blockchain et sauvegarde la contre-valeur correspondante $CV_{BN,name_j,k}$ qui va lui permettre de vérifier son vote ultérieurement. Pour construire son vote, chaque votant choisit un candidat avec un pseudo ID C_j et le chiffre en utilisant le numéro de son bulletin de vote BN. Ainsi, chaque vote chiffré a la forme suivante : $VoteChiffré = E_{Q_{C_j}}(BN)$ avec $Q_{C_j} = H_1(C_j)$.

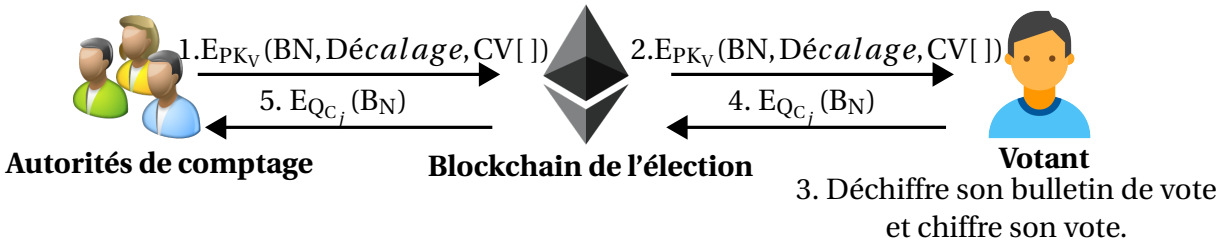


FIGURE 3.4 – Phase de vote.

Pour implémenter cette phase, nous développons deux fonctions qui sont décrites par les algorithmes 4 et 5.
  — ChiffrerBulletin (Algo4) : Cette fonction met en œuvre le rôle des ACs pendant la phase de vote. Elle prend en paramètre un bulletin de vote et une clé publique d'un votant éligible et renvoie le bulletin de vote chiffré.

Algorithme 4 $ChiffrerBulletin$

Paramètres : $Bulletin$ = [BN, Décalage, $ListeNoms$[ ], $ListeCV$[ ]], PK_V
Résultat : $BulletinChiffré$ = [BN$Chiffré$, Décalage$Chiffré$, $ListeNoms$[ ], $ListeCV$[ ]]

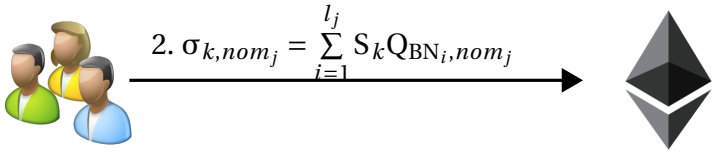
1 : $BulletinChiffré$[ ] ← $Bulletin$[ ] ;
2 : $BulletinChiffré$[0] ← $ChiffrementPaillier$($Bulletin$[0], PK_V) ;
3 : $BulletinChiffré$[1] ← $ChiffrementPaillier$($Bulletin$[1], PK_V) ;
4 : retourner($BulletinChiffré$[ ]) ;

  — Voter (Algo5) : Cette fonction permet aux électeurs de voter pour un candidat donné. Elle prend en entrée un bulletin de vote chiffré et la clé maîtresse publique du cryptosystème à base d'identité MPK, déchiffre le bulletin de vote en utilisant la clé secrète du votant SK_V et retourne un vote chiffré et la contre-valeur correspondante. Pour chiffrer un vote, nous utilisons le système de chiffrement à base d'identité (Voir le troisième point de la sous-section 1.2.1), importé du paquet Charm. Un bulletin de vote est représenté par la liste suivante : $Bulletin$ = [BN, Décalage, $[nom_1, nom_2, ..., nom_m]$, $[CV_{BN,nom_1,1}, CV_{BN,nom_2,2}, ..., CV_{BN,nom_m,m}]$].

Algorithme 5 $Voter$

Paramètres : $BulletinChiffré[\,]$, MPK, SK_V
Résultat : $VoteChiffré$, $CV_{BN,nom_j,k}$
1 : $Bulletin[\,] \leftarrow BulletinChiffré[\,]$;
2 : $Bulletin[0] \leftarrow DéchiffrementPaillier(BulletinChiffré[0], SK_V)$;
3 : $Bulletin[1] \leftarrow DéchiffrementPaillier(BulletinChiffré[1], SK_V)$;
4 : $C_j \leftarrow$ saisie de l'indice du candidat choisi;
5 : $BN \leftarrow Bulletin[0]$;
6 : $VoteChiffré \leftarrow Chiffrement(MPK, C_j, BN)$;
7 : $CV_{BN,nom_j,k} \leftarrow Bulletin[3][C_j]$;
8 : retourner$(VoteChiffré, CV_{BN,nom_j,k})$;

— *Phase de comptage :* Durant cette phase, les autorités de comptage déchiffrent tous les votes éligibles, en utilisant leurs clés secrètes, pour obtenir les numéros des bulletins BN. Chaque autorité de comptage est dédiée au calcul du nombre de votes pour un pseudo ID (C_j) spécifique : par exemple la première autorité de comptage "AC_1" déchiffre, avec sa clé secrète $S_1 \cdot Q_{C1}$, tous les bulletins qui ont été chiffrés avec la clé publique Q_{C1}. Chaque AC est responsable de la génération de sa propre clé secrète pour déchiffrer les votes en exécutant la fonction $Extract$ de la cryptographie à base d'identité. Cela signifie que chaque AC joue le rôle d'un PKG pour elle même. Cette étape peut être effectuée avant le début de l'élection. A partir de chaque numéro de bulletin de vote BN et de sa valeur de décalage correspondante, les ACs reconstruisent les bulletins, identifient les candidats choisis et incrémentent les compteurs. Une fois que tous les votes ont été déchiffrés et comptés, les ACs publient, sur la Blockchain de l'élection, le résultat final de l'élection ainsi que le décompte de chaque candidat, en utilisant la formule suivante : $\sigma_{k,nom_j} = \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i,nom_j}$ où l_j est le nombre de votes reçus par le candidat de nom nom_j, S_k est la clé secrète de l'autorité de comptage AC_k, $Q_{BN_i,nom_j} = H_1(BN_{i,nom_j})$ et BN_{i,nom_j} est le numéro de bulletin du vote i qui correspond au candidat de nom nom_j.
Cette phase est décrite par la Figure 3.5 et mise en œuvre par deux fonctions, données par les algorithmes 6 et 7.



Autorités de comptage
1. Déchiffrent les votes pour révéler les numéros des bulletins, reconstruisent les bulletins de vote, identifient les candidats choisis et incrémentent les compteurs.

Blockchain de l'élection

FIGURE 3.5 – Phase de comptage.

— Compter (Algo6) : Cette fonction prend en compte un vote chiffré $VoteChiffré$, la clé secrète de l'AC_k S_K, la liste des numéros des bulletins $ListeBN[\,]$, la liste de leurs valeurs de décalage correspondantes $ListeDécalage[\,]$ et la liste de noms des candidats $ListeNoms[\,]$. Elle renvoie le résultat de l'élection après

avoir incrémenté les compteurs.

Algorithme 6 $Compter$

Paramètres : $VoteChiffré$, S_K, $ListeBN[\,]$, $ListeDécalage[\,]$, $ListeNoms[\,]$
Résultat : $Résultat$

```
1 : VoteDéchiffré ← Déchiffrer(S_K, VoteChiffré);
2 : Pour i ← 1 à longueur(ListeBN[ ])
3 :     Si (VoteDéchiffré = ListeBN[i])
4 :         Décalage ← ListeDécalage[i];
5 :     Fin Si
6 : Fin Pour
7 : Résultat ← IncrementerCompteur(Décalage, ListeNoms[ ]);
8 : retourner(Résultat);
```

— CalculSigma (Algo7) : Implémente la formule $\sigma_{k,name_j} = \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i,name_j}$ (lignes 2, 3 et 4 de Algo7) pour calculer le décompte de chaque candidat. Elle prend en paramètre le nom d'un candidat nom_j, la liste des numéros de bulletins de vote qui contiennent des votes pour ce candidat $ListeBN[\,]$ et la clé secrète de l'autorité de comptage S_k.

Algorithme 7 $CalculSigma$

Paramètres : nom_j, $ListeBN[\,]$, S_k
Résultat : σ_{k,nom_j}

```
1 : σ_{k,nom_j} ← 0;
2 : Pour i ← 1 à longueur(ListeBN[ ])
3 :     σ_{k,nom_j} ← σ_{k,nom_j} + S_k · H_1(ListeBN[i]);
4 : Fin Pour
5 : retourner(σ_{k,nom_j});
```

— *Phase de vérification :* Cette phase permet aux votants de vérifier que leurs votes ont été pris en considération correctement (vérification individuelle) et que le résultat final de l'élection correspond bien à la somme de tous les votes éligibles (vérification universelle). Elle comprend deux sous-phases :
  — Au cours de la première sous-phase, les ACs recalculent les contre-valeurs à partir de chaque numéro de bulletin et du nom du candidat choisi et les publient sur la Blockchain. Ainsi, chaque votant, souhaitant vérifier que son dernier vote valide a été inclus dans le résultat final, peut accéder à la Blockchain de l'élection et vérifier l'existence de sa contre-valeur, qu'il a sauvegardée pendant la phase de vote, dans la liste des contre-valeurs reconstruites. Cette sous-phase est illustrée par la Figure 3.6 et mise en œuvre par les fonctions décrites par les algorithmes 8 et 9 :
    — ReconstruireCV (Algo8) : Cette fonction prend en paramètres un numéro de bulletin de vote BN_i, le nom du candidat choisi nom_j et la clé secrète de l'autorité de comptage S_k. Elle retourne la contre-valeur correspondante $CV_{BN_i,nom_j,k}$.

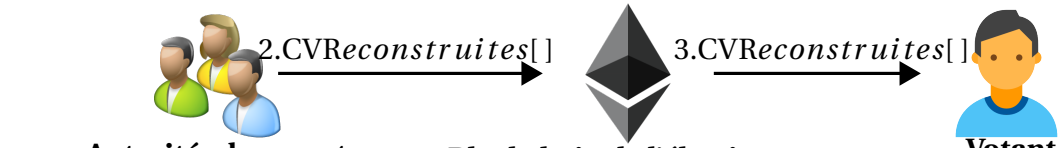
Algorithme 8 $Reconstruire$CV

Paramètres : BN_i, nom_j, S_k
Résultat : $\mathrm{CV}_{\mathrm{BN}_i,nom_j,k}$
1 : $\mathrm{Q}_{\mathrm{BN}} \leftarrow \mathrm{H}_1(\mathrm{BN})$;
2 : $\mathrm{Q}_{nom_j} \leftarrow \mathrm{H}_1(nom_j)$;
3 : $\mathrm{CV}_{\mathrm{BN}_i,nom_j,k} \leftarrow \mathrm{pairing}(\mathrm{Q}_{nom_j}, \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}})$;
4 : retourner($\mathrm{CV}_{\mathrm{BN}_i,nom_j,k}$) ;

— VérifierCV (Algo9) : Cette fonction prend en paramètre la liste des contre-valeurs reconstruites CVR$econstruites$[ ] ainsi que la contre-valeur CV du votant et vérifie son existence dans la liste. Elle retourne Vrai si CVR$ec-onstruites$[ ] contient la valeur de CV et Faux sinon.

Algorithme 9 $Vérifier$CV

Paramètres : CVR$econstruites$[ ], CV
Résultat : Valeur booléenne
1 : Pour i ← 1 à longueur(CVR$econstruites$[ ])
2 : Si (CVR$econstruites$[i]= CV)
3 : retourner(Vrai) ;
4 : Fin Si
5 : Fin Pour
6 : retourner(Faux) ;



1. Recalculent les contre-valeurs
4. Vérifie l'existence de sa CV

FIGURE 3.6 – Première sous-phase de la vérification.

— La deuxième sous-phase permet aux votants et à toutes autres parties de vérifier l'exactitude du résultat final à partir de la liste des contre-valeurs et le décompte de chaque candidat comme suit :

$$\begin{aligned}\prod_{i=1}^{l} \mathrm{CV}_{\mathrm{BN}_i} &= \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} \mathrm{CV}_{\mathrm{BN}_{i,nom_j},k} = \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} e(\mathrm{Q}_{nom_j}, \mathrm{S}_k.\mathrm{Q}_{\mathrm{BN}_i,nom_j}) \\ &= \prod_{k=1}^{m}\prod_{j=1}^{m} e(\mathrm{Q}_{nom_j}, \sum_{i=1}^{l_j} \mathrm{S}_k.\mathrm{Q}_{\mathrm{BN}_i,nom_j}) \\ &= \prod_{k=1}^{m}\prod_{j=1}^{m} e(\mathrm{Q}_{nom_j}, \sigma_{k,nom_j}) \qquad (3.1)\end{aligned}$$

Où $l = \sum_{j=1}^{m} l_j$ est le nombre total de votes. Ces égalités utilisent la propriété d'homomorphisme des pairings :

$$\prod_{i=1}^{l_j} e(\mathrm{Q}_{nom_j}, \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i,nom_j}) = e(\mathrm{Q}_{nom_j}, \sum_{i=1}^{l_j} \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i,nom_j})$$

Nous développons la fonction décrite par l'algorithme 10 pour mettre en œuvre cette sous-phase :

— VerifierRésultat (Algo10) : Elle prend en paramètres la liste des contre-valeurs reconstruites CVR$econstruites[\,]$, la liste de décompte des candidats $\sigma_{k,nom_j}[\,][\,]$ ainsi que la liste de leurs noms L$iste$N$oms[\,]$, et retourne Vrai si l'équation 4.1 est vérifiée et Faux sinon.

Algorithme 10 V$erifier$Rés$ultat$

Paramètres : CVR$econstruites[\,]$, $\sigma_{k,nom_j}[\,][\,]$, L$iste$N$oms[\,]$
Résultat : Valeur booléenne
//Calcul du produit des contre-valeurs reconstruites
1 : Prod_CV= 1 ;
2 : Pour i ← 1 à longueur(CVR$econstruites[\,]$)
3 : Prod_CV ← Prod_CV · CVR$econstruites[i]$;
4 : Fin Pour
//Calcul du $\prod_{k=1}^{m}\prod_{j=1}^{m} e(Q_{nom_j},\sigma_{k,nom_j})$
5 : Prod_P$airing$ ← 1
6 : Pour k ← 1 à m
7 : Pour j ← 1 à m
8 : Prod_P$airing$ ← Prod_Pairing · pairing(H(L$iste$N$oms[j]$), $\sigma_{k,nom_j}[k][j]$) ;
9 : Fin Pour
10 : Fin Pour
11 : Si (Prod_CV = Prod_P$airing$)
12 : retourner(Vrai) ;
13 : Sinon
14 : retourner(Faux) ;
15 : Fin Si

3.3 Évaluation des performances du VYV

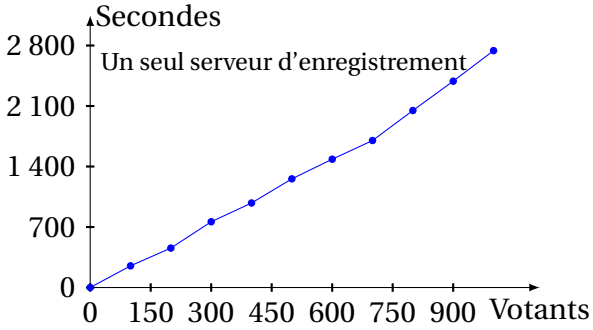
Après avoir implémenté notre protocole, nous procédons à l'évaluation de ses performances. Nous faisons varier *le nombre de votants, de candidats et de serveurs,* et nous mesurons, pour chaque valeur de ces paramètres, le temps d'exécution de chaque phase de notre système. Lorsque nous disposons de plus d'un serveur par autorité pour mettre en place l'élection, nous utilisons une stratégie d'équilibrage de charge qui nous permet d'améliorer les performances et d'exploiter toutes les ressources dont nous disposons. En utilisant les valeurs obtenues, nous générons des courbes qui montrent la variation des temps d'exécution en fonction de ces paramètres. Les courbes obtenues peuvent être divisées en deux catégories : (1) **les courbes strictement monotones** (soit croissantes, lorsqu'elles représentent les variations du temps en fonction du nombre de votant, soit décroissantes lorsqu'elles représentent les variations du temps en fonction du nombre de serveurs) et (2) **les courbes croissantes avec des parties constantes** (lorsqu'elles représentent la variation du temps en fonction du nombre de candidats, qui est égal au nombre des ACs). Ces parties constantes sont obtenues lorsque le nombre de serveurs n'est pas multiple du nombre de candidats et que le nombre minimum de serveurs par AC est le même pour des valeurs proches du nombre d'ACs. Pour chaque valeur expérimentale, nous répétons l'exécution *100* fois et prenons la moyenne des *100* valeurs obtenues. Nous évaluons les performances de notre système avec un nombre de votants allant de

0 à 1000, un nombre de candidats qui varie entre *1 et 31* et sur un seul ordinateur (dont les caractéristiques sont mentionnées dans la sous-section 3.2.3). Ensuite, nous prenons deux exemples d'élections réelles, qui sont la dernière élection présidentielle de la Tunisie (2019) et celle de la France (2017), et nous calculons le coût de ces élections si elles ont été exécutées sur notre système.

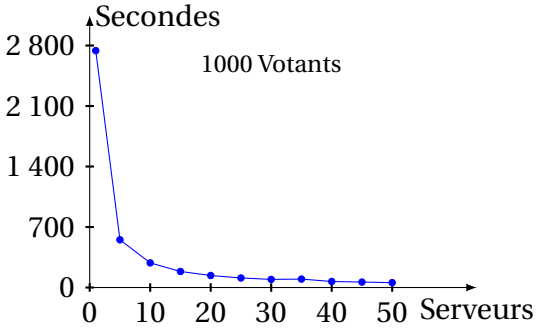
3.3.1 Évaluation du temps d'enregistrement

La phase d'enregistrement est hors ligne (et hors Blockchain). Chaque votant se déplace vers le bureau de vote le plus proche pour s'inscrire et obtenir ses paramètres d'authentification. La durée totale de cette phase dépend uniquement du nombre de votant et du nombre de serveurs dont nous disposons pour faire fonctionner notre système. Lorsque notre protocole s'exécute sur un seul serveur et avec des valeurs différentes du nombre de votants, nous obtenons la courbe représentée dans la Figure 3.7.a. Comme nous remarquons, un votant met environ 2,61 secondes pour obtenir un login et un mot de passe à partir du serveur d'enregistrement et cette période de temps augmente linéairement avec le nombre de votants.

Pour évaluer les performances de notre protocole lorsque nous faisons varier le nombre de serveurs, nous fixons le nombre d'électeurs à *1000* et mesurons le temps que prend notre protocole pour enregistrer ces électeurs. La Figure 3.7.b montre la variation de ce temps en fonction du nombre de serveurs.



(a) Temps d'enreg. en fct. du nbr. de votants.

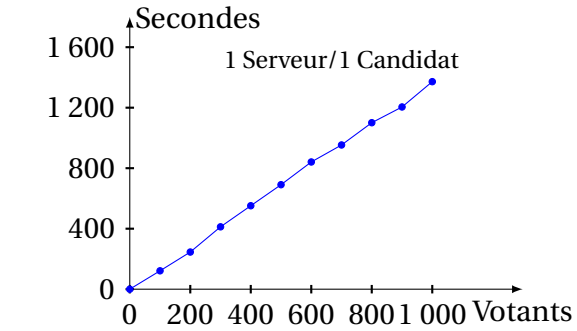


(b) Temps d'enreg. en fct du nbr. de serveurs.

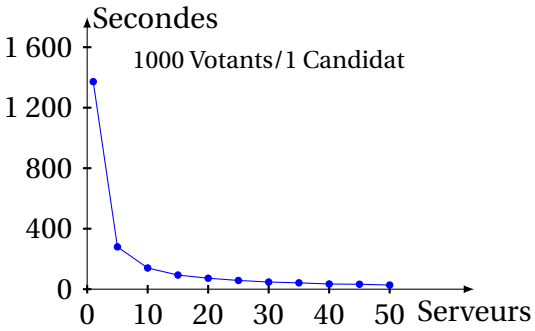
FIGURE 3.7 – Évaluation du temps d'enregistrement.

3.3.2 Évaluation du temps de configuration

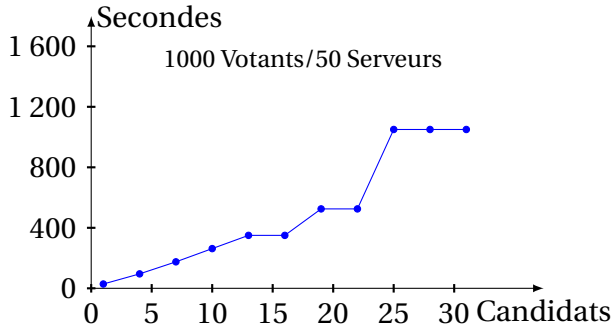
Pendant la phase de configuration, l'administrateur génère tous les numéros de bulletins de vote et leurs valeurs de décalage correspondantes. Ensuite, les ACs calculent les contre-valeurs. La durée totale de cette phase dépend du nombre de bulletins à générer, qui est égal au nombre d'électeurs inscrits (Figure 3.8.a), du nombre de serveurs dont nous disposons pour exécuter cette phase (Figure 3.8.b) et du nombre de candidats de l'élection (Figure 3.8.c).



(a) Temps de config. en fct du nbr. de votants.



(b) Temps de config. en fct du nbr. de serveurs.

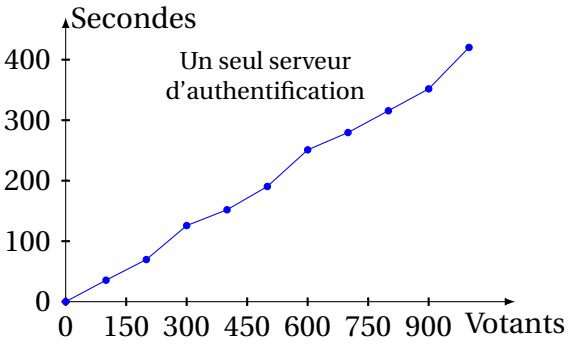


(c) Temps de config. en fct du nbr. de candidats.

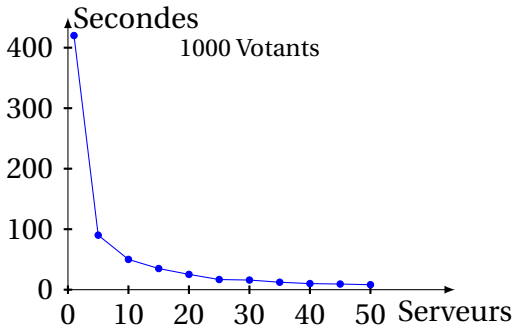
FIGURE 3.8 – Évaluation du temps de configuration.

3.3.3 Évaluation du temps d'authentification

La majorité des systèmes de vote électronique existants qui comportent une phase d'authentification, vérifient les paramètres d'authentification de chaque votant en cherchant leur existence dans une liste qui contient les paramètres d'authentification de tous les votants inscrits. Cette vérification a une complexité quadratique. Dans notre cas, la phase d'authentification a une complexité linéaire puisque nous vérifions la validité des paramètres d'authentification de chaque votant en n'exécutant qu'une seule opération de multiplication. Nous évaluons le temps que prend cette phase en fonction du nombre de votants (Figure 3.9.a) et du nombre de serveurs (Figure 3.9.b). Le nombre de candidats n'a aucune influence sur le temps d'authentification.



(a) Temps d'auth. en fct. du nbr. de votants.

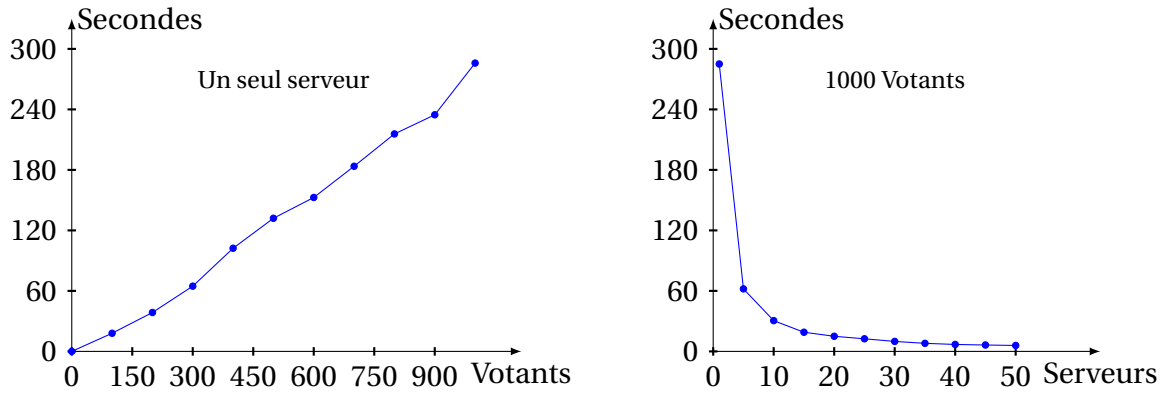


(b) Temps d'auth. en fct. du nbr. de serveurs.

FIGURE 3.9 – Évaluation du temps d'authentification.

3.3.4 Évaluation du temps de vote

Pendant la phase de vote, les ACs chiffrent et envoient un bulletin de vote à chaque votant, qui le déchiffre, choisit son candidat préféré, chiffre son vote et l'envoie aux ACs via la Blockchain de l'élection. La durée totale de la phase du vote dépend du nombre total de bulletins à chiffrer (qui est égal au nombre de votants), et du nombre de serveurs. Cette variation est illustrée par la Figure 3.10.



(a) Temps de vote en fct. du nbr. de votants. (b) Temps de vote en fct. du nbr. de serveurs.

FIGURE 3.10 – Évaluation du temps de vote.

3.3.5 Évaluation du temps de comptage

La phase de dépouillement comprend le déchiffrement des votes, l'identification des candidats choisis et l'incrémentation des compteurs correspondants. La durée totale de cette phase dépend du nombre de votes, du nombre de serveurs, de la manière dont les votants ont voté et du nombre d'ACs (qui est égal au nombre de candidats) (Figure 3.11). En effet, chaque AC est dédiée au calcul du nombre de votes pour un pseudo-ID spécifique. Dans le meilleur cas, les ACs comptent un nombre égal de votes. Dans le pire cas, une seule AC compte tous les votes (tous les votes sont pour un pseudo-ID unique).

Lorsque chaque AC dispose de plus d'un serveur pour compter les votes, elle utilise une stratégie d'équilibrage de charge pour répartir son travail et exploiter tous les serveurs dont elle dispose.

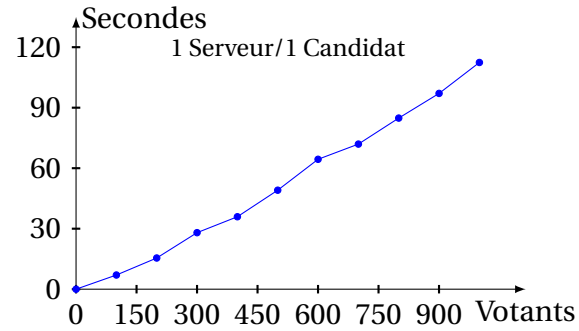
3.3.6 Évaluation du temps de vérification

La phase de vérification est facultative et comprend :

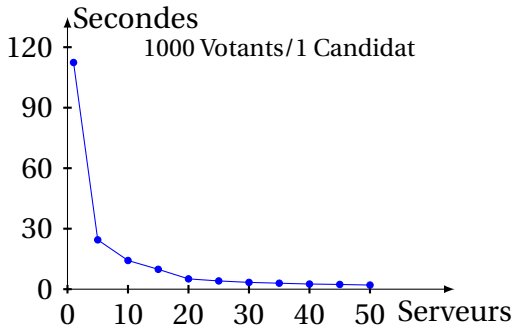
— La reconstruction des reçus des votants (par les ACs). La durée de cette étape dépend du nombre de votes (Figure 3.12.a), du nombre de serveurs (Figure 3.12.b), du nombre d'AC (ou de candidats) et de la manière dont les votants ont voté (Figure 3.12.c).

— La vérification de l'existence de la contre-valeur d'un votant dans la liste des contre-valeurs reconstruites. Cette étape est effectuée par les votants qui souhaitent vérifier que leurs votes ont été pris en compte correctement, sur leurs propre ordinateurs. Le temps nécessaire pour cette vérification est d'environ 0.07 seconde.

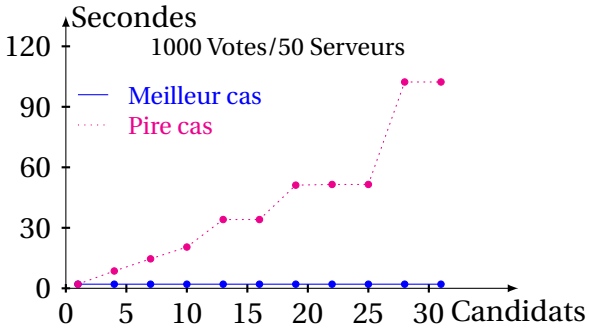
— La vérification de l'équation (4.1) pour vérifier l'exactitude du décompte final. Le temps nécessaire pour cette étape dépend du nombre de votes, du nombre de serveurs que disposent le vérifieur pour vérifier l'équation et du nombre de candidats. La Figure 3.13 illustre ces variations.



(a) Temps de compt. en fct. du nbr. de votants.

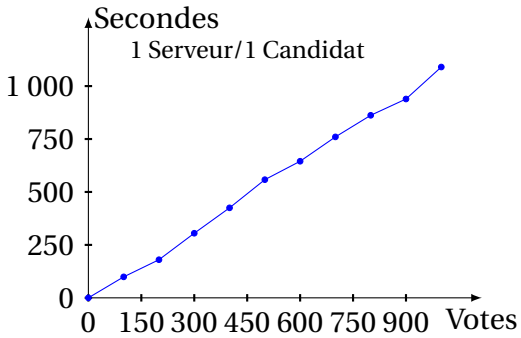


(b) Temps de compt. en fct. . du nbr. de serveurs

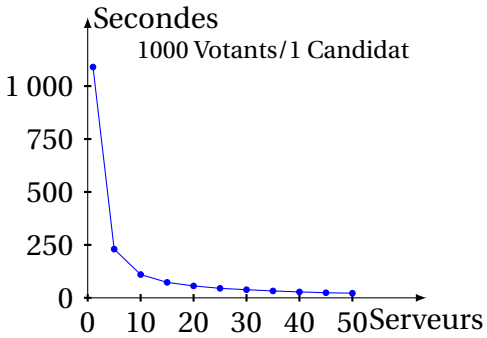


(c) Temps de compt. en fct. du nbr. de candidats.

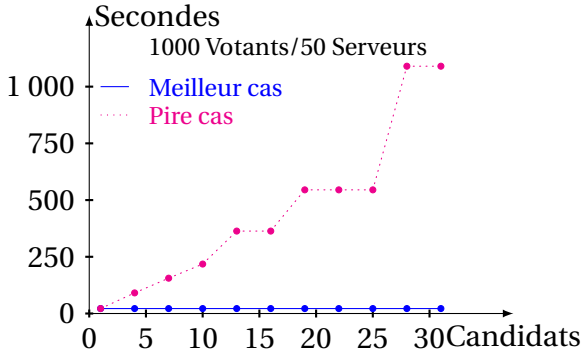
FIGURE 3.11 – Évaluation du temps de comptage.



(a) Temps de recons. des CV en fct. du nbr. de votants.

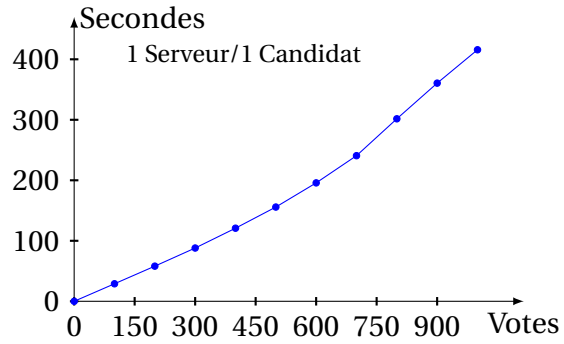


(b) Temps de recons. des CV en fct. du nbr. de serveurs.

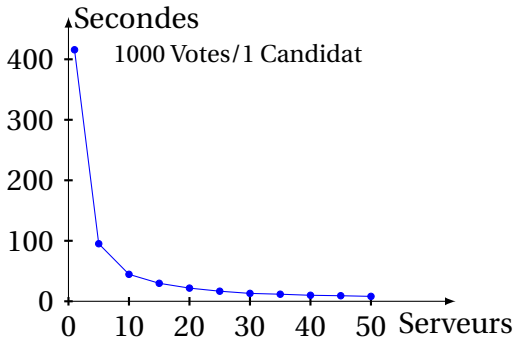


(c) Temps de recons. des CV en fct. du nbr. de candidats.

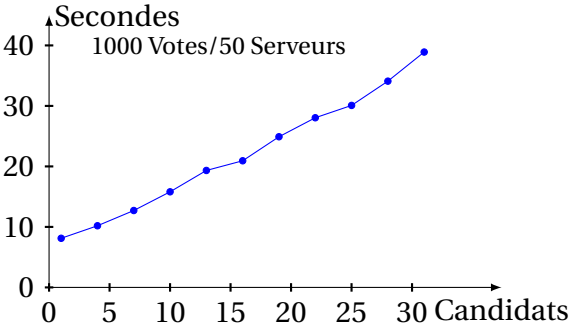
FIGURE 3.12 – Évaluation du temps de reconstruction des contre-valeurs.



(a) Temps de vérif. univ. en fct. du nbr. de votants.



(b) Temps de vérif. univ. en fct. du nbr. de serveurs.



(c) Temps de vérif. univ. en fct. du nbr. de candidats.

FIGURE 3.13 – Évaluation du temps de vérification universelle.

3.3.7 Exemple 1 : Élection présidentielle de la Tunisie de 2019

Dans cette partie, nous évaluons les performance de notre protocole dans un contexte réel. Nous prenons l'exemple de l'élection présidentielle tunisienne de 2019. Les statistiques de cette élection sont données dans les tableaux 3.2 et 3.3 [7] et nous donnons le temps que prend chaque phase de notre protocole pour exécuter l'élection tunisienne dans les tableaux 3.4 et 3.5. Le calcul est effectué sur la base des mesures expérimentales présentées dans les sous-sections précédentes. Pour exécuter notre système, nous supposons que nous disposons d'un nombre de serveurs égal au nombre de bureaux de vote de l'élection.

Nombre de votants enregistrés	Nombre de bureaux de vote	Nombre de votants 1er tour	Nombre de candidats 1er tour
7,074,566	4,567	3,465,184	26

TABLEAU 3.2 – Statistiques de l'élection présidentielle de la Tunisie de 2019 (1er tour).

Nombre de votants 2éme tour	Nombre de candidats 2éme tour
3,892,085	2

TABLEAU 3.3 – Statistiques de l'élection présidentielle de la Tunisie de 2019 (2éme tour).

7. http://www.isie.tn/

Nombre de serveurs	Temps de configuration	Temps d'enregistrement
4,567	11h 55min 38s	1h 7min 23s

TABLEAU 3.4 – Exécution de l'élection présidentielle de la Tunisie de 2019 avec le protocole VYV (Partie 1).

Temps d'authentification 1er / 2éme tour	Temps de vote 1er / 2éme tour	Temps de comptage 1er / 2éme tour	Temps de vérification 1er / 2éme tour
5min 3s / 5min 40s	3min 9s / 3min 32s	1min 17s / 1min 27s	32min 14s / 19min 52s

TABLEAU 3.5 – Exécution de l'élection présidentielle de la Tunisie de 2019 avec le protocole VYV (Partie 2).

3.3.8 Exemple 2 : Élection présidentielle de la France de 2017

Nous prenons un autre exemple d'une élection avec un nombre plus important de votant, qui est la dernière élection présidentielle de la France (2017). Les statistiques de cette élection sont représentées dans les tableaux 3.6 et 3.7 [8].

Nombre de votants enregistrés	Nombre de cantons	Nombre de votants 1er tour	Nombre de candidats 1er tour
47,582,183	2,054	37,003,728	11

TABLEAU 3.6 – Statistiques de l'élection présidentielle de la France de 2017 (1er tour).

Nombre de votants 2ème tour	Nombre de candidats 2ème tour
35,467,327	2

TABLEAU 3.7 – Statistiques de l'élection présidentielle de la France de 2017 (2ème tour).

Nous considérons que nous disposons d'un nombre de serveurs égal au nombre de cantons en France pour exécuter cette élection avec notre système de vote. Les résultats sont présentés dans les tableaux 3.8 et 3.9.

Nombre de serveurs	Temps de configuration 1er / 2ème tour	Temps d'enregistrement
2,054	3d 4h 15min 12s / 15h 32min 27s	16h 47min 42s

TABLEAU 3.8 – Exécution de l'élection présidentielle de la France de 2017 avec le protocole VYV (Partie1).

8. https://www.insee.fr/fr/statistiques/3540007

Temps d'authentification 1er / 2ème tour	Temps de vote 1er / 2ème tour	Temps de comptage 1er / 2ème tour	Temps de vérification 1er / 2ème tour
2h 6s / 1h 55min 6s	1h 14min 45s / 1h 11min 39s	30min 43s / 29min 26s	4h 18min 4s / 1h 57min 51s

TABLEAU 3.9 – Exécution de l'élection présidentielle de la France de 2017 avec le protocole VYV (Partie2).

3.4 Évaluation de la sécurité du VYV

Dans cette section, nous évaluons informellement puis formellement la sécurité du protocole de vote VYV.

3.4.1 Évaluation informelle

Dans cette partie, nous évaluons la sécurité de notre protocole contre la liste des exigences de sécurité que nous avons définie dans le Chapitre 1.

— *Éligibilité :* Notre protocole veille à ce que seuls les électeurs éligibles rejoignent la Blockchain de l'élection et participent au processus de vote grâce aux phases d'enregistrement et d'authentification. Lors de la phase d'enregistrement, les autorités vérifient l'identité de chaque électeur et seuls les électeurs éligibles reçoivent des paramètres d'authentification. La phase d'authentification garantit que seuls les électeurs enregistrés ont accès à la Blockchain de l'élection, créent leurs comptes et publient leurs clés publiques et leurs adresses. Un autre avantage de notre protocole est la complexité linéaire de la phase d'authentification. Dans la plupart des cas, les systèmes de vote électronique en ligne vérifient l'éligibilité de chaque électeur en comparant ses paramètres d'authentification avec une liste contenant les paramètres des votants inscrits. Ainsi, si une élection comprend n électeurs, nous avons n^2 opérations de comparaison. Alors que, dans notre cas, nous vérifions pour chaque électeur si $P_{PWi} = S_{SE\_A}.S_{PWi}$; Ainsi, si une élection compte n électeurs, nous n'effectuons que n opérations de vérification.

— *Pas de résultat partiel :* Les votes sont chiffrés avant d'être envoyés, de sorte que nous ne pouvons pas obtenir des résultats partiels avant le décompte officiel.

— *Robustesse :* Notre système de vote tolère le mauvais comportement des votants malveillants.

— *Intégrité :* Le fait d'exprimer et de conserver des votes dans la Blockchain de vote empêche qu'ils soient modifiés ou supprimés grâce à la propriété d'immuabilité des Blockchains. Puisque nous utilisons la Blockchain Ethereum, cette propriété est respectée par notre protocole si, au minimum, 50% + 1 du nombre total de mineurs dans notre infrastructure sont honnêtes.

— *Vérifiabilité individuelle :* Cette propriété est respectée par notre protocole grâce à la structure de nos bulletins de vote qui comprend les contre-valeurs CV_j. Ces valeurs servent de reçus pour les électeurs et permettent de vérifier que leurs votes ont été exprimés comme voulu sans révéler pour qui ils ont voté.

— *Vérifiabilité universelle :* Chaque autorité de comptage publie sur la Blockchain de l'élection le décompte de chaque candidat et les contre-valeurs. À partir de ces paramètres, chacun peut vérifier l'exactitude du résultat final en vérifiant l'équation(4.1). Cette équation vérifie l'égalité entre le produit des CV_j et la somme des décomptes publiés par les autorités de comptage. Si chaque électeur trouve son

reçu dans la liste de toutes les contre-valeurs et qu'aucune réclamation n'a été détectée et si l'égalité est vérifiée, alors nous sommes sûrs que le résultat final de l'élection est correct. Ainsi, les autorités de comptage ne peuvent pas modifier, supprimer ou ajouter des contre-valeurs puisque le nombre des votants participant à l'élection est publique. Si une ou plusieurs autorités de comptage tentent de modifier le décompte d'un ou plusieurs candidats, l'équation ne sera pas vérifiée et donc toute tentative de triche est détectée.

— *Anonymat :* Dans VYV, on ne peut pas établir un lien entre un électeur et son vote, car les votes sont envoyés après avoir été chiffrés et enregistrés à l'aide de la technologie Blockchain. En fait, chaque électeur est identifié dans la Blockchain électorale par une clé publique et une adresse qui n'ont aucun rapport avec son identité réelle. L'anonymat des votes est assurée même si toutes les autorités présentes dans notre approche collaborent. En effet, elles ne peuvent pas établir la relation entre un électeur et son vote : le serveur d'enregistrement peut établir la correspondance entre l'identité réelle de l'électeur et ses paramètres d'authentification, les autorités de comptage peuvent établir la correspondance entre un vote et l'adresse publique qui envoie ce vote mais personne, sauf l'électeur lui-même, ne peut faire le lien entre l'adresse publique d'un électeur et ses paramètres d'authentification puisque la création du compte dans la Blockchain est effectuée par l'électeur lui-même.
— *Sans-reçu :* Dans VYV, l'électeur ne peut pas trouver son vote à partir de la contre-valeur CV_j et des autres paramètres publics. Il ne peut pas donc prouver qu'il a voté pour un candidat donné.
— *Résistance à la coercition :* Notre protocole n'est pas résistant à la coercition. Un attaquant peut forcer un électeur à voter pour un candidat donné et vérifier sa soumission plus tard en utilisant la contre-valeur.
— *Voter et quitter :* VYV n'a pas besoin que l'électeur attende la fin de la phase de vote ni qu'il déclenche le décompte. Il peut tout simplement voter et quitter le système de vote.
— *Politique de vote :* Notre approche donne aux électeurs éligibles la possibilité de voter plus qu'une fois avant la fin de la phase de vote et seuls leurs derniers votes valides sont comptabilisés.

Le Tableau 3.10 résume les propriétés de sécurité de VYV.

VYV	
Éligibilité	✓
Pas de résultat partiel	✓
Robustesse	✓
Intégrité	50% + 1 MC
Vérifiabilité individuelle	✓
Vérifiabilité universelle	✓
Anonymat	✓
Sans-reçu	✓
Résistance à la coercition	X
Voter et quitter	✓
Politique de vote	Multiple

TABLEAU 3.10 – Évaluation de la sécurité de VYV.

3.4.2 Évaluation formelle

Dans cette partie, nous effectuons une analyse formelle de la sécurité de notre protocole en utilisant l'outil de vérification automatique ProVerif. Pour ce faire, nous avons besoin, tout d'abord, d'écrire le protocole de vote en utilisant le langage de modélisation formelle Applied Pi-Calculus.

Le code est donné dans l'Annexe A. On définit les requêtes suivantes pour prouver *le secret de vote, l'authentification des électeurs* et *la confidentialité des votes* et on donne les résultats de l'exécution des codes, ainsi que le temps que prend ProVerif pour prouver ces propriétés dans le Tableau 3.11.

— *Vérification du secret de vote :* Afin d'intercepter la valeur d'un vote donné, un attaquant doit intercepter les valeurs de deux paramètres : le numéro de bulletin de vote BN et le pseudo ID du candidat choisi Cj. Nous utilisons donc les requêtes suivantes :

```
query attacker(BN).
query attacker(Cj).
```

— *Vérification de l'authentification des électeurs :* Cette propriété est vérifié à l'aide *d'assertions de correspondance*. Le protocole vise à garantir que l'administrateur authentifie tous les électeurs en vérifiant la validité de leurs paramètres d'authentification. Par conséquent, nous définissons les évènements suivants :
  — **event acceptedAuthentication(bitstring, bitstring) :** utilisé par l'électeur pour enregistrer le fait qu'il a été authentifié avec succès,
  — **event VerifiesParameters(bitstring, bitstring) :** utilisé par l'administrateur pour enregistrer le fait qu'il a vérifié les paramètres d'authentification des électeurs.

  On définit aussi la requête suivante :

```
query a: bitstring, b: bitstring;
event acceptedAuthentication(a,b)==> event VerifiesParameters(a,b).
```

— *Vérification de la confidentialité des votes :* Pour exprimer la confidentialité des votes, nous prouvons la propriété d'équivalence observationnelle entre deux instances de notre processus qui ne diffèrent que par le choix des votes. Pour ce faire, nous utilisons `choice[V1,V2]` pour représenter les termes qui diffèrent entre les deux instances.

Propriétés	Résultat	Temps d'exécution
Secret du vote	Prouvée	0.019 s
Authentification des votants	Prouvée	0.017 s
Confidentialité du vote	Prouvée	0.038 s

TABLEAU 3.11 – Résultats et temps d'exécution des codes ProVerif de VYV.

3.5 Conclusion

Dans ce chapitre, nous avons proposé un protocole de vote électronique en ligne sécurisé et basé sur une variété de primitives cryptographiques, à savoir la cryptographie à base de courbes elliptiques, la cryptographie à base d'identité, les pairings et le cryptosystème de Paillier. Appelé "Verify-Your-Vote", ce protocole utilise la Blockchain Ethereum

comme un tableau d'affichage public. Nous avons implémenté ce protocole et évalué ses performances en termes de temps, de coût et de nombre de votants et de candidats pouvant être supportés. Notre évaluation prend en compte le coût lié au protocole de vote lui-même, qui est plus significatif que celui lié à l'envoi des données via la Blockchain. Notre protocole garantit l'éligibilité des votants, la vérifiabilité individuelle et universelle, le secret du vote, la confidentialité, le sans-reçu et la robustesse. Nous avons évalué, formellement, la sécurité de notre approche à l'aide de l'outil de vérification automatique ProVerif et le langage de modélisation formelle Applied Pi-Calculus. Cependant, notre protocole ne garantit pas la résistance à la coercition. Aussi, l'intégrité des données est garantie sous l'hypothèse que 50% +1 des mineurs sont honnêtes. Par conséquent, les futurs travaux seront consacrés à l'amélioration de la sécurité de ce protocole afin d'assurer la résistance à la coercition et de garantir l'intégrité des données sans faire confiance à aucune partie.

Chapitre 4

"DABSTERS" : Un protocole de vote électronique en ligne préservant l'anonymat et basé sur la Blockchain Hyperledger Fabric

Sommaire

4.1 Introduction

Avec la propriété d'immuabilité et l'architecture décentralisée, la technologie Blockchain est présentée comme une révolution dans plusieurs domaines et pour de nombreuses technologies existantes et émergentes. Pour le vote électronique, elle peut être utilisée pour garantir l'intégrité des votes et la vérifiabilité des résultats. Toutefois, les avantages des Blockchains publiques s'accompagnent par des inconvénients importants, tels qu'une importante complexité et une scalabilité limitée. Mais sa plus grande faiblesse est l'inefficacité du mécanisme utilisé pour éviter les comportements des mineurs malhonnêtes. En fait, pour chaque nouveau bloc, il est nécessaire de suivre un mécanisme de consensus pour l'ajouter à la chaîne, comme la preuve du travail et la preuve de l'enjeu. Dans ces mécanismes de consensus, les mineurs collectent les transactions, les valident et les organisent dans un bloc. Une fois qu'un mineur a passé avec succès le mécanisme de sélection imposé par l'algorithme de consensus pour proposer un nouveau bloc, le mineur sélectionné diffuse le bloc au réseau pour être validé par le reste des mineurs et

être ensuite ajouté à la chaîne. Les mineurs malhonnêtes peuvent modifier les données incluses dans les transactions avant de les stocker dans des blocs. Ce problème s'aggrave surtout si les données échangées sont sensibles ou importantes. C'est le cas des systèmes de vote électronique où les données échangées sont des votes. Les mineurs malhonnêtes peuvent invalider les élections en modifiant ces transactions. En outre, les Blockchains publiques ne sont pas adaptées à la gestion d'accès aux données et à la définition de plusieurs profils d'utilisateurs. Ces exigences particulières peuvent être satisfaites par l'utilisation des Blockchains privées.

En raison de la prolifération des implémentations de la technologie Blockchain, l'observatoire et le forum de la blockchain de l'union européenne [1] a publié un rapport technique dans lequel il recommande l'utilisation des Blockchains privées pour le stockage de données sensibles, comme dans le cas des systèmes de vote électronique. Dans cette architecture Blockchain, les informations d'identification de l'utilisateur sont générées par une autorité de certification (AC). Les utilisateurs doivent donc être authentifiés au système par l'autorité de certification avant de rejoindre le réseau. L'avantage d'avoir un niveau de confiance minimum grâce à la connaissance des participants est que nous pouvons sécuriser le processus de réplication de la Blockchain en utilisant l'accord byzantin comme mécanisme de consensus.

Bien que les Blockchains privées présentent plusieurs caractéristiques adaptées aux services qui impliquent des données sensibles, telles que les informations personnelles des utilisateurs, elles présentent des inconvénients liés aux transactions et à la possibilité d'établir des liens avec les utilisateurs. Cela est dû au fait que chaque paramètre d'authentification, clé publique et certificat sont délivrés à des utilisateurs spécifiques qui étaient précédemment authentifiés auprès de l'AC. Afin de surmonter cet inconvénient, nous utilisons, dans notre approche, le schéma de signature en aveugle d'Okamoto-Schnorr [42] pour signer les transactions d'une manière anonyme. Ce modèle permet de valider les transactions sans exposer l'identifiant des votants, et donc de préserver l'anonymat des votes.

Dans ce chapitre, nous visons à concevoir un protocole de vote électronique en ligne sécurisé, qui traite les exigences de sécurité mentionnées ci-dessus, basé sur une Blockchain privée et de diverses primitives cryptographiques. Appelé *DABSTERS*, notre protocole utilise une nouvelle architecture basée sur la Blockchain Hyperledger Fabric [26] et le schéma de signature en aveugle d'Okamoto-Schnorr [42]. Il répond aux propriétés de sécurité suivantes : éligibilité, pas de résultat partiel, robustesse, intégrité, vérifiabilité individuelle, vérifiabilité universelle, anonymat des votes et sans-reçu. Nos contributions peuvent être résumées comme suit :

1. Une nouvelle architecture de confiance pour les systèmes de vote électronique. Cette architecture est basée sur une Blockchain privée et sur un nouveau consensus qui assure l'anonymat des votants et l'intégrité des votes.
2. Un protocole de vote électronique en ligne sécurisé et entièrement distribué, basé sur l'architecture que nous proposons. Appelé *DABSTERS*, ce protocole utilise la même structure de bulletin de vote que le protocole VYV (Chapitre3), mais qui s'accompagne d'une multitudes d'améliorations significatives pour remédier aux faiblesses de VYV (Voir Section4.3 pour plus de détails).

Ces contributions faisaient l'objet de deux publications internationales [43, 44].

1. https://www.eublockchainforum.eu/

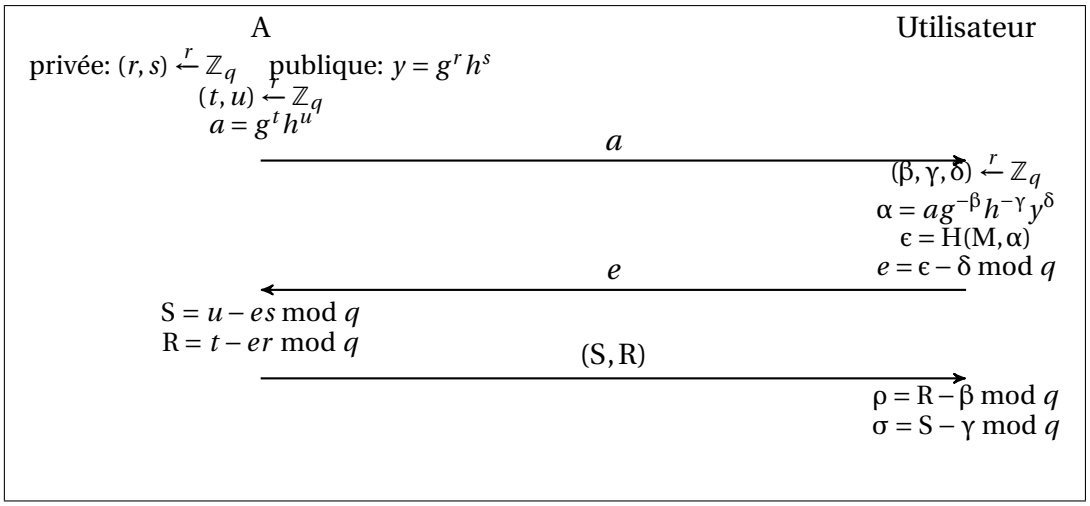


FIGURE 4.1 – Diagramme de la signature en aveugle d'Okamoto-Schnorr.

4.2 Préliminaires

Dans cette partie, nous présentons le schéma de signature d'Okamoto-Schnorr [42] ainsi que l'algorithme de consensus basé sur la tolérance aux pannes byzantines (Byzantine Fault Tolerance BFT).

4.2.1 Schéma de signature en aveugle d'Okamoto-Schnorr

Soit p et q deux grands nombres premiers avec $q|p-1$ (q divise $p-1$). Soit $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q, et g et h deux générateurs de $\mathbb{G}$. Soit $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ une fonction de hachage. Le schéma de signature en aveugle d'Okamoto-schnorr [42] se compose des 3 étapes suivantes et est illustré par la Figure 4.1.

Génération de clés : Soit $(r,s) \xleftarrow{r} \mathbb{Z}_q$ et $y = g^r h^s$ les clés privées et publiques, respectivement, de l'autorité A.

Signature en aveugle : Elle se déroule en 4 étapes :

1. A choisit $(t,u) \xleftarrow{r} \mathbb{Z}_q$, calcule $a = g^t h^u$, et envoie a à l'utilisateur.
2. L'utilisateur choisit $(\beta,\gamma,\delta) \xleftarrow{r} \mathbb{Z}_q$ et calcule la version aveugle de a qui est $\alpha = a g^{-\beta} h^{-\gamma} y^{\delta}$, et $\epsilon = H(M,\alpha)$. Ensuite, il calcule $e = \epsilon - \delta \bmod q$, et envoie e à A.
3. A calcule $S = u - es \bmod q$ et $R = t - er \bmod q$ et envoie (S,R) à l'utilisateur.
4. L'utilisateur calcule $\rho = R - \beta \bmod q$ et $\sigma = S - \gamma \bmod q$.

Vérification : Étant donné un message $M \in \{0,1\}^*$ et une signature (ρ,σ,ϵ), on a $\alpha = g^{\rho} h^{\sigma} y^{\epsilon} \bmod p$.

4.2.2 Algorithme de consensus basé sur la BFT

Considérons un algorithme de consensus avec autorisation et basé sur la tolérance aux fautes byzantines (BFT) comme celui introduit dans Hyperledger Fabric [26]. Dans ce protocole, les utilisateurs sont authentifiés avec leurs signatures numériques, sans protéger leurs identités. Ainsi, afin de concevoir un protocole de consensus préservant l'anonymat des utilisateurs, nous devons ajouter les éléments suivants à l'algorithme de consensus basé sur la BFT :

— Alice envoie une transaction nouvellement signée à l'autorité d'enregistrement (AE) qui est responsable de l'enregistrement d'Alice.
— La signature d'Alice n'est validée que par l'AE.
— L'AE rend l'identité d'Alice anonyme.
— L'AE signe la transaction envoyée par Alice au réseau.
— Tous les nœuds du processus de validation des transactions peuvent valider la signature de l'AE.
— La signature de l'AE ne peut pas être dupliquée.

Pour préserver l'anonymat des utilisateurs et des entités impliqués dans le processus transactionnel, nous devons cacher leurs identités et rendre leurs signatures aveugles. Pour ce faire, nous remplaçons le mécanisme de signature utilisé dans le protocole original par le système de signature en aveugle d'Okamoto-Schnorr. Afin de maintenir la cohérence et la vivacité du protocole, nous conservons le flux des transactions. Cependant, les étapes sont modifiées afin d'accepter le nouveau système de signature en aveugle pour authentifier les utilisateurs. Le processus transactionnel basé sur l'algorithme de consensus BFT avec la signature en aveugle comprend les étapes suivantes :

1. *Initiation d'une transaction :* Un client C génère un message M pour exécuter une opération O_C dans le réseau avec une signature en aveugle en utilisant la fonction *SignerAveugle*$((M,\beta,\sigma,\gamma),y)$.
2. *Proposition de la transaction :* L'entité qui soumet la demande de transaction, notée SP, reçoit le message M du client C, vérifie la signature en utilisant la fonction *VérifierSignatureAveugle(M,*$(\rho,\delta,\epsilon),y$) et propose une transaction avec l'opération du client O_C.
3. *Approbation de la transaction :* Les entités endosseurs, notées EP, vérifient la signature en aveugle du client en utilisant *VérifierSignatureAveugle(M,*$(\rho,\delta,\epsilon),y$) et vérifient si la transaction est valide ou pas en simulant l'opération O_C sur leurs versions locales de la Blockchain. Ensuite, elles génèrent des transactions signées avec le résultat du processus de validation et l'envoient à SP.
4. *Diffusion au consensus :* L'entité SP collecte l'approbation des EP connectées au réseau. Une fois que SP a recueilli suffisamment de réponses valides de la part des EP, elle diffuse la transaction au service d'ordonnancement.
5. *Validation :* Pendant cette étape, toutes les transactions sont regroupées dans un bloc et sont validées avec leurs endossements respectifs. Ensuite, le nouveau bloc est diffusé sur le réseau pour être validé par les autres entités.

4.3 Description du protocole proposé : DABSTERS

Notre protocole utilise la même structure de bulletin de vote que le protocole VYV (Chapitre 3), illustrée par la Figure 4.1, mais avec une architecture de système totalement différente. En effet, cette nouvelle architecture de confiance est basée sur un algorithme de consensus qui intègre le schéma de signature en aveugle avec l'algorithme BFT.

Grâce à cette architecture de confiance, notre protocole *DABSTERS* élimine le risque d'invalider l'élection à cause de mineurs malhonnêtes qui modifient les transactions avant de les stocker dans des blocs. Nous proposons également une phase d'enregistrement distribuée pour réduire la confiance à l'autorité d'enregistrement et nous imposons la publication de la liste des votants inscrits à la fin de la phase d'enregistrement. Cette liste peut être vérifiable par toutes parties intéressées.

DABSTERS contient 5 phases. Il commence par *une phase d'enregistrement* au cours de laquelle les autorités d'enregistrement (AEs) vérifient l'éligibilité des votants en vé-

Numéro du bulletin BN			
Pseudo ID "C_j"	**Nom du candidat** "nom_j"	**Choix**	**Contre-valeurs** "$CV_{BN,nom_j,k}$"
0	Paul	☐	$CV_{BN,nom_0,0}$
1	Nico	☐	$CV_{BN,nom_1,1}$
2	Joel	☐	$CV_{BN,nom_2,2}$

TABLEAU 4.1 – Structure du bulletin de vote.

rifiant l'existence de leurs noms et de leurs numéros de carte d'identité dans une liste publiée au préalable et contenant les noms de toutes les personnes ayant le droit de voter. Ensuite, tous les votants éligibles sont enregistrés et munis de leurs paramètres d'authentification. La phase d'enregistrement est hors ligne. À la fin de cette phase, les AEs forment une liste contenant les noms de tous les votants enregistrés et leurs numéros de carte d'identité. Cette liste peut être rejetée ou publiée sur le Blockchain de l'élection pendant *la phase de validation*. Une fois la liste est validée, on passe à la troisième étape qui est *la phase de vote*. Chaque votant éligible (V_i) initie une transaction dans laquelle il met son vote chiffré, signe la transaction à l'aide de son identifiant et l'envoie aux AEs pour vérifier sa signature et la rendre aveugle. Ensuite, le votant envoie la transaction avec sa signature aveuglée et son identifiant anonyme aux entités du consensus pour être validée et stockée dans la Blockchain de l'élection d'une manière anonyme. Après avoir validé et stocké tous les votes dans notre Blockchain, les autorités de comptage (ACs) récupèrent ces votes chiffrés, les déchiffrent et procèdent au comptage, durant *la phase de comptage*. La dernière étape est *la phase de vérification*. Au cours de cette phase, les votants peuvent vérifier que leurs votes ont été pris en compte correctement et que le résultat final de l'élection correspond à la somme de tous les votes éligibles. La vérifiabilité individuelle est assurée grâce à la structure de nos bulletins de vote et la vérification universelle est assurée grâce à la propriété d'homomorphisme des pairings. À l'exception de la phase d'enregistrement, toutes les phases de notre protocole utilisent la Blockchain. Nous faisons appel, ainsi, au protocole de consensus basé sur la BFT avec chaque transaction initiée par les autorités, et le nouvel algorithme de consensus, que nous proposons, avec chaque transaction initiée par les votants éligibles, car nous n'avons pas besoin de cacher l'identité de nos autorités mais nous devons garantir l'anonymat des votants. Dans ce qui suit, nous donnons une description détaillée des rôles des entités de notre protocole ainsi que ses différentes phases et les deux algorithmes de consensus.

4.3.1 Entités du protocole

DABSTERS implique trois entités principales :

— *Autorités d'enregistrement (AEs) :* Elles vérifient l'éligibilité de toute personne souhaitant s'enregistrer à l'élection et fournissent aux votants leurs paramètres d'authentification, qui sont construits par coopération entre toutes les AEs.

— *Votants éligibles (V) :* Chaque votant éligible (V_i) a le droit de voter plus d'une fois avant la fin de la phase de vote et seul son dernier vote valide est comptabilisé. Les votants ont la possibilité de vérifier que leurs votes sont exprimés comme prévu et qu'ils sont comptés comme exprimés, pendant la phase de vérification. Ils peuvent également vérifier l'exactitude du résultat final de l'élection, mais ils ne sont pas obligés de participer à la phase de vérification (ils peuvent voter et quitter le système de vote).

— *Autorités de comptage (ACs) :* Le protocole fait appel à autant d'autorités de comptage que de candidats. Avant la phase de vote, elles construisent n bulletins de vote, où n est le nombre de votants inscrits. Ainsi, chaque votant a un bulletin de vote unique qui se distingue des autres par son numéro et l'ordre des candidats. Les ACs chiffrent les bulletins de vote et les envoient aux votants pendant la phase de vote. Elles déchiffrent les votes et calculent le résultat final de l'élection pendant la phase de comptage et publient les différentes valeurs qui permettent aux électeurs de vérifier l'exactitude du comptage pendant la phase de vérification.

DABSTERS fait appel, également, à des observateurs et des organisateurs d'élections qui ont le droit d'être des entités dans la Blockchain, afin de s'assurer de la bonne exécution du protocole.

4.3.2 Phases du protocole

Notre approche comprend les phases suivantes :

Phase d'enregistrement

Toute personne, qui a le droit de voter et qui souhaite le faire, se rend physiquement au bureau de vote le plus proche. Elle fournit sa carte d'identité nationale aux autorités d'enregistrement, qui vérifient son éligibilité en vérifiant si son nom et son numéro de carte d'identité figurent sur une liste, préalablement publiée, qui contient toutes les personnes pouvant participer à l'élection. S'il est éligible, les AEs enregistrent le numéro de sa carte d'identité et lui fournissent un identifiant anonyme qui lui permet de participer à l'élection. Les identifiants anonymes des votants sont calculés en utilisant la cryptographie à base de courbe elliptiques et ont la forme suivante :

$$\mathrm{ID}anonyme_{\mathrm{V}_i} = \mathrm{S_M} \cdot \mathrm{H_1}(\mathrm{ID}_{\mathrm{V}_i})$$

Où :

— $\mathrm{S_M} = \mathrm{S_1} \cdot \mathrm{S_2} \ldots, \mathrm{S_R}$ est une clé maitresse secrète calculée par coopération entre toutes les AEs. Chaque AE participe avec son fragment secret S_r ; $r \in \{1 \ldots \mathrm{R}\}$,
— $\mathrm{H_1}$ est une fonction de hachage, définie comme suit : $\mathrm{H_1} : \{0,1\}^* \rightarrow \mathrm{G_1}$; $\mathrm{G_1}$ est un groupe cyclique additif d'ordre un nombre premier q,
— $\mathrm{ID}_{\mathrm{V}_i}$ est le numéro de la carte d'identité du votant V_i.

Phase de validation

Après avoir enregistré tous les votants éligibles, les AEs forment une liste contenant les noms et les numéros des cartes d'identité de tous les votants enregistrés. Cette liste doit être consultée et vérifiée par les votants, les observateurs et les organisateurs des élections. Ainsi, les AEs envoient cette liste dans une transaction sur la Blockchain de l'élection. Cette transaction passe par les cinq étapes du protocole de consensus basé sur la BFT, pour être validée si la liste est valide ou rejetée si la liste contient des noms de votants inéligibles. Nous détaillons, dans ce qui suit, les différentes étapes du consensus basé sur la BFT par lesquelles passe une transaction pour être validée et ajoutée à la Blockchain de l'élection.

1. *Étape 1 : Initiation de la transaction.* Les AEs génèrent la liste des votants éligibles à valider par les entités du réseau. La liste est envoyée à l'entité du consensus BFT qui soumet la transaction (SP). Dans le cas d'une SP hors ligne ou qui se comporte

mal, les AEs envoient la transaction à la prochaine SP. Cette étape est illustrée par la Figure 4.2.

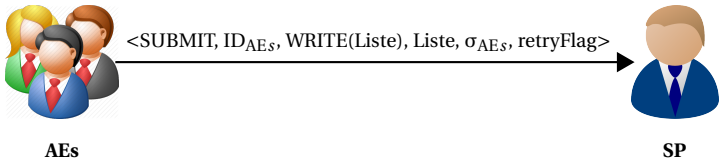


FIGURE 4.2 – Étape 1 : Initiation d'une transaction.

- ID_{AEs} est l'identifiant des autorités d'enregistrement,
- $Write(Liste)$ est l'opération invoquée par les AEs pour être exécutée par le réseau. Elle consiste à écrire la liste des votants éligibles et leurs numéros de carte d'identité sur la Blockchain de l'élection,
- $Liste$ est la liste des votants enregistrés,
- σ_{RA} est la signature des autorités d'enregistrement,
- $retryFlag$ est une variable booléenne permettant d'identifier s'il faut, ou pas, re-soumettre la transaction en cas d'échec de celle-ci.

2. *Étape 2 : Proposition de la transaction.* SP reçoit la transaction et vérifie la signature des AEs. Elle prépare ensuite une proposition de transaction à envoyer aux EP. Ces dernières peuvent être composées de votants, d'organisateurs d'élections et d'observateurs. Cette étape est décrite dans la Figure 4.3.

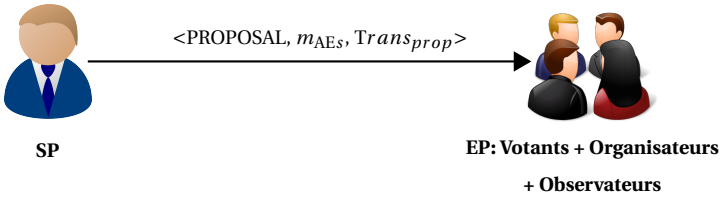


FIGURE 4.3 – Étape 2 : Proposition de la transaction.

- $m_{AEs} = (ID_{AEs}, Write(Liste), Liste, \sigma_{AEs})$
- $Trans_{prop} = (SP, Write(Liste), Liste, StateUpdate, VerDep)$ avec :
  - *StateUpdate* correspond à l'état de la Blockchain après avoir simulé, localement, l'opération $Write(List)$.
  - *VerDep* est la version associée à la variable à créer ou à modifier. Elle est utilisée pour maintenir la cohérence des variables entre les différentes versions d'état de la Blockchain.

3. *Étape 3 : Approbation de la transaction.* Chaque EP vérifie la signature des AEs σ_{RA} venant dans m_{AEs} et vérifie que la liste des votants éligibles incluse dans m_{AEs} et $Trans_{prop}$ est la même. Ensuite, chaque EP vérifie l'éligibilité de tous les noms et numéros de cartes d'identité figurant sur la liste. S'ils sont tous valides, l'EP génère un message de *transaction valide* à envoyer au SP (Figure 4.4). Si la liste comprend des votants inéligibles, l'EP génère un message de *transaction invalide* (Figure 4.5).
   - Tx_{ID} est l'identifiant de la transaction,
   - σ_{EP} est la signatures de l'EP.
   - $Erreur$: peut avoir les valeurs suivantes :
     - ÉTAT-INCORRECTE : lorsque l'EP tente de valider la transaction avec une version locale de la Blockchain différente de celle qui figure dans la proposition de la transaction.

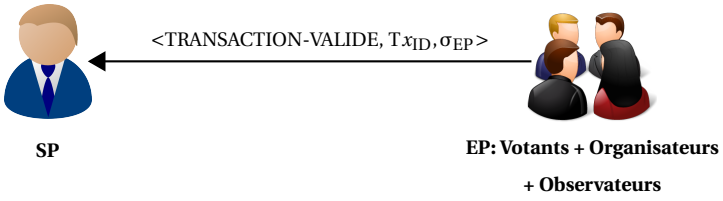


FIGURE 4.4 – Étape 3 : Approbation de la transaction dans le cas d'une transaction valide.

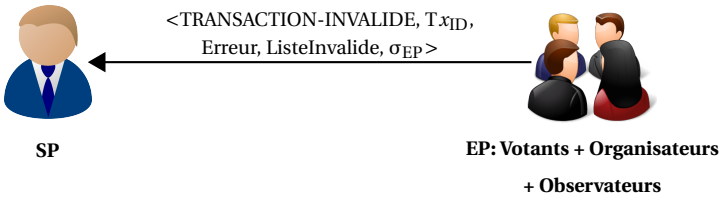


FIGURE 4.5 – Étape 3 : Approbation de la transaction dans le cas d'une transaction invalide.

- VERSION-INCORRECTE : lorsque la version de la variable dans laquelle la liste sera enregistrée diffère de celle mentionnée dans la proposition de la transaction.
- REJETÉE : pour toute autre raison.

— ListeInvalide : est la liste des noms inéligibles qui ont été inclus dans la liste envoyée par les AEs.

4. *Étape 4 : Diffusion au consensus.* Quand SP reçoit un nombre suffisant de messages de *transaction valide* adéquatement signés par les EP, elle stocke les signatures d'approbation dans un paquet appelé "endossement". Une fois que la transaction est considérée comme approuvée, SP invoque les services de consensus en utilisant la fonction $broadcast(blob)$, où $blob = (Trans_{prop}, endossement)$ (Figure4.6).
Le nombre de réponses et d'approbations pour considérer la proposition de transaction comme approuvée est configuré au préalable. Dans notre cas, on considère qur ce nombre égal à 50% + 1 du nombre total des EP. Si la transaction n'a pas recueilli suffisamment d'endossements, SP abandonne cette transaction et informe les AEs.

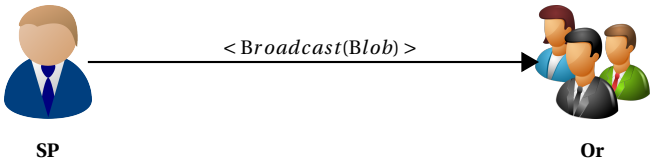


FIGURE 4.6 – Étape 4 : Diffusion au consensus.

5. *Étape 5 : Validation.* Une fois que SP diffuse la transaction au consensus, les services d'ordonnancement la mettent dans le bloc actuel, qui sera envoyé à toutes les entités du consensus, une fois construit. Enfin, si la transaction n'a pas été validée, les autorités d'enregistrement sont informées par SP.
Dans le cas d'une liste de votant enregistrés invalide, les AEs doivent rectifier la liste et relancer la phase de validation. Nous ne passons à la phase suivante (qui est la phase de vote) que lorsque nous obtenons une liste valide ne contenant que des votants éligibles.

Phase de vote

Deux entités participent à cette phase :

— Les autorités de comptage qui ont construit les bulletins de vote avant le début de la phase de vote. Pour construire un bulletin de vote, les ACs calculent, en locale, le numéro de bulletin $BN = \{g, D\}_{PK_{TA}}$, la valeur de décalage $Décalage = H(g) \mod m$ et les contre-valeurs $CV_{BN,nom_j,k} = e(Q_{nom_j}, S_k \cdot Q_{BN})$, où g est un générateur d'un groupe cyclique additif $\mathbb{G}$ d'ordre un nombre premier, D est un nombre aléatoire, PK_{ACs} est la clé publique des ACs, m est le nombre de candidats, $e(.,.)$ est la fonction de pairing, S_k est la clé secrète de l'autorité de comptage AC_k, $Q_{nom_j} = H_1(nom_j)$ et $Q_{BN} = H_1(BN)$ sont deux points d'une courbe elliptique E. Ensuite, les ACs choisissent, au hasard, un bulletin de vote vierge, le chiffrent avec la clé publique d'un votant et la transmettre au votant correspondant via la Blockchain. Les bulletins de vote sont envoyés chiffrés parce qu'ils contiennent des informations secrètes comme le numéro de bulletin BN et la valeur de décalage. Pour envoyer des bulletins de vote chiffrés aux votants via la Blockchain, les ACs interagissent avec les entités du consensus BFT. Ces interactions se déroulent en cinq étapes, les mêmes que celles présentées dans la sous section 4.3.2 et décrites dans la Figure 4.7.

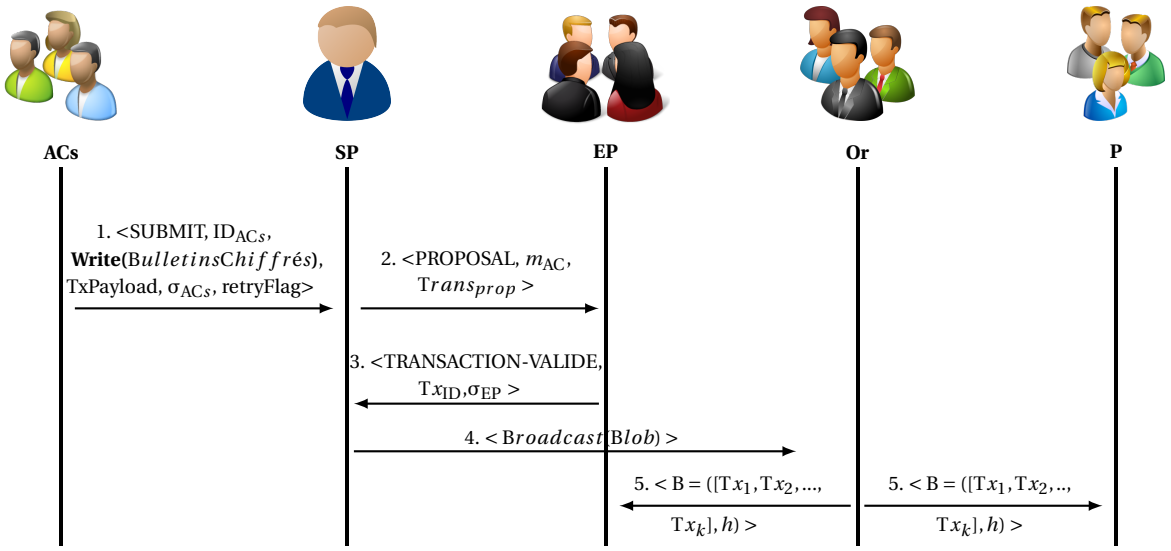


FIGURE 4.7 – Interactions entre ACs et les entités du consensus basé sur la BFT.

1. *Initiation de la transaction.* Les ACs initient une transaction et l'envoient à SP. La transaction contient leur ID (ID_{ACs}), la liste des bulletin chiffrés, la charge de la transaction $TxPayload$, leur signature (σ_{ACs}) et la valeur de la variable $retryFlag$.
2. *Proposition de la transaction.* SP vérifie la signature des ACs et prépare une proposition de transaction $Trans_{prop}$ = (SP, $Write$(BulletinsChiffrés), BulletinsChiffrés, $stateUpdate$, $VerDep$) à envoyer aux EP avec le message des ACs m_{ACs} = (ID_{ACs}, $Write$(BulletinsChiffrés), BulletinsChiffrés, σ_{ACs}).
3. *Approbation de la transaction.* Chaque EP vérifie la signature σ_{ACs} venant dans le message m_{ACs}, simule la proposition de transaction et vérifie la validité de $stateUpdate$ et $verDep$. Si le processus de validation est réussi, l'EP génère un message de *Transaction valide* à envoyer à SP.

4. *Diffusion au consensus.* Une fois SP reçoit un nombre de message de *Transaction valide* égale à 50% + 1 du nombre totale des EP, adéquatement signés, elle stocke les signatures dans le paquet "endossement" et invoque le service de consensus à l'aide de la fonction $broadcast(blob)$; avec $blob = (\mathrm{T}rans_{prop}, endossement)$.
5. *Validation.* Les services d'ordonnancement (Or) ajoute la transaction à un bloc. Une fois qu'ils collectent suffisamment de transactions, ils calculent l'entête du bloc et l'envoient à toutes les autres entités du consensus. Un bloc B a la forme suivante : $\mathrm{B} = ([tx_1, tx_2, \ldots, tx_k]; h)$ avec h est l'entête du bloc, contenant le hash du bloc précédent.

— Chaque votant éligible récupère son bulletin de vote, le déchiffre avec sa clé privée, choisit son candidat préféré, chiffre son vote et l'envoie dans une transaction via la Blockchain de l'élection. Pour chiffrer son vote, le votant utilise la cryptographie à base d'identité [12, 13] et chiffre le numéro de son bulletin avec $\mathrm{Q}_{\mathrm{C}_j} = \mathrm{H}_1(\mathrm{C}_j)$ avec C_j est le pseudo ID du candidat choisi. Ainsi, un vote chiffré a la forme suivante : $VoteChiffré = \{\mathrm{BN}\}_{\mathrm{Q}_{\mathrm{C}_j}}$.
Pour être écrites sur la Blockchain de l'élection, les transaction des votants passent par notre nouvel algorithme de consensus qui utilise la signature en aveugle afin d'assurer l'anonymat des votants. Nous modélisons les étapes à travers lesquelles passe une transaction d'un votant éligible dans la Figure 4.8. Nous prenons un exemple d'une transaction contenant une vote chiffré.

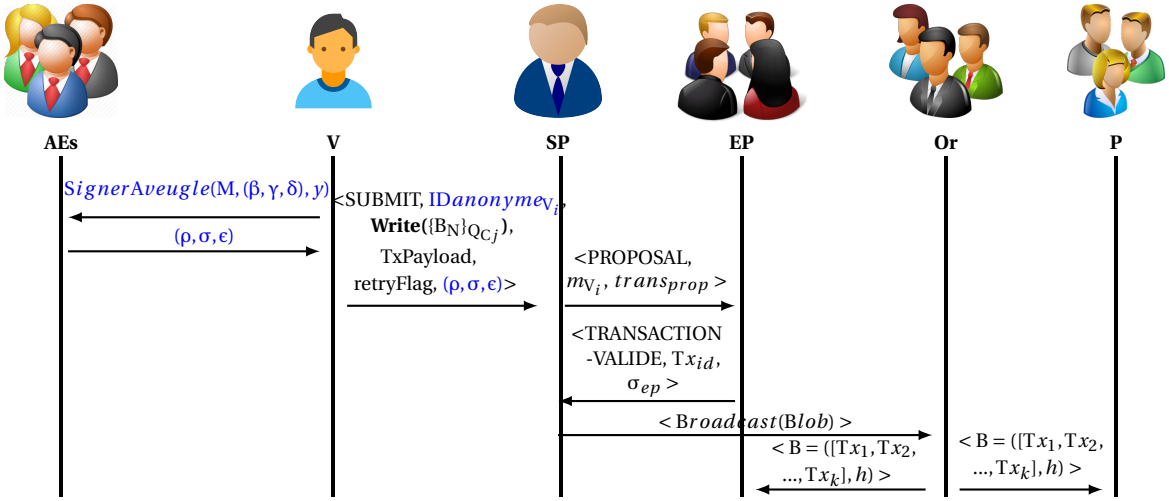


FIGURE 4.8 – Interactions entre un votant éligible et les entités du nouveau consensus.

Durant les interactions entre les ACs et les entités du consensus basé sur la BFT, on utilise la signature digitale comme une méthode d'authentification sans protéger ou cacher l'identité des ACs puisque nous n'avons pas besoin de masquer les identités des autorités de notre protocole. Cependant, quand il s'agit des interactions entre les votants et les entités de la Blockchain, il est impératif de préserver l'anonymat des votants en utilisant les signatures en aveugle. Note nouveau consensus ajoute deux étapes au consensus basé sur la BFT :

— La signature de chaque votant est aveuglée automatiquement, après avoir envoyé un vote par la fonction $SignerAveugle(\mathrm{M}, (\beta, \gamma, \delta), \mathrm{PK}_{\mathrm{AE}s})$, avec :
  — $\mathrm{M} = (\mathrm{ID}anonyme_{\mathrm{V}_i} || Write(VoteChiffré) || VoteChiffré || retryFlag)$ est le message à signer,

  - (β, γ, δ) sont des valeurs secrètes choisies aléatoirement par le le votant,
  - PK_{AEs} est la clé publique des AEs.
- AEs masquent la signature de chaque votant éligible en lui fournissant le n-uplet (R, S), permettant au votant de construire sa nouvelle signature (ρ, σ, ϵ) qui va être utilisée durant ses interactions avec les entités du consensus.

Les autres étapes sont analogues aux étapes du consensus basé sur la BFT, mais au lieu d'utiliser leurs signatures, les votants utilisent les signatures en aveugle que les AEs leurs fournissent.

1. *Initiation de la transaction :*
   <SUBMIT, $ID anonyme_{V_i}$, $Write(VoteChiffré)$, VoteChiffré, retryFlag, (ρ, σ, ϵ)>
2. *Proposition de la transaction :* <PROPOSAL, m_{V_i}, $trans_{prop}$ >
3. *Approbation de la transaction :* < TRANSACTION-VALIDE, Tx_{id}, σ_{EP} >
4. *Diffusion au consensus :* <Broadcast(Blob)>
5. *Validation :* < B = ([$Tx_1, Tx_2, \ldots, Tx_k$], h) >

Tout votant souhaitant vérifier que son vote a été compté correctement, doit mémoriser la contre-valeur correspondante à son candidat préféré.

Phase de comptage

Après avoir envoyé tous les votes, les ACs procèdent au comptage. Nous rappelons ici que nous avons autant d'ACs que de candidats. Chaque autorité de comptage AC_k calcule le nombre de vote reçu par un certain pseudo ID. Par exemple : la première autorité de comptage AC_1 déchiffre, avec sa clé privée $S_1 \cdot Q_{C_1}$, tous les votes qui ont été chiffrés avec la clé publique Q_{C_1}. AC_k. Elle commence par initier une transaction pour récupérer les votes chiffrés à partir de la Blockchain de l'élection. Cette transaction passe par les cinq étapes du consensus basé sur la BFT. Ensuite, elle déchiffre les votes avec sa clé secrète S_k afin de révéler le numéro de bulletin BN. À partir de ce numéro, elle reconstruit le bulletin de vote, identifie le candidat choisi et incrémente le compteur correspondant. À la fin de cette phase, TA_k calcule et publie le décompte de chaque candidat en utilisant la formule suivante : $\sigma_{k,nom_j} = \sum_{i=1}^{l_j} S_k \cdot Q_{BN_i}$; Où l_j est le nombre de vote reçu par le candidat j, S_k est la clé privée de TA_k, $Q_{BN_i} = H_1(BN_i)$ et BN_i est le numéro de bulletin de vote i qui correspond au candidat de nom nom_j.

Phase de vérification

Cette phase permet aux votants de vérifier que leurs votes ont été comptabilisés comme exprimés et que le résultat final de l'élection correspond à la somme de tous les votes éligibles. Elle comprend deux sous-phases. Au cours de la première, les ACs calculent la liste des contre-valeurs choisies à partir de chaque numéro de bulletin et du nom du candidat choisi, et publient cette liste sur la Blockchain. Chaque votant éligible peut lire cette liste et vérifier l'existence de sa contre-valeur pour s'assurer que son vote a été compté correctement. La deuxième sous-phase utilise la propriété d'homomorphisme des pairings pour vérifier l'exactitude du résultat final de l'élection. En utilisant les décomptes publiés par les ACs et les contre-valeurs reconstituées, nous pouvons vérifier l'exactitude

du résultat final comme suit :

$$\begin{aligned}\prod_{i=1}^{l} \mathrm{CV}_{\mathrm{BN}_i} &= \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} \mathrm{CV}_{\mathrm{BN}_{i,nom_j},k} = \prod_{k=1}^{m}\prod_{j=1}^{m}\prod_{i=1}^{l_j} e(\mathrm{Q}_{nom_j}, \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i}) \\ &= \prod_{k=1}^{m}\prod_{j=1}^{m} e(\mathrm{Q}_{nom_j}, \sum_{i=1}^{l_j} \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i}) = \prod_{k=1}^{m}\prod_{j=1}^{m} e(\mathrm{Q}_{nom_j}, \sigma_{k,nom_j}) \end{aligned} \tag{4.1}$$

Où $l = \sum_{j=1}^{m} l_j$ est le nombre totale de votes.
Ces égalités utilisent la propriété d'homomorphisme des pairings :

$$\prod_{i=1}^{l_j} e(\mathrm{Q}_{nom_j}, \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i}) = e(\mathrm{Q}_{nom_j}, \sum_{i=1}^{l_j} \mathrm{S}_k \cdot \mathrm{Q}_{\mathrm{BN}_i})$$

4.4 Évaluation de la sécurité de DABSTERS

Dans cette section, nous évaluons, informellement puis formellement, la sécurité de notre protocole DABSTERS.

4.4.1 Évaluation informelle

Dans cette partie, nous évaluons la sécurité de notre protocole contre la liste des exigences de sécurité que nous avons définie dans le Chapitre 1.

— *Éligibilité :* Les phases d'enregistrement et de validation de notre protocole garantissent que seuls les électeurs éligibles participent au processus de vote. Lors de la phase d'enregistrement, les autorités vérifient l'identité de chaque votant et seuls ceux qui sont éligibles reçoivent des paramètres d'authentification. Pendant la phase de validation, les AEs envoient la liste des électeurs inscrits aux entités du consensus, qui sont composés de votants, d'organisateurs d'élections et d'observateurs, afin de vérifier l'éligibilité de tous les électeurs inscrits et de valider ou de rejeter cette liste.

— *Pas de résultat partiel :* Pendant la phase de vote, chaque électeur éligible chiffre son bulletin de vote BN avec $\mathrm{Q}_{\mathrm{C}_j} = \mathrm{H}_1(\mathrm{C}_j)$ où C_j est le pseudo-ID du candidat préféré. Les numéros des bulletins sont secrets et les pseudo-ID des candidats ne reflètent pas l'identité réelle des candidats grâce à la valeur de décalage, de sorte que personne ne peut identifier le candidat choisi à partir du vote chiffré. Ainsi, nous ne pouvons pas obtenir de résultats partiels avant que le résultat officiel.

— *Robustesse :* DABSTERS tolère le mauvais comportement des votants malveillants.

— *Intégrité :* Le consensus basé sur la BFT et les signatures en aveugle empêchent les votes d'être modifiés tout en préservant le secret de l'électeur. Chaque transaction est stockée dans la Blockchain après avoir été validée par 50 % + 1 des EP. Cela élimine le risque de modifier les transactions avant de les stocker. Nous mentionnons ici que le consensus de la BFT est basé sur l'hypothèse que 2/3 des entités qui ont approuvé la transaction sont honnêtes.

— *Vérifiabilité individuelle :* Cette propriété est assurée par notre protocole grâce à la structure des bulletins de vote qui comprend les contre-valeurs. Ces valeurs servent de reçus aux électeurs et leur permettent de vérifier que leurs votes ont été exprimés comme prévu sans révéler pour qui ils ont voté. En fait, les contre-valeurs

sont calculées en utilisant la formule suivante : $CV_{BN,name_j,k} = e(Q_{name_j}, S_k \cdot Q_{BN})$. Ainsi, nous ne pouvons pas obtenir le nom du candidat à partir de la valeur de $CV_{BN,name_j,k}$.

— *Vérifiabilité universelle :* A partir des paramètres publiés par les ACs lors de la phase de vérification, chacun peut vérifier l'exactitude du résultat final en vérifiant l'équation (4.1).
— *Anonymat :* Cette propriété est assurée grâce à notre nouveau consensus. Avant d'interagir avec les entités de la Blockchain, les AEs rendent aveugle la signature de tous les électeurs éligibles pour cacher leurs identités réelles.
— *Sans-reçu :* Dans notre cas, un votant ne peut pas trouver son vote à partir de la contre-valeur $CV_{BN,name_j,k}$ et des autres paramètres publiques. Il ne peut donc pas prouver qu'il a voté pour un candidat donné.
— *Résistance à la coercition :* Notre protocole n'est pas résistant à la coercition. Un attaquant peut forcer un électeur à voter pour un certain candidat et vérifier sa soumission plus tard en utilisant la contre-valeur.
— *Voter et quitter :* DABSTERS n'exige pas que le votant attend la fin de la phase de vote ni qu'il déclenche le décompte. Il peut tout simplement voter et quitter le système de vote.
— *Politique de vote :* Notre approche donne aux électeurs éligibles la possibilité de voter plus qu'une fois avant la fin de la phase de vote et seuls leurs derniers votes valides sont comptabilisés.

Le Tableau 4.2 résume les propriétés de sécurité de DABSTERS.

DABSTERS	
Éligibilité	✓
Pas de résultat partiel	✓
Robustesse	✓
Intégrité	✓
Vérifiabilité individuelle	✓
Vérifiabilité universelle	✓
Anonymat	✓
Sans-reçu	✓
Résistance à la coercition	X
Voter et quitter	✓
Politique de vote	Multiple

TABLEAU 4.2 – Évaluation de la sécurité de DABSTERS.

4.4.2 Évaluation formelle

Dans cette partie, nous effectuons une analyse formelle de la sécurité de notre protocole en utilisant l'outil de vérification automatique ProVerif. Pour ce faire, nous avons besoin, tout d'abord, d'écrire le protocole de vote en utilisant le langage de modélisation formelle Applied Pi-Calculus.

Le code est donné dans l'Annexe B. On définit les requêtes suivantes pour prouver *le secret de vote, l'authentification des électeurs* et *la confidentialité des votes* et on donne les résultats de l'exécution des codes, ainsi que le temps que prend ProVerif pour prouver ces propriétés dans le Tableau 4.3.

— *Vérification du secret de vote :* Afin d'intercepter la valeur d'un vote donné, un attaquant doit intercepter les valeurs de deux paramètres : le numéro de bulletin de vote BN et le pseudo ID du candidat choisi Cj. Nous utilisons donc les requêtes suivantes :

```
query attacker(BN).
query attacker(Cj).
```

— *Vérification de l'authentification des électeurs :* Cette propriété est vérifié à l'aide *d'assertions de correspondance*. Le protocole vise à garantir que les autorités vérifient la validité des paramètres d'authentification des votants. Par conséquent, nous définissons les évènements suivants :
  — **event acceptedAuth(bitstring) :** utilisé par l'électeur pour enregistrer le fait qu'il a été authentifié avec succès,
  — **event VerifiesParam(bitstring) :** utilisé par les autorités pour enregistrer le fait qu'il a vérifié les paramètres d'authentification des électeurs.

  On définit aussi la requête suivante :

```
query a: bitstring;
event acceptedAuthentication(a)==> event VerifiesParameters(a).
```

— *Vérification de la confidentialité des votes :* Pour exprimer la confidentialité des votes, nous prouvons la propriété d'équivalence observationnelle entre deux instances de notre processus qui ne diffèrent que par le choix des votes. Pour ce faire, nous utilisons `choice[V1,V2]` pour représenter les termes qui diffèrent entre les deux instances.

Propriétés	Résultat	Temps d'exécution
Secret du vote	Prouvée	0.012 s
Authentification des votants	Prouvée	0.010 s
Confidentialité du vote	Prouvée	0.024 s

TABLEAU 4.3 – Résultats et temps d'exécution des codes ProVerif de DABSTERS.

4.5 Conclusion

Nous avons proposé un système de vote électronique entièrement décentralisé qui combine plusieurs propriétés de sécurité, appelé DABSTERS. Il utilise une nouvelle architecture qui permet de renforcer la sécurité des systèmes de vote électronique et de garantir la fiabilité requise par les votants et les organisateurs des élections. Il est conçu pour être implémenté sur des Blockchains privées et utilise un nouvel algorithme de consensus qui garantie l'intégrité des votes et l'anonymat des votants grâce à l'utilisation des signatures en aveugle. Nous avons également évalué la sécurité de notre approche et prouvé, formellement, qu'elle garantie le secret et la confidentialité des votes ainsi que l'authentification des votants. Nos futurs travaux seront consacrés à l'évaluation des performances et de la scalabilité de ce protocole.

Chapitre 5

"LOKI Vote" : Un protocole de vote électronique en ligne résistant à la coercition et basé sur la Blockchain LOKI

Sommaire

5.1 Introduction

La création d'un système de vote électronique en ligne qui assure à la fois la résistance à la coercition et la vérifiabilité de bout en bout, a constitué un véritable défi pendant une longue période. La notion de résistance à la coercition a été introduite, pour la première fois, par Juels, Catalano et Jakobsson (JCJ) en 2005 [45]. Depuis ce temps, plusieurs travaux de recherches ont été proposés dans le but de remédier au plus grand inconvénient de l'approche de JCJ, qui est la complexité quadratique de la vérification des paramètres d'authentification des votants. La majorité de ces systèmes repose sur l'utilisation d'un tableau d'affichage public, où sont affichés les votes et d'autres paramètres publics. Malgré cette exigence répandue, la notion d'un tableau d'affichage reste flou, et aucune méthode de construction d'un tel tableau n'a été proposée dans ces travaux. Ils partent toujours du principe que ce tableau d'affichage public garantit la vérifiabilité de bout en bout, l'intégrité et l'exactitude du processus électoral. Ainsi, les tableaux d'affichage publics doivent avoir les propriétés suivantes : (1) *Une architecture distribuée* pour

résister aux attaques par déni de service distribué (DDOS), (2) *Horodaté* afin de référencer les données par leurs dates de publication, (3) *Immuable* pour protéger les données de la suppression ou l'altération et enfin (4) *Vérifiable de manière universelle* pour garantir un niveau élevé de transparence. Ce sont exactement les principales caractéristiques de la technologie Blockchain. Dans ce chapitre, on comble le vide en proposant un protocole de vote électronique en ligne, qui est à la fois vérifiable de bout en bout et résistant à la coercition, basé sur la technologie Blockchain. Notre proposition s'inspire du schéma proposé par Araùjo et Traoré en 2013 [46], qui est basé sur les travaux de JCJ mais avec une complexité linéaire. Appelé LOKI Vote, notre système est pratique pour les élections à grande échelle et garantit une forte confidentialité pour les électeurs en utilisant une variété de primitives cryptographiques. En outre, notre protocole améliore la complexité des anciens systèmes résistants à la coercition en utilisant un nouveau réseau de mixage, appelé *"Low Latency Anonymous Routing Protocol"*, qui se caractérise par une complexité moindre et un niveau de sécurité plus élevé par rapport aux réseaux de mixage existants. Enfin, nous prouvons, formellement, la sécurité de LOKI Vote en utilisant l'outil de vérification automatique, ProVerif, et le langage de modélisation formelle Applied Pi-Calculus.

5.2 Étude de l'existant

Dans cette section, nous donnons un aperçu sur certains systèmes de vote électronique qui sont, ou prétendent être, résistants à la coercition. Nous commençons par décrire trois protocoles de la littérature qui sont intéressants pour notre travail et qui n'utilisent pas la technologie Blockchain (1, 2 et 3). Ensuite, nous présentons deux systèmes de vote électronique en ligne basés sur la technologie Blockchain et prétendus être résistants à la coercition (4 et 5). Nous évaluons la sécurité de ces systèmes dans le Tableau 5.2.

1. *Coercion Resistant Electronic Elections (CREE) [45] :* Dans leur article, Juels, Catalano et Jakobsson (JCJ) donnent la première définition formelle de la résistance à la coercition et proposent le premier système de vote électronique qui respecte cette propriété. Leur système repose sur une chaîne de caractères aléatoire et secrète, notée "σ", qui sert de paramètre d'authentification anonyme pour les électeurs éligibles. Chaque électeur éligible reçoit son paramètre d'authentification valide et anonyme pendant la phase d'enregistrement, après vérification de son éligibilité par une autorité appelé "*Registrar (R)*". Pour voter, chaque électeur chiffre son paramètre d'authentification, en utilisant une version modifiée du cryptosystème d'El-Gamal, et l'envoie avec son bulletin de vote à un tableau d'affichage public. Les auteurs partent du principe que le tableau d'affichage est universellement accessible, sur lequel chaque parti peut écrire et lire des données mais personne ne peut modifier ou supprimer des informations. À la fin de la phase de vote, une autorité appelée "*Talliers (T)*" effectue une comparaison aveugle (en utilisant le test d'équivalence de texte en clair PET [47, 48]) entre les paramètres d'authentification chiffrés et une liste L, publiée par R, contenant les paramètres d'authentification chiffrés de chaque votant enregistré avec son nom en texte clair. La liste des paramètres d'authentification chiffrés et la liste L passent par un réseau de mixage par rechiffrement [35, 49] avant d'être comparées entre elles. T ne retient que les votes dont les paramètres d'authentification correspondants correspondent à un élément de L. Enfin, T déchiffrent tous les votes valides et éligibles et calculent le résultat final.
   Le système de JCJ garantit la résistance à la coercition grâce à l'utilisation des paramètres d'authentification anonymes σ. En effet, lorsqu'un votant V_i est sous coerci-

tion, il peut tout simplement sélectionner et révéler un élément aléatoire du groupe σ'_i, en prétendant qu'il s'agit du σ_i. Puisque l'attaquant n'est pas en mesure de faire la distinction entre les paramètres d'authentification valides et ceux invalides, il ne peut pas être sûr si l'électeur a obéi à sa demande ou pas.
Le principal inconvénient du système de JCJ est sa complexité quadratique en fonction du nombre d'électeurs pendant la phase de comptage (lors de la vérification de la validité des paramètres d'authentification). Ce problème rend le système irréaliste car il ne peut pas être utilisé dans des élections à large échelle. Malgré cela, le protocole est largement discuté et pris comme point de départ pour de nouvelles contributions [46, 50, 51, 52, 53, 54].

2. *Towards Practical and Secure Coercion Resistant Electronic Elections (TPSCREE) [52] :* Pour surmonter l'inconvénient du système de JCJ, les auteurs proposent une nouvelle approche avec une complexité linéaire. Cette solution s'appuie sur le schéma de signature de groupe de BBS [55]. Dans leur article, les auteurs décrivent d'abord une attaque contre le système de Schweisgut [56] (qui est également basé sur le travail de JCJ) et prouvent qu'il n'est pas résistant à la coercition comme on le prétend puisqu'un attaquant peut vérifier plus tard si l'électeur a obéi à sa demande ou pas. Ensuite, ils proposent leur système de vote et prouvent formellement qu'il est résistant à la coercition et adapté aux élections à grande échelle. Le protocole proposé est basé sur les mêmes primitives cryptographiques que la proposition du JCJ, à savoir : un tableau d'affichage public [57], une version modifiée du cryptosystème d'El-Gamal, proposé par JCJ [45], un réseau de mixage universellement vérifiable [35, 58], un ensemble de preuves à divulgation nulle de connaissance [42, 59] et le PET [47]. Il comporte les phases suivantes :
   — *Phase d'enregistrement :* les autorités d'enregistrement (R) vérifient l'éligibilité de chaque électeur et lui fournissent ses paramètres d'authentification qui ont la forme suivante (A, r, x) où $\mathrm{A} = (g_1 g_3^x)^{1/(y+r)}$, g_1 et g_3 sont des paramètres publics, x est une valeur secrète, y est la clé privée de R et r est une valeur aléatoire.
   — *Phase de vote :* chaque électeur chiffre son vote et ses paramètres d'authentification et les envoie via un tableau d'affichage public, accompagnés d'un ensemble de preuves à divulgation nulle de connaissance pour prouver la validité du n-uplet de vote.
   — *Phase de comptage :* les autorités de comptage (T) récupèrent les n-uplets de vote du tableau d'affichage, vérifient la validité de chacun d'entre eux, éliminent les doublons et les n-uplets dont les paramètres d'authentification sont invalides, puis déchiffrent les votes restants et comptent le résultat final de l'élection.

   Lorsqu'il est sous coercition, l'électeur donne de faux paramètres d'authentification. Ces dernières ont la forme suivante (A, r, x') où $x' \neq x$. Ce protocole présente deux problèmes. (1) Un ensemble d'autorités d'enregistrement malveillantes ont la possibilité de fournir aux électeurs inéligibles des paramètres d'authentification valides. Ainsi, le décompte final peut inclure des votes valides mais illégitimes. (2) Il est impossible d'organiser une autre élection, dont la liste des électeurs éligibles est différente de la première, sans exécuter la phase d'enregistrement une autre fois, car les autorités n'ont pas la possibilité de révoquer les paramètres d'authentification qui ne sont plus éligibles.
3. *A Practical Coercion Resistant Voting Scheme Revisited (PCRVSR) [46] :* En 2013, Araùjo et Traoré ont souligné les inconvénients du système précédent [52] et ont proposé

une version révisée pour surmonter ces problèmes. Ils ont ajouté quelques modifications dans le processus électoral afin de rendre possible la vérification de l'éligibilité des votes et la révocation des paramètres d'authentification. Pour résoudre le premier problème, les autorités d'enregistrement construisent et publient une liste L_2, lors de la phase d'enregistrement, qui contient le couple $< E_T[A], ID_{votant} >$ pour chaque électeur inscrit, où T est la clé publique des autorités de comptage. Durant la phase de comptage, les autorités comparent les paramètres d'authentification valides venants dans les n-uplets de vote avec la liste L_2 et ne compte que les votes dont les paramètres d'authentification correspondent à un élément de L_2. Pour résoudre le second problème, les autorités d'enregistrement génèrent, pour chaque nouvelle élection, une nouvelle paire de clés et l'utilisent pour générer de nouveaux paramètres d'authentification. Elles calculent les nouveaux paramètres d'authentification à partir de la nouvelle clé privée et d'une liste L_1 contenant le couple $< E_R[g_1 g_3^x], ID_{votant} >$ pour chaque électeur. La liste L_1 est publiée sur le tableau d'affichage public lors de la première élection. Les nouveaux paramètres d'authentification ont la forme suivante : $(A' = (g_1 g_3^x)^{1/(y'+r')}, r', x)$, où r' est un nombre aléatoire, y' est la nouvelle clé secrète des autorités d'enregistrement et x est la même valeur secrète donnée à l'électeur lors de sa première inscription.

4. *Platform-Independent Secure Blockchain-based Voting System (PISBVS) [33] :* Il s'agit d'un système de vote électronique implémenté sur une Blockchain, basée sur un consensus de tolérance aux fautes byzantines [26]. Les auteurs prétendent que leur solution ne repose pas sur une autorité de confiance centralisée pour calculer et publier le résultat final de l'élection, alors qu'ils doivent faire confiance à un administrateur pour déchiffrer la somme des votes et publier le résultat sur la Blockchain de l'élection. Ils utilisent le cryptosystème Paillier pour chiffrer les votes avant de les publier sur la Blockchain. Ils font appel à la preuve à divulgation nulle de connaissance pour assurer l'exactitude et la cohérence des votes, et "*Short Linkable Ring Signature (SLRS)*" pour garantir la confidentialité des électeurs. Toutefois, ce protocole ne garantit pas l'éligibilité des électeurs puisqu'un électeur peut s'inscrire en fournissant son adresse e-mail, le numéro de sa carte d'identité ou une URL d'invitation avec un mot de passe et ces mécanismes ne sont pas suffisants pour vérifier l'éligibilité d'un électeur. En outre, les auteurs prétendent garantir la résistance à la coercition en partant de l'hypothèse suivante : "*on suppose que personne ne se tient derrière un électeur ou n'utilise de dispositifs numériques pour enregistrer le processus de vote. Nous ne prenons pas en considération la sécurité de l'environnement physique du vote*". Ainsi, si on se réfère à la définition de la résistance à la coercition donnée par JCJ [45], ce protocole n'est pas résistant à la coercition. Un attaquant peut voter à la place d'un électeur s'il connaît la clé secrète de ce dernier. L'électeur sous-coercition ne peut pas fournir une fausse clé secrète à l'attaquant car un vote avec une fausse clé secrète est rejeté par le contrat intelligent de vote.
5. *Efficient, Coercion-free and Universally Verifiable Blockchain-based Voting (ECFUVBV) [60] :* Il s'agit d'un protocole de vote électronique basé sur une Blockchain, prétendu être sécurisé et résistant à la coercition. Il utilise un jeton aléatoire, un dispositif inviolable qui peut être instancié avec des cartes à puce ou des dispositifs compatibles avec *Trusted Platform Module (TPM)* [61]. Les auteurs utilisent la Blockchain Bitcoin pour assurer la vérifiabilité. Sa phase de comptage a une complexité linéaire. Le protocole se déroule en plusieurs phases :
   — *Configuration :* l'autorité électorale génère ses clés publiques et privées ainsi que d'autres paramètres du système.

— *Enregistrement :* un électeur V_i interagit avec le bureau d'enregistrement R pour obtenir une paire de clés publique/privée ainsi qu'un engagement signé C_i sur les valeurs s_i, r_i, qui sont générées par le générateur aléatoire de jetons du votant V_i. Les paramètres d'authentification de l'électeur sont la version signée de l'engagement avec la clé privée de l'électeur.
— *Vote :* chaque électeur chiffre son choix v à l'aide d'une clé à usage unique K_i. Ensuite, il calcule une preuve π_i pour prouver sa connaissance de r_i en utilisant *zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs)* [62]. Il envoi un n-uplet qui a la forme suivante $< \pi_i, s_i, E_{K_i}(v) >$ via la Blockchain de l'élection.
— *Tally :* l'autorité électorale vérifie la validité de chaque bulletin de vote publié sur la Blockchain en utilisant π_i et élimine les bulletins dont les preuves ne sont pas valides. Elle élimine également les doublons en utilisant l'élément s_i. Ensuite, elle déchiffre les votes à l'aide de l'élément K_i, qui sont publiés par les électeurs avec la valeur de s_i pour faciliter la comparaison de la clé avec son vote chiffré transmis précédemment, et calcule le résultat final.

Dans cet article, les auteurs supposent que l'attaquant et les électeurs ne sont pas côte à côte. Tout ce que l'attaquant peut faire, c'est donner des instructions et demander une preuve de leur respect. Conformément à la définition de la résistance à la coercition donnée par JCJ [45], plus le fait que l'électeur ne peut voter qu'une seule fois, ce système n'est pas résistant à la coercition.

5.3 Préliminaire

Dans cette section, nous donnons un aperçu sur les principales primitives cryptographiques et technologies utilisées dans notre protocole.

5.3.1 Version modifiée du cryptosystème d'El-Gamal

LOKI Vote utilise une version modifiée du cryptosystème d'El-Gamal [3] (voir cinquième point de la sous-section 1.2.1) proposé par JCJ [45]. Dans ce schéma, la paire de clés est construite par coopération entre n autorités. Il faut t autorités parmi n pour déchiffrer un message. Comme démontré dans [45], cette version modifiée d'El-Gamal est sécurisée sous l'hypothèse de *Decision Diffie Hellman (DDH)* [6] (Voir le paragraphe Hypothèses de sécurité de la sous-section 1.2.1). Cette variante se déroule en trois étapes :

— **Génération de clés :** Soit $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q, dans lequel l'hypothèse DDH tient. On dénote la clé publique par y et elle est représentée par le triplé suivant : $y = (g_1, g_2, h)$; avec $h = g_1^{x_1} g_2^{x_2}$. La clé privée correspondante est représentée par le couple (x_1, x_2) ; avec $x_1, x_2 \in \mathbb{Z}_q$.
— **Chiffrement :** Le texte chiffré d'un message $m \in \mathbb{G}$ est représenté par le triplé suivant :

$$E_y[m] = (\alpha, \beta, \gamma) = (m \cdot h^r, g_1^r, g_2^r)$$

Où r est un nombre aléatoire de $\mathbb{Z}_q$.
— **Déchiffrement :** m est obtenu à partir de (α, β, γ) en utilisant la formule suivante :

$$m = \alpha / (\beta^{x_1} \gamma^{x_2})$$

5.3.2 Schéma de signature de groupe de Boneh, Boyen et Shacham

Dans leur article [55], Boneh, Boyen et Shacham (BBS) ont présenté un schéma de signature de groupe. Sa sécurité repose sur l'hypothèse de *q-Strong Diffie-Hellman (q-SDH)* [7] (Voir le paragraphe Hypothèses de sécurité de la sous-section 1.2.1).

L'approche que nous allons présenter dans ce chapitre utilise le schéma de signature de groupe de BBS, présenté dans la section 8 de l'article [55], comme paramètres d'authentification anonymes pour les votants éligibles. Ce schéma de signature peut être décrit comme suit : Soit $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q, dans lequel l'hypothèse de *Decisional Diffie–Hellman (DDH)* tient, g_1, $g_2 \in \mathbb{G}$ sont deux générateurs aléatoires, y est une clé secrète, et $r, x \in \mathbb{Z}_q$ sont deux nombres aléatoires. La signature est représentée par le triplé (A, r, x) où $A = (g_1 g_2^x)^{1/(y+r)}$.

5.3.3 La Blockchain Loki

Loki [1] est une plate-forme basée sur la Blockchain Monero [2] (Voir la sous-section1.3.5). Elle propose d'importantes modifications au niveau du code source de Monero afin de garantir un haut degré de confidentialité et de fournir un modèle pour des transactions anonymes et une communication décentralisée. Les principaux inconvénients de la Blockchain Monero sont la quantité importante de bande passante et d'espace disque que ses nœuds opérateurs ont besoin, plus le fait qu'ils ne sont pas récompensés pour leur travail. Pour remédier à ce problème, Loki propose un nouveau système de récompense qui fournit des incitations économiques aux nœuds opérateurs, appelé *nœuds de services*. Ces nœuds de services assurent la confidentialité et la sécurité du réseau. Cette technologie a été proposée pour promouvoir la neutralité de l'internet, l'anonymat numérique et une suite d'outils résistant à la censure, permettant aux gens de communiquer d'une manière privée et sécurisée. C'est pour ces atouts que Loki peut être utilisée dans divers domaines, en particulier lorsque nous devons garantir un niveau élevé d'anonymat, comme dans le cas des systèmes de vote électronique.

Étant basée sur la Blockchain Monero, Loki fait appel à plusieurs primitives cryptographiques, à savoir *la signature en anneau* [29] pour masquer l'historique des transactions, *l'adresse furtive* [28] pour couper le lien entre la clé publique réelle du destinataire et ses transactions, et *les transactions confidentielles en anneau* [27] pour masquer le montant des transactions. Cette plate-forme, basée sur la technologie Blockchain, utilise également l'algorithme de consensus de preuve de travail pour valider les transactions et construire des blocs. Elle opte pour un mode différent de distribution de récompenses du bloc : 45% des récompenses sont réservées aux mineurs, 50% aux nœuds de service et 5% aux opérations de gouvernance. Le rôle principal des nœuds de service est de faire fonctionner le protocole de routage anonyme à faible latence (*Low Latency Anonymous Routing Protocol LLARP*) [3], qui est un mixnet anonyme, et de former le Lokinet, qui est un réseau entièrement décentralisé ne reposant sur aucune autorité de confiance. Le protocole de routage anonyme à faible latence (LLARP) est une couche de routage privée créée par Loki. Il s'agit d'un hybride entre TOR (*The Onion Routing*) [4] et I2P (*Invisible Internet Protocol*) [5]. Il corrige les vulnérabilités des protocoles TOR et I2P et offre un niveau de

1. https://loki.network/wp-content/uploads/2018/10/LokiWhitepaperV3_1.pdf
2. https://www.allcryptowhitepapers.com/wp-content/uploads/2018/05/monero-whitepaper.pdf
3. https://github.com/loki-project/loki-network
4. https://www.torproject.org/
5. https://geti2p.net/en/

sécurité et de distribution plus élevé que tout autre protocole de routage existant. Pour mieux comprendre le fonctionnement du LLARP, nous donnons un aperçu sur les protocoles TOR et I2P. Les avantages et les inconvénients de chaque protocole sont résumés dans le tableau 5.1.

	TOR	**I2P**
Avantages	+ Fournit un réseau anonyme, + Préserve la vie privée sur internet, + Meilleure performance pour contourner les pare-feu de l'état, + Garantie un haut niveau de résistance à la censure.	+ Fournit un réseau anonyme, + Utilise une table de hachage distribuée au lieu des autorités de l'annuaire, + Permet l'acheminement des trafics TCP et UDP.
Inconvénients	- Il s'agit d'un réseau hiérarchique, - S'appuie sur un groupe d'autorités d'annuaire (serveurs centralisés), - N'autorise que le trafic TCP, -Irrésistant aux attaques *Sybil* [63, 64].	- Problèmes de performances et manque de bande passante, - Les tunnels sont de courte durée, - Irrésistant aux attaques *Sybil* [63, 65].

TABLEAU 5.1 – Avantages et inconvénients de TOR et I2P

TOR et I2P sont tous deux gérés par des volontaires, ce qui peut poser des problèmes de sécurité, de fiabilité et de performance. En fait, un réseau construit à partir d'incitations financières peut atteindre une plus grande résilience contre les attaques, tout en fournissant un service plus fiable. C'est ce que propose le LLARP en utilisant une table de hachage distribuée (DHT) basée sur la technologie Blockchain. Cette DHT permet aux nœuds de service d'agir comme routeurs dans le réseau et ils sont rémunérés pour leur travail. Le LLARP opte également pour un routage basé sur la commutation de paquets plutôt que sur un routage par tunnel afin de permettre un meilleur équilibrage de charges et une meilleure redondance dans le réseau. Pour éviter les attaques *Sybil*, LLARP permet uniquement aux nœuds de service d'acheminer les paquets, et ils sont récompensés pour leur honnêteté.

5.4 Description du protocole proposé : LOKI Vote

Nous proposons un système de vote électronique en ligne résistant à la coercition qui utilise la technologie Blockchain et qui est conçu pour être implémenté sur Loki. Appelé *LOKI Vote*, le protocole offre une vérifiabilité de bout en bout en utilisant un tableau d'affichage public basé sur la technologie Blockchain pour afficher toutes les valeurs publiques et offrir une vue permanente à tous les électeurs. Dans cette section, nous présentons les différentes entités impliquées dans LOKI Vote ainsi que ses différentes phases.

5.4.1 Entités du protocole

Notre protocole implique trois entités principales :

— *Autorités d'enregistrement (AEs) :* Elles coopèrent et génèrent une nouvelle paire de clés (R, R') pour chaque nouvelle élection, génèrent et publient les paramètres électoraux sur la Blockchain pendant la phase de configuration, vérifient l'éligibilité de

chaque personne souhaitant s'inscrire à l'élection, pendant la phase d'enregistrement et de ne fournir que les électeurs éligibles par des paramètres d'authentification anonymes, qui sont générés par coopération entre toutes les autorités d'enregistrement. En outre, elles coopèrent pour établir et publier deux listes L_1 et L_2 qui serviront plus tard, respectivement, à la révocation des paramètres d'authentification et à la vérification de l'éligibilité des votes. Enfin, elles aident les autorités de comptage à vérifier la validité des votes pendant la phase de comptage.

— *Autorités de comptage (ACs) :* Elles coopèrent et génèrent une paire de clés $(\mathscr{T}, \mathscr{T}')$ pendant la phase de configuration, récupèrent les n-uplets de vote de la Blockchain de l'élection, vérifient, déchiffrent et calculent les votes valides et éligibles pendant la phase de comptage. Enfin, elles publient le résultat final sur la Blockchain.

— *Électeurs éligibles (V) :* Après s'être enregistré à une élection, chaque électeur éligible (V_i) dispose de ses paramètres d'authentification anonymes et valides et peut générer un nombre illimité de faux paramètres pour les utiliser lorsqu'il est sous coercition. Il a le droit de voter plus d'une fois avant la fin de la phase de vote et seul son dernier vote valide est comptabilisé.

Chaque entité de notre protocole a un accès en lecture et en écriture à la Blockchain de l'élection, qui est considérée comme un tableau d'affichage public et une urne. De plus, les observateurs et les organisateurs des élections ont le droit d'accéder à la Blockchain et de superviser l'élection afin de s'assurer de la régularité du processus électoral.

5.4.2 Phases du protocole

Notre protocole inclue quatre phases : configuration, enregistrement, vote et comptage. Il y a deux façons d'effectuer les phases de configuration et d'enregistrement, ça dépend si c'est la première fois que le protocole est exécuté (la première élection) ou plus.

Phase de configuration

Configuration de la première élection : Cette phase est décrite par la Figure 5.1.

1. Les AEs commencent par générer les paramètres de l'élection suivants et les publient sur la Blockchain : $\mathbb{G}$ un groupe cyclique d'ordre un nombre premier q, dans lequel se tient le problème de *Decisional Diffie Hellman*; g_1, g_2, g_3 et $o \in \mathbb{G}$ quatre générateurs aléatoires. Elles coopèrent et génèrent leur paire de clés $((R, R')$, où $R = g_3^y$ est la clé publique et $R' = y$ est la clé privée). Une autre paire de clés $(\mathscr{R}, \mathscr{R}')$ est également générée par coopération entre toutes les AEs en utilisant la version modifiée du cryptosystème d'El-Gamal [45]. Enfin, elles publient les parties publiques de ces paramètres sur la Blockchain de l'élection.
2. Les ACs coopèrent et génèrent une paire de clés en utilisant le cryptosystème à seuil d'El-Gamal, noté $(\mathscr{T}, \mathscr{T}')$, où $\mathscr{T} = (g_1, g_2, h = g_1^{x_1} g_2^{x_2})$ est la partie publique et $\mathscr{T}' = (x_1, x_2)$ est la partie secrète. Elles publient leur clé publique sur la Blockchain.

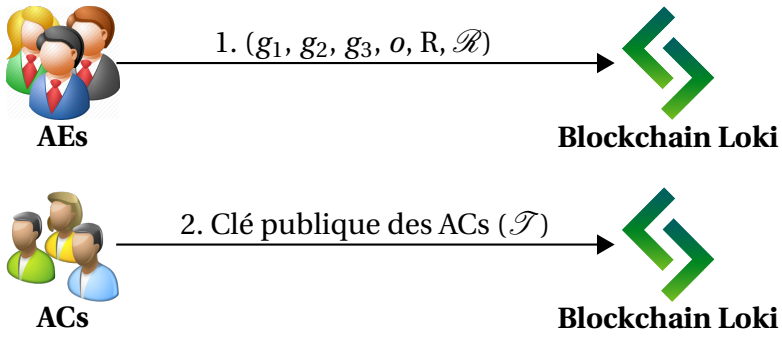


FIGURE 5.1 – Phase de configuration, 1ère élection.

Configuration de la deuxième (ou plus) élection : Pour chaque nouvelle élection, les AEs créent un nouveau générateur aléatoire $o' \in \mathbb{G}$. Si nous n'avons pas besoin de révoquer les anciens paramètres d'authentification, les AEs publient les mêmes paramètres que la première élection, en remplaçant o par o' (Figure 5.2).

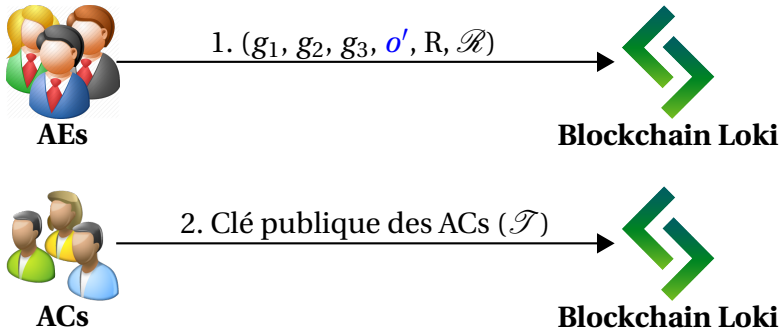


FIGURE 5.2 – Phase de configuration, 2éme (ou plus) élection, sans révocation.

Sinon, elles génèrent une nouvelle paire de clés (R_1, R'_1), où $R_1 = g_3^{y_1}$ et $R'_1 = y_1$ et publient tous les paramètres publics sur la Blockchain (Figure 5.3). La nouvelle paire de clés est utilisée pour la révocation des paramètres d'authentification.

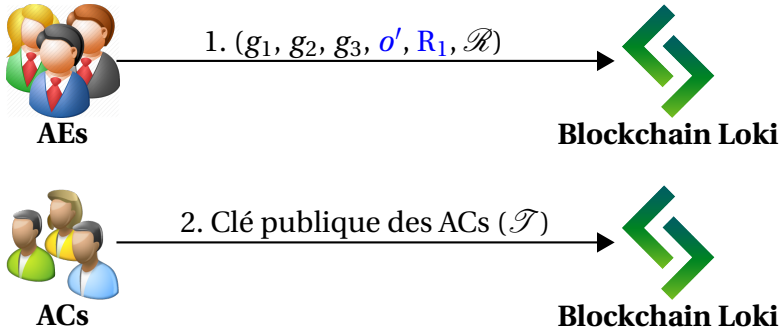


FIGURE 5.3 – Phase de configuration, 2ème (ou plus) élection, avec révocation.

Phase d'enregistrement

Enregistrement pour la première élection : Toute personne ayant le droit de voter et souhaitant le faire, se rend au bureau de vote le plus proche et fournit sa carte d'iden-

tité aux autorités d'enregistrement (AEs). Ces autorités vérifient son éligibilité et lui fournissent des paramètres d'authentification valides et anonymes, s'il est éligible pour participer à l'élection. Dans le cas contraire, la phase d'enregistrement échoue. La Figure 5.4 illustre une phase d'enregistrement réussie. Les paramètres d'authentification sont calculés par coopération entre les autorités d'enregistrement et sont utilisés par l'électeur pour voter. Pour calculer ces paramètres, les AEs génèrent deux nombres aléatoires $r, x \in \mathbb{Z}_q$, utilisent leur clé privée partagée y et calculent $A = (g_1 g_3^x)^{1/(y+r)}$. Les paramètres d'authentification d'un votant éligible sont représentés par le triplé (A, r, x), où x est la partie secrète. Après avoir enregistré tous les électeurs éligibles, les AEs coopèrent et génèrent deux listes :

— $L_1 =< E_{\mathscr{R}}[g_1 g_3^x], ID_{votant} >$ contient, pour chaque électeur, le chiffré de $(g_1 g_3^x)$ en utilisant leur clé publique $\mathscr{R}$ et l'identifiant unique de l'électeur correspondant ID_{votant}. Cette liste servira plus tard pour la révocation des paramètres d'authentification.

— $L2 =< E_{\mathscr{T}}[A], ID_{votant} >$ contient, pour chaque électeur, le chiffré de A à l'aide de la clé publique des ACs $\mathscr{T}$ et l'identifiant unique de l'électeur correspondant ID_{votant}. Cette liste sert à vérifier l'éligibilité des paramètres d'authentification.

Enfin, les AEs publient L_1 et L_2 sur la Blockchain de l'élection.

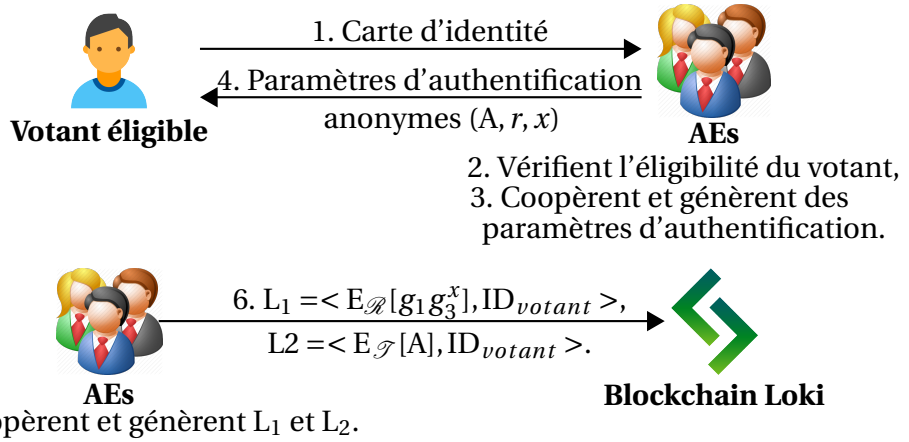
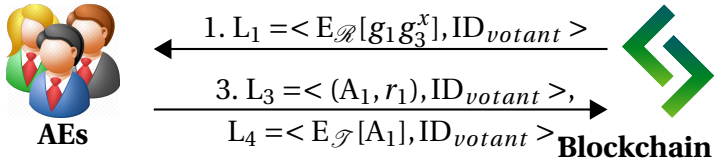


FIGURE 5.4 – Phase d'enregistrement, 1ère élection.

Enregistrement pour la deuxième (ou plus) élection : Pour chaque nouvelle élection, et s'il y a des paramètres d'authentification à révoquer, les AEs doivent mettre à jour les paramètres d'authentification des électeurs qui ont encore le droit de voter. À partir de la liste L_1 et de leur nouvelle clé privée partagée y_1, elles calculent les nouveaux paramètres d'authentification anonymes. En se basant sur les valeurs des ID_{votant}, les AEs identifient les votants qui sont encore éligibles pour participer à la nouvelle élection. Pour chacun de ces électeurs, les AEs choisissent, au hasard, $r_1 \in \mathbb{Z}_q$ et calculent ses nouveaux paramètres d'authentification valides $\sigma_1 = (A_1, r_1, x)$, où $A_1 = (g_1 g_3^x)^{1/(y_1+r_1)}$, et x est la même valeur secrète donnée à l'électeur lors de sa première inscription. À la fin de cette phase, les AEs publient sur la Blockchain de l'élection les listes $L_3 =< (A_1, r_1), ID_{votant} >$ et $L_4 =< E_{\mathscr{T}}[A_1], ID_{votant} >$. Cette phase est illustrée par la Figure 5.5.

Phase de vote

Pour voter, chaque électeur éligible construit un n-uplet de vote qui contient son vote chiffré,ses paramètres d'authentification chiffrés et un ensemble de preuves à divulgation



2. Coopèrent et génèrent L_3 et L_4.

FIGURE 5.5 – Phase d'enregistrement, 2ème (ou plus) élection

nulle de connaissance non interactives (NI-ZKP) qui prouvent l'exactitude du n-uplet. Il se présente sous la forme suivante :

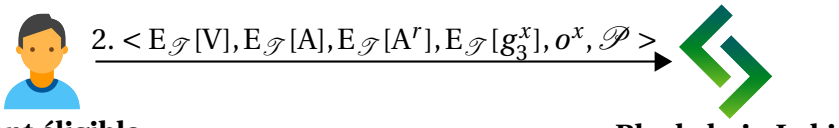
$$< E_{\mathscr{T}}[V], E_{\mathscr{T}}[A], E_{\mathscr{T}}[A^r], E_{\mathscr{T}}[g_3^x], o^x, \mathscr{P} >$$

Où $\mathscr{T}$ est la clé publique des ACs, V est le choix du votant, A, r et x sont les paramètres d'authentification du votant et $\mathscr{P}$ est composé d'un ensemble de NI-ZKP. Ces preuves sont construites en utilisant des techniques standards telle que [42] et contiennent :

— P_1 : Preuve de validité du vote chiffré V,
— P_2 : Preuve de connaissance du texte en clair relatif à $E_{\mathscr{T}}[A]$,
— P_3 : Preuve de connaissance du texte en clair relatif à $E_{\mathscr{T}}[A^r]$,
— P_4 : Preuve de connaissance du texte en clair relatif à $E_{\mathscr{T}}[g_3^x]$,
— P_5 : Preuve de connaissance de la valeur de A et qu'elle est différente de 1,
— P_6 : Preuve de connaissance du logarithme discret du o^x dans la base o et son égalité avec le logarithme discret du texte en clair relatif à $E_{\mathscr{T}}[g_3^x]$ dans la base g_3.

Cette phase est illustrée par la Figure 5.6. Le votant a le droit de voter plusieurs fois avant la fin de la phase de vote et seul son dernier vote valide est comptabilisé. Lorsqu'il est sous coercition, l'électeur génère $x' \neq x$ et vote en utilisant la valeur de x' au lieu de x.

Si ce n'est pas la première élection, l'électeur utilise o' au lieu de o ainsi que ses nouveaux paramètres d'authentification qu'il a reçus de la part des AEs pendant la phase d'enregistrement.



1. Génère et construit le n-uplet de vote.

FIGURE 5.6 – Phase de vote

Phase de comptage

À la fin de la phase de vote, les autorités de comptage récupèrent tous les n-uplets de vote de notre Blockchain électoral et procèdent au décompte. Elles commencent par vérifier la validité de chaque n-uplet et éliminent ceux dont les preuves ne sont pas valides. Ensuite, elles éliminent les doublons en utilisant l'attribut o^x (ou o'^x si ce n'est pas la première élection) inclus dans chaque n-uplet, en utilisant une table de hachage. Comme tous les n-uplets de vote ont été acheminés à travers la Blockchain Loki, ils ont été passés par le réseau de mixage LLARP (voir la sous section 5.3.3 pour plus de détails). À ce stade, chaque n-uplet de vote a la forme suivante :

$$< E'_{\mathscr{T}}[V], E'_{\mathscr{T}}[A], E'_{\mathscr{T}}[A^r], E'_{\mathscr{T}}[g_3^x] >$$

Où $E'_{\mathscr{T}}[.]$ et le chiffré de $E_{\mathscr{T}}[.]$.
En utilisant les trois derniers éléments de chaque n-uplet, les ACs coopèrent avec les AEs et vérifient la validité des paramètres d'authentification anonymes. Ils procèdent comme suit :

— En utilisant leur clé secrète partagée y, les AEs coopèrent et calculent $E'_{\mathscr{T}}[A]^y$ qui est égal à $E'_{\mathscr{T}}[A^y]$ grâce à la propriété homomorphisme d'El-Gamal. Ensuite, elles effectuent la multiplication suivante :

$$E'_{\mathscr{T}}[A^y] \cdot E'_{\mathscr{T}}[A^r] = E'_{\mathscr{T}}[A^y \cdot A^r] = E'_{\mathscr{T}}[A^{y+r}]$$

La première égalité est obtenue en utilisant la propriété d'homomorphisme du cryptosystème d'El-Gamal.

— Les ACs coopèrent et effectuent la multiplication suivante, dans laquelle ils utilisent également la propriété d'homomorphisme d'El-Gamal :

$$E'_{\mathscr{T}}[A^{y+r}] \cdot E'_{\mathscr{T}}[g_1]^{-1} \cdot E'_{\mathscr{T}}[g_3^x]^{-1} = E'_{\mathscr{T}}[A^{y+r} \cdot g_1^{-1} \cdot g_3^{-x}]$$

Le résultat $E'_{\mathscr{T}}[A^{y+r} \cdot g_1^{-1} \cdot g_3^{-x}]$ est noté C. Ensuite, les ACs exécutent le PET pour déterminer si C est un chiffrement de 1 ou non. Si c'est le cas, les paramètres d'authentification sont jugés valides et le n-uplet de vote correspondant passe à l'étape suivante. En effet, des paramètres d'authentification valides ont la forme suivante $\sigma = (A, r, x)$ où $A = (g_1 \cdot g_3^x)^{1/(y+r)}$ donc nous avons $A^{y+r} = g_1 \cdot g_3^x$ et par conséquent $A^{y+r} \cdot g_1^{-1} \cdot g_3^{-x} = 1$.
Dans le cas contraire, les paramètres d'authentification sont jugés invalides et le n-uplet de vote correspondant est rejeté.

L'étape suivante consiste à vérifier l'éligibilité des votes en utilisant l'élément $E'_{\mathscr{T}}[A]$, inclus dans chaque n-uplet de vote, et la liste L_2. On rappelle que $L_2 =< E_{\mathscr{T}}[A], ID_{votant} >$ a été publié sur la Blockchain de l'élection par les AEs lors de la phase d'enregistrement. À cette étape, et après avoir passé par le réseau de mixage LLARP, nous obtenons $L'_2 =< E'_{\mathscr{T}}[A], ID'_{voter} >$. En utilisant une table de hachage, les ACs comparent les $E'_{\mathscr{T}}[A]$, venant dans chaque n-uplet de vote, avec $E'_{\mathscr{T}}[A]$, figurant sur la liste L'_2, et ne maintiennent que les n-uplets qui correspondent à un élément de L'_2. Enfin, les ACs coopèrent et déchiffrent tous les votes de la liste retenue, en utilisant leur clé secrète partagée $\mathscr{T}'$, et calculent le résultat final de l'élection.

Nous mentionnons que si ce n'est pas la première élection, y est remplacée par y_1, A et r sont remplacées, respectivement, par A_1 et r_1 et L_2 par L_4.

5.5 Évaluation de la sécurité de LOKI Vote

Dans cette section, nous discutons, de manière informelle puis formelle, la sécurité du système que nous proposons.

5.5.1 Évaluation informelle

Nous commençons par évaluer notre protocole par rapport à la liste des exigences de sécurité présentée dans le Chapitre 1. Nous reprenons cette évaluation dans le Tableau 5.3.

— *Éligibilité :* LOKI Vote comprend une phase d'enregistrement, au cours de laquelle les autorités d'enregistrement vérifient l'éligibilité de chaque électeur et ne fournissent que les personnes éligibles par des paramètres d'authentification. À la fin

de cette phase, les AEs publient la liste L_2 de tous les électeurs inscrits. Ainsi, tout le monde peut vérifier la validité de cette liste. En outre, pendant la phase de comptage, les ACs ne comptent que les votes qui correspondent à un élément de la liste L_2.

— *Pas de résultat partiel :* Tous les votes sont chiffrés, à l'aide de la clé publique des ACs $\mathscr{T}$, avant d'être envoyés via la Blockchain. Ainsi, personne, sauf les ACs, n'a la possibilité de déchiffrer les votes et d'obtenir des résultats partiels avant le décompte officiel. Nous mentionnons ici que la clé privée de déchiffrement est construite par coopération entre toutes les ACs. Nous devons donc faire confiance à une seule AC pour garantir qu'il y aura pas de résultat partiel.

— *Robustesse :* Le protocole que nous proposons résiste au mauvais comportement des électeurs malveillants grâce à l'utilisation des preuves incluses dans chaque n-uplet de vote. En effet, les ACs écartent du décompte final tous les n-uplets de vote comportant des preuves invalides.

— *Intégrité :* Le fait d'émettre et de stocker les votes et les autres données de vote dans la Blockchain, les protège contre toute modification ou suppression grâce à la propriété d'immutabilité de la technologie Blockchain.

— *Vérifiabilité universelle :* Cette propriété est assurée par le fait d'utiliser la technologie Blockchain comme un tableau d'affichage public. À l'exception de la phase d'enregistrement, toutes les phases du protocole utilisent la Blockchain. Ainsi, les votants, les organisateurs des élections, les observateurs et toute partie intéressée ont la possibilité d'observer et de vérifier l'exactitude de chaque étape ainsi que le décompte final de l'élection.

— *Anonymat :* Cette propriété est assurée par la plate-forme Loki, qui est basée sur la Blockchain Monero. Monero se caractérise par l'anonymat de ses transactions puisqu'il utilise des primitives cryptographiques telles que les transactions confidentielles en anneau et la signature en anneau. Ainsi, on ne peut pas établir un lien entre une transaction et son émetteur. Par conséquent, on ne peut pas faire le lien entre un votant à son vote.

— *Sans-reçu :* À partir de toutes les données publiées sur la Blockchain de l'élection, le votant ne peut pas construire un reçu qui reflète son vote.

— *Résistance à la coercition :* LOKI Vote s'inspire du système [46], dont la résistance à la coercition est formellement prouvée. Cette propriété est assurée grâce à l'utilisation du schéma de signature de BBS, $\sigma = (A, r, x)$, comme paramètres d'authentification anonymes pour les votants éligibles. Lorsqu'ils sont sous coercition, les électeurs génèrent une valeur aléatoire x' au lieu de x et donne à l'attaquant $\sigma' = (A, r, x')$. Comme l'électeur a le droit de voter plus d'une fois, il a la possibilité de voter une autre fois lorsqu'il est seul et utilise σ. L'attaquant n'a aucun moyen de vérifier si l'électeur a obéi à ses instructions ou non.

— *Voter et quitter :* LOKI Vote n'a pas besoin que l'électeur attende la fin de la phase de vote ni qu'il déclenche le décompte. Il peut tout simplement voter et quitter le système de vote.

— *Politique de vote :* Si on n'a pas encore atteint la fin de la phase de vote et qu'un votant éligible souhaite changer son avis après avoir voter, il a le droit de le faire. On mentionne qu'on ne prend en considération que son dernier vote valide.

	CREE	TPSCREE	PCRVSR	PISBVS	ECFUVBV
Éligibilité	X	X	✓	X	✓
Pas de résultat partiel	✓	✓	✓	✓	✓
Robustesse	X	X	X	✓	✓
Intégrité	X	X	X	✓	✓
Vérifiabilité universelle	✓	✓	✓	✓	✓
Anonymat	✓	✓	✓	✓	✓
Sans-reçu	✓	✓	✓	✓	✓
Résistance à la coercition	✓	✓	✓	X	X
Voter et quitter	✓	✓	✓	✓	✓
Politique de vote	Multiple	Multiple	Multiple	Un seul vote	Multiple

TABLEAU 5.2 – Évaluation de la sécurité de CREE, TPSCREE, PCRVSR, PISBVS et ECFUVBV

	LOKI Vote
Éligibilité	✓
Pas de résultat partiel	✓
Robustesse	✓
Intégrité	✓
Vérifiabilité universelle	✓
Anonymat	✓
Sans-reçu	✓
Résistance à la coercition	✓
Voter et quitter	✓
Politique de vote	Multiple

TABLEAU 5.3 – Évaluation de la sécurité de *LOKI Vote*

5.5.2 Évaluation formelle

Dans cette partie, nous effectuons une analyse de sécurité automatisée en utilisant l'outil de vérification ProVerif et le langage de modélisation Applied Pi-Calculus.

Le code est donné dans l'Annexe C. On définit les requêtes suivantes pour prouver *le secret de vote*, *l'authentification des électeurs* et *la confidentialité des votes* et on donne les résultats de l'exécution des codes, ainsi que le temps que prend ProVerif pour prouver ces propriétés dans le Tableau 5.4.

— *Vérification du secret de vote :* Afin d'intercepter la valeur d'un vote donné, un attaquant doit intercepter la valeur du paramètre $Vote$. Nous utilisons donc la requête suivante :

```
query attacker(Vote).
```

— *Vérification de l'authentification des électeurs :* Cette propriété est vérifiée à l'aide *d'assertions de correspondance*. Le protocole vise à garantir que les ACs vérifient l'éligibilité de tous les électeurs en vérifiant la validité de leurs paramètres d'authentification. Par conséquent, nous définissons les requêtes suivantes :

```
event ValidCredential.
event CredentialVerification.
query event(ValidCredential)==>event(CredentialVerification).
```

— *Vérification de la confidentialité des votes :* Pour exprimer la confidentialité des votes, nous prouvons la propriété d'équivalence d'observation entre deux instances de notre processus qui ne diffèrent que par le choix des votes. Pour ce faire, nous utilisons `choice[V1,V2]` pour représenter les termes qui diffèrent entre les deux instances.

Propriétés	Résultat	Temps d'exécution
Secret du vote	Prouvée	0.007 s
Authentification des votants	Prouvée	0.009 s
Confidentialité du vote	Prouvée	0.089 s

TABLEAU 5.4 – Résultats et temps d'exécution des codes ProVerif de LOKI Vote.

5.6 Conclusion

Dans ce chapitre, nous avons proposé un protocole de vote électronique en ligne vérifiable de bout en bout, résistant à la coercition, sécurisé et basé sur la technologie Blockchain. LOKI Vote est basé sur le travail d'Araùjo et Traoré de 2013 et utilise la plate-forme Loki. Il fait appel à plusieurs primitives cryptographiques à savoir NI-ZKP, une version modifiée du cryptosystème El-Gamal, la signature de BBS et le réseau de mixage LLARP. Il a une complexité linéaire qui le rend pratique pour les élections à grande échelle. Nous avons également prouvé, de manière formelle en utilisant ProVerif, la sécurité de notre protocole. L'implémentation et l'évaluation des performances de ce système font parties de nos présents axes de recherches.

Chapitre 6

Conclusion et Perspectives

Nos travaux de recherches consistaient à étudier la technologie Blockchain et l'appliquer dans le domaine de vote électronique en ligne, afin d'atteindre un niveau de sécurité satisfaisant les votants et les organisateurs des élections.

Nous avons commencé par étudier l'existant et cerner les problématiques actuelles. Ensuite, nous avons proposé un protocole de vote en ligne vérifiable de bout-en-bout grâce à l'utilisation de la Blockchain Ethereum. Intitulé "Verify-Your-Vote", ce protocole fait appel à une variété de primitives cryptographiques à savoir la cryptographie à base de courbes elliptique, la cryptographie à base d'identité, les pairings et le cryptosystème de Paillier. Nous avons implémenté ce protocole et évalué ces performances en terme de temps, de coût et de nombre de votants et de candidats pouvant être supportés.

Le principal défaut de ce protocole est lié à l'utilisation de la Blockchain Ethereum. En fait, les mineurs peuvent invalider l'élection s'ils modifient les transactions contenant les votes avant de les stocker sur la Blockchain de l'élection. Pour remédier à ce problème inhérent, nous avons proposé, dans le chapitre suivant, un protocole de vote en ligne sécurisé et basé sur une Blockchain privée qui est Hyperledger Fabric. Intitulé "DABSTERS", ce protocole utilise un nouvel algorithme de consensus basé sur la BFT et les signatures en aveugle pour assurer l'anonymat des votants.

Ces deux protocoles proposés offrent un niveau élevé de sécurité. En effet, ils respectent les propriétés de sécurité suivantes : éligibilité, pas de résultat partiel, robustesse, vérifiabilité individuelle et universelle, anonymat et sans reçu. Cependant, ils n'offrent aucun mécanisme permettant d'assurer la résistance à la coercition et empêcher l'achat des votes. Pour résoudre ce problème, nous avons proposé, dans le cinquième chapitre, un protocole de vote électronique en ligne qui répond à toute les propriétés de sécurité, énumérés ci-dessus, et qui assure la résistance à la coercition. Appelé "LOKI Vote", ce protocole est basé sur la Blockchain Loki et une variété de primitives cryptographiques, à savoir le schéma de signature de Boneh, Boyen et Shacham, une version modifiée du cryptosystème d'El-Gamal, les signatures en anneau, les transactions confidentielles en anneau, les adresses furtives ainsi qu'une nouvelle approche de réseau de mixage (LLARP).

En guise de perspective, nous comptons implémenter et évaluer les performances des deux approches DABSTERS et LOKI Vote. Une comparaison entre nos deux protocoles Verify-Your-Vote et DABSTERS serait aussi très utiles pour nos futurs travaux de recherche.

Bibliographie

[1] Madise, Ü., Martens, T. : E-voting in estonia 2005. the first practice of country-wide binding internet voting in the world. In Krimmer, R., ed. : Electronic Voting 2006 : 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC, August, 2nd - 4th, 2006 in Castle Hofen, Bregenz, Austria. Volume P-86 of LNI., GI (2006) 15–26 5

[2] Achieng, M., Ruhode, E. : The adoption and challenges of electronic voting technologies within the south african context. CoRR **abs/1312.2406** (2013) 5

[3] Gamal, T.E. : A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory **31**(4) (1985) 469–472 6, 9, 31, 77

[4] Diffie, W., Hellman, M.E. : Special feature exhaustive cryptanalysis of the NBS data encryption standard. Computer **10**(6) (1977) 74–84 6

[5] Camenisch, J., Shoup, V. : Practical verifiable encryption and decryption of discrete logarithms. In Boneh, D., ed. : Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings. Volume 2729 of Lecture Notes in Computer Science., Springer (2003) 126–144 7

[6] Boneh, D. : The decision diffie-hellman problem. In Buhler, J., ed. : Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings. Volume 1423 of Lecture Notes in Computer Science., Springer (1998) 48–63 7, 77

[7] Boneh, D., Boyen, X. : Short signatures without random oracles. In Cachin, C., Camenisch, J., eds. : Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. Volume 3027 of Lecture Notes in Computer Science., Springer (2004) 56–73 7, 78

[8] Koblitz, N. : Constructing elliptic curve cryptosystems in characteristic 2. In Menezes, A., Vanstone, S.A., eds. : Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings. Volume 537 of Lecture Notes in Computer Science., Springer (1990) 156–167 7, 39

[9] Blakley, G.R., Chaum, D., eds. : Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Volume 196 of Lecture Notes in Computer Science., Springer (1985) 7, 90

[10] Koblitz, N. : Elliptic curve cryptosystems. Mathematics of Computation **48**(177) (January 1987) 203–209 7

[11] Boneh, D. : Pairing-based cryptography : Past, present, and future. In Wang, X., Sako, K., eds. : Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings. Volume 7658 of Lecture Notes in Computer Science., Springer (2012) 1 8, 39

[12] Boneh, D., Franklin, M.K. : Identity-based encryption from the weil pairing. [66] 213–229 8, 39, 68

[13] Shamir, A. : Identity-based cryptosystems and signature schemes. [9] 47–53 8, 39, 68

[14] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T. : Secure distributed key generation for discrete-log based cryptosystems. In Stern, J., ed. : Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 295–310 8

[15] Paillier, P. : Public-key cryptosystems based on composite degree residuosity classes. In Stern, J., ed. : Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding. Volume 1592 of Lecture Notes in Computer Science., Springer (1999) 223–238 9, 39

[16] Goldwasser, S., Micali, S., Rackoff, C. : The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1) (1989) 186–208 9

[17] Blum, M., Feldman, P., Micali, S. : Non-interactive zero-knowledge and its applications (extended abstract). In Simon, J., ed. : Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, ACM (1988) 103–112 9

[18] Blum, M., Santis, A.D., Micali, S., Persiano, G. : Noninteractive zero-knowledge. SIAM J. Comput. **20**(6) (1991) 1084–1118 9

[19] Dreier, J., Lafourcade, P., Lakhnech, Y. : A formal taxonomy of privacy in voting protocols. In : Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012, IEEE (2012) 6710–6715 10

[20] Abadi, M., Blanchet, B., Fournet, C. : The applied pi calculus : Mobile values, new names, and secure communication. J. ACM **65**(1) (2018) 1 :1–1 :41 11

[21] Abadi, M., Fournet, C. : Mobile values, new names, and secure communication. In Hankin, C., Schmidt, D., eds. : Conference Record of POPL 2001 : The 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, London, UK, January 17-19, 2001, ACM (2001) 104–115 11

[22] Blanchet, B. : An efficient cryptographic protocol verifier based on prolog rules. In : 14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada, IEEE Computer Society (2001) 82–96 11

[23] Dolev, D., Yao, A.C. : On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2) (1983) 198–207 11

[24] Turek, J., Shasha, D.E. : The many faces of consensus in distributed systems. Computer **25**(6) (1992) 8–17 17

[25] Merkle, R.C. : Protocols for public key cryptosystems. In : Proceedings of the 1980 IEEE Symposium on Security and Privacy, Oakland, California, USA, April 14-16, 1980, IEEE Computer Society (1980) 122–134 17

[26] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., Caro, A.D., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al. : Hyperledger fabric : a distributed operating system for permissioned blockchains. In : Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018, ACM (2018) 30 :1–30 :15 19, 60, 61, 76

[27] Noether, S., Mackenzie, A. : Ring confidential transactions. Ledger **1** (2016) 1–18 20, 78

[28] Diffie, W., Hellman, M.E. : New directions in cryptography. IEEE Trans. Information Theory **22**(6) (1976) 644–654 20, 78

[29] Rivest, R.L., Shamir, A., Tauman, Y. : How to leak a secret : Theory and applications of ring signatures. In Goldreich, O., Rosenberg, A.L., Selman, A.L., eds. : Theoretical Computer Science, Essays in Memory of Shimon Even. Volume 3895 of Lecture Notes in Computer Science., Springer (2006) 164–186 20, 78

[30] Dagher, G.G., Marella, P.B., Milojkovic, M., Mohler, J. : Broncovote : Secure voting system using ethereum's blockchain. In Mori, P., Furnell, S., Camp, O., eds. : Proceedings of the 4th International Conference on Information Systems Security and Privacy, ICISSP 2018, Funchal, Madeira - Portugal, January 22-24, 2018., SciTePress (2018) 96–107 23

[31] Hjalmarsson, F.P., Hreioarsson, G.K., Hamdaqa, M., Hjalmtysson, G. : Blockchain-based e-voting system. In : 11th IEEE International Conference on Cloud Computing, CLOUD 2018, San Francisco, CA, USA, July 2-7, 2018, IEEE Computer Society (2018) 983–986 24

[32] McCorry, P., Shahandashti, S.F., Hao, F. : A smart contract for boardroom voting with maximum voter privacy. In Kiayias, A., ed. : Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers. Volume 10322 of Lecture Notes in Computer Science., Springer (2017) 357–375 26

[33] Yu, B., Liu, J.K., Sakzad, A., Nepal, S., Steinfeld, R., Rimba, P., Au, M.H. : Platform-independent secure blockchain-based voting system. In Chen, L., Manulis, M., Schneider, S., eds. : Information Security - 21st International Conference, ISC 2018, Guildford, UK, September 9-12, 2018, Proceedings. Volume 11060 of Lecture Notes in Computer Science., Springer (2018) 369–386 28, 76

[34] Gamal, T.E. : A public key cryptosystem and a signature scheme based on discrete logarithms. In Blakley, G.R., Chaum, D., eds. : Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings. Volume 196 of Lecture Notes in Computer Science., Springer (1984) 10–18 31

[35] Neff, C.A. : A verifiable secret shuffle and its application to e-voting. In Reiter, M.K., Samarati, P., eds. : CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001, ACM (2001) 116–125 31, 74, 75

[36] Nikitin, K., Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Gasser, L., Khoffi, I., Cappos, J., Ford, B. : CHAINIAC : proactive software-update transparency via collectively signed skipchains and verified builds. In Kirda, E., Ristenpart, T., eds. : 26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017, USENIX Association (2017) 1271–1287 31

[37] Tomescu, A., Devadas, S. : Catena : Efficient non-equivocation via bitcoin. In : 2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017, IEEE Computer Society (2017) 393–409 31

[38] Chaieb, M., Yousfi, S., Lafourcade, P., Robbana, R. : Verify-your-vote : A verifiable blockchain-based online voting protocol. In Themistocleous, M., da Cunha, P.R., eds. : Information Systems - 15th European, Mediterranean, and Middle Eastern Conference, EMCIS 2018, Limassol, Cyprus, October 4-5, 2018, Proceedings. Volume 341 of Lecture Notes in Business Information Processing., Springer (2018) 16–30 39

[39] Chaieb, M., Yousfi, S., Lafourcade, P., Robbana, R. : Design and practical implementation of verify-your-vote protocol. Concurr. Comput. Pract. Exp. **32** (2020) 39

[40] Chaum, D., Ryan, P.Y.A., Schneider, S.A. : A practical voter-verifiable election scheme. In di Vimercati, S.D.C., Syverson, P.F., Gollmann, D., eds. : Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings. Volume 3679 of Lecture Notes in Computer Science., Springer (2005) 118–139 40

[41] Akinyele, J.A., Garman, C., Miers, I., Pagano, M.W., Rushanan, M., Green, M., Rubin, A.D. : Charm : a framework for rapidly prototyping cryptosystems. J. Cryptographic Engineering **3**(2) (2013) 111–128 41

[42] Okamoto, T. : Provably secure and practical identification schemes and corresponding signature schemes. In Brickell, E.F., ed. : Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Volume 740 of Lecture Notes in Computer Science., Springer (1992) 31–53 60, 61, 75, 83

[43] Chaieb, M., Koscina, M., Yousfi, S., Lafourcade, P., Robbana, R. : DABSTERS : distributed authorities using blind signature to effect robust security in e-voting. In Obaidat, M.S., Samarati, P., eds. : Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2 : SECRYPT, Prague, Czech Republic, July 26-28, 2019, SciTePress (2019) 228–235 60

[44] Chaieb, M., Koscina, M., Yousfi, S., Lafourcade, P., Robbana, R. : DABSTERS : A privacy preserving e-voting protocol for permissioned blockchain. In Hierons, R.M., Mosbah, M., eds. : Theoretical Aspects of Computing - ICTAC 2019 - 16th International Colloquium, Hammamet, Tunisia, October 31 - November 4, 2019, Proceedings. Volume 11884 of Lecture Notes in Computer Science., Springer (2019) 292–312 60

[45] Juels, A., Catalano, D., Jakobsson, M. : Coercion-resistant electronic elections. In Atluri, V., di Vimercati, S.D.C., Dingledine, R., eds. : Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, WPES 2005, Alexandria, VA, USA, November 7, 2005, ACM (2005) 61–70 73, 74, 75, 76, 77, 80

[46] Araújo, R., Traoré, J. : A practical coercion resistant voting scheme revisited. In Heather, J., Schneider, S.A., Teague, V., eds. : E-Voting and Identify - 4th International Conference, Vote-ID 2013, Guildford, UK, July 17-19, 2013. Proceedings. Volume 7985 of Lecture Notes in Computer Science., Springer (2013) 193–209 74, 75, 85

[47] Jakobsson, M., Juels, A. : Mix and match : Secure function evaluation via ciphertexts. In Okamoto, T., ed. : Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Volume 1976 of Lecture Notes in Computer Science., Springer (2000) 162–177 74, 75

[48] MacKenzie, P.D., Shrimpton, T., Jakobsson, M. : Threshold password-authenticated key exchange. In Yung, M., ed. : Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings. Volume 2442 of Lecture Notes in Computer Science., Springer (2002) 385–400 74

[49] Furukawa, J., Sako, K. : An efficient scheme for proving a shuffle. [66] 368–387 74

[50] Weber, S.G., Araújo, R., Buchmann, J.A. : On coercion-resistant electronic elections with linear work. In : Proceedings of the The Second International Conference on Availability, Reliability and Security, ARES 2007, The International Dependability Conference - Bridging Theory and Practice, April 10-13 2007, Vienna, Austria, IEEE Computer Society (2007) 908–916 75

[51] Clarkson, M.R., Chong, S., Myers, A.C. : Civitas : Toward a secure voting system. In : 2008 IEEE Symposium on Security and Privacy (S&P 2008), 18-21 May 2008, Oakland, California, USA, IEEE Computer Society (2008) 354–368 75

[52] Araújo, R., Rajeb, N.B., Robbana, R., Traoré, J., Yousfi, S. : Towards practical and secure coercion-resistant electronic elections. In Heng, S., Wright, R.N., Goi, B., eds. : Cryptology and Network Security - 9th International Conference, CANS 2010, Kuala Lumpur, Malaysia, December 12-14, 2010. Proceedings. Volume 6467 of Lecture Notes in Computer Science., Springer (2010) 278–297 75

[53] Spycher, O., Koenig, R.E., Haenni, R., Schläpfer, M. : A new approach towards coercion-resistant remote e-voting in linear time. In Danezis, G., ed. : Financial Cryptography and Data Security - 15th International Conference, FC 2011, Gros Islet, St. Lucia, February 28 - March 4, 2011, Revised Selected Papers. Volume 7035 of Lecture Notes in Computer Science., Springer (2011) 182–189 75

[54] Rønne, P.B., Atashpendar, A., Gjøsteen, K., Ryan, P.Y.A. : Coercion-resistant voting in linear time via fully homomorphic encryption : Towards a quantum-safe scheme. CoRR **abs/1901.02560** (2019) 75

[55] Boneh, D., Boyen, X., Shacham, H. : Short group signatures. In Franklin, M.K., ed. : Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings. Volume 3152 of Lecture Notes in Computer Science., Springer (2004) 41–55 75, 78

[56] Schweisgut, J. : Coercion-resistant electronic elections with observer. In Krimmer, R., ed. : Electronic Voting 2006 : 2nd International Workshop, Co-organized by Council of Europe, ESF TED, IFIP WG 8.6 and E-Voting.CC, August, 2nd - 4th, 2006 in Castle Hofen, Bregenz, Austria. Volume P-86 of LNI., GI (2006) 171–177 75

[57] Cachin, C., Kursawe, K., Shoup, V. : Random oracles in constantinople : Practical asynchronous byzantine agreement using cryptography. J. Cryptology **18**(3) (2005) 219–246 75

[58] Furukawa, J., Sako, K. : An efficient publicly verifiable mix-net for long inputs. IEICE Transactions **90-A**(1) (2007) 113–127 75

[59] Chaum, D., Pedersen, T.P. : Wallet databases with observers. In Brickell, E.F., ed. : Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings. Volume 740 of Lecture Notes in Computer Science., Springer (1992) 89–105 75

[60] Dimtiriou, T. : Efficient, coercion-free and universally verifiable blockchain-based voting. IACR Cryptology ePrint Archive **2019** (2019) 1406 76

[61] Brickell, E.F., Camenisch, J., Chen, L. : Direct anonymous attestation. In Atluri, V., Pfitzmann, B., McDaniel, P.D., eds. : Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS 2004, Washington, DC, USA, October 25-29, 2004, ACM (2004) 132–145 76

[62] Gennaro, R., Gentry, C., Parno, B., Raykova, M. : Quadratic span programs and succinct nizks without pcps. In Johansson, T., Nguyen, P.Q., eds. : Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Volume 7881 of Lecture Notes in Computer Science., Springer (2013) 626–645 77

[63] Douceur, J.R. : The sybil attack. In Druschel, P., Kaashoek, M.F., Rowstron, A.I.T., eds. : Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers. Volume 2429 of Lecture Notes in Computer Science., Springer (2002) 251–260 79

[64] Winter, P., Ensafi, R., Loesing, K., Feamster, N. : Identifying and characterizing sybils in the tor network. In Holz, T., Savage, S., eds. : 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016, USENIX Association (2016) 1169–1185 79

[65] Egger, C., Schlumberger, J., Kruegel, C., Vigna, G. : Practical attacks against the I2P network. In Stolfo, S.J., Stavrou, A., Wright, C.V., eds. : Research in Attacks, Intrusions, and Defenses - 16th International Symposium, RAID 2013, Rodney Bay, St. Lucia, October 23-25, 2013. Proceedings. Volume 8145 of Lecture Notes in Computer Science., Springer (2013) 432–451 79

[66] Kilian, J., ed. : Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings. Volume 2139 of Lecture Notes in Computer Science., Springer (2001) 90, 93

Annexe A

Codes ProVerif du protocole *Verify-Your-Vote*

Pour écrire notre protocole avec le langage de modélisation formelle Applied Pi-Calculus, nous devons définir un ensemble de noms, de variables et de symboles de fonctions. Ces symboles de fonction ont des arités et des types. Pour représenter les opérations de chiffrement, de déchiffrement, de signature numérique et de hachage, nous utilisons les symboles de fonction suivants :
pk(skey), aenc(x,pk(skey)), adec(x,skey), spk(sskey), sign(x,sskey), checksign(x,spk(sskey)), H1(x).

Intuitivement, la fonction *pk* génère la clé publique correspondante à partir d'une clé secrète donnée, *aenc* et *adec* représentent, respectivement, le chiffrement et le déchiffrement asymétriques, *aenc* et *adec* suivent cette équation :

$$adec(aenc(x,pk(y)),y)=x$$

La fonction *spk* génère la clé publique correspondante à partir d'une clé secrète de signature donnée, *sign* et *checksign* fournissent, respectivement, la signature d'un message donné et la vérification de la signature. Elles respectent l'équation suivante :

$$checksign(sign(x,y),spk(y))=x$$

Nous supposons également l'opération de hachage qui est désignée par la fonction *H1*.

A.1 Code relatif à la vérification de la propriété "Secret de vote"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre le serveur d'enregistrement et le votant *)
free Crs_v: channel [private].

(* Un canal public entre l'administrateur et le votant *)
free Ca_v: channel.

(* Pseudo ID d'un candidat C1*)
```

```
const C1: bitstring [private].

(* Cryptosysteme asymetrique *)
type skey .
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(*Signatures numeriques*)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incre-
mente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.

(* Requetes de secret de vote *)
query attacker(Bn).
query attacker(C1).

(* Un numero de bulletin de vote *)
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa est la cle secrete d'un votant A. *)
free SKa, SKta: skey [private].

let VYV(SKv: skey, Cj: bitstring)=
new SKrs_a: skey;
new SKadmin: skey;
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
```

```
(* Le votant genere un mot de passe et l'envoie au serveur d'enregistre-
ment via le canal prive Crs_v *)
new PW: bitstring;
out(Crs_v, PW);
(* Le serveur d'enregistrement recoit le mot de passe, calcule les para-
metres d'authentification et les envoie au votant via le meme canal *)
in(Crs_v, XPW: bitstring);
let Spw = prod(SKrs_a,H1(XPW)) in
let Ppw = H1(XPW) in
out(Crs_v, (Spw, Ppw));
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Crs_v, (XSpw: bitstring, XPpw: bitstring));

(*** Phase d'authentification ***)
(* Le votant chiffre ses parametres d'authentification avec la cle pub-
lique de l'administrateur, les signe avec sa cle secrete de signature
"SSKv" et les envoie a l'administrateur via le canal public Ca_v *)
let Enc_Spw= sign(aenc(XSpw, pk(SKadmin)), SSKv) in
let Enc_Ppw= sign(aenc(XPpw, pk(SKadmin)), SSKv) in
out(Ca_v, (Enc_Spw, Enc_Ppw));
(* L'administrateur recoit les parametres d'authentification du votant,
verifie la validite de la signature et les dechiffre *)
in(Ca_v,(XXSpw: bitstring, XXPpw: bitstring));
let (X1: bitstring)= checksign(adec(XXSpw, SKadmin),spk(SSKv)) in
let (X2: bitstring)= checksign(adec(XXPpw, SKadmin),spk(SSKv)) in
(* Il verifie la validite des parametres d'authentification du votant,
si les parametres d'authentification sont valides on passe a la phase de
vote *)
if X1=prod(SKrs_a, X2) then
(
(*** Phase de vote ***)
(* Les autorites de comptage chiffrent un bulletin de vote avec la cle
publique du votant pour le lui envoyer via la Blockchain, representee
ici par la canal public C *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(*Le votant recoit son bulletin de vote, le dechiffre avec sa cle pri-
vee, chiffre son vote et l'envoie via la canal public C *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
out(C, Enc_Vote);

(*** Phase de Comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre puis incremente
le compteur correspondant *)
in(C, Xvote: bitstring);
let Vote= adec(Xvote, SKta) in
```

```
let result= increment_counter(Vote) in
out(C, result)).

process
VYV(SKa, C1)
```

A.2 Code relatif à la vérification de la propriété "Authentification des électeurs"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre le serveur d'enregistrement et le votant *)
free Crs_v: channel [private].

(* Un canal public entre l'administrateur et le votant *)
free Ca_v: channel.

(* Pseudo ID d'un candidat C1*)
const C1: bitstring [private].

(* Cryptosysteme asymetrique *)
type skey .
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(*Signatures numeriques*)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incremente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.
```

```
(* Requetes d'authentification *)
event acceptedAuthentication(bitstring, bitstring).
event VerifiesParameters(bitstring, bitstring).
query a: bitstring, b: bitstring;
event (acceptedAuthentication(a,b))==>event(VerifiesParameters(a, b)).

(* Un numero de bulletin de vote *)
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa est la cle secrete d'un votant A. *)
free SKa, SKta: skey [private].

let VYV(SKv: skey, Cj: bitstring)=
new SKrs_a: skey;
new SKadmin: skey;
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
(* Le votant genere un mot de passe et l'envoie au serveur d'enregistre-
ment via le canal prive Crs_v *)
new PW: bitstring;
out(Crs_v, PW);
(* Le serveur d'enregistrement recoit le mot de passe, calcule les para-
metres d'authentification et les envoie au votant via le meme canal *)
in(Crs_v, XPW: bitstring);
let Spw = prod(SKrs_a,H1(XPW))in
let Ppw = H1(XPW)in
out(Crs_v, (Spw, Ppw));
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Crs_v, (XSpw: bitstring, XPpw: bitstring));

(*** Phase d'authentification ***)
(* Le votant chiffre ses parametres d'authentification avec la cle pub-
lique de l'administrateur, les signe avec sa cle secrete de signature
"SSKv" et les envoie a l'administrateur via le canal public Ca_v *)
let Enc_Spw= sign(aenc(XSpw, pk(SKadmin)), SSKv)in
let Enc_Ppw= sign(aenc(XPpw, pk(SKadmin)), SSKv) in
out(Ca_v, (Enc_Spw, Enc_Ppw));
(* L'administrateur recoit les parametres d'authentification du votant,
verifie la validite de la signature et les dechiffre *)
in(Ca_v,(XXSpw: bitstring, XXPpw: bitstring));
let (X1: bitstring)= checksign(adec(XXSpw, SKadmin),spk(SSKv)) in
let (X2: bitstring)= checksign(adec(XXPpw, SKadmin),spk(SSKv)) in
(* Il verifie la validite des parametres d'authentification du votant,
emet l'evenement "VerifiesParameters" si les parametres d'authentifica-
```

```
tion sont valides et envoie un message chiffre pour notifier le votant *)
if X1=prod(SKrs_a, X2) then
(event VerifiesParameters(XXSpw, XXPpw);
out(Ca_v, aenc(msg1, pk(SKv)));
(* Le votant dechiffre le message et emet l'evennement "acceptedAuthen-
tication" pour enregistrer le fait qu'il a ete authentifie avec succes *)
in(Ca_v, msg: bitstring);
let msg_dec= adec(msg, SKv) in
event acceptedAuthentication(Enc_Spw, Enc_Ppw);

(*** Phase de vote ***)
(* Les autorites de comptage chiffrent un bulletin de vote avec la cle
publique du votant pour le lui envoyer via la Blockchain, representee
ici par la canal public C *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(*Le votant recoit son bulletin de vote, le dechiffre avec sa cle pri-
vee, chiffre son vote et l'envoie via la canal public C *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
out(C, Enc_Vote);

(*** Phase de Comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre puis incremente
le compteur correspondant *)
in(C, Xvote: bitstring);
let Vote= adec(Xvote, SKta) in
let result= increment_counter(Vote) in
out(C, result)).

process
VYV(SKa, C1)
```

A.3 Code relatif à la vérification de la propriété "Confidentialité des votes"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre le serveur d'enregistrement et le votant *)
free Crs_v: channel [private].

(* Un canal public entre l'administrateur et le votant *)
free Ca_v: channel.

(* Deux Pseudo ID possibles C1 et C2 *)
const C1, C2: bitstring [private].
```

```
(* Cryptosysteme asymetrique *)
type skey .
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(*Signatures numeriques*)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incre-
mente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.

(* Numero d'un bulletin de vote *)
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa et SKb sont les cles secretes de deux votants A et B. *)
free SKa, SKb, SKta: skey [private].

let VYV(SKv: skey, Cj: bitstring)=
new SKrs_a: skey;
new SKadmin: skey;
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
(* Le votant genere un mot de passe et l'envoie au serveur d'enregistre-
ment via le canal prive Crs_v *)
new PW: bitstring;
out(Crs_v, PW);
(* Le serveur d'enregistrement recoit le mot de passe, calcule les para-
metres d'authentification et les envoie au votant via le meme canal *)
```

```
in(Crs_v, XPW: bitstring);
let Spw = prod(SKrs_a,H1(XPW))in
let Ppw = H1(XPW)in
out(Crs_v, (Spw, Ppw));
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Crs_v, (XSpw: bitstring, XPpw: bitstring));

(*** Phase d'authentification ***)
(* Le votant chiffre ses parametres d'authentification avec la cle pub-
lique de l'administrateur, les signe avec sa cle secrete de signature
"SSKv" et les envoie a l'administrateur via le canal public Ca_v *)
let Enc_Spw= sign(aenc(XSpw, pk(SKadmin)), SSKv)in
let Enc_Ppw= sign(aenc(XPpw, pk(SKadmin)), SSKv) in
out(Ca_v, (Enc_Spw, Enc_Ppw));
(* L'administrateur recoit les parametres d'authentification du votant,
verifie la validite de la signature et les dechiffre *)
in(Ca_v,(XXSpw: bitstring, XXPpw: bitstring));
let (X1: bitstring)= checksign(adec(XXSpw, SKadmin),spk(SSKv)) in
let (X2: bitstring)= checksign(adec(XXPpw, SKadmin),spk(SSKv)) in
(* Il verifie la validite des parametres d'authentification du votant,
si les parametres d'authentification sont valides on passe a la phase de
vote *)
if X1=prod(SKrs_a, X2) then
(
(*** Phase de vote ***)
(* Les autorites de comptage chiffrent un bulletin de vote avec la cle
publique du votant pour le lui envoyer via la Blockchain, representee
ici par la canal public C *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(*Le votant recoit son bulletin de vote, le dechiffre avec sa cle pri-
vee, chiffre son vote et l'envoie via la canal public C *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
sync 1;
out(C, Enc_Vote);
sync 2;

(*** Phase de Comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre puis incremente
le compteur correspondant *)
in(C, Xvote: bitstring);
let Vote= adec(Xvote, SKta) in
let result= increment_counter(Vote) in
out(C, result)).

process
```

```
VYV(SKa, choice[C1, C2]) | VYV(SKb, choice[C2, C1])
```

Annexe B

Codes ProVerif du protocole *DABSTERS*

Pour modéliser notre protocole avec le langage de modélisation formelle Applied Pi-Calculus, nous utilisons les mêmes noms, variables et symboles de fonctions définis dans l'Annexe A.

B.1 Code relatif à la vérification de la propriété "Secret de vote"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre l'autorite d'enregistrement et le votant *)
free Cra_v: channel [private].

(* Pseudo ID d'un candidat *)
const C1: bitstring [private].

(* Crytposysteme asymetrique *)
type skey .
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(* Signatures numeriques *)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Signature en aveugle *)
```

```
fun BlindSign(bitstring, bitstring, bitstring, bitstring, bitstring,
pkey): bitstring.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incre-
mente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.
fun Valid(bitstring): bool.

(* Requetes de secret de vote *)
query attacker(Bn).
query attacker(C1).

(* Numero d'un bulletin de vote *)
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa est la cle secrete d'un votant A. *)
free SKa, SKta: skey [private].


let DABSTERS(SKv: skey, Cj: bitstring)=
new Sm: skey; (* Cle secrete maitresse des autorites d'enregistrement *)
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement via le canal prive Cra_v *)
new ID: bitstring;
out(Cra_v, ID);
(* Les autorites d'enregistrement verifie l'eligibilite du votant, cal-
cule ses parametres d'authentification et les envient au votant via le
canal prive Cra_v *)
let Cred = prod(Sm,H1(ID))in
out(Cra_v, Cred);
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Cra_v, (Credential: bitstring));

(*** Phase de validation ***)
(* Les autorites d'enregistrement publient la liste des votants enregis-
tres sur la Blockchain, representee ici par le canal public C *)
```

```
out(C, ID);

(*** Phase de vote ***)
(* Les autorites de comptage chiffre un bulletin de vote et l'envoie au
votant *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(* Le votant recoit son bulletin de vote, le dechiffre, chiffre son vote
et l'envoie aux autorites d'enregistrement pour anonymiser son vote et
masquer sa signature *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
new beta: bitstring;
new gama: bitstring;
new delta: bitstring;
out(Cra_v, (Credential, Enc_Vote, beta,gama,delta));
(* Les autorites d'enregistrement recoivent le message du votant, veri-
fient ses parametres d'authentification et masquent sa signature s'il
est eligible pour voter *)
in(Cra_v, (XCredential: bitstring, XEnc_Vote:bitstring, Xbeta:bitstring,
Xgama:bitstring, Xdelta:bitstring));
if Valid(XCredential) then
(new BlindedSig: bitstring;
let BlindedSig=BlindSign(XCredential, XEnc_Vote, Xbeta, Xgama, Xdelta,
pk(Sm)) in
out(Cra_v, BlindedSig);
(* Le votant recoit sa signature aveugle et envoie son vote via le canal
public C *)
in(Cra_v, BlindedSignature: bitstring);
out(C, (Credential, Enc_Vote, BlindedSignature));

(*** Phase de comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre et incremente le
compteur correspondant *)
in(C, (XXCredential: bitstring, XEnc_Vote: bitstring, XBlindedSignature:
bitstring));
if Valid(XBlindedSignature) then
(new Vote: bitstring;
let Vote= adec(XEnc_Vote, SKta) in
let result= increment_counter(Vote) in
out(C, result))).

process
DABSTERS(SKa, C1)
```

B.2 Code relatif à la vérification de la propriété "Authentification des électeurs"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre l'autorite d'enregistrement et le votant *)
free Cra_v: channel [private].

(* Pseudo ID d'un candidat *)
const C1: bitstring [private].

(* Crytposysteme asymetrique *)
type skey .
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(* Signatures numeriques *)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Signature en aveugle *)
fun BlindSign(bitstring, bitstring, bitstring, bitstring, bitstring,
pkey): bitstring.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incre-
mente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.
fun Valid(bitstring): bool.

(* Requetes d'authentification *)
event acceptedAuth(bitstring).
event VerifiesParam(bitstring).
query a: bitstring;
event (acceptedAuth(a))==>event(VerifiesParam(a)).

(* Numero d'un bulletin de vote *)
```

```
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa est la cle secrete d'un votant A. *)
free SKa, SKta: skey [private].


let DABSTERS(SKv: skey, Cj: bitstring)=
new Sm: skey; (* Cle secrete maitresse des autorites d'enregistrement *)
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement via le canal prive Cra_v *)
new ID: bitstring;
out(Cra_v, ID);
(* Les autorites d'enregistrement verifie l'eligibilite du votant, cal-
cule ses parametres d'authentification et les envient au votant via le
canal prive Cra_v *)
let Cred = prod(Sm,H1(ID))in
out(Cra_v, Cred);
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Cra_v, (Credential: bitstring));

(*** Phase de validation ***)
(* Les autorites d'enregistrement publient la liste des votants enregis-
tres sur la Blockchain, representee ici par le canal public C *)
out(C, ID);

(*** Phase de vote ***)
(* Les autorites de comptage chiffre un bulletin de vote et l'envoie au
votant *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(* Le votant recoit son bulletin de vote, le dechiffre, chiffre son vote
et l'envoie aux autorites d'enregistrement pour anonymiser son vote et
masquer sa signature *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
new beta: bitstring;
new gama: bitstring;
new delta: bitstring;
out(Cra_v, (Credential, Enc_Vote, beta,gama,delta));
(* Les autorites d'enregistrement recoivent le message du votant, veri-
fient ses parametres d'authentification et masquent sa signature s'il
```

```
est eligible pour voter *)
in(Cra_v, (XCredential: bitstring, XEnc_Vote:bitstring, Xbeta:bitstring,
Xgama:bitstring, Xdelta:bitstring));
if Valid(XCredential) then
(event VerifiesParam(XCredential);
out(Cra_v, aenc(msg1, pk(SKv)));
(* Le votant dechiffre le message et emet l'evennement "acceptedAuthen-
tication" pour enregistrer le fait qu'il a ete authentifie avec succes *)
in(Cra_v, msg: bitstring);
let msg_dec= adec(msg, SKv) in
event acceptedAuth(Credential);

new BlindedSig: bitstring;
let BlindedSig=BlindSign(XCredential, XEnc_Vote, Xbeta, Xgama, Xdelta,
pk(Sm)) in
out(Cra_v, BlindedSig);
(* Le votant recoit sa signature aveugle et envoie son vote via le canal
public C *)
in(Cra_v, BlindedSignature: bitstring);
out(C, (Credential, Enc_Vote, BlindedSignature));

(*** Phase de comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre et incremente le
compteur correspondant *)
in(C, (XXCredential: bitstring, XEnc_Vote: bitstring, XBlindedSignature:
bitstring));
if Valid(XBlindedSignature) then
(new Vote: bitstring;
let Vote= adec(XEnc_Vote, SKta) in
let result= increment_counter(Vote) in
out(C, result))).

process
DABSTERS(SKa, C1)
```

B.3 Code relatif à la vérification de la propriété "Confidentialité des votes"

```
(* Un canal public, represente la Blockchain de l'election *)
free C:channel.

(* Un canal prive entre l'autorite d'enregistrement et le votant *)
free Cra_v: channel [private].

(* Deux Pseudo ID possibles C1 et C2 *)
const C1, C2: bitstring [private].

(* Crytposysteme asymetrique *)
type skey .
```

```
type pkey .
fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

(* Signatures numeriques *)
type spkey.
type sskey.
fun spk(sskey): spkey.
fun sign(bitstring, sskey): bitstring.
reduc forall x: bitstring, y: sskey;
getmess(sign(x,y)) = x.
reduc forall x: bitstring, y: sskey;
checksign(sign(x,y), spk(y)) = x.

(* Signature en aveugle *)
fun BlindSign(bitstring, bitstring, bitstring, bitstring, bitstring,
pkey): bitstring.

(* Fonctions de hashage *)
fun h1(bitstring): pkey.
fun H1(bitstring): bitstring.

(* La fonction "increment_counter" prend en parametre un vote et incre-
mente le compteur du candidat choisi *)
fun increment_counter(bitstring): bitstring.

fun prod(skey, bitstring): bitstring.
fun Valid(bitstring): bool.

(* Numero d'un bulletin de vote *)
free Bn: bitstring [private].

(* Cles secretes:
    SKta est la cle secrete des autorites de comptage
    SKa et SKb sont les cles secretes des votants A et B. *)
free SKa, SKb, SKta: skey [private].


let DABSTERS(SKv: skey, Cj: bitstring)=
new Sm: skey; (* Cle secrete maitresse des autorites d'enregistrement *)
new SSKv: sskey;
new msg1: bitstring;

(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement via le canal prive Cra_v *)
new ID: bitstring;
```

```
out(Cra_v, ID);
(* Les autorites d'enregistrement verifie l'eligibilite du votant, cal-
cule ses parametres d'authentification et les envient au votant via le
canal prive Cra_v *)
let Cred = prod(Sm,H1(ID))in
out(Cra_v, Cred);
(* Le votant recoit ses parametres d'authentification via le meme canal
prive *)
in(Cra_v, (Credential: bitstring));

(*** Phase de validation ***)
(* Les autorites d'enregistrement publient la liste des votants enregis-
tres sur la Blockchain, representee ici par le canal public C *)
out(C, ID);

(*** Phase de vote ***)
(* Les autorites de comptage chiffre un bulletin de vote et l'envoie au
votant *)
let Enc_Ballot= aenc(Bn, pk(SKv)) in
out(C, Enc_Ballot);
(* Le votant recoit son bulletin de vote, le dechiffre, chiffre son vote
et l'envoie aux autorites d'enregistrement pour anonymiser son vote et
masquer sa signature *)
in(C, X:bitstring);
let Bn= adec(X, SKv) in
let Enc_Vote= aenc(Bn, h1(Cj)) in
new beta: bitstring;
new gama: bitstring;
new delta: bitstring;
out(Cra_v, (Credential, Enc_Vote, beta,gama,delta));
(* Les autorites d'enregistrement recoivent le message du votant, veri-
fient ses parametres d'authentification et masquent sa signature s'il
est eligible pour voter *)
in(Cra_v, (XCredential: bitstring, XEnc_Vote:bitstring, Xbeta:bitstring,
Xgama:bitstring, Xdelta:bitstring));
if Valid(XCredential) then
(new BlindedSig: bitstring;
let BlindedSig=BlindSign(XCredential, XEnc_Vote, Xbeta, Xgama, Xdelta,
pk(Sm)) in
out(Cra_v, BlindedSig);
(* Le votant recoit sa signature aveugle et envoie son vote via le canal
public C *)
in(Cra_v, BlindedSignature: bitstring);
sync 1;
out(C, (Credential, Enc_Vote, BlindedSignature));
sync 2;

(*** Phase de comptage ***)
(* L'autorite de comptage recoit le vote, le dechiffre et incremente le
```

```
compteur correspondant *)
in(C, (XXCredential: bitstring, XEnc_Vote: bitstring, XBlindedSignature:
bitstring));
if Valid(XBlindedSignature) then
(new Vote: bitstring;
let Vote= adec(XEnc_Vote, SKta) in
let result= increment_counter(Vote) in
out(C, result))).

process
DABSTERS(SKa, choice[C1, C2]) | DABSTERS(SKb, choice[C2, C1])
```

Annexe C

Codes ProVerif du protocole *LOKI Vote*

Pour écrire notre protocole avec le langage de modélisation formelle Applied Pi-Calculus, nous définissons les fonctions cryptographiques suivantes :
pk(skey), aenc(x,pk(skey)) et adec(x,skey).

La fonction *pk* génère la clé publique correspondante à partir d'une clé secrète donnée, *aenc* et *adec* représentent, respectivement, le chiffrement et le déchiffrement asymétriques, *aenc* et *adec* suivent cette équation :

$$adec(aenc(x,pk(y)),y)=x$$

C.1 Code relatif à la vérification de la propriété "Secret de vote"

```
(* Un canal public qui represente la Blockchain de l'election *)
free C:channel.

(* Un canal public entre les autorites d'enregistrement et le votant *)
free Cra_vPub: channel.

(* Un canal prive entre les autorites d'enregistrement et le votant *)
free Cra_v: channel [private].

type skey .
type pkey .
type integer.

fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

fun exist(bitstring, bitstring): bool.
fun GenL1(bitstring, bitstring, integer, pkey, bitstring): bitstring.
fun GenL2(bitstring, pkey, bitstring): bitstring.
fun GenProofs(bitstring, bitstring, bitstring, bitstring, bitstring,
bitstring): bitstring.
```

```
fun power(bitstring, integer):bitstring.
fun verifyProof(bitstring): bool.
fun verifyCredential(bitstring, bitstring, bitstring): bool.
fun Duplicate(bitstring): bool.
fun incrementCounter(bitstring): bitstring.
fun Genrandom(): integer.
fun GenA(bitstring, bitstring, integer, integer, skey): bitstring.

free vote: bitstring [private].

(* Requete de secret de vote *)
query attacker(vote).


let LokiVote()=
(*** Phase de configuration ***)
(* Les autorites d'enregistrement generent les parametres publics de
l'election g1, g2, g3, o ainsi que deux paires de cles (R, R') et
(M_ElGamalR, M_ElGamalR') et publient les parties publiques sur la
Blockchain *)
new g1 : bitstring;
new g2 : bitstring;
new g3 : bitstring;
new o : bitstring;
new R' : skey;
new M_ElGamalR': skey;
let R= pk(R') in
let M_ElGamalR= pk(M_ElGamalR') in
out(C, (g1, g2, g3, o, R, M_ElGamalR));
(* Les autorites de comptage generent leur paire de cles et publient la
cle publique sur la Blockchain de l'election *)
new M_ElGamalT': skey;
let M_ElGamalT= pk(M_ElGamalT') in
out(C, M_ElGamalT);
(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement, via le canal prive Cra_v, qui verifient l'eligibilite
du votant et lui fournit ses parametres d'authentification *)
new ID: bitstring;
out(Cra_v, ID);
new List: bitstring; (* La liste des numeros de cartes d'identite des
votants eligibles *)
if exist(ID, List)=true then
(
let r= Genrandom() in
let x= Genrandom() in
let A= GenA(g1, g3, x, r, M_ElGamalR') in
(* La partie publique des parametres d'authentification (A,r) est envo-
yee a travers le canal public Cra_vPub *)
```

```
out(Cra_vPub, (A, r));
(* La partie privee des parametres d'authentification (x) est envoyee
via le canal prive Cra_v *)
out(Cra_v, x);
let L1= GenL1(g1, g3, x, M_ElGamalR, ID)in
let L2= GenL2(A, M_ElGamalT, ID)in
out(C, (L1, L2));

(*** Phase de vote ***)
(* Le votant genere son n-uplet de vote *)
let EncVote=aenc(vote, M_ElGamalT) in
let EncA=aenc(A, M_ElGamalT) in
let Ar= power(A, r) in
let EncAr= aenc(Ar, M_ElGamalT) in
let g3x= power(g3, x) in
let Encg3x = aenc(g3x, M_ElGamalT) in
let Ox= power(o, x) in
let proofs= GenProofs(vote, EncA, EncAr, Encg3x, A, o) in
out(C, (EncVote, EncA, EncAr, Encg3x, Ox, proofs));

(*** Phase de comptage ***)
(* Les autorites de comptage verifient la validite des preuves de cha-
que n-uplet de vote *)
if verifyProof(proofs) = true then
(
        (* Si les preuves sont valides, les autorites de comptage veri-
        fient si le votant a deja vote *)
        if Duplicate(Ox)= false then
        (
                (* Finalement, elles verifient la validite des paramet-
                res d'authentification *)
                (* Si ces parametres sont valides, les autorites de com-
                ptage dechiffrent le vote *)
                if verifyCredential(EncA, EncAr, Encg3x)=true then
                (
                        let DecVote = adec(EncVote, M_ElGamalT') in
                        let result= incrementCounter(DecVote) in
                        out(C, result)
                )
        )
)
).

process
LokiVote()
```

C.2 Code relatif à la vérification de la propriété "Authentification des électeurs"

```
(* Un canal public qui represente la Blockchain de l'election *)
free C:channel.

(* Un canal public entre les autorites d'enregistrement et le votant *)
free Cra_vPub: channel.

(* Un canal prive entre les autorites d'enregistrement et le votant *)
free Cra_v: channel [private].

type skey .
type pkey .
type integer.

fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

fun exist(bitstring, bitstring): bool.
fun GenL1(bitstring, bitstring, integer, pkey, bitstring): bitstring.
fun GenL2(bitstring, pkey, bitstring): bitstring.
fun GenProofs(bitstring, bitstring, bitstring, bitstring, bitstring,
bitstring): bitstring.
fun power(bitstring, integer):bitstring.
fun verifyProof(bitstring): bool.
fun verifyCredential(bitstring, bitstring, bitstring): bool.
fun Duplicate(bitstring): bool.
fun incrementCounter(bitstring): bitstring.
fun Genrandom(): integer.
fun GenA(bitstring, bitstring, integer, integer, skey): bitstring.

free vote: bitstring [private].

(* Requetes d'authentification *)
event ValidCred.
event CredentialVerification.
query event(ValidCred)==>event(CredentialVerification).



let LokiVote()=
(*** Phase de configuration ***)
(* Les autorites d'enregistrement generent les parametres publics de
l'election g1, g2, g3, o ainsi que deux paires de cles (R, R') et
(M_ElGamalR, M_ElGamalR') et publient les parties publiques sur la
Blockchain *)
new g1 : bitstring;
new g2 : bitstring;
new g3 : bitstring;
```

```
new o : bitstring;
new R' : skey;
new M_ElGamalR': skey;
let R= pk(R') in
let M_ElGamalR= pk(M_ElGamalR') in
out(C, (g1, g2, g3, o, R, M_ElGamalR));
(* Les autorites de comptage generent leur paire de cles et publient la
cle publique sur la Blockchain de l'election *)
new M_ElGamalT': skey;
let M_ElGamalT= pk(M_ElGamalT') in
out(C, M_ElGamalT);
(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement, via le canal prive Cra_v, qui verifient l'eligibilite
du votant et lui fournit ses parametres d'authentification *)
new ID: bitstring;
out(Cra_v, ID);
new List: bitstring; (* La liste des numeros de cartes d'identite des
votants eligibles *)
if exist(ID, List)=true then
(
let r= Genrandom() in
let x= Genrandom() in
let A= GenA(g1, g3, x, r, M_ElGamalR') in
(* La partie publique des parametres d'authentification (A,r) est envo-
yee a travers le canal public Cra_vPub *)
out(Cra_vPub, (A, r));
(* La partie privee des parametres d'authentification (x) est envoyee
via le canal prive Cra_v *)
out(Cra_v, x);
let L1= GenL1(g1, g3, x, M_ElGamalR, ID)in
let L2= GenL2(A, M_ElGamalT, ID)in
out(C, (L1, L2));

(*** Phase de vote ***)
(* Le votant genere son n-uplet de vote *)
let EncVote=aenc(vote, M_ElGamalT) in
let EncA=aenc(A, M_ElGamalT) in
let Ar= power(A, r) in
let EncAr= aenc(Ar, M_ElGamalT) in
let g3x= power(g3, x) in
let Encg3x = aenc(g3x, M_ElGamalT) in
let Ox= power(o, x) in
let proofs= GenProofs(vote, EncA, EncAr, Encg3x, A, o) in
out(C, (EncVote, EncA, EncAr, Encg3x, Ox, proofs));

(*** Phase de comptage ***)
(* Les autorites de comptage verifient la validite des preuves de cha-
que n-uplet de vote *)
```

```
if verifyProof(proofs) = true then
(
        (* Si les preuves sont valides, les autorites de comptage veri-
        fient si le votant a deja vote *)
        if Duplicate(Ox)= false then
        (
                (* Finalement, elles verifient la validite des paramet-
                res d'authentification *)
                (* Si ces parametres sont valides, les autorites de com-
                ptage dechiffrent le vote *)
                if verifyCredential(EncA, EncAr, Encg3x)=true then
                (
                        event CredentialVerification();
                        let DecVote = adec(EncVote, M_ElGamalT') in
                        let result= incrementCounter(DecVote) in
                        out(C, result);
                        event ValidCred()
                )
        )
)
).

process
LokiVote()
```

C.3 Code relatif à la vérification de la propriété "Confidentialité des votes"

```
(* Un canal public qui represente la Blockchain de l'election *)
free C:channel.

(* Un canal public entre les autorites d'enregistrement et le votant *)
free Cra_vPub: channel.

(* Un canal prive entre les autorites d'enregistrement et le votant *)
free Cra_v: channel [private].

type skey .
type pkey .
type integer.

fun pk(skey): pkey.
fun aenc (bitstring , pkey): bitstring.
reduc forall m: bitstring, sk: skey;
adec(aenc(m, pk(sk)), sk) = m.

fun exist(bitstring, bitstring): bool.
fun GenL1(bitstring, bitstring, integer, pkey, bitstring): bitstring.
fun GenL2(bitstring, pkey, bitstring): bitstring.
```

```
fun GenProofs(bitstring, bitstring, bitstring, bitstring, bitstring,
bitstring): bitstring.
fun power(bitstring, integer):bitstring.
fun verifyProof(bitstring): bool.
fun verifyCredential(bitstring, bitstring, bitstring): bool.
fun Duplicate(bitstring): bool.
fun incrementCounter(bitstring): bitstring.
fun Genrandom(): integer.
fun GenA(bitstring, bitstring, integer, integer, skey): bitstring.

(* Deux votes possibles V1 et V2 *)
const V1, V2: bitstring [private].


let LokiVote()=
(*** Phase de configuration ***)
(* Les autorites d'enregistrement generent les parametres publics de
l'election g1, g2, g3, o ainsi que deux paires de cles (R, R') et
(M_ElGamalR, M_ElGamalR') et publient les parties publiques sur la
Blockchain *)
new g1 : bitstring;
new g2 : bitstring;
new g3 : bitstring;
new o : bitstring;
new R' : skey;
new M_ElGamalR': skey;
let R= pk(R') in
let M_ElGamalR= pk(M_ElGamalR') in
out(C, (g1, g2, g3, o, R, M_ElGamalR));
(* Les autorites de comptage generent leur paire de cles et publient la
cle publique sur la Blockchain de l'election *)
new M_ElGamalT': skey;
let M_ElGamalT= pk(M_ElGamalT') in
out(C, M_ElGamalT);
(*** Phase d'enregistrement ***)
(* Le votant fournit le numero de sa carte d'identite aux autorites
d'enregistrement, via le canal prive Cra_v, qui verifient l'eligibilite
du votant et lui fournit ses parametres d'authentification *)
new ID: bitstring;
out(Cra_v, ID);
new List: bitstring; (* La liste des numeros de cartes d'identite des
votants eligibles *)
if exist(ID, List)=true then
(
let r= Genrandom() in
let x= Genrandom() in
let A= GenA(g1, g3, x, r, M_ElGamalR') in
(* La partie publique des parametres d'authentification (A,r) est envo-
yee a travers le canal public Cra_vPub *)
```

```
out(Cra_vPub, (A, r));
(* La partie privee des parametres d'authentification (x) est envoyee
via le canal prive Cra_v *)
out(Cra_v, x);
let L1= GenL1(g1, g3, x, M_ElGamalR, ID)in
let L2= GenL2(A, M_ElGamalT, ID)in
out(C, (L1, L2));

(*** Phase de vote ***)
(* Le votant genere son n-uplet de vote *)
let EncVote=aenc(vote, M_ElGamalT) in
let EncA=aenc(A, M_ElGamalT) in
let Ar= power(A, r) in
let EncAr= aenc(Ar, M_ElGamalT) in
let g3x= power(g3, x) in
let Encg3x = aenc(g3x, M_ElGamalT) in
let Ox= power(o, x) in
let proofs= GenProofs(vote, EncA, EncAr, Encg3x, A, o) in
sync 1;
out(C, (EncVote, EncA, EncAr, Encg3x, Ox, proofs));
sync 2;

(*** Phase de comptage ***)
(* Les autorites de comptage verifient la validite des preuves de cha-
que n-uplet de vote *)
if verifyProof(proofs) = true then
(
        (* Si les preuves sont valides, les autorites de comptage veri-
        fient si le votant a deja vote *)
        if Duplicate(Ox)= false then
        (
                (* Finalement, elles verifient la validite des paramet-
                res d'authentification *)
                (* Si ces parametres sont valides, les autorites de com-
                ptage dechiffrent le vote *)
                if verifyCredential(EncA, EncAr, Encg3x)=true then
                (
                        let DecVote = adec(EncVote, M_ElGamalT') in
                        let result= incrementCounter(DecVote) in
                        out(C, result)
                )
        )
)
).

process
LokiVote(choice[V1, V2]) | LokiVote(choice[V2, V1])
```