N^o d'ordre : 4149 ANNÉE 2010

THÈSE DE DOCTORAT

Université de Rennes 1 / Université 7 Novembre à Carthage
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention : Informatique
École doctorale : Matisse

et

DOCTEUR DE L'ÉCOLE SUPÉRIEURE DES COMMUNICATIONS DE TUNIS
Mention : Technologie de l'information et de la communication

présenté par

Mehdi Talbi

Spécification et vérification automatique des propriétés des protocoles de vote électronique en utilisant la logique ADM

Thèse soutenue à Rennes
le 8 Octobre 2010

devant le jury composé de :

Kamel ADI
Professeur, Université du Québec en Outaouais / rapporteur

Riadh ROBBANA
Professeur, INSAT Tunis / rapporteur

Sihem GUEMARA EL FATMI
Professeur, SUPCOM Tunis / examinatrice

Sophie PINCHINAT
Professeur, IRISA / examinatrice

Adel BOUHOULA
Professeur, SUPCOM Tunis / directeur de thèse

Ludovic MÉ
Professeur, SUPÉLEC Rennes / directeur de thèse

Valérie VIET TRIEM TONG
Enseignant-Chercheur, SUPÉLEC Rennes / co-directeur de thèse

« *À mes parents* »
« *À mes soeurs Lilia et Amel* »
« *Et à leurs enfants Zohra, Nina, Ismael, Selma et Sara* »

Remerciements

Je tiens à remercier d'abord Ludovic Mé de m'avoir donné la chance de faire ma thèse à Supélec en m'acceptant dans son équipe de recherche SSIR. Le travail de recherche n'aura pas été le même sans l'ambiance et la bonne humeur dans les pauses cafés que tu as instaurées.

Je remercie bien évidemment Valérie Viet Triem Tong et Christophe Bidan d'avoir dirigé mes travaux de thèse et de m'avoir supporté pendant ces trois dernières années. Ce fut laborieux mais on a réussi. Je remercie également Benjamin Morin de m'avoir donné le premier l'opportunité de faire ma thèse à Supélec et d'avoir suivi mes travaux de recherche pendant ma première année. Je remercie aussi Adel Bouhoula d'avoir assuré mon encadrement à Sup'Com.

Je tiens à remercier Mohamed Mejri de m'avoir initié à la recherche et de m'avoir donné le goût de poursuivre avec une thèse de doctorat. Co-concepteur de la logique ADM, il est avant tout un ami très cher dont les conseils et idées m'ont largement inspiré dans l'accomplissement de ce travail.

Je remercie Kamel Adi et Riadh Robbana d'avoir accepté d'évaluer mes travaux de recherche, ainsi que Sophie Pinchinat d'avoir présidé le jury de thèse. Je remercie également Sihem Guemara d'avoir accepté de prendre part au jury.

Je remercie mon épouse Meriam Ben Ghorbel pour son soutien constant tout au long de cette thèse. Nos longues discussions que ce soit sur la modélisation des protocoles ou bien sur les propriétés de sécurité ont contribué à l'amélioration et à l'aboutissement du travail accompli durant cette thèse.

Je remercie et dédie cette thèse à mes parents.

Je remercie tous les amis que j'ai connus durant mes séjours à Rennes et avec qui j'ai noué de solides liens : Dali, Sofiène, Safa, Rayène & Insaf, Iyadh & Amira, Wiem (wiwi), Mohamed (gass), Neila, Anis et Hanieh.

Je remercie tous les thésards/stagiaires que j'ai côtoyés à Supélec. En vrac et en espérant n'oublier personne [1] : François, Milin, Olivier, Jonathan, Asma, Guillaume, Christopher, Stéphane, Ilhem, Adrien, Daniel, Ming, Laurent, Loïc, ...

1. Merci Milin pour la formule

Je remercie tous les membres de l'équipe SSIR pour leur disponibilité, leur soutien et pour l'accueil chaleureux qu'ils m'ont réservé. Je remercie également le personnel de Supélec et en particulier Myriam Andrieux, Clairette Place, Ophélie Morvan et Gregory Cadeau.

Finalement, je tiens à remercier le ministère de l'enseignement supérieur tunisien d'avoir financé ma première année de thèse, et l'institut français de coopération d'avoir pris le relais. Je tiens particulièrement à remercier Karine Elie, Mireille Dauvier et Imen Annabi pour leur gentillesse et leur accueil chaleureux.

Table des matières

Liste des tableaux

Table des figures

Introduction

Motivations et problématiques

Ces dernières années, plusieurs gouvernements (Allemagne, États-Unis, Corée du Sud) ont multiplié les expériences en vue d'adopter le vote électronique, tout en espérant réduire les coûts engendrés par les élections. L'automatisation du processus du vote électoral permet d'avoir les résultats plus rapidement, et de minimiser par la même occasion les risques d'erreurs dûs au facteur humain. Les systèmes de vote électronique se déclinent en deux versions : le vote en kiosque et le vote en ligne. Dans le premier cas, le votant est amené à se déplacer dans un bureau de vote où est mis à disposition un ensemble de terminaux permettant aux électeurs d'établir leurs préférences et de les émettre ensuite. Dans le cas de la seconde approche, tout votant disposant d'un accès au réseau Internet, peut participer à l'élection sans besoin de se déplacer de chez lui. Cette seconde forme suscite un intérêt particulier, étant donné qu'elle permettrait de faire face à un taux d'abstention de plus en plus élevé du fait de l'accessibilité grandissante au réseau Internet. Cependant, le pas à franchir n'est pas des plus simples étant donné les enjeux abrités par de tels systèmes d'un côté, et la sécurité des réseaux souvent mises en échec d'un autre côté. De plus les systèmes de vote électronique se doivent de garantir les objectifs des élections traditionnelles (anonymat, éligibilité, équité) tout en offrant un certain nombre de fonctionnalités supplémentaires (recomptage des voix). Les estoniens ont été les premiers à avoir épousé le vote par Internet en 2005 pour des élections locales. L'expérience a été réitérée ensuite en 2007, et tout récemment en 2009 où le taux de votes émis à distance a atteint un total de 9,5% [Com09].

De nos jours, le vote électronique suscite encore de la méfiance de la part de ses détracteurs du fait de l'opacité des systèmes proposés, de la non maturité de la technologie utilisée, et surtout des problèmes récurrents de sécurité. De plus, les soupçons de fraudes qui entachent souvent les élections électroniques ne font qu'attiser davantage les interrogations des électeurs. En 2003, la mise à disposition par accident du code source des machines de vote Diebold, a permis à une équipe de chercheurs d'analyser et d'établir un rapport accablant [KSRW04] sur les trous de sécurité du système de vote, largement déployé durant les élections présidentielles aux Etats-Unis en 2004. Parmi les attaques repertoriées, il était possible de reprogrammer la carte à puce donnant ainsi accès à de multiples votes.

De manière similaire au commerce électronique, les systèmes de vote en ligne sont basés sur des protocoles cryptographiques. Ces derniers consistent en des opérations de communication et de traitements internes permettant aux entités impliquées de faire usage de la cryptographie dans le but d'atteindre un objectif particulier : partage d'un secret, authentification mutuelle,

ou bien émission d'un vote. Dans la littérature, on distingue deux schémas de vote : ceux basés sur la signature en aveugle [Boy89, FOO92, Oka96, Oka97, JL97] et ceux basés sur le chiffrement homomorphe [Ben87, CGS97, Sch99, HS00, PAB+04]. Il s'agit de deux schémas, tentant de deux manières différentes de satisfaire les principales propriétés des protocoles de vote électronique : anonymat, vérifiabilité, éligibilité, équité, etc. Les propriétés à garantir sont assez complexes et certaines d'entre elles peuvent même sembler à première vue contradictoires. À titre d'exemple, la propriété de vérifiabilité individuelle stipule que le protocole doit permettre au votant de vérifier la prise en compte de son bulletin, tandis que la propriété sans reçu implique que les informations mises à disposition du votant ne doivent pas lui permettre de prouver le contenu de son bulletin.

Il est bien établi aujourd'hui que la conception des protocoles cryptographiques est difficile et sujette aux vulnérabilités. Ces failles ne sont pas uniquement dûes aux algorithmes cryptographiques sous-jacents mais également à la manière dont ils sont utilisés pour construire les protocoles, et à l'environnement supposé hostile dans lequel les objectifs de sécurité doivent être atteints. Pour garantir la correction d'un protocole, on raisonne toujours par rapport au pire scénario dans lequel on suppose que le réseau de communication est sous le contrôle d'un intrus capable d'usurper l'identité d'honnêtes participants, d'intercepter les messages, de les rejouer, ou bien d'en fabriquer de nouveaux, et de les injecter. Sous ces conditions il est bien difficile de déceler les éventuelles attaques mêmes pour les plus basiques des protocoles. Bien des protocoles ont été publiés, appliqués, puis plus tard démontrés faillibles à des attaques [CJ97]. L'exemple le plus parlant est celui du protocole d'authentification à clé publique proposé par Needham et Schroeder [NS78] démontré faillible dix-sept ans après sa publication à l'attaque de l'intercepteur (*man in the middle attack*) [Low95].

Dans ce contexte, l'usage des méthodes formelles pour montrer que les propriétés prétendues satisfaites par un protocole sont réellement garanties devrait être incontournable. Leur usage est d'autant plus important dans les applications critiques telles que le commerce électronique où l'argent est en jeu, ou bien le vote électronique où la démocratie est en jeu. Les méthodes formelles peuvent être définies comme un ensemble de formalismes mathématiques et/ou logiques pour la description des systèmes et de leurs spécifications, et de procédures permettant de s'interroger quant au respect de ces spécifications par le système. Leurs applications dans le domaine de spécification et d'analyse des protocoles cryptographiques remontent probablement aux travaux de Dolev et al. [DY83, DEK82] mais n'ont réellement émergées qu'au début des années quatre-vingt dix avec la logique BAN [BAN90]. De nos jours, on dénombre une variété de méthodes et d'outils associés à la vérification formelle des protocoles cryptographiques. Notre travail dans cette thèse est axé principalement sur la spécification formelle des propriétés des protocoles de vote électronique, à la modélisation de ces protocoles, et à la vérification automatique des propriétés de sécurité spécifiées par rapport au modèle du protocole établi.

Objectifs et contributions

La littérature est riche en travaux portant sur la vérification formelle des protocoles cryptographiques, cependant, très peu d'entre eux portent sur l'analyse des protocoles de vote électronique. Force est de constater que la principale technique employée pour la vérification de tels protocoles repose sur du *model checking*. Notre approche repose également sur du *model checking* consistant en la vérification de la relation de satisfaction $\mathscr{M} \models \phi$, où $\mathscr{M}$ désigne un mo-

dèle à base de traces, et ϕ une propriété de sécurité exprimée au moyen de la logique modale ADM [ADM03]. Plus précisément, notre contribution dans cette thèse est triple : (i) proposer tout d'abord un ensemble de notations pour la spécification et la modélisation des protocoles de vote ($\mathscr{M}$), (ii) spécifier ensuite les principales propriétés de sécurité (ϕ), et (iii) implémenter finalement un outil pour la vérification automatique des propriétés spécifiées par rapport au modèle défini ($\mathscr{M} \models \phi$). Chacune de ces étapes est validée par rapport à une étude de cas portant sur le protocole de vote FOO [FOO92].

Notre première contribution dans cette thèse concerne la modélisation des protocoles de vote électronique. Ces derniers font appel à des primitives cryptographiques avancées (schéma d'engagement, signature en aveugle, chiffrement homomorphe), et supposent l'existence de canaux de communication (anonymes, privés) plus élaborés que ceux habituellement employés par les protocoles cryptographiques classiques (authentification, échange de clés). Afin d'affiner l'analyse des propriétés de sécurité, il est primordial de disposer d'une représentation simple des protocoles qui tient compte des spécificités introduites par le vote électronique. Notre objectif a été d'enrichir dans un premier temps les notations existantes afin de représenter au mieux les protocoles de vote électronique, et de les utiliser ensuite afin de définir le modèle par rapport auquel seront vérifiées ultérieurement les propriétés de sécurité. Dans notre approche, nous avons opté pour un modèle à base de traces, où une trace représente une exécution dynamique du protocole. Ce choix est motivé par le fait qu'une faille dans un protocole cryptographique est souvent mise en évidence par une trace représentant la violation d'une propriété donnée. Notre modélisation des protocoles tient compte de la présence d'un intrus actif dont le pouvoir de déduction a été ajusté conformément à l'algèbre de messages que nous avons défini.

La seconde partie de la thèse est relative à la spécification des propriétés de sécurité, et constitue la majeure contribution de notre travail [TMT+08a, TMT+08b, TTB09]. Cette partie consiste en la spécification au moyen de la logique ADM d'une sélection de propriétés de sécurité : droit d'expression, éligibilité, non réutilisabilité, exactitude, équité, anonymat et sans-reçu. La vérification ensuite de ces propriétés par rapport au modèle du protocole FOO, construit à l'issue de la phase de modélisation, nous a permis de montrer la faillibilité de ce protocole quant à la satisfaction de certains des objectifs du vote électronique. Le choix de la logique ADM comme langage de formalisation des propriétés de sécurité a été motivé par le fait que ses constructeurs s'interprètent par rapport à des traces, et qu'elle offre un certain nombre de caractéristiques intéressantes pour l'analyse de ces traces (modalités temporelles et existentielles, récursivité, linéarité). De plus, la logique est dotée d'un système de preuve basée sur la méthode des tableaux [GLM97] permettant une implémentation efficace d'un outil automatisant la tâche de la vérification formelle des propriétés spécifiées.

La troisième et ultime partie de la thèse a été consacrée à l'implémentation de la sémantique associée à la logique ADM afin de permettre la vérification automatique des propriétés de sécurité. Cet outil, dont nous avons entamé le développement dans [GTM07, TTM08], est dénommé ADM-Checker. Il offre à l'utilisateur final la possibilité de spécifier, par l'intermédiaire d'une interface graphique conviviale, les propriétés de sécurité et leurs paramètres associés, de vérifier ensuite leur syntaxe conformément à la grammaire de la logique ADM, et de les vérifier par la suite par rapport à des traces du protocole analysé. L'outil ADM-Checker permettra notamment de confirmer les résultats de sécurité énoncés à l'issue de la phase de spécification des propriétés.

Organisation du mémoire

La suite de cette thèse est organisée de la manière suivante.

Tout d'abord, le premier chapitre introduit les protocoles de vote électronique. Nous présentons en premier lieu les propriétés de sécurité qu'un protocole de vote se doit de garantir. Nous donnons ensuite un panorama des principaux concepts et schémas cryptographiques utilisés dans le but d'atteindre ces objectifs de sécurité. Nous donnons finalement, dans chaque classe de protocole de vote identifiée, des exemples de protocoles. Nous détaillons en particulier les étapes du protocole FOO, objet de notre étude de cas tout au long de ce manuscrit de thèse.

Le chapitre 2 établit d'abord un état des lieux des principaux formalismes utilisés dans l'analyse des protocoles de vote électronique. Cet état de l'art est organisé en deux parties : la première partie est dédiée aux algèbres de processus, et la seconde est consacrée aux logiques modales. Pour chacune de ces approches, nous donnons des exemples illustratifs, et discutons des principaux travaux formels ciblant la vérification des protocoles de vote électronique. Nous introduisons ensuite la logique ADM par l'intermédiaire de sa syntaxe et sa sémantique et motivons son intérêt dans la vérification des protocoles de vote électronique.

Le chapitre 3 est dédié à la modélisation ($\mathscr{M}$) des protocoles de vote électronique. Ce chapitre introduit le modèle du protocole que nous avons adopté, et face auquel seront vérifiées ensuite les propriétés de sécurité spécifiées. Ce modèle tient compte d'un intrus dont nous avons ajusté le pouvoir conformément aux spécificités introduites par le vote électronique. Il sera particulièrement question de la modélisation du protocole FOO dont nous proposons une analyse dans le présent manuscrit.

Le chapitre 4 est réservé à la spécification des propriétés (ϕ) des protocoles de vote électronique. Nous donnons la spécification, au moyen de la logique ADM, d'une sélection de propriétés de sécurité (équité, démocratie, exactitude, etc.). Notre formalisation est adaptée particulièrement au protocole FOO dont nous proposons une analyse détaillée de sa sécurité en conclusion de ce chapitre.

Le chapitre 5 présente l'outil ADM-Checker dont la fonctionnalité principale est la vérification des propriétés spécifiées (ϕ) par rapport au modèle du protocole ($\mathscr{M}$). Le chapitre est divisé en deux parties : la première partie introduit le système de preuve qui accompagne la logique ADM, et décrit son mode opératoire, dans la vérification de la relation de satisfaction ($\mathscr{M} \models \phi$), à travers deux exemples illustratifs. La seconde partie présente l'architecture et les fonctionnalités de l'outil ADM-Checker.

Finalement, nous concluons cette thèse par un récapitulatif des principales réalisations, et par la proposition d'une série de perspectives permettant de compléter et d'améliorer davantage les travaux accomplis durant cette thèse.

Chapitre 1

Protocoles de vote électronique

Le vote électronique se décline en deux versions : le vote en kiosque et le vote en ligne. Dans le premier cas, le votant est amené à se déplacer dans un bureau de vote où est mis à disposition un ensemble de terminaux permettant aux votants d'effectuer leurs choix et de les émettre ensuite. Dans la seconde approche, tout votant disposant d'un accès au réseau Internet, peut participer à l'élection sans besoin de se déplacer de chez lui. Bien que la seconde approche semble plus attrayante, elle doit faire face aux problèmes de sécurité inhérents au réseau Internet (en particulier, le déni de service).

De manière similaire au commerce électronique, les systèmes de vote en ligne sont basés sur des protocoles cryptographiques. Ces derniers consistent en des opérations de communication permettant aux entités impliquées de faire usage de la cryptographie dans le but d'atteindre un objectif particulier : partage d'un secret, authentification mutuelle, ou bien émission d'un vote. Dans la littérature on distingue deux schémas de vote : ceux basés sur la signature en aveugle [Boy89, FOO92, Oka96, Oka97, JL97] et ceux utilisant le chiffrement homomorphe [Ben87, CGS97, Sch99, HS00, PAB+04]. Il s'agit de deux schémas tentant de deux manières différentes de satisfaire les principales propriétés des protocoles de vote (anonymat, équité, éligibilité).

Dans ce qui suit, nous présentons en premier lieu les propriétés de sécurité qu'un protocole de vote doit satisfaire. Ensuite, nous introduisons les principales primitives cryptographiques ainsi que les canaux de communication impliqués dans les protocoles de vote. Finalement, nous présentons par l'intermédiaire d'exemples de protocoles, les schémas de vote existants.

1.1 Propriétés de sécurité

Les protocoles de vote se veulent une représentation la plus fidèle possible des élections papier traditionnelles tout en offrant un certain nombre de fonctionnalités supplémentaires tels que le recomptage des votes. Il est ainsi crucial que le protocole de vote réponde à un certain nombre de propriétés de sécurité. Ces propriétés dépendent fortement de la nature de l'élection et des lois constitutionnelles de chaque gouvernement. À titre d'exemple, il n'est pas requis que le vote soit anonyme au Royaume-Uni ou en Nouvelle Zélande [JP06] alors que c'est le cas dans la majorité des autres pays.

Un rapport établi par l'IPI (*Internet Policy Institute*) identifie un ensemble de propriétés de sécurité qu'un protocole de vote électronique se doit de satisfaire. Dans [Saf00, Gri02, CC97, Cet08], il est fait mention de propriétés similaires (ou adaptées). Nous donnons ci-après une description de ces propriétés :

1.1.1 Anonymat

La propriété d'anonymat signifie que le lien entre le votant et son vote ne peut être établi par une tierce personne (observateur).

1.1.2 Démocratie

Cette propriété englobe trois sous propriétés :

- **Éligibilité** (*eligibility property*). Cette propriété exprime le fait que seuls les votants légitimes (inscrits sur les listes électorales) peuvent voter.
- **Droit d'expression**. Cette propriété garantit que chaque votant légitime peut participer à l'élection et voter.
- **Non réutilisabilité** (*non-reusability property*). Cette propriété signifie qu'un votant légitime ne peut voter plus d'une fois. Une variante de cette dernière est appelée **unicité** (*uniqueness property*). Dans ce cas, le votant peut participer à plusieurs reprises à la phase de vote, mais seul un vote par votant doit être comptabilisé (par exemple, le dernier émis).

1.1.3 Exactitude

La propriété d'exactitude (*accuracy property*) exprime le fait que les résultats publiés doivent correspondre exactement aux choix des électeurs. Le protocole doit ainsi garantir que tous les votes légitimes ont été correctement comptabilisés, qu'aucun des votes n'a été dupliqué, modifié, ou supprimé, et qu'aucun faux bulletin n'a pu être inséré par une entité malicieuse.

1.1.4 Équité

La propriété d'équité (*fairness property*) signifie que le protocole de vote ne doit pas permettre la divulgation de résultats partiels. De tels résultats pourraient influencer le choix des votants qui ne se sont pas encore exprimés.

1.1.5 Vérifiabilité

Dans la littérature, on distingue deux formes de vérifiabilité :

- **Vérifiabilité individuelle** (*individual verifiability property*). Elle exprime le fait que le protocole doit permettre au votant de vérifier que son vote a été réellement pris en compte.
- **Vérifiabilité universelle** (*universal verifiability property*). Elle signifie que tous les votants peuvent être convaincus de la sincérité du scrutin. Cette propriété peut être vue comme la prouvabilité de la propriété d'exactitude.

1.1.6 Sans-Reçu

La propriété sans-reçu (*receipt-freeness property*) a été introduite par Benaloh et Tuinstra dans [BT94]. Elle exprime le fait que le protocole ne doit pas fournir de reçu permettant au votant de prouver à une tierce personne qu'il a voté comme convenu. Cette propriété permet d'empêcher la vente de vote. Une version plus robuste de la propriété est appelée **résistance à la coercition** (*coercion-resistance property*) [DKR06, JCJ05] et qui signifie que le protocole doit également empêcher le vote sous pression même si on considère que le votant interagit durant certaines phases du protocole avec la personne le forçant à voter d'une certaine manière.

1.1.7 Propriétés auxiliaires.

Des propriétés d'ordre pratique ont également été identifiées pour les systèmes de vote électronique. À titre d'exemple, nous pouvons citer :

- **Flexibilité** (*flexibility property*). Le protocole de vote doit pouvoir s'utiliser pour diverses types d'élection.
- **Commodité** (*convenience property*). Les votants doivent pouvoir voter sans besoin de compétences particulières.
- **Mobilité** (*mobility property*). Le votant n'est pas restreint à une localisation physique particulière pour émettre son choix.

1.2 Primitives cryptographiques

Contrairement aux protocoles cryptographiques classiques (échange de clés, authentification), les protocoles de vote électronique font appel à des primitives cryptographiques plus élaborées. Ci-après est donné un panorama des principales fonctions utilisées dans les protocoles de vote électronique.

1.2.1 Systèmes cryptographiques

Un système cryptographique (ou cryptosystème) peut être défini comme un triplet $(\mathscr{K}, \mathscr{E}, \mathscr{D})$ où $\mathscr{K}$ est un algorithme de génération de clés, $\mathscr{E}$ est un algorithme de chiffrement permettant de transformer un texte clair en un texte inintelligible, et $\mathscr{D}$ est un algorithme de déchiffrement permettant de retrouver le texte en clair à partir du chiffré et de la clé de chiffrement. On distingue deux grandes familles de cryptosystèmes : les systèmes symétriques ou à clé secrète (DES (*Data Encryption Standard*) [46-99]), AES (*Advanced Encryption Standard*) [19701]) et les systèmes asymétriques ou à clé publique (RSA [RSA78], El-Gamal [Gam85]). Les systèmes symétriques font appel à la même clé pour le chiffrement et le déchiffrement. Dans le cas des systèmes asymétriques, une clé est utilisée pour le chiffrement (clé publique) et une seconde est utilisée pour le déchiffrement (clé privée).

Ci-après, nous nous intéressons particulièrement aux systèmes asymétriques que nous présentons via les cryptosystèmes RSA et El-Gamal. Ces systèmes serviront de base pour introduire des concepts cryptographiques plus avancés tels que la signature en aveugle et le chiffrement homomorphe.

Cryptosystème RSA

Génération des clés ($\mathscr{K}$). Soient p et q deux "grands" nombres premiers distincts. Le processus de génération suit les étapes suivantes :

1. Calcul du module RSA $n = p \times q$.
2. Calcul de la fonction d'Euler $\phi(n) = (p-1) \times (q-1)$.
3. Choix d'un exposant de chiffrement e tel que $pgcd(e, \phi(n)) = 1$ et $e < \phi(n)$.
4. Calcul de l'exposant de déchiffrement d tel que $e \times d \equiv 1[\phi(n)]$.

La clé publique est donnée par la paire (e, n), et la clé privée est d.

Chiffrement ($\mathscr{E}$). Le chiffrement du message m est calculé comme suit :

$$c = m^e \, mod \, n$$

Déchiffrement ($\mathscr{D}$). Le déchiffrement du cryptogramme c est donné par :

$$m = c^d \, mod \, n$$

Cryptosystème El-Gamal

Génération des clés ($\mathscr{K}$). Soit p un "grand" nombre premier. La génération des clés suit les étapes suivantes :

1. Choix d'un générateur $g \in Z_p$ (ensemble des entiers modulo p).
2. Choix d'une clé secrète s telle que $0 < s < p-1$.
3. Calcul de $h = g^s \, mod \, p$.

La clé publique est donnée par le triplet (p, g, h). La clé privée est s.

Chiffrement ($\mathscr{E}$). Le chiffrement du message m est calculé comme suit :

1. Choix d'un nombre aléatoire k tel que $pgcd(k, p-1) = 1$.
2. Calcul du couple $(c_1, c_2) = (g^k \, mod \, p, m \times h^k \, mod \, p)$.

Déchiffrement ($\mathscr{D}$). Pour retrouver le message m à partir du cryptogramme $c = (c_1, c_2)$, on procède comme suit :

$$m = \frac{c_2}{c_1^s} \, mod \, p$$

Le cryptosystème El-Gamal a ses avantages dans un système de vote électronique. L'algorithme de chiffrement étant probabiliste, le chiffré d'un choix donné ("oui", "non") peut prendre plusieurs formes. Ceci permet d'éviter qu'un attaquant puisse retrouver le vote en clair par comparaison du chiffré observé au chiffrement des différentes possibilités de choix.

1.2.2 Schéma d'engagement

Un schéma d'engagement (*bit-commitment scheme*) est un protocole faisant intervenir deux entités V et A se trouvant face au problème suivant :

1. V souhaite communiquer un bit de donnée b à A mais désire que ce dernier ne puisse en connaître la valeur qu'ultérieurement (*concealing property*).
2. De son côté, A désire que V ne puisse tricher plus tard sur la valeur gagée (*binding property*).

Pour résoudre ce problème, V et A peuvent dérouler le protocole illustré par la figure 1.1. V chiffre d'une certaine manière l'information à transmettre, et envoie le résultat à A (étape 1 du protocole). Ce dernier pourra prendre connaissance de la valeur gagée lorsque l'entité V lui fournira la clé de chiffrement (étape 2).

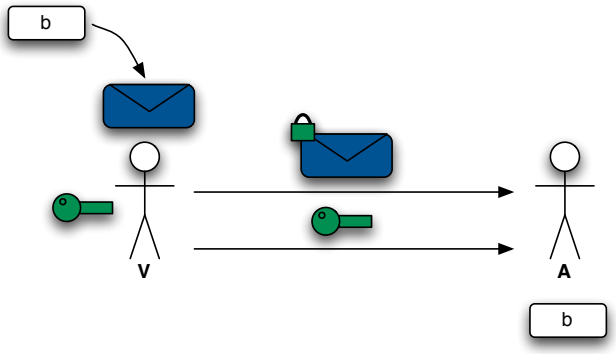


FIGURE 1.1 – Schéma d'engagement

Dans un système de vote, un schéma d'engagement s'avère utile lorsque l'on souhaite garantir une propriété telle que celle de l'équité. Les étapes 1 et 2 correspondent dans ce cas respectivement à la phase de vote et à la phase d'ouverture des bulletins. Les entités V et A jouent respectivement le rôle du votant et de l'entité responsable de la collecte des bulletins. Finalement, b encode le vote dans le cas d'un simple référendum (0 pour "non" et 1 pour "oui").

Lorsque l'entité V peut tricher sur la valeur gagée, le schéma d'engagement est dit à trappe (*trapdoor bit-commitment*). Notons que l'existence d'une trappe peut s'avérer utile si l'on souhaite empêcher la coercition. Les protocoles proposés par Okamoto [Oka97] reposent en partie sur un tel schéma. L'idée est de donner au votant la possibilité d'ouvrir son bulletin de différentes manières, et ainsi faire croire à une tierce personne C (*coercer*) qu'il a voté $1-b$ au lieu de b. Ceci peut être mis en évidence par l'exemple suivant extrait de [Oka96] :

Soit la fonction de mise en gage $\mathscr{E}()$ donnée ci après, où g et G sont des générateurs de Z_p, et k la clé de chiffrement :

$$\mathscr{E}(b,k) = g^k G^b$$

La valeur b peut être retrouvée au moyen de la clé k en calculant la fraction $r = \frac{\mathscr{E}(b,k)}{g^k}$:

$$b = \begin{cases} 1 & \text{si} \quad r = G \\ 0 & \text{si} \quad r = 1 \end{cases}$$

Disposant du logarithme discret de G en base g ($\alpha = log_g(G)$), le votant V peut faire croire à C qu'il a voté comme convenu. À titre d'exemple, si le votant a voté "oui", et veut faire croire à C le contraire, il lui transmet une fausse clé $k' = k + \alpha$. En calculant la fraction $r = \frac{\mathscr{E}(b,k)}{g^{k'}}$, l'entité C est induite en erreur étant donné qu'elle en conclut par ce résultat ($r = 1$) que V a voté "non".

Il est à noter que le schéma d'engagement peut aisément être étendu à une séquence de bits [Rja02].

1.2.3 Signature en aveugle

La signature numérique est un procédé permettant d'assurer l'intégrité de l'information transmise et de garantir son authenticité. Dans un cryptosystème à clé publique, la signature d'un message est obtenue par son chiffrement au moyen de la clé privée. Dans le système RSA décrit précédemment, la signature d'un message m est simplement donnée par [1] :

$$s = m^d \, mod \, n$$

Dans le cadre des protocoles de vote, la signature en aveugle entre en jeu lorsqu'une entité V (votant) désire obtenir une signature sur un message m (bulletin) de la part d'une autorité A sans rien révéler du contenu de son message. Pour arriver à ses fins, l'entité V commence par masquer le message (il produit un message m') avant de le transmettre à A. L'autorité A signe en aveugle le message reçu et le renvoie à V qui extrait alors une signature sur m à partir de la signature sur le message masqué m'. Cette procédure peut être représentée par la figure 1.2 où le votant V glisse son bulletin dans une enveloppe accompagné d'un papier carbone. La signature apposée sur la partie externe de l'enveloppe est retranscrite grâce au papier carbone sur le bulletin.

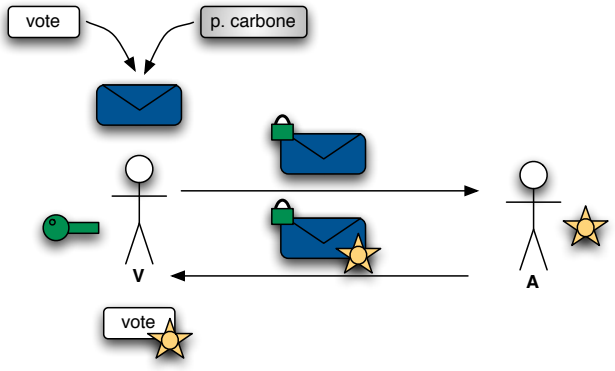


FIGURE 1.2 – Signature en aveugle

1. Dans la pratique, une fonction de hachage à sens unique est appliquée sur le message avant son chiffrement au moyen de la clé privée.

À titre d'exemple, soit $((e,n),d)$ la paire de clés publique/privée RSA associée à l'autorité A. Le votant V peut obtenir une signature sur son message m en procédant comme suit :

1. Le votant masque le message m à l'aide d'un facteur r choisi aléatoirement. Il prépare le message $m' = m \times r^e \, mod \, n$.
2. L'autorité A signe le message et renvoie la signature $s' = m'^d \, mod \, n$ au votant.
3. Le votant calcule la fraction $\frac{s'}{r}$ afin d'en extraire la signature $s = m^d \, mod \, n$.

1.2.4 Chiffrement homomorphe

L'homomorphisme est une propriété de certains cryptosystèmes tels que RSA, Paillier [Pai99] ou BGN [BGN05] selon laquelle la primitive de chiffrement $\mathscr{E} : \mathscr{P} \rightarrow \mathscr{C}$ satisfait la relation définie ci-après, avec $\odot$ et $\otimes$ les lois de composition interne régissant respectivement les groupes $\mathscr{P}$ et $\mathscr{C}$:

$$\mathscr{E}(m_1 \odot m_2) = \mathscr{E}(m_1) \otimes \mathscr{E}(m_2)$$

L'exemple donné ci-après illustre la propriété d'homomorphisme dans le cas particulier du cryptosystème El-Gamal :

$$\begin{aligned} \mathscr{E}(m_1,k_1) \times \mathscr{E}(m_2,k_2) &= (g^{k_1} \, mod \, p, m_1 \times h^{k_1} \, mod \, p) \times (g^{k_2} \, mod \, p, m_2 \times h^{k_2} \, mod \, p) \\ &= (g^{k_1+k_2} \, mod \, p, m_1 \times m_2 \times h^{k_1+k_2} \, mod \, p) \\ &= \mathscr{E}(m_1 \times m_2, k_1 + k_2) \end{aligned}$$

Il s'agit là d'un homomorphisme multiplicatif vérifiant l'équation sur les messages suivante :

$$\prod_{i=1}^{n} \mathscr{E}(m_i) = \mathscr{E}(\prod_{i=1}^{n} m_i) \tag{1.1}$$

Le cryptosystème de Paillier met en évidence quant à lui un homomorphisme additif où le produit des messages chiffrés équivaut au chiffrement de la somme des messages en clair :

$$\prod_{i=1}^{n} \mathscr{E}(m_i) = \mathscr{E}(\sum_{i=1}^{n} m_i) \tag{1.2}$$

Dans un système de vote électronique, la propriété d'homomorphisme intervient dans le comptage des voix et est exploitée de la manière suivante : tout d'abord, les votes chiffrés transmis par les votants sont multipliés entre eux. Ensuite, le résultat de l'élection (somme des votes dans le cas d'un homomorphisme additif) est obtenu par déchiffrement du produit des votes chiffrés. À titre d'exemple, le nombre de personnes ayant approuvé une proposition lors d'un référendum peut être directement obtenu à partir du produit des votes chiffrés, si les choix possibles sont encodés de la manière suivante : 0 pour "non" et 1 pour "oui". Il est à noter que le fait que les bulletins ne soient pas déchiffrés individuellement, participe à garantir une propriété telle que l'anonymat.

1.2.5 Preuves à divulgation nulle de connaissance

Une preuve interactive à divulgation nulle de connaissance ZKIP (*Zero-Knowledge Interactive Proof*) permet à une entité A de convaincre une entité B qu'une proposition est vraie ("Je dispose d'une clé secrète s") sans toutefois compromettre les éléments sur lesquels cette proposition est fondée. Les éléments de preuve exposés à B n'apprennent rien de plus que la véracité de la proposition. Les preuves constituent des éléments essentiels dans les protocoles de vote électronique, et en particulier dans ceux faisant appel au chiffrement homomorphe. À titre d'exemple, dans le protocole CGS [CGS97], le votant est amené à prouver que son vote est valide (le vote est un chiffrement de 0 ou de 1) sans révéler le contenu de son bulletin. Les preuves jouent également un rôle dans le but de garantir des propriétés de sécurité telles que celle de la vérifiabilité universelle. Toujours dans ce même exemple de protocole CGS, les autorités gérant l'élection sont amenées à fournir des preuves garantissant la fiabilité des résultats publiés. Plus précisément, cette preuve est basée sur l'égalité des logarithmes discrets, et est énoncée comme exemple dans la Table 1.1 [2]. Dans cet exemple, l'entité A prouve à l'entité B qu'elle dispose de la valeur $\alpha \in Z_p$ satisfaisant l'égalité $(x = g^{\alpha}, y = h^{\alpha})$ avec $x, y, g, h \in Z_p$ des éléments publiques à disposition également de B.

TABLE 1.1 – Preuve interactive basée sur l'égalité des logarithmes discrets

Éléments de Preuve :	$[(x, y, g, h)] / [(x = g^{\alpha}) \wedge (y = h^{\alpha})]$
Preuve (de A) :	$[log_g(x) = log_h(y)]$
1.	A $\longrightarrow$ B : $(a = g^w, b = h^w)$
2.	B $\longrightarrow$ A : c
3.	A $\longrightarrow$ B : $(r = w + \alpha c)$
Vérification (par B) :	$[(g^r = a \times x^c) \wedge (h^r = b \times y^c)]$

Dans la littérature, il existe une variété de preuves interactives. Elles sont souvent basées sur le même modèle : (engagement (e), challenge (c), réponse (r)). L'entité A s'engage d'abord à fournir une preuve de la proposition p au moyen de l'élément e. L'entité B met au défi A en lui proposant d'inclure le challenge c dans la preuve. A fournit alors la preuve représentée par la réponse r calculée sur la base des éléments p, e et c. Les preuves de cette nature peuvent se transformer en preuves non interactives si l'entité A se met au défi par elle même. Ce défi peut être généré en employant la technique proposée par Fiat et Shamir dans [FS86]. Dans ce cas, le défi c est obtenu par application d'une fonction de hachage sur les éléments p et e ($c = h(p, e)$). Une version non interactive du protocole présenté en Table 1.1 est donnée dans [Rja02].

1.2.6 Schéma de partage de secret

Un schéma de partage de secret (t, n) permet à une entité T (autorité de confiance, votant) de partager un secret s (clé privée, vote) entre n autorités de telle sorte que le secret s ne peut être reconstruit que par la coopération d'un minimum de t entités. Ci-dessous, est décrit le schéma proposé par Shamir dans [Sha79] :

2. Le protocole énoncé par la Table 1.1, et dû à Chaum et Pedersen [CP92], n'est à divulgation nulle de connaissance que si l'entité B, responsable de la vérification de la preuve, est une entité de confiance. La sécurité du protocole est discuté plus en détail par les auteurs du protocole de vote CGS qui optent pour l'usage de la version non interactive de la preuve.

1. L'entité T génère aléatoirement un polynôme $p(x)$ de degré $t-1$. Le polynôme n'est pas divulgué et le secret s est la valeur du polynôme au point 0 ($s = p(0)$) :
$$p(x) = \sum_{k=0}^{t-1} \alpha_k x^k$$
2. L'entité T transfère ensuite à chaque autorité A_j ($j \in \{1 \dots n\}$) sa part du secret représentée par :
$$s_j = p(j)$$
3. Un ensemble minimal $\Delta_t \subset \{1 \dots n\}$ de t parts (fragments) du secret est nécessaire pour reconstruire le secret s par interpolation de Lagrange :
$$s = p(0) = \sum_{j \in \Delta_t} s_j \times L_{j,\Delta_t} \qquad L_{j,\Delta_t} = \prod_{k \in \Delta_t \setminus \{j\}} \frac{k}{k-j}$$

Une modification apportée par Schoenmakers dans [Sch99] permet de transformer le schéma (t, n) en un schéma vérifiable. Plus précisément, des preuves basées sur l'égalité des logarithmes discrets sont introduites afin d'assurer que l'entité T distribue des parts de secret valides, et permettant de garantir que la reconstruction du secret par les autorités se fait au moyen de fragments valides.

1.3 Communication

Dans les protocoles cryptographiques classiques, la communication est souvent assurée par des canaux publiques où l'information échangée transite par l'intrus. Afin de garantir certaines des propriétés décrites plus haut (anonymat, sans-reçu), les protocoles de vote électronique supposent l'existence de canaux de communication plus élaborés ayant des caractéristiques particulières (anonymat de l'expéditeur, sécurité physique de la communication). Dans ce qui suit, nous présentons les canaux usuellement utilisés dans ces protocoles.

1.3.1 Canal publique

Il s'agit du canal de communication "standard" utilisé dans les protocoles cryptographiques. Le canal est non sûr, et est sous le contrôle d'un intrus ayant le pouvoir d'intercepter tout message transitant via le canal.

1.3.2 Canal anonyme

Il s'agit d'un canal où l'information en transit peut être interceptée par intrus, mais ne peut être tracée pour remonter à l'identité de l'émetteur. Cette caractéristique peut être obtenue au moyen des réseaux de mélangeurs (*Mix_Net*) introduits par Chaum dans [Cha81]. L'idée repose sur un cryptosystème à clé publique et consiste à intercaler un ensemble de n entités entre les éventuels émetteurs et destinataires des messages. Lorsqu'une entité V désire envoyer un message v à une entité A, elle le chiffre successivement avec chacune des clés publiques pk_i ($i = 1 \dots n$) des entités MX_i constituant le *Mix_Net*, et le transmet ensuite au réseau de mélangeurs qui attend la réception d'une série de messages avant d'entamer le "mélange". Ce processus se déroule au niveau de chaque étage MX_i du *Mix_Net* selon le principe suivant :

1. MX_i choisit une permutation p_i et l'applique à la liste des éléments reçue de la part de MX_{i-1}.
2. MX_i enlève ensuite une couche de déchiffrement de chacun des éléments constituant la liste. Elle déchiffre les messages au moyen de sa clé privée sk_i.
3. MX_i transmet finalement la liste ainsi produite à l'étage MX_{i+1}.

Ce procédé est illustré par la figure 1.3 où les votes v_1, v_2, et v_3 propres respectivement aux votants V_1 , V_2, et V_3 sont transmis de manière anonyme par l'intermédiaire d'un réseau de mélangeurs à trois entités. Plus précisément, le schéma placé en haut de la figure représente le chiffrement de chaque vote v_j ($j = 1,2,3$) successivement au moyen des clés publiques pk_i ($i = 1,2,3$) propres aux différents étages MX_i composant le *Mix_Net*. La figure du bas représente le passage des messages ainsi préparés par les votants au travers du *Mix_Net*. Une couche de déchiffrement et une permutation sont appliquées au niveau de chaque entité MX_i. L'anonymat des votants est garanti étant donné qu'un observateur externe (un intrus) ne peut faire correspondre les éléments des listes fournis en entrée à ceux produits en sortie.

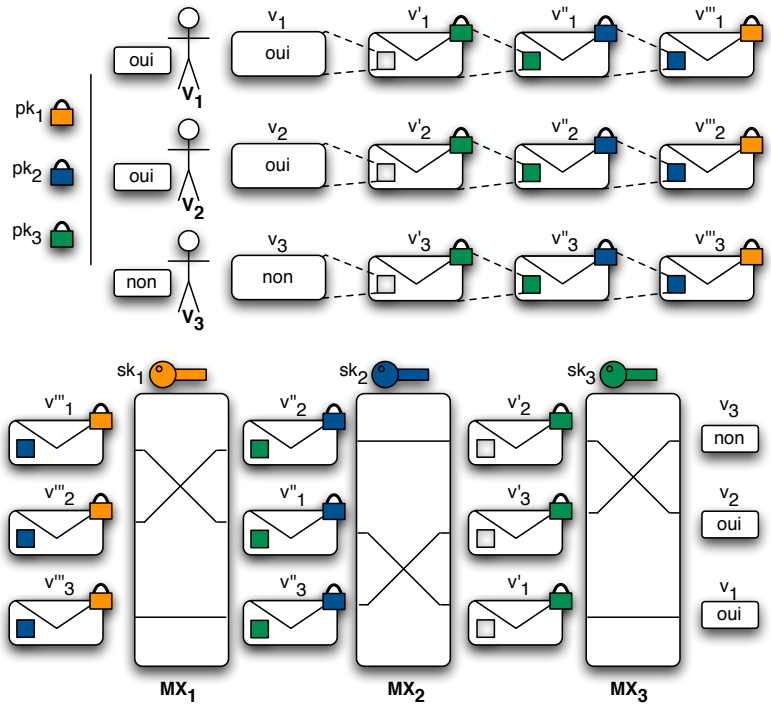


FIGURE 1.3 – Réseaux de mélangeurs

1.3.3 Canal privé

Un canal privé est un canal où l'information en transit est hors de portée de l'intrus. Plusieurs protocoles de vote [BT94, SK95, Oka97, HS00] en supposent l'existence dans le but d'empêcher la coercition et la vente de vote. Certains d'entre eux (schémas de vote proposés dans [Oka97]) nécessitent même que le canal soit également anonyme.

1.4 Schémas de vote électronique

Les protocoles de vote électronique peuvent être classés en deux catégories : ceux basés sur la signature en aveugle, et ceux utilisant un schéma de chiffrement homomorphe. La première catégorie de protocoles est plus adaptée pour des élections à grande échelle. À l'inverse, les protocoles appartenant à la seconde famille impliquent souvent des calculs complexes, et par conséquent souvent applicables que pour des petites élections. Cependant, ils permettent d'aboutir à des objectifs de sécurité (vérifiabilité universelle) difficile à atteindre dans les protocoles faisant appel à la signature en aveugle. Dans ce qui suit, nous présentons, au travers d'exemples de protocoles, les principes de base utilisés dans chacun de ces schémas.

1.4.1 Schémas basés sur la signature en aveugle

Les protocoles utilisant la signature en aveugle suivent le même principe : le votant contacte dans un premier temps l'autorité gérant l'élection afin d'obtenir une signature sur un message préalablement masqué par le votant. La signature en aveugle apposée par l'autorité représente le jeton permettant au votant de poster son vote. Durant la phase de vote, le jeton est émis via un canal anonyme et ne doit contenir aucune information (pseudonyme) permettant de remonter à l'identité de l'émetteur.

Le protocole FOO

Dans cette section, nous présentons le protocole FOO [FOO92] du nom de ces auteurs Fujioka, Okamoto et Ohta.

Entités. Le protocole FOO fait intervenir trois profils :

- Les votants $V_i \in L_v, i \in \{1 \dots n\}$. L_v représente la liste des votants légitimes
- Un administrateur A dont le rôle est de fournir aux votants légitimes les jetons (signatures en aveugle) leurs permettant d'émettre leurs votes.
- Un collecteur C responsable de la collecte des votes et de la publication du résultat final de l'élection.

Nous considérons une entité supplémentaire B jouant le rôle du babillard (*bulletin board*). Plus précisément, cette entité est responsable de l'affichage des résultats publiées par le collecteur C.

Primitives cryptographiques. Le protocole FOO emploie les primitives cryptographiques suivantes :

- **Engagement.** La mise en gage d'un message m par la clé r est notée $\mathscr{C}(m,r)$.
- **Aveuglement.** Le masquage d'un message m par la clé b est représentée par $\mathscr{B}(m,b)$.
- **Signature numérique.** La signature du message m par l'autorité A est représentée par $\mathscr{S}(m,sk_A)$, avec (pk_A, sk_A) la paire de clés publique/privée associée à l'administrateur A. De même, à chaque votant $V_i \in L_v$ correspond une paire de clés (pk_{V_i}, sk_{V_i}).

Canaux de communication. Les entités du protocole FOO peuvent communiquer via deux types de canaux :

- Canaux publiques servant à l'enregistrement des votants et à la publication des résultats.
- Canaux anonymes établis entre les votants et l'entité responsable de la collecte des votes.

Étapes du protocole. Le protocole FOO se déroule en six étapes :

1. **Préparation.** Le votant V_i choisit tout d'abord parmi la liste des choix possibles L_c son vote v_i. Il le met ensuite en gage au moyen d'une clé aléatoire r_i ($x_i = \mathscr{C}(v_i, r_i)$). Le vote gagé est par la suite aveuglé au moyen d'une deuxième clé aléatoire b_i ($e_i = \mathscr{B}(x_i, b_i)$), et le tout est signé avec la clé privée du votant sk_{V_i} ($s_i = \mathscr{S}(e_i, sk_{V_i})$). La signature est finalement transmise à l'administrateur A accompagnée du message e_i et de l'identité du votant.

   Votant

   $$\begin{array}{lcl} x_i & = & \mathscr{C}(v_i, r_i) \\ e_i & = & \mathscr{B}(x_i, b_i) \\ s_i & = & \mathscr{S}(e_i, sk_{V_i}) \end{array}$$

   $$1. \quad V_i \longrightarrow A : V_i, e_i, s_i$$

2. **Administration.** À la réception de la requête d'enregistrement transmise par V_i, l'autorité A vérifie si ce dernier est un votant légitime ($V_i \in L_v$), s'il ne s'est pas déjà enregistré ($V_i \notin L_1$ avec L_1 la liste des votants acquittés), et si sa signature est valide. Si ces conditions sont satisfaites, A enregistre le votant V_i, signe en aveugle le vote gagé par le votant, puis lui transmet cette signature ($d_i = \mathscr{S}(e_i, sk_A)$). Cette signature représente le jeton permettant au votant de poster son vote durant la prochaine étape du protocole.

   Administrateur

   $$\begin{array}{lcl} V_i & \stackrel{?}{\in} & L_v \\ V_i & \stackrel{?}{\in} & L_1 \\ e_i & \stackrel{?}{=} & \mathscr{S}^{-1}(s_i, pk_{V_i}) \\ d_i & = & \mathscr{S}(e_i, sk_A) \end{array}$$

   $$2. \quad A \longrightarrow V_i : d_i$$

3. **Vote.** La phase de vote ne débute qu'après qu'une certaine *deadline* ait été atteinte (tous les votants se sont enregistrés). Le votant démasque la signature reçue précédemment de A après avoir vérifié sa validité. Il obtient ainsi une signature de A sur le vote gagé à la première étape du protocole. Cette signature ($y_i = \mathscr{S}(x_i, sk_A)$) est transmise ensuite par le votant au collecteur C via un canal anonyme.

   Votant

   $$\begin{array}{lcl} e_i & \stackrel{?}{=} & \mathscr{S}^{-1}(d_i, pk_A) \\ y_i & = & \mathscr{B}^{-1}(d_i, b_i) = \mathscr{S}(x_i, sk_A) \end{array}$$

   $$3. \quad V_i \longrightarrow C : x_i, y_i$$

4. **Notification.** Si le vote reçu dispose d'une signature valide, le collecteur C le sauvegarde comme $l^{ème}$ élément de la liste des votes reçus. Après une seconde *deadline* (tous les votants se sont prononcés), C publie auprès de l'entité B la liste des votes reçus (liste L_2). Un élément de cette liste est un triplet (l_j, x_j, y_j).

   Collecteur

   $x_i \stackrel{?}{=} \mathscr{S}^{-1}(y_i, pk_A)$

   $$4. \quad C \longrightarrow B : \ldots, (l_j, x_j, y_j), \ldots$$

5. **Ouverture des bulletins.** Le votant V_i vérifie si son vote figure parmi ceux publiés par le collecteur à l'étape précédente. Si c'est le cas, le votant transmet alors anonymement au collecteur C l'élément l_i identifiant son vote dans la liste, ainsi que la clé r_i permettant l'ouverture de son bulletin.

   Votant

   $(x_i, y_i) \stackrel{?}{\in} L_2$

   $$5. \quad V_i \longrightarrow C : l_i, r_i$$

6. **Publication des résultats.** Le collecteur ouvre le $l_i^{ème}$ bulletin en le déchiffrant au moyen de la clé r_i reçue anonymement. Si le vote est valide ($v_i \in L_c$), C l'enregistre dans une liste L_3 dont les éléments sont des tuples (x_j, y_j, r_j, v_j), avec x_j, y_j, r_j, et v_j représentant respectivement le vote gagée, sa version signée, la clé d'engagement et le vote correspondant. Cette liste est finalement publiée, après une troisième *deadline* marquant la fin de la phase d'ouverture des bulletins, dans le but de permettre à chaque votant de vérifier la prise en compte de son vote.

   Collecteur

   $v_i = \mathscr{C}^{-1}(x_i)$

   $v_i \stackrel{?}{\in} L_c$

   $$6. \quad C \longrightarrow B : \ldots, (x_j, y_j, r_j, v_j), \ldots$$

Il est à noter que nous avons présenté ici les grandes lignes du protocole FOO. Dans la version originale, il est fait mention de procédures permettant aux votants de porter réclamation auprès des autorités dans le cas d'irrégularités (en particulier, lorsque le vote d'un votant ne figure pas dans la liste des votes publiés).

1.4.2 Schémas basés sur le chiffrement homomorphe

Dans les protocoles de vote électronique basés sur le chiffrement homomorphe, l'élection est gérée par un ensemble d'autorités, et le processus de vote repose sur un système de partage de secret (voir section 1.2.6) pouvant être utilisé dans différents contextes : (i) le votant partage son secret (vote) entre les autorités gérant l'élection [Ben87, Sch99], ou (ii) une entité de

confiance distribue à chaque autorité sa part du secret représentant la clé privée du cryptosystème [CGS97, HS00]. Selon la seconde approche, tous les votes sont envoyés chiffrés au moyen de la clé publique correspondante, et les autorités coopèrent ensuite afin de déchiffrer le produit des votes chiffrés. Le résultat de l'élection (somme des votes) est calculé en exploitant la propriété du chiffrement homomorphe (voir section 1.2.4).

Les protocoles de vote électronique utilisant le chiffrement homomorphe impliquent généralement trois phases : initialisation, vote, et comptage des voix. Ci-après, est décrit brièvement, à travers deux exemples de protocoles, les principes de base employés dans ces protocoles.

Protocole A

Le protocole CGS proposé dans [CGS97] est bâti sur le principe suivant :

Initialisation. Les autorités mettent en place un système de partage de secret afin de partager une clé privée sk.

Vote. Le votant V_i chiffre son vote au moyen de la clé publique correspondante pk, et d'un aléa k_i (chiffrement probabiliste). Le vote chiffré est ensuite rendu publique par le votant.

Comptage des voix. Les votes chiffrés sont en premier lieu multipliés. Ensuite, les autorités coopèrent afin de déchiffrer la somme des votes (homomorphisme additif). Il est à noter que le processus de déchiffrement est réalisé sans reconstruction de la clé secrète sk. Ceci peut être assuré en ayant recours à un cryptosystème à seuil [CGS97].

Les phases 1 et 3 sont également communes au protocole [HS00] qui propose une version modifiée du schéma de vote permettant de transformer le protocole CGS en un protocole sans-reçu. Plus précisément, le votant est assisté, durant la phase 2 du protocole, par l'ensemble des autorités pour construire une version chiffrée de son vote. Ne disposant plus de reçu matérialisé dans le protocole CGS par le paramètre aléatoire k_i, le votant V_i n'est plus apte à prouver le contenu de son bulletin.

Protocole B

Les protocoles proposés dans [Ben87, Sch99] suivent les principes décrits ci-après :

Initialisation. Chaque autorité A_j génère une paire de clés (pk_j, sk_j).

Vote. Le votant V_i fait appel à un système de partage de secret (t,n) afin de partager son vote (v_i) entre les n entités du système. Pour ce faire, le votant envoie à chaque autorité A_j sa part du secret s_{ij} chiffré au moyen de la clé publique pk_j et d'un aléa k_{ij} (chiffrement probabiliste).

Comptage des voix. Chaque autorité A_j multiplie les parts des secrets reçus, puis déchiffre le produit afin d'obtenir la somme des parts S_j (homomorphisme additif). Le résultat de l'élection peut être finalement calculée par chacun au moyen des parts S_j $(j = 1 \ldots n)$ publiés par les autorités. Plus précisément, t fragments sont nécessaires pour calculer le résultat de l'élection (somme des votes) par interpolation de Lagrange.

Il est à noter que les preuves à divulgation nulle de connaissance sont souvent utilisés dans les schémas de protocoles utilisant le chiffrement homomorphe. À titre d'exemple, le votant est souvent amené à prouver que son vote est valide. De même les autorités doivent également produire des preuves de la validité des résultats publiés.

1.5 Conclusion

Comme décrit tout au long de ce chapitre, les protocoles de vote électronique font appel à des primitives cryptographiques toutes aussi variées que complexes (partage de secret, signature en aveugle) dans le but de satisfaire un ensemble de propriétés de sécurité (anonymat, éligibilité, équité). Cependant, il est très rare de trouver un protocole garantissant la totalité des principales exigences du vote électronique. Il arrive également souvent qu'un protocole clame satisfaire une propriété [Oka96, BT94] et s'avérer par la suite ne pas la garantir. À titre d'exemple, Okamoto a mis au point dans [Oka96] un protocole empêchant la coercition et démontré un an plus tard [Oka97] que son protocole ne disposait pas réellement de cette propriété. Ces erreurs sont sans doute dûes aux contraintes très strictes imposées par le vote électronique dont certaines peuvent même sembler à première vue contradictoires. À titre d'exemple, le votant doit être capable de vérifier que son vote a été pris en considération (vérifiabilité individuelle), et en même temps être incapable de prouver le contenu de son bulletin (propriété sans-reçu).

Dans ce contexte, l'usage des méthodes formelles, pour montrer que les propriétés prétendues satisfaites par un protocole sont réellement garanties, devrait être incontournable. Dans ce qui suit, nous faisons un état des lieux des différentes approches formelles s'articulant autour de la vérification des protocoles de vote électronique, et présentons celle que nous avons adopté. Les protocoles introduits dans ce chapitre, et notamment le protocole FOO serviront de base d'exemple pour illustrer notre approche tout au long de cette thèse.

Chapitre 2

Vérification des protocoles de vote électronique

Les protocoles de vote électronique peuvent être vus comme un exemple particulier de protocoles cryptographiques. Ces derniers consistent en des opérations de communication et de traitements internes (vérification de la validité d'une signature) permettant à des entités communicantes de s'échanger des messages, via des canaux de communication (publiques, privés, anonymes), dans le but d'atteindre un objectif particulier (authentification mutuelle, échange de clés de session, partage d'un secret).

De nos jours, il est bien établi que la conception de tels protocoles est difficile et sujette aux vulnérabilités. Ces failles ne sont pas dûes uniquement aux algorithmes cryptographiques sous-jacents mais également à la manière dont ils ont été utilisés pour construire les protocoles, et à l'environnement supposé hostile dans lequel les objectifs de sécurité doivent être atteints. Pour garantir la correction d'un protocole cryptographique, il faut toujours raisonner par rapport au pire scénario dans lequel on suppose que le réseau de communication est sous le contrôle d'un intrus capable d'usurper l'identité d'honnêtes participants, d'intercepter les messages, de les rejouer, ou bien d'en fabriquer de nouveaux et de les injecter. Sous ces conditions, il est bien souvent difficile de déceler les éventuels attaques même pour les plus basiques des protocoles. Bien des protocoles ont été publiés, appliqués, puis plus tard démontrés faillibles à des attaques [CJ97]. L'exemple le plus parlant est celui du protocole d'authentification à clé publique proposé par Needham et Schroeder [NS78] démontré faillible dix-sept ans après sa publication à l'attaque de l'intercepteur (*man in the middle attack*) [Low95].

Dans ce contexte, l'usage des méthodes formelles pour montrer que les propriétés prétendues satisfaites par un protocole sont réellement garanties devrait être incontournable. Leur usage est d'autant plus important dans les applications critiques telles que le commerce électronique où l'argent est en jeu, ou bien le vote électronique où la démocratie est en jeu.

Les méthodes formelles peuvent être définies comme un ensemble de formalismes mathématiques et/ou logiques pour la description des systèmes et de leurs spécifications, et de procédures permettant de s'interroger quant au respect de ces spécifications par le système. Leurs applications dans le domaine de la spécification et de l'analyse des protocoles cryptographiques remontent probablement aux travaux de Dolev et al. [DEK82, DY83] mais n'ont réellement émer-

gées qu'au début des années quatre-vingt dix avec la logique BAN [BAN90]. De nos jours, il existe une variété de méthodes et d'outils associés à la vérification formelle des protocoles. Dans [Mea03] on peut trouver une étude sur les différentes approches formelles employées dans l'analyse des protocoles cryptographiques.

Bien que le problème soit connu pour être généralement indécidable [CDL+99, CS02], recourir aux méthodes formelles afin de détecter certaines failles, ou bien de confirmer la sécurité d'un protocole cryptographique particulier constitue un résultat plus que satisfaisant. La littérature est riche en travaux portant sur la vérification formelle des protocoles cryptographiques. Cependant, comparativement il n'existe que très peu de travaux sur l'analyse formelle des protocoles de vote électronique [KR05, DKR06, BHM08, COPD06, MVdV07, JP06, EO07, BRS07]. Les algèbres de processus et les logiques modales, particulièrement les logiques de connaissance, constituent les principaux formalismes utilisés dans ces travaux. Selon la première approche, le rôle de chaque entité dans le protocole est modélisé par un processus spécifié par une grammaire donnée, et les propriétés sont ensuite généralement exprimées en terme de relation d'équivalence entre processus (équivalences observationnelles). Dans le cas de la seconde approche, les propriétés sont exprimées au moyen d'une logique disposant de constructeurs permettant de s'interroger sur la possibilité que certains événements du protocole se produisent, sur l'ordre d'exécution de ses événements (logiques temporelles), sur la croyance des agents (logiques doxastiques) et/ou sur la connaissance de ces agents (logiques épistémiques) par rapport à certaines propositions ("Bob a voté oui").

Il est à noter que ces deux approches ne sont pas indépendantes l'une de l'autre. La première met l'accent sur la spécification du protocole. La seconde se focalise sur la spécification des propriétés pouvant être éventuellement vérifiées par rapport à un modèle de protocole spécifié par une algèbre de processus.

Dans ce qui suit, nous présentons ces deux approches via des exemples illustratifs. Nous faisons également un état des lieux des travaux formels ciblant la vérification des protocoles de vote électronique. Nous présentons ensuite l'approche que nous avons adoptée. Plus précisément, nous introduisons la logique ADM à travers sa syntaxe et sa sémantique et discutons de son intérêt dans la vérification des protocoles cryptographiques en général et des protocoles de vote électronique en particulier. Nous évoquons brièvement la difficulté liée à la vérification de tels protocoles. Finalement, nous clôturons ce chapitre par préciser les principales réalisations envisagées dans cette thèse.

2.1 Algèbres de processus

Une algèbre de processus est un langage pour la spécification des systèmes concurrents. Le terme algèbre fait référence aux opérations permises par la grammaire sur les processus telles que la réplication d'un processus ou la composition parallèle de deux processus.

Roscoe et Lowe ont été les premiers à suggérer respectivement dans [Ros95] et [Low96] l'usage des algèbres de processus pour l'analyse des protocoles cryptographiques. L'idée est de spécifier le protocole comme la composition parallèle de processus décrivant le rôle des entités du protocole et leurs interactions via l'échange de messages. L'analyse du protocole s'opère ensuite par l'extraction à partir de cette spécification d'un modèle à états finis, le plus souvent un système de transition étiqueté LTS (*Labelled Transition System* [Plo81]), sur lequel seront vérifiées les propriétés de sécurité. Celles-ci pourront être exprimées au moyen d'une logique donnée

(LTL [HR00], μ-calcul [Sti96]) ou définies en des termes propres à l'algèbre elle même. Nous nous intéressons ici particulièrement à la notion d'équivalence étant donné qu'elle a été largement exploitée dans l'analyse des protocoles de vote, notamment pour la formalisation des propriétés telles que l'anonymat, le sans-reçu ou la résistance à la coercition.

Dans la littérature, il existe une variété d'algèbres de processus : CCS [Mil82], ACP [BK85] et leurs multiples extensions : π-calcul [Mil99], μCRL [GR01]. Nous présentons ci-après deux d'entre elles : l'algèbre CSP [Hoa85], et le π-calcul appliqué [AF01]. Nous nous intéressons particulièrement à la notion d'équivalence observationnelle dans ces algèbres, et donnons comment la propriété d'anonymat peut être décrite en se basant sur cette relation d'équivalence. Nous synthétisons ensuite les travaux ayant trait à la modélisation et l'analyse des protocoles de vote au moyen des algèbres de processus tout en décrivant les diverses propriétés de sécurité analysées.

2.1.1 CSP

L'algèbre CSP (*Communicating Sequential Processes*) offre un langage permettant de décrire le comportement des systèmes concurrents en tant que composantes (processus) interagissant entre elles via l'échange de messages. Elle semble dès lors appropriée pour l'analyse des protocoles cryptographiques.

Dans CSP, un processus peut être représenté par un système de transition décrivant les diverses possibilités d'évolution du processus. Un branchement de ce modèle arborescent dénote une séquence d'événements (trace) sur lesquels le processus peut s'engager. Dans sa forme la plus simple, un événement peut être abstrait aux symboles a, b ou c. Pour les systèmes communicants (protocoles), l'algèbre prévoit les actions $c!m$ (action d'envoi) et $c?x$ (action de réception) pour représenter respectivement l'envoi du message m et sa récupération dans x via le canal c. L'ensemble des événements possibles pour un processus P constitue son alphabet noté αP.

Syntaxe. La syntaxe de CSP [1] [Hoa85] est donnée par la grammaire BNF [2] définie en Table 2.1.

TABLE 2.1 – Syntaxe de CSP

P	::=	$a \longrightarrow P$	|	$P \Box Q$	|	$P \sqcap Q$	|	$P\|[A]\|Q$	|	$P \backslash A$	|	$f_A(P)$	|	$STOP$

Tout d'abord, $a \longrightarrow P$ représente le processus qui peut exécuter l'action a et se comporter ensuite comme le processus P. Le processus $P \Box Q$ représente le choix externe et désigne le processus qui peut se comporter soit comme P soit comme Q et c'est l'environnement qui effectue ce choix. L'environnement choisit le processus qui peut s'exécuter. Si P et Q peuvent tous deux s'exécuter, le système en choisit un de manière non déterministe. Le processus $P \sqcap Q$ désigne le choix interne (non déterministe) sur lequel l'environnement n'a aucun contrôle. Le processus $P|[A]|Q$ désigne la composition parallèle de P et Q avec la condition que les deux processus se synchronisent sur les événements de l'ensemble d'actions A. Une variante de ce constructeur est $P \,|||\, Q$, qui désigne le processus où P et Q s'exécutent en parallèle et indépendamment l'un de l'autre. Le processus $P \backslash A$ désigne le masquage et représente le processus où l'ensemble des

1. La syntaxe définie par Hoare dans [Hoa85] est plus riche. Nous avons présenté uniquement les principaux constructeurs de la grammaire.

2. *Backus Naur Form*. Notation, dû à John Backus et Peter Naur, définie initialement afin d'introduire la syntaxe des langages de programmation.

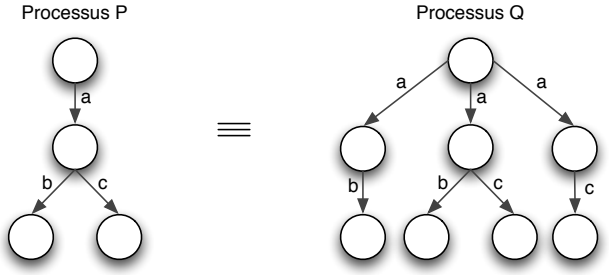
actions dans A ne sont pas observables. Le processus $f_A(P)$ représente le processus P où tout événement $a \in A$ est renommé par son image par la fonction $f : \alpha P \longrightarrow \alpha P$. Finalement, le processus $STOP$ est le processus qui ne peut s'engager sur aucune action.

Équivalence. Dans l'algèbre CSP, trois notions d'équivalence ont été définies : équivalence par trace, équivalence par refus et équivalence par refus/divergence. Dans ce qui suit, nous nous intéressons uniquement à la première forme d'équivalence dans laquelle deux processus sont équivalents s'ils exhibent le même ensemble de traces. L'exemple 2.1.1 représente deux processus équivalents par trace.

Exemple 2.1.1

Soit les deux processus P et Q définis comme suit :

$$\begin{aligned} P &= a \longrightarrow (b \longrightarrow STOP \,\Box\, c \longrightarrow STOP) \\ Q &= a \longrightarrow (b \longrightarrow STOP \,\Box\, c \longrightarrow STOP) \\ &\Box \quad a \longrightarrow b \longrightarrow STOP \\ &\Box \quad a \longrightarrow c \longrightarrow STOP \end{aligned}$$



Les processus P et Q sont équivalents par trace étant donné l'égalité suivante :

$$Traces(P) = \{\langle\rangle, \langle a\rangle, \langle a,b\rangle, \langle a,c\rangle\} = Traces(Q)$$

Note : La notation $\langle\rangle$ désigne la trace vide dans CSP.

L'anonymat dans CSP. Une première formalisation de la propriété d'anonymat a été donnée par Schneider et Sidiropoulos dans [SS96]. Leur définition s'articule autour d'un ensemble Ag d'agents du système (votants) désirant être anonymes par rapport à un événement particulier evt (vote). Ce dernier se présente selon la forme composée $i.e$ où i représente l'identité de l'agent ayant initié l'événement e. L'idée est que si un événement de l'ensemble $A = \{i.e \mid i \in Ag\}$ se produit, il pourrait alors être confondu avec n'importe quel autre événement de ce même ensemble. Cette idée est énoncée en terme d'équivalence par trace comme suit :

$$Traces(P) = Traces(f_A^{-1}(f_A(P)))$$

$$f_A(x) = \begin{cases} \alpha & \text{si} \quad x \in A \\ x & \text{sinon} \end{cases}$$

La fonction $f_A()$ est une fonction de renommage où tout événement de l'ensemble A est substitué par un événement donné α. Si P désigne le processus modélisant le système à analyser, le processus $f_A(P)$ représente le système qui rend l'événement α possible à chaque fois qu'un événement de A est possible dans le processus original P. En suivant le même raisonnement, le processus $f_A^{-1}(f_A(P))$ est celui qui rend tout événement de A possible à chaque fois qu'un événement particulier de ce même ensemble est possible dans P. Selon la définition énoncée ci-dessus, un système modélisé par un processus P satisfait la propriété d'anonymat s'il est capable de s'engager dans les mêmes séquences d'actions que celles pouvant être générées par le processus $f_A^{-1}(f_A(P))$.

Dans [SS96], cette définition de l'anonymat est mise en application sur le protocole du dîner des cryptographes énoncé par Chaum [Cha88], et illustré sur un exemple de protocole basique que nous reprenons ci-après :

Exemple 2.1.2

Soit le protocole de donation où deux entités, dont les identités sont volontairement masquées par a_1 et a_2, peuvent faire acte de charité. Si le premier fais un don, il est toujours de l'ordre de 5€. Si c'est le second qui y participe, le montant est toujours le double (10€). Le protocole est spécifié dans CSP comme processus P :

$$P = a_1.don \longrightarrow 5€ \longrightarrow STOP \,\square\, a_2.don \longrightarrow 10€ \longrightarrow STOP$$

L'anonymat étant requis pour les entités a_1 et a_2 quant à l'action de donation don. Il n'est pas très difficile de voir que la propriété d'anonymat définie par rapport à l'ensemble $A = \{a_1.don, a_2.don\}$ n'est pas réellement garantie. En effet, un observateur externe, pouvant faire la distinction entre les exécutions liées aux événements de l'ensemble A, est capable de remonter à l'identité du donateur. Plus précisément, connaissant les habitudes des donateurs, et observant les montants transférés, l'intrus peut en déduire l'identité du donateur. Ceci est démontré par le processus $Q = f_A^{-1}(f_A(P))$ capable de générer les traces $\langle a_1.don, 10€\rangle$ et $\langle a_2.don, 5€\rangle$, impossibles dans le processus P :

$$\begin{aligned} Q &= f_A^{-1}(f_A(P)) \\ &= a_1.don \longrightarrow 5€ \longrightarrow STOP \,\square\, a_1.don \longrightarrow 10€ \longrightarrow STOP \\ &\square\ a_2.don \longrightarrow 5€ \longrightarrow STOP \,\square\, a_2.don \longrightarrow 10€ \longrightarrow STOP \end{aligned}$$

2.1.2 π-calcul appliqué

Le π-calcul appliqué [AF01] a été proposé par Abadi et Fournet comme extension à l'algèbre du π-calcul. Il s'agit d'un langage pour la spécification de processus concurrents interagissant entre eux via l'échange de messages. Sa conception a été orientée à l'analyse des protocoles cryptographiques pouvant être automatisée au moyen de l'outil ProVerif [Bla01]. Nous avons choisi de le présenter ici étant donné qu'il a été largement exploité dans la littérature pour l'analyse des protocoles de vote électronique [KR05, DKR06, BHM08].

Termes et théorie équationnelle. La syntaxe des processus décrits en π-calcul appliqué nécessite de définir au préalable l'algèbre des termes manipulés par ces processus. Ces termes sont typés, et constructibles à partir d'un ensemble $\mathcal{V}$ de variables, d'un ensemble $\mathcal{N}$ de noms, et d'une signature Σ. Les éléments de l'ensemble $\mathcal{N}$ dénomment essentiellement les canaux de communication, mais peuvent caractériser tout autre type de donnée. La signature $\Sigma = \{(f_1, a_1), \ldots (f_n, a_n)\}$ consiste quant à elle en un nombre fini de symboles de fonctions f_i, d'arité a_i, applicables sur les termes de l'algèbre. À noter ici qu'une fonction d'arité nulle désigne une constante.

Les termes de l'algèbre sont équipés d'une théorie équationnelle E consistant en un ensemble fini d'équations de la forme $M_i = N_i$ où M_i et N_i désignent des termes dépourvus de noms. L'exemple 2.1.3 illustre une signature Σ et une théorie équationnelle pouvant être utilisée dans le cadre de l'analyse des protocoles cryptographiques.

Exemple 2.1.3 (*Signature et théorie équationnelle*)

Dans la signature Σ donnée ci-après, la fonction $pair$ permet de former la paire de messages (m_1, m_2) à partir des messages m_1 et m_2. Ensuite, les fonctions fst et snd permettent d'extraire respectivement le premier et le second élément d'une paire de messages. Finalement, la primitive $symenc$ modélise le chiffrement symétrique, et représente le chiffrement d'un message m au moyen d'une clé k. La primitive $symdec$ modélise quant à elle le processus inverse.

$$\Sigma = \{(pair, 2), (fst, 1), (snd, 1), (symenc, 2), (symdec, 2)\}$$

La théorie équationnelle E associée à la signature Σ est définie par :

$$fst(pair(x, y)) = x \tag{2.1}$$
$$snd(pair(x, y)) = y \tag{2.2}$$
$$symdec(symenc(x, y), y) = x \tag{2.3}$$

Soient maintenant les termes M_1 et M_2 définis comme suit :

$$M_1 = symenc(symdec(symenc(m, k_2), k_2), k_1)$$
$$M_2 = symenc(m, k_1)$$

Nous remarquons ici que $E \vdash M_1 = M_2$ (par application de l'équation (2.3)), signifiant que le terme M_1 équivaut le terme M_2 modulo la théorie équationnelle E.

Syntaxe. Dans le π-calcul appliqué, nous distinguons deux types de processus : les processus simples (*plain processes*) et les processus étendus (*extended processes*).

La syntaxe des processus simples est donnée par la Table 2.2.

TABLE 2.2 – Syntaxe du π-calcul appliqué (Processus simple)

P	::=	$P \mid Q$	$\mid !P \mid$	$\nu n.P \mid$	*if* $M = N$ *then* P *else* $Q \mid$	$u(x).P \mid$	$\bar{u}\langle N \rangle.P \mid$	0

Tout d'abord, le processus $P \mid Q$ représente la composition parallèle des processus P et Q. Le processus $!P$ désigne la réplication et se comporte comme l'exécution parallèle d'une infinité

de copies du processus P. Le processus $\nu n.P$ (restriction de noms) génère un nouveau nom n (privé), et se comporte ensuite comme le processus P. Le processus *if* $M = N$ *then* P *else* Q est celui qui se comporte comme processus P si le terme M équivaut au terme N modulo la théorie E ($E \vdash M = N$), et comme le processus Q dans le cas contraire. Le processus $u(x).P$ (noté aussi $in(x).P$) dénote celui qui est prêt à recevoir une donnée sur le canal u, et qui se comporte ensuite comme le processus P où la variable x est substituée par la valeur reçue. Le processus $\bar{u}\langle N\rangle.P$ (noté également $out(u,N).P$) est celui qui envoie le message N sur le canal u, et qui se comporte ensuite comme processus P. Finalement, le processus 0 (*null process*) est identique au processus $STOP$ de CSP.

La syntaxe d'un processus étendu est donnée en Table 2.3, où P désigne un processus simple et $A \mid B$ la composition parallèle de processus étendus A et B. $\nu n.A$ et $\nu x.A$ représentent respectivement la restriction de noms et de variables. Finalement, $\{^M/_x\}$ est une substitution active qui remplace la variable x par le terme M. Son usage dans le processus $\nu x.(\{^M/_x\} \mid P)$ est assimilable à "*let* $x = M$ *in* P" et se traduit comme le processus P dans lequel le terme M est affecté à la variable x. Un processus étendu est dit clos si toutes ses variables sont soit liées soit impliquées dans une substitution active.

TABLE 2.3 – Syntaxe du π-calcul appliqué (Processus étendu)

A	::=	P	$A \mid B$	$\nu n.A$	$\nu x.A$	$\{^M/_x\}$	

On définit également, le domaine d'un processus A (noté $dom(A)$) par l'intermédiaire du domaine de son cadre. Un cadre (*frame*) est un processus étendu obtenu à partir du processus 0 et des substitutions actives par application de la composition parallèle et de la restriction. Le cadre associé à un processus A (noté $\varphi(A)$) est obtenu par remplacement de tout sous processus simple dans A, par le processus 0. Son domaine (noté $dom(\varphi(A))$) est l'ensemble des variables non restreintes x pour lesquels il dispose d'une substitution $\{^M/_x\}$.

Un contexte C est défini maintenant comme un processus qui contient un trou (noté []), et on désigne par $C[P]$, l'expression obtenue par remplacement du trou [] dans le contexte C par le processus P.

Exemple 2.1.4

Soient le contexte C et le processus P définis comme suit :

$$C = \bar{a}\langle M\rangle.0 \mid []$$
$$P = \bar{b}\langle N\rangle.0$$

Le processus $C[P]$ est donné par :

$$C[P] = \bar{a}.\langle M\rangle.0 \mid \bar{b}\langle N\rangle.0$$

Un contexte est dit d'évaluation lorsque le trou [] n'est pas impliqué par une réplication, une condition, un envoi ou une réception.

Sémantique. La sémantique opérationnelle du π-calcul appliqué est définie au moyen de règles de congruence structurelle ($\equiv$) et de règles de réduction ($\rightarrow$) (voir Annexe A, Tables A.1 et A.2).

Équivalence. La définition de l'équivalence observationnelle est donnée ci-après. La notation $A \Downarrow a$, employée dans cette définition, signifie que le processus A peut envoyer un message sur le canal a.

Définition 2.1.1 (*Équivalence observationnelle*)

L'équivalence observationnelle (notée $\approx$) est la plus grande relation symétrique $\mathscr{R}$ entre processus étendus clos et de même domaine tel que $A \mathscr{R} B$ implique :

1. *Si $A \Downarrow a$ alors $B \Downarrow a$.*
2. *Si $A \rightarrow^* A'$, alors il existe B' tel que $B \rightarrow^* B'$ et $A' \mathscr{R} B'$.*
3. *$C[A] \mathscr{R} C[B]$ pout tout contexte d'évaluation C tel que $C[A]$ et $C[B]$ sont clos.*

L'anonymat dans le π-calcul appliqué. Dans [KR05], Kremer et Ryan proposent une formalisation de l'anonymat basée sur l'équivalence observationnelle. Leur définition a été illustrée sur le protocole FOO, puis généralisée, dans [DKR09], aux protocoles de vote électronique dont ils considèrent leur modélisation donnée par le processus VP :

$$VP \equiv \nu \tilde{n}.(V\sigma_1 \mid \ldots \mid V\sigma_n \mid A_1 \mid \ldots \mid A_m)$$

Le processus VP est un processus simple et clos, et consiste en la composition parallèle de processus $V\sigma_i$, et de processus A_j caractérisant respectivement le rôle des votants V_i ($i = 1 \ldots n$) et des autorités A_j ($j = 1 \ldots m$) dans le protocole de vote. Dans cette configuration, $\tilde{n}$ désigne une séquence de canaux de communication privés, et il est supposé que les autorités A_j ($j = 1 \ldots m$) sont honnêtes, et qu'il existe une variable v, du domaine de la substitution σ_i ($v \in dom(\sigma_i)$) représentant le vote.

La formalisation de la propriété d'anonymat est maintenant donnée par l'équivalence suivante, où V_A et V_B sont deux processus modélisant chacun un votant honnête, et S est un contexte d'évaluation correspondant au processus VP au niveau duquel un sous processus $(V\sigma_k \mid V\sigma_{k+1})$ $(1 \leqslant k \leqslant n-1)$ a été remplacé par un trou [] :

$$S[V_A\{^{v_a}/_v\} \mid V_B\{^{v_b}/_v\}] \approx S[V_A\{^{v_b}/_v\} \mid V_B\{^{v_a}/_v\}] \tag{2.4}$$

Cette définition signifie qu'un protocole de vote (modélisé par un processus VP) satisfait la propriété d'anonymat, si le processus où deux votants votent respectivement v_a et v_b est équivalent observationnellement au processus où les deux votants inversent leurs votes. Plus précisément, elle exprime l'incapacité de l'intrus à différencier la situation actuelle de celle où les deux votants ont interverti leurs votes.

Afin de donner un exemple d'application de cette formalisation, nous reprenons ci-après les grandes lignes de l'étude portée dans [DKR09] sur le protocole FOO.

La première étape dans l'analyse du protocole FOO est d'identifier les primitives cryptographiques utilisées dans ce protocole (voir section 1.4.1), et de définir ensuite la signature et la théorie équationnelle qui leurs sont associées. Celles-ci sont données en Table 2.4.

TABLE 2.4 – Signature et théorie équationnelle - Protocole FOO

(Signature)	Σ	$=$	$\left\{ \begin{array}{c} (pair,2),(fst,1),(snd,1), \\ (host,1),(pk,1),(getpk,1), \\ (sign,2),(checksign,2), \\ (commit,2),(open,2), \\ (blind,2),(unblind,2) \end{array} \right\}$
$fst(pair(x,y))$	$=$	x	
$snd(pair(x,y))$	$=$	y	
$getpk(host(x))$	$=$	x	
$checksign(sign(x,y),pk(y))$	$=$	x	(Théorie équationnelle)
$open(commit(x,y),y)$	$=$	x	
$unblind(blind(x,y),y)$	$=$	x	
$unblind(sign(blind(x,z),y),z)$	$=$	$sign(x,y)$	

La signature est d'abord formée par les symboles de fonction *pair, fst* et *snd* décrits précédemment. Ensuite, elle est complétée par la primitive *host* qui permet de retrouver l'identité associée à une clé publique. Les primitives *pk* et *getpk* permettent de retrouver la clé publique à partir respectivement de la clé privée et de l'identité du détenteur de la clé. Les primitives *sign* et *checksign* correspondant respectivement à la signature numérique et à sa vérification. La primitive *commit* est relative à la mise en gage (voir section 1.2.2), et la primitive *open* assure la fonction inverse. Finalement, la primitive *blind* est relative à l'aveuglement (voir section 1.2.3), et *unblind* assure l'opération inverse.

La théorie équationnelle englobe d'abord les équations spécifiques à la composition de messages. L'équation qui manipule ensuite les entités du protocole et leurs clés publiques est triviale. L'équation $checksign(sign(x,y),pk(y)) = x$ permet d'extraire le message à partir de sa signature. La mise en gage et l'aveuglement sont assimilés au chiffrement symétrique, et de ce fait les deux avant dernières équations sont définies de manière similaire à celle relative au chiffrement symétrique (voir exemple 2.1.3). Finalement, la dernière équation est spécifique à l'aveuglement, et permet d'extraire la signature d'un message à partir de celle apposée sur un message aveuglé.

La seconde étape est de spécifier, dans l'algèbre du π-calcul appliqué, le rôle joué par chacune des entités du protocole FOO. Un exemple d'une telle spécification est donné par le processus $processV$ modélisant le rôle du votant dans le protocole. La spécification complète est donnée en Annexe A.

Processus votant

let $processV =$
in$(skvch, skv)$.
let $hostv = host(pk(skv))$ **in**
in$(pkach_1, pubka)$.
$vb.\,vr.$
let $cv = commit(v,r)$ **in**
$\vdots$

$\vdots$
let $bcv = blind(cv, b)$ **in**
out$(ch, (hostv, sign(bcv, skv)))$.
in(ch, m_2).
let $res = checksign(m_2, pubka)$ **in**
if $res = bcv$ **then**
let $scv = unblind(m_2, b)$ **in**
$phase\,1$.
out$(ch, (cv, scv))$.
in$(ch, (l, = (cv, scv)))$
$phase\,2$.
out$(ch, (l, r))$

Dans leur modélisation du protocole FOO, les auteurs de [DKR09] considèrent une entité supplémentaire K faisant office de PKI (Public Key Infrastructure), et dont le rôle est la génération et la distribution des clés cryptographiques aux entités concernées. Ainsi, la spécification du processus $processV$ débute par une phase d'initialisation durant laquelle le votant reçoit sa clé privée skv (via le canal restreint $skvch$), dont il extrait son identité $hostv$ ($hostv = host(pk(skv))$). Le votant reçoit également, durant cette phase d'initialisation, la clé publique de l'administrateur ($in(pkach_1, pubka)$). La spécification se poursuit par la génération des paramètres aléatoires b et r servant respectivement à l'aveuglement et à la mise en gage. Le votant construit ensuite une version signée de son vote, par appel successif aux primitives $commit$, $blind$ et $sign$, qu'il transmet accompagnée de son identité $hostv$ à travers le canal ch. Le votant reçoit sur ce même canal la signature produite par l'administrateur dont il vérifie la validité, et à partir de laquelle il extrait une signature sur son vote gagé. Les lignes qui suivent décrivent la transmission du vote $out(ch, (cv, scv))$ et la réception du message l identifiant le bulletin transmis (la déclaration $in(ch, (m_1, = m_2))$ se traduit comme le processus qui lit sur le canal ch une paire de messages dont le second argument est m_2). Finalement, la spécification se termine par la transmission de la clé r permettant l'ouverture du bulletin identifié par l.

À noter ici qu'un canal est par défaut anonyme en π-calcul appliqué, et de ce fait le bulletin ainsi que la clé permettant son ouverture sont transmis de manière anonyme. La spécification du processus $processV$ est donnée dans un style proche de ProVerif dont il hérite certaines de ces déclarations telles que $pahse$ qui permet de modéliser les *deadlines* du protocole FOO. Ainsi, $phase\ 1$ permet de délimiter les instructions de la phase d'enregistrement de la phase de vote. De même, pour $phase\,2$ qui délimite la phase de vote de celle d'ouverture des bulletins.

Vérifier maintenant si le protocole FOO garantit la propriété d'anonymat, revient à vérifier si l'équivalence suivante, adaptée à partir de celle donnée en (2.4) est établie :

$$S[processV\{^{skvach}/_{skvch}\}\{^{v_a}/_{v}\} \mid processV\{^{skvbch}/_{skvch}\}\{^{v_b}/_{v}\}]$$
$$\approx$$
$$S[processV\{^{skvach}/_{skvch}\}\{^{v_b}/_{v}\} \mid processV\{^{skvbch}/_{skvch}\}\{^{v_a}/_{v}\}]$$

Dû à l'incomplétude de l'outil ProVerif, quant à la vérification de certaines équivalences observationnelles, les auteurs se sont rabattus sur des techniques de preuve manuelles afin de démontrer la garantie de l'anonymat dans le protocole FOO.

2.1.3 Travaux connexes

Parmi les algèbres de processus existantes, le π-calcul appliqué est celui qui a été le plus sollicité pour l'analyse des protocoles de vote électronique. Kremer et Ryan [KR05] ont été les premiers à suggérer son usage pour la modélisation du protocole FOO. L'analyse formelle a porté sur trois propriétés de sécurité : équité, éligibilité et anonymat.

La première de ces propriétés a été ramenée à une propriété de confidentialité exprimant l'incapacité d'un observateur externe (intrus) à déduire les choix des votants avant la phase d'ouverture des bulletins. Intuitivement, un vote reste secret tant que l'intrus n'est pas apte à dégager le contenu en clair du bulletin à partir de sa version chiffrée. Cette notion de secret a été exprimée sous la forme d'une propriété d'accessibilité (ou atteignabilité) consistant à vérifier si le processus, modélisant le protocole FOO, est capable d'atteindre un état (précédant la phase d'ouverture des bulletins) où le vote est en clair.

Cependant, un vote peut être déduit sans besoin de déchiffrer le contenu du bulletin. En effet, si on considère que les votes, pris dans un ensemble restreint de choix connus publiquement, sont chiffrés au moyen d'un algorithme de chiffrement asymétrique déterministe, l'intrus serait alors capable de déduire le contenu du bulletin sans recours au déchiffrement. Un exemple simple est celui où le votant a le choix entre deux possibilités : "oui" et "non", et où les bulletins sont transmis chiffrés au moyen de la clé publique de l'entité responsable de la collecte des votes. Pour déduire un vote donné, l'intrus fabrique à son tour un bulletin chiffré et le compare ensuite au bulletin observé. Si les deux bulletins coïncident, alors le choix de l'intrus est le même que celui observé. Dans le cas contraire, il s'agit de l'autre possibilité de choix. Cette notion de secret où l'attaquant est mis au défi de deviner la valeur du vote a été formalisée en terme de relation d'équivalence.

La propriété d'éligibilité a elle aussi été ramenée à une propriété de confidentialité ; le secret étant un paramètre divulgué uniquement si l'attaquant est capable d'insérer un vote particulier (vote challenge). La propriété a été définie donc en terme d'accessibilité consistant à vérifier si le processus peut atteindre un état où le paramètre secret est connu.

Les deux propriétés discutées ci-dessus ont été également analysées dans le cas d'un administrateur corrompu. Dans ce cas précis, il a été prouvé que seule la première propriété est garantie dans le protocole FOO.

Finalement, la propriété d'anonymat a été exprimée en terme de relation d'équivalence comme illustrée par l'exemple de la section 2.1.2. Dans [DKR06], les mêmes auteurs récidivent par la proposition de définitions formelles similaires à celles de l'anonymat. Dans cet article, il est question de la formalisation des propriétés sans-reçu (équivalence observationnelle) et résistance à la coercition (simulation adaptive), et de la relation d'implication liant ces propriétés à l'anonymat : un protocole qui garantirait la propriété de résistance à la coercition impliquerait également la propriété sans-reçu qui impliquerait à son tour l'anonymat. Bien qu'il a été possible d'automatiser la vérification des propriétés d'équité et d'éligibilité au moyen de l'outil ProVerif, les auteurs ont eu recours à des techniques de preuves manuelles pour prouver le restant des propriétés. Ce problème a été néanmoins résolu dans [BHM08] où Backes et al. ont proposé une formalisation alternative des propriétés sans-reçu et résistance à la coercition. Leur formalisation, basée exclusivement sur les équivalences observationnelles, est propice à l'automatisation au moyen de l'outil ProVerif.

L'algèbre ACP et son extension μ-CRL ont été également utilisées respectivement dans les travaux [MVdV07] et [COPD06] pour la modélisation et l'analyse du protocole FOO. Dans ces travaux, l'étude est axée principalement sur la propriété d'anonymat, où les auteurs proposent des moyens, s'appuyant sur des relations d'équivalence, permettant de mesurer jusqu'à quel point le votant est anonyme. À titre d'exemple, dans [MVdV07], l'anonymat est mesuré par l'intermédiaire de ce que les auteurs appellent l'"ensemble d'attribution" du votant. Cet ensemble représente les différents votes pouvant être attribués à un votant. Plus cet ensemble est grand, plus robuste est l'anonymat du votant.

2.2 Logiques modales

Une logique est dite modale lorsqu'elle est dotée des modalités $\Box$ et $\Diamond$ représentant usuellement la nécessité et la possibilité, respectivement. Nous nous intéressons ici particulièrement aux logiques épistémiques où la modalité de nécessité est translaté en un opérateur de connaissance noté le plus souvent $K_a p$. Il s'agit là d'une abréviation du constructeur $\Box_a p$ désignant le fait que l'agent a sait que la proposition p est vraie. Un tel pouvoir d'expression conféré à la logique semble intéressant pour la spécification des propriétés telles que l'anonymat. En effet, exprimer la propriété d'anonymat reviendrait à spécifier des propriétés du type $K_i p$ permettant de s'interroger sur la connaissance d'un observateur externe i par rapport à des propositions p du type "le votant v a voté pour le candidat c".

Dans ce qui suit, nous nous intéressons à cet opérateur dans la logique multimodale propositionnelle que nous appellerons plus simplement, dans ce contexte de connaissance, par logique épistémique propositionnelle.

2.2.1 Logique épistémique propositionnelle

Syntaxe. La syntaxe de la logique est définie dans sa forme minimale par la grammaire BNF donnée en Table 2.5 où p désigne une proposition atomique, $\neg$ et $\wedge$ représentent respectivement la négation et la conjonction, et $\Box_a p$ désigne l'opérateur modal discuté plus haut.

TABLE 2.5 – Syntaxe de la logique épistémique propositionnelle

ϕ	::=	p	$\mid$	$\neg\phi$	$\mid$	$\phi_1 \wedge \phi_2$	$\mid$	$\Box_a p$

Sémantique. La sémantique associée à l'opérateur de connaissance peut être illustrée par rapport à une structure de Kripke défini comme suit :

Définition 2.2.1 (*Structure de Kripke pour une logique multimodale*)

Soit Ag un ensemble d'agents et P un ensemble de propositions atomiques. Une structure de Kripke est un triplet $\mathcal{M} = (S, L, R)$ où :

- *S est un ensemble d'états fini.*
- *$L : P \longrightarrow \mathcal{P}(S)$ est une fonction qui associe à chaque proposition $p \in P$ l'ensemble des états où la proposition est vraie.*
- *$R : Ag \longrightarrow \mathcal{P}(S^2)$ est une fonction qui associe à chaque agent $a \in Ag$ une relation d'accessibilité $\xrightarrow{a}$.*

On s'intéresse ici particulièrement à une structure de Kripke S5 où la relation d'accessibilité $\xrightarrow{a}$ est une relation d'équivalence satisfaisant les propriétés de réflexivité, de transitivité et de symétrie.

La formule $K_a p$ est maintenant évaluée à vrai dans un état si la proposition p est vraie dans tous les états accessibles via la transition a.

Anonymat dans la logique épistémique propositionnelle. L'exemple de protocole donné ci-après, et inspiré de [EO07], permet d'illustrer comment la propriété d'anonymat peut être spécifiée au moyen des opérateurs de la logique épistémique propositionnelle.

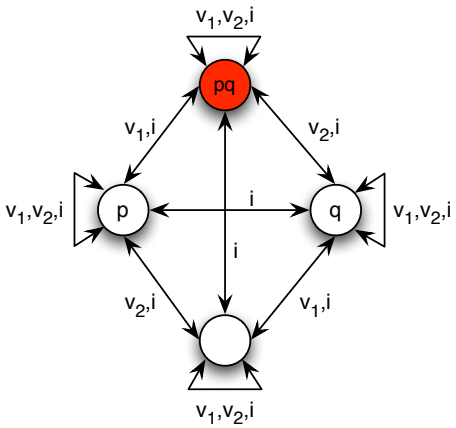
Exemple 2.2.1

Soit un protocole de vote où deux votants v_1 et v_2 ont le choix entre deux possibilités "oui" et "non". La modélisation du protocole nécessite au préalable d'identifier l'ensemble des agents Ag du système et l'ensemble des propositions atomiques P :

- *Agents du protocole. L'ensemble des agents comprends les deux votants v_1 et v_2 ainsi que l'intrus i face auquel l'anonymat est requis.*
- *Propositions atomiques. On s'intéresse ici aux propositions p et q définies comme suit :*

$$p = \text{"Le choix du votant } v_1 \text{ est oui"}$$
$$q = \text{"Le choix du votant } v_2 \text{ est oui"}$$

Le protocole peut être maintenant modélisé par une structure de Kripke à 4 états correspondant à la véracité ou non des propositions p et q. La figure ci-dessous illustre le modèle du protocole où l'état courant est celui où les deux votants ont opté pour le choix "oui" (état colorié en rouge).



Si on considère maintenant que cette configuration de choix a donné lieu à la distribution de connaissance telle qu'illustrée par la figure du dessus, il est possible de vérifier que :

- *le votant v_1 sait qu'il a voté "oui" : la formule $K_{v_1} p$ est évaluée à vrai dans l'état courant.*
- *le votant v_2 est anonyme par rapport à l'intrus i : la formule $K_i q \vee K_i \neg q$ est évaluée à faux dans l'état courant. Cette formule exprime le fait que l'intrus ne sait pas si la proposition q est vraie ou pas.*

Au delà des simples formulations de type $K_i p$, la logique offre la possibilité aux agents de s'interroger sur la connaissance des autres agents quant à la véracité ou non de certaines propositions. À titre d'exemple, la formule $K_i(K_{v_1} p \vee K_{v_1} \neg p)$, évaluée à vrai dans l'état courant, exprime le fait que l'intrus i sait que le votant v_1 peut évaluer la proposition p.

2.2.2 Travaux connexes

Parmi les travaux faisant usage de logiques épistémiques pour la spécification des protocoles de vote électronique, on peut citer [EO07, JP06, BRS07]. Dans [EO07], les auteurs proposent d'utiliser la logique DEL (*Dynamic Epistemic Logic*) pour la spécification de la propriété d'anonymat. La logique sert à la fois à la spécification des propriétés et du protocole à analyser. L'évolution de ce dernier est décrite au moyen de formulations logiques faisant transiter le modèle du protocole d'une configuration à une autre. Plus précisément, le modèle évolue à chaque étape du protocole en fonction de l'évolution de la connaissance des agents du protocole. À titre d'exemple, si on considère que dans le protocole de l'exemple 2.2.1, les résultats sont annoncés selon la forme suivante : (i) "les votes sont identiques" ou (ii) "les votes sont différents", alors l'évolution du modèle de la configuration illustrée par la figure de l'exemple 2.2.1 à celle donnée par la figure 2.1, serait donnée par la formule $p \leftrightarrow q$. L'inconvénient d'une telle méthodologie est que la logique pourrait être expressive pour spécifier les propriétés de sécurité, mais l'est généralement moins pour décrire le comportement du protocole.

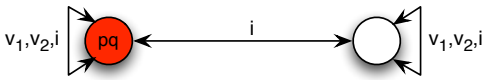


FIGURE 2.1 – Évolution du modèle de l'exemple 2.2.1

Dans [JP06], les auteurs proposent une spécification de la propriété sans-reçu au moyen d'une logique épistémique. La sémantique qu'ils associent à l'opérateur de connaissance $K_a p$ est basée sur des équivalences observationnelles entre exécutions du protocole du point de vue d'un agent a. Une approche similaire a été adoptée par [BRS07]. La logique, dotée également de modalités temporelles, a été utilisée pour la spécification des propriétés d'équité, d'anonymat, de sans-reçcu et de vérifiabilité individuelle. L'analyse formelle a porté sur une version simpliste du protocole FOO qui a été prouvée défaillante par rapport à la propriété sans-reçu.

La Table 2.6 classifie selon l'approche utilisée les travaux détaillés tout au long de ce chapitre. Plus précisément, elle présente pour chacun de ces travaux, le formalisme utilisé, le protocole analysé, les propriétés spécifiées, et les outils associés à l'éventuel automatisation de la vérification de ces propriétés.

TABLE 2.6 – Vérification des protocoles de vote électronique

		Algèbre de processus					Logique modale		
		[KR05]	[DKR06]	[COPD06]	[MVdV07]	[BHM08]	[JP06]	[EO07]	[BRS07]
	Formalisme	π-calcul appliqué	π-calcul appliqué	μ-CRL	ACP	π-calcul appliqué		DEL	
Propriétés	Équité	×							×
	Droit d'expression								
	Éligibilité	×				×			
	Non réutilisabilité					×			
	Exactitude								
	Vérifiabilité individuelle								×
	Vérifiabilité universelle								
	Anonymat	×	×	×	×			×	×
	Sans-Reçu		×			×	×		×
	Résistance à la coercition		×			×			
	Autre					Inaltérabilité			
	Protocole	FOO	LBDKYY [LBD+03][a]	FOO	FOO	JCJ [JCJ05]	-	FOO[a]	FOO[a]
	Automatisation	Partielle[b]	Partielle[b]	Oui	Non	Oui	Non	Oui	Non
	(**Outils**)	ProVerif	ProVerif	Outils μ-CRL		ProVerif		CADP LYS	

a. L'analyse est portée sur une version simplifiée du protocole.
b. Certaines propriétés sont prouvées manuellement.

Dans l'approche basée sur les algèbres de processus, l'accent est mis sur la spécification du protocole, tandis que dans la seconde approche, il est mis sur la spécification des propriétés. Il est à noter que, tel qu'il est illustré par la figure 2.2, les deux approches ne sont pas totalement indissociables. Un protocole peut être spécifié au moyen de l'algèbre la plus adéquate, et les propriétés de sécurité peuvent être spécifiées au moyen de la logique la plus appropriée.

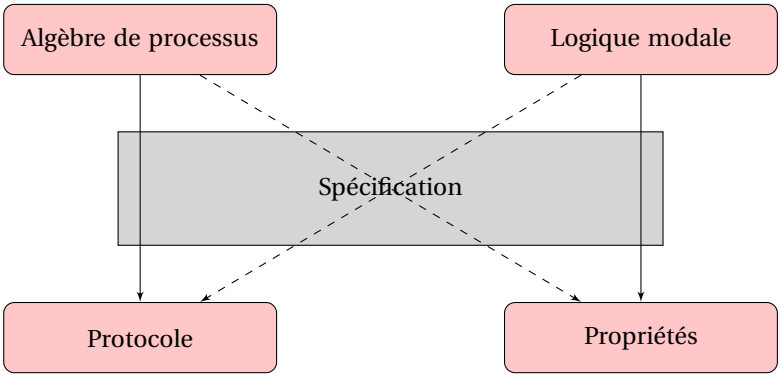


FIGURE 2.2 – Spécification du protocole et des propriétés

Force est de constater que la principale technique employée pour la vérification des propriétés des protocoles de vote électronique repose sur du *model checking* consistant à vérifier des relations de type $\mathscr{M} \approx \mathscr{M}'$ ou bien $\mathscr{M} \models \phi$. La première ($\mathscr{M} \approx \mathscr{M}'$) fait référence à la relation d'équivalence observationnelle et consiste à vérifier si une implémentation du protocole (modèle $\mathscr{M}$) est équivalente à sa spécification (modèle $\mathscr{M}$'). La seconde ($\mathscr{M} \models \phi$) se résume à vérifier si le modèle du protocole $\mathscr{M}$ satisfait une certaine spécification logique ϕ.

Notre approche pour l'analyse des protocoles de vote électronique repose aussi sur la vérification de la relation de satisfaction $\mathscr{M} \models \phi$, où $\mathscr{M}$ désigne un modèle à base de traces et ϕ une propriété de sécurité exprimée au moyen de la logique modale ADM [ADM03]. Dans ce qui suit, nous introduisons cette logique par l'intermédiaire de sa syntaxe et sa sémantique et décrivons son usage dans la spécification des propriétés de sécurité et l'analyse des protocoles cryptographiques.

2.3 Analyse des protocoles cryptographiques au moyen de la logique ADM

Dans [ADM03], Adi, Debbabi et Mejri ont proposé la logique ADM dans le but d'analyser les protocoles de commerce électronique. L'approche se résume à la spécification au moyen de cette logique des propriétés de sécurité (ϕ), et la vérification de ces propriétés par rapport à des traces d'exécution du protocole. Lorsqu'une trace t du protocole analysé ne satisfait pas une formule ϕ ($t \not\models \phi$), celle-ci est considérée comme une preuve de la violation de la propriété de sécurité exprimée par la formule ϕ. Il en résulte que la sécurité d'un protocole quant à la garantie d'une propriété donnée ne peut être affirmée que lorsque l'ensemble de ses traces est analysé.

2.3.1 Modèle du protocole

Dans un modèle à base de traces, une trace consiste en une séquence d'événements et est définie de manière formelle comme suit :

$$t \quad ::= \quad \epsilon \quad | \quad t.a$$

Dans cette définition, ϵ désigne la trace vide, et '.' représente l'opérateur de concaténation des actions a du protocole. Dans le cadre de l'analyse des protocoles cryptographiques, deux types d'actions ont été identifiés dans [ADM03] :

- Une action interne est une opération de traitement interne. Elle prend la forme suivante $\alpha.i\text{-}act(A, p_1, \ldots, p_n)$ et dénote l'exécution de l'action act par l'entité A sur l'ensemble des paramètres $p_1, \ldots, p_n$ à l'étape i de la session α du protocole.
- Une action externe est une opération de communication. Elle se présente selon la forme $\alpha.i\text{-}A \triangleright B : m$ (respectivement $\alpha.i\text{-}A \triangleleft B : m$) pour dénoter une opération d'envoi (respectivement de réception) durant laquelle une entité A envoie à (respectivement reçoit de) une entité B un message m à l'étape i de la session α du protocole.

Le format des messages échangés est quant à lui donné par la grammaire BNF suivante :

$$m \quad ::= \quad M \quad | \quad m_1, m_2 \quad | \quad \{m\}_k \quad | \quad k \quad | \quad cte \quad | \quad f(m)$$

Dans cette algèbre des messages, une lettre majuscule M identifie l'entité communicante, m_1, m_2 désigne une paire de message, et $\{m\}_k$ représente le chiffrement du message m par la clé k. Finalement, par opposition à cte (constante), $f(m)$ représente les messages non constants comme application de fonctions f sur un message m. À titre d'exemple, le message $N(A, \alpha)$ peut être utilisé pour désigner un *nonce* généré par une entité A à la session α du protocole.

L'analyse des protocoles s'opère dans cette approche par la vérification des propriétés spécifiées par rapport à des traces valides du protocole. Une trace est dite valide si les entités honnêtes agissent conformément aux règles du protocole, et si les messages transmis par l'intrus sont déductibles à partir de sa connaissance initiale et des messages interceptés par écoute du trafic. Il existe plusieurs approches pour construire l'ensemble des traces valides. Dans le chapitre 3 nous détaillons celle adoptée par Paulson dans [Pau98] tout en la mettant en application sur le protocole FOO décrit dans le chapitre précédent (voir section 1.4.1).

Notons que le modèle de l'intrus communément adopté est celui de Dolev-Yao [DY83], par référence à ces concepteurs. Ce modèle suppose que le réseau est sous le contrôle d'un intrus ("l'intrus est le réseau"), disposant par conséquent d'un contrôle total des canaux publiques sur lesquels il est capable d'effectuer certaines opérations telles que l'écoute, l'interception ou bien le rejeu de messages. L'intrus peut également injecter des messages sur le réseau si jamais il est capable de les construire à partir de sa connaissance cumulée. Celle-ci englobe sa connaissance initiale et celle déduite par écoute du trafic. Dans le modèle Dolev-Yao, l'intrus peut dériver de nouveaux messages par application de simples primitives (chiffrement, concaténation) sur les messages qu'il détient. Cette faculté est formalisée par des règles d'inférence telles que celles données en Table 2.7 où la notation $M \vdash_K m$ signifie que l'intrus est capable de dériver le message m à partir de l'ensemble des messages M, de sa connaissance initiale K, et de ses règles $\vdash$. La règle $compose$ modélise par exemple la capacité de l'intrus à former la paire de messages m_1, m_2

s'il a en sa possession à la fois les messages m_1 et m_2. La règle $symenc$ signifie que l'intrus est capable de construire le message $\{m\}_k$ s'il dispose en sa connaissance des messages m et k. Dans ce modèle, la cryptographie est supposée parfaite, ce qui signifie que l'intrus ne peut déduire le message m à partir de sa version chiffrée $\{m\}_k$ sans la connaissance de la clé k.

TABLE 2.7 – Règles de déduction

$(\boldsymbol{init})$	$\frac{}{M \vdash_K m} \quad [m \in M \cup K]$
$(\boldsymbol{compose})$	$\frac{M \vdash_K m_1 \quad M \vdash_K m_2}{M \vdash_K m_1, m_2}$
$(\boldsymbol{decompose})$	$\frac{M \vdash_K m_1, m_2}{M \vdash_K m_i} \quad i = 1,2$
$(\boldsymbol{symenc})$	$\frac{M \vdash_K m \quad M \vdash_K k}{M \vdash_K \{m\}_k}$
$(\boldsymbol{symdec})$	$\frac{M \vdash_K \{m\}_k \quad M \vdash_K k}{M \vdash_K m}$

2.3.2 La logique ADM

La logique ADM peut être vue comme une variante du μ-calcul modal [Sti96]. Initialement conçue pour la spécification des propriétés des protocoles de commerce électronique (atomicité des biens (*good atomicity property*), atomicité de l'argent (*money atomicity property*)), elle permet également la spécification des propriétés de sécurité classiques des protocoles cryptographiques (authentification, confidentialité).

Syntaxe. La syntaxe de la logique ADM est basée sur des *patterns*. Un *pattern* est une abstraction de la trace où certaines actions sont remplacées par des variables. La syntaxe d'un *pattern* est donnée par la grammaire BNF de la Table 2.8 où ϵ, x et a dénotent respectivement un *pattern* vide, une variable, et une action.

TABLE 2.8 – Syntaxe d'un *pattern*

p	::=	$\epsilon \mid x.p \mid a.p$

Selon la syntaxe de la Table 2.8, un *pattern* est défini comme une séquence de variables et d'actions. Ces derniers peuvent contenir à leur tour des variables. À titre d'exemple, dans le *pattern* p donné ci-après, nous identifions des variables "pattern" (x_1 et x_2), une variable "session" (x_α), une variable "étape" (x_s), une variable "entité" (X_ν) et une variable "message" (x_m).

$$p = x_1.(x_\alpha.x_s - X_\nu \rhd A : x_m).x_2$$

Un *pattern* représente l'ingrédient de base des formules spécifiées dans la logique ADM. Afin de simplifier la spécification des propriétés de sécurité, les concepteurs de la logique ont introduit les abréviations de *pattern* données en Table 2.9.

TABLE 2.9 – Abréviations (*patterns*)

ϵ^+	$\equiv$	x	ϵ^-	$\equiv$	ϵ
$(x.p)^+$	$\equiv$	$x.p^+$	$(x.p)^-$	$\equiv$	$x.p^-$
$(a.p)^+$	$\equiv$	$x.a.p^+$	$(a.p)^-$	$\equiv$	p^-

Comme nous le verrons dans le chapitre 4, ces abréviations contribuent notamment à la simplification de la spécification des propriétés telles que la démocratie ou l'exactitude.

Exemple 2.3.1 (*Abréviations de pattern*)

Dans les patterns p_a et p_b définis ci-dessous, x, y et z désignent des variables, et a et b représentent deux actions abstraites :

p_a	$=$	$a.a$	p_b	$=$	$x.b.y.b.z$
p_a^+	$=$	$x.a.y.a.z$	p_b^-	$=$	$x.y.z$

La syntaxe de la logique ADM est définie par la grammaire BNF donnée en Table 2.10. Les symboles $\neg$ et $\wedge$ représentent respectivement la négation et la conjonction. X est une variable propositionnelle. $\nu X.\phi$ est une formule récursive. Finalement, $[p_1 \looparrowright p_2]$ est un opérateur modal indexé par les *patterns* p_1 et p_2, avec la condition que les variables contenues dans le second *pattern* doivent êtres incluses dans le *pattern* p_1. Une trace t satisfait la formule $[p_1 \looparrowright p_2]\phi$, si pour toutes les substitutions σ telles que $p_1\sigma = t$, la nouvelle version de la trace donnée par $t' = p_2\sigma$ satisfait la seconde partie de la formule (ϕ).

TABLE 2.10 – Syntaxe de la logique ADM

ϕ	$::=$	X	$\mid$	$\neg\phi$	$\mid$	$\phi_1 \wedge \phi_2$	$\mid$	$[p_1 \looparrowright p_2]\phi$	$\mid$	$\nu X.\phi$

Il est à noter ici que la syntaxe de la logique ADM est donnée sous sa forme minimale, et que d'autres opérateurs peuvent être dérivés à partir de celle-ci. La Table 2.11 illustre les abréviations de formules associées à la logique ADM.

TABLE 2.11 – Abréviations (formules)

tt	$\equiv$	$\nu X.X$
ff	$\equiv$	$\mu X.X$
$\langle p_1 \looparrowright p_2\rangle\phi$	$\equiv$	$\neg[p_1 \looparrowright p_2]\neg\phi$
$\mu X.\phi$	$\equiv$	$\neg\nu X.\neg\phi[\neg X/X]$
$\phi_1 \vee \phi_2$	$\equiv$	$\neg(\neg\phi_1 \wedge \neg\phi_2)$
$\phi_1 \rightarrow \phi_2$	$\equiv$	$\neg\phi_1 \vee \phi_2$
$\phi_1 \leftrightarrow \phi_2$	$\equiv$	$\phi_1 \rightarrow \phi_2 \wedge \phi_2 \rightarrow \phi_1$

Les opérateurs $\vee$, $\rightarrow$ et $\leftrightarrow$ correspondent respectivement à la disjonction, à l'implication et à l'équivalence, et peuvent être utilisés dans les formules avec leur sens usuel. À partir de l'opérateur modal $[-]$, il est possible de dériver l'opérateur dual $\langle-\rangle$. La formule $\langle p_1 \looparrowright p_2\rangle\phi$ s'interprète

par rapport à une trace t de la manière suivante : (i) vérifier s'il existe au moins une substitution σ qui fait coïncider le *pattern* p_1 à la trace t ($t = p_1\sigma$), et (ii) vérifier ensuite si la trace t', donnée par application de la substitution trouvée sur le *pattern* p_2 ($t' = p_2\sigma$), satisfait le restant de la formule (ϕ). Finalement, à partir du plus grand point fixe ν, il est possible de dériver la tautologie tt, et le plus petit point fixe μ ($\mu X.\phi \equiv \neg\nu X.\neg\phi[\neg X/X]$). Le terme $\phi[\Gamma/X]$ représente le remplacement simultanée de toutes les occurrences de la variable X dans la formule ϕ par Γ.

Sémantique dénotationnelle. La sémantique de la logique ADM est illustrée en Table 2.12 où l'interprétation des formules est donnée par la fonction suivante :

$$[\![-]\!]^{\_,\_}_{\_} : \mathscr{L} \times \mathscr{T} \times Sub \times Env \to 2^{\mathscr{T}}$$

- L'ensemble $\mathscr{L}$ représente l'ensemble des formules de la logique.
- L'ensemble $\mathscr{T}$ désigne l'ensemble des traces valides du protocole.
- L'ensemble Sub dénote l'ensemble des substitutions σ telle que :

$$\sigma \in [\mathscr{V}_p \to \mathscr{T}] \circ [\mathscr{V}_{ses} \to I_{ses}] \circ [\mathscr{V}_{stp} \to I_{stp}] \circ [\mathscr{V}_{prp} \to I_{prp}] \circ [\mathscr{V}_{msg} \to M]$$

  Où $\mathscr{V}_p$, $\mathscr{V}_{ses}$, $\mathscr{V}_{stp}$, $\mathscr{V}_{prp}$ et $\mathscr{V}_{msg}$ représentent respectivement l'ensemble des variables "pattern", l'ensemble des variables "session", l'ensemble des variables "étape", l'ensemble des variables "entité", et l'ensemble des variables "message". Les ensembles d'arrivée I_{ses}, I_{stp} et I_{prp} représentent respectivement l'ensemble des sessions, l'ensemble des étapes et l'ensemble des entités du protocole. L'ensemble M désigne quant à lui l'ensemble des messages. Il est à noter que la définition de l'ensemble Sub implique que chaque variable d'un ensemble donné $\mathscr{V}_x$ ne peut être substitituée que par des éléments d'un ensemble particulier $\mathscr{I}_x$.

  L'exemple 2.3.2 illustre une substitution σ faisant coïncider un pattern p à une trace t :

Exemple 2.3.2

Soient le pattern p et la trace t définis comme suit :

$$\begin{cases} p &= x_1.(x_\alpha.x_s - X_s \rhd X_r : x_m).x_2 \\ t &= (\alpha.1 - A \rhd B : m).(\alpha.1 - B \lhd A : m) \end{cases}$$

Dans le pattern p, nous distinguons deux variables "pattern" ($x_1, x_2 \in \mathscr{V}_p$), une variable "session" ($x_\alpha \in \mathscr{V}_{ses}$), une variable "étape" ($x_s \in \mathscr{V}_{stp}$), deux variables "entité" ($X_s, X_r \in \mathscr{V}_{prp}$), et une variable "message" ($x_m \in \mathscr{V}_{msg}$).

La trace t décrit une séquence d'événements durant lesquels un message m a été envoyé par une entité A et reçu par une entité B à l'étape 1 de la session α d'un protocole donné P.

La substitution $\sigma \in Sub$ suivante permet de faire coïncider le pattern p à la trace t ($t = p\sigma$) :

$$\sigma = \{x_1 \mapsto \epsilon, x_\alpha \mapsto \alpha, x_s \mapsto 1, X_s \mapsto A, X_r \mapsto B, x_m \mapsto m, x_2 \mapsto (\alpha.1 - B \lhd A : m)\}$$

– L'ensemble Env représente l'ensemble des environnements possibles dans $[\mathcal{V} \to 2^{\mathcal{T}}]$, où $\mathcal{V}$ est l'ensemble des variables propositionnelles. L'environnement associe à chaque variable $X \in \mathcal{V}$ l'ensemble des traces qui la satisfont.
La notation auxiliaire $e[X \mapsto U]$ est utilisée dans la sémantique des formules récursives (voir Table 2.12), et est introduite dans le but de désigner un environnement e' qui se distingue d'un environnement initial e par l'ensemble de traces U qui est associé à la variable X dans l'environnement e'. Plus précisément, e' est construit à partir de e comme suit :

$$\begin{aligned} e'(Y) &= e(Y) \quad \text{Si } Y \neq X \\ e'(X) &= U \end{aligned}$$

Maintenant, une trace t satisfait une formule ϕ si $t \in [\![\phi]\!]_e^{t,\sigma}$ pour une substitution $\sigma \in Sub$ et un environnement $e \in Env$ initialement vide. Plus précisément, $[\![\phi]\!]_e^{t,\sigma}$ désigne l'ensemble des traces issues de $\tilde{t}$ et satisfaisant la formule ϕ. La notation $\tilde{t}$ désigne l'ensemble des traces pouvant être extraites de t par élimination des actions se trouvant au début, à la fin et/ou au milieu de la trace. Formellement, l'ensemble $\tilde{t}$ est défini de manière inductive comme suit :

$$\begin{aligned} &\text{(i)} \quad t \in \tilde{t}, \\ &\text{(ii)} \quad t_1.a.t_2 \in \tilde{t} \Rightarrow t_1.t_2 \in \tilde{t} \end{aligned}$$

Exemple 2.3.3

$$\begin{aligned} t &= a.b.c \\ \tilde{t} &= \{\epsilon, a, b, c, a.b, a.c, b.c, a.b.c\} \end{aligned}$$

La sémantique dénotationnelle est basée sur un paradigme de *model checking* global où la vérification de la relation de satisfaction $t \models \phi$ consiste à rechercher d'abord l'ensemble des traces de $\tilde{t}$ satisfaisant la formule ϕ ($[\![\phi]\!]_e^{t,\sigma}$), et vérifier ensuite si t est un élément de l'ensemble résultant ou pas.

La Table 2.12 donne la construction de l'ensemble $[\![\phi]\!]_e^{t,\sigma}$ pour chacune des formules ϕ de logique ADM donnée en Table 2.10, à savoir les formules X, $\neg\phi$, $\phi_1 \wedge \phi_2$, $[p_1 \looparrowright p_2]\phi$ et $\nu X.\phi$:

TABLE 2.12 – Sémantique dénotationnelle de la logique ADM

$$\begin{aligned} [\![X]\!]_e^{t,\sigma} &= e(X) \\ [\![\neg\phi]\!]_e^{t,\sigma} &= \tilde{t} - [\![\phi]\!]_e^{t,\sigma} \\ [\![\phi_1 \wedge \phi_2]\!]_e^{t,\sigma} &= [\![\phi_1]\!]_e^{t,\sigma} \cap [\![\phi_2]\!]_e^{t,\sigma} \\ [\![[p_1 \looparrowright p_2]\phi]\!]_e^{t,\sigma} &= \{u \in \tilde{t} \mid \forall \sigma' : p_1\sigma\sigma' = u \Rightarrow p_2\sigma\sigma' \in [\![\phi]\!]_e^{p_2\sigma\sigma', \sigma'\circ\sigma}\} \\ [\![\nu X.\phi]\!]_e^{t,\sigma} &= \nu f \quad \text{avec} \left\{ \begin{array}{ccccc} f & : & 2^{\mathcal{T}} & \to & 2^{\mathcal{T}} \\ & & U & \mapsto & [\![\phi]\!]_{e[X\mapsto U]}^{t,\sigma} \end{array} \right. \end{aligned}$$

- Une trace t satisfait la variable propositionnelle X ($t \in [\![X]\!]_e^{t,\sigma}$), si elle appartient à l'environnement de celle-ci ($t \in e(X)$).
- Intuitivement une trace t satisfait une formule $\neg\phi$ ($t \in [\![\neg\phi]\!]_e^{t,\sigma}$) si la trace t ne satisfait pas la formule ϕ ($t \notin [\![\phi]\!]_e^{t,\sigma}$). La sémantique associée à la négation ($\neg\phi$) se traduit donc par toute trace $t' \in \tilde{t}$ ne satisfaisant pas les conditions requises par la formule ϕ.
- Une trace t satisfait la formule $\phi_1 \wedge \phi_2$ ($t \in [\![\phi_1 \wedge \phi_2]\!]_e^{t,\sigma}$) si la trace t satisfait à la fois les formules ϕ_1 et ϕ_2.
- Une trace t satisfait une formule de la forme $[p_1 \looparrowright p_2]\phi$ ($t \in [\![[p_1 \looparrowright p_2]]\!]_e^{t,\sigma}$) si pour toutes les substitutions σ' vérifiant $p_1\sigma\sigma' = t$, la trace $t' = p_2\sigma\sigma'$ satisfait ϕ.
- Finalement, la sémantique de la formule récursive $\nu X.\phi$ est donnée par le plus grand point fixe d'une fonction f. Une trace t satisfait la formule récursive $\nu X.\phi$ ($t \in [\![\nu X.\phi]\!]_e^{t,\sigma}$) si la trace t est un élément du plus grand ensemble de traces $U \subseteq \tilde{t}$ tel que U soit inclut dans son image par f ($U \subseteq [\![\phi]\!]_{e[X \mapsto U]}^{t,\sigma}$).

  Dans [ADM03], les auteurs montrent que la fonction f est monotone pour une certaine restriction sur la syntaxe des formules récursives ($\nu X.\phi$) nécessitant que toutes les occurrences libres de X dans ϕ apparaissent dans la portée d'un nombre pair de négation. D'après le théorème de Knaster-Tarski [Tar55], la monotonicité de la fonction f sur le treillis $(2^{\mathcal{T}}, \subseteq, \emptyset, \mathcal{T}, \cup, \cap)$ garantit l'existence du plus grand point fixe νf :

$$\nu f \quad = \quad \bigcup \{U \subseteq \mathcal{T} \mid U \subseteq f(U)\}$$

  De même, la fonction f admet un plus petit point fixe (μf) défini comme suit :

$$\mu f \quad = \quad \bigcap \{U \subseteq \mathcal{T} \mid f(U) \subseteq U\}$$

Exemples. Les exemples donnés ci-dessous mettent en évidence certaines des caractéristiques de la logique ADM, et permettent de mieux assimiler la sémantique associée à ses constructeurs.

La logique ADM est dotée à la fois de modalités temporelles permettant de s'interroger sur l'ordre d'occurrence des événements, et de modalités existentielles permettant de s'interroger sur la possibilité de ces événements. L'exemple suivant illustre ces deux caractéristiques :

Exemple 2.3.4 (*Modalités temporelles & existentielles*)

Soient la trace t et la formule ϕ suivantes :

$$\begin{cases} t & = & b.a.d \\ \phi & = & \langle x_1.b.x_2.d.x_3 \looparrowright \epsilon \rangle\, tt \end{cases}$$

La formule ϕ consiste à vérifier si la trace analysée contient les actions abstraites b et d telle que l'action b précède l'action d.

Le processus de vérification s'opère en deux étapes :

1. *La première partie de la formule ϕ ($\langle x_1.b.x_2.d.x_3 \looparrowright \epsilon \rangle$) permet de vérifier s'il existe une substitution, qui appliqué au premier pattern ($p_1 = x_1.b.x_2.d.x_3$), coïnciderait avec la trace t.*

Cette partie de la formule est vérifiée étant donné que la substitution suivante

$$\sigma \quad = \quad \{x_1 \mapsto \epsilon, x_2 \mapsto a, x_3 \mapsto \epsilon\}$$

garantit la condition requise ($t = p_1\sigma$) :

$$t \quad = \quad (x_1.b.x_2.d.x_3)\sigma = b.a.d$$

La substitution σ est ensuite appliquée sur le second pattern $p_2 = \epsilon$, donnant lieu à une trace $t' = p_2\sigma = \epsilon$.

2. *La seconde partie de la formule ϕ (tt) est évaluée par rapport à la trace t'. Cette trace satisfait cette partie de la formule, étant donné que la formule tt est, par définition, satisfaite par toutes les traces.*

Nous en concluons donc que la trace t satisfait la formule ϕ ($t \models \phi$). À noter que ce résultat peut être retrouvé en vérifiant que :

$$t \quad \in \quad [\![\phi]\!]^{t,\emptyset}_{\emptyset}$$

Étant donné que la Table 2.12 ne présente que la sémantique des opérateurs de base de la logique ADM, nous devons transformer au préalable la formule ϕ au moyen des abréviations données en Table 2.11 :

$$\langle x_1.b.x_2.d.x_3 \looparrowright \epsilon \rangle tt \quad \equiv \quad \neg[x_1.b.x_2.d.x_3 \looparrowright \epsilon]\neg \nu X.X$$

L'ensemble des traces issues de $\tilde{t} = \{\epsilon, b, a, d, b.a, b.d, a.d, b.a.d\}$ et vérifiant la formule ϕ est obtenu comme suit :

$$\begin{aligned} [\![\phi]\!]^{t,\emptyset}_{\emptyset} &= [\![\neg[x_1.b.x_2.d.x_3 \looparrowright \epsilon]\neg \nu X.X]\!]^{t,\emptyset}_{\emptyset} \\ &= \tilde{t} - [\![[x_1.b.x_2.d.x_3 \looparrowright \epsilon]\neg \nu X.X]\!]^{t,\emptyset}_{\emptyset} \\ &= \tilde{t} - \{\epsilon, b, a, d, b.a, a.d\} \\ &= \{b.d, b.a.d\} \end{aligned}$$

Ce calcul prouve que la trace t satisfait la formule ϕ étant donné que la trace $t = b.a.d \in [\![\phi]\!]^{t,\emptyset}_{\emptyset}$.

L'exemple suivant est plus élaboré, et met en évidence une autre caractéristique intéressante de la logique : la linéarité. Comparé au μ-calcul modal, la logique ADM est doté d'un opérateur spécial "$\looparrowright$" la transformant en une logique linéaire. Cette caractéristique fait référence à la consommation de ressources dans la trace. Concrètement, elle lui confère la possibilité d'effectuer des modifications dynamiques à la trace (supprimer des actions, ajouter des actions, ou bien substituer certaines actions par d'autres). Comme nous le verrons dans le chapitre 4, cette caractéristique est fondamentale pour la spécification des propriétés des protocoles de vote électronique (équité, anonymat, etc.), et particulièrement lorsqu'elle est couplée aux opérateurs de récursivité.

Exemple 2.3.5 (*Linéarité*)

Soient la trace t et la formule ϕ suivantes :

$$\begin{cases} t & = & a.d.b.c.b.a \\ \phi & = & [x_1.a.x_2.b.x_3 \looparrowright x_1.x_2.x_3]\langle x_4.c.x_5.a.x_6 \looparrowright x_4.e.x_5.x_6\rangle\, tt \end{cases}$$

Le processus de vérification s'opère selon les étapes suivantes :

1. *La première partie de la formule ($[x_1.a.x_2.b.x_3 \looparrowright x_1.x_2.x_3]$) permet d'extraire toutes les substitutions faisant coïncider le pattern $x_1.a.x_2.b.x_3$ avec la trace t. Les substitutions trouvées (σ_1 et σ_2) sont ensuite appliquées sur le pattern de droite ($x_1.x_2.x_3$) qui définit les modifications devant être apportées à la trace. Dans notre cas, une occurrence de chacune des actions a et b est supprimée de la trace.*

$$\begin{cases} \sigma_1 = \{x_1 \mapsto \epsilon, x_2 \mapsto d, x_3 \mapsto c.b.a\} & t_1 = (x_1.x_2.x_3)\sigma_1 = d.c.b.a \\ \sigma_2 = \{x_1 \mapsto \epsilon, x_2 \mapsto d.b.c, x_3 \mapsto a\} & t_2 = (x_1.x_2.x_3)\sigma_2 = d.b.c.a \end{cases}$$

2. *La seconde partie de la formule ($\langle x_4.c.x_5.a.x_6 \looparrowright x_4.e.x_5.x_6\rangle$) doit être maintenant évaluée par rapport aux traces t_1 et t_2. La trace t_1 satisfait cette partie de la formule étant donné qu'il existe une substitution σ_3 telle que $t_1 = (x_1.c.x_2.a.x_3)\sigma_3$. Il en est de même pour la trace t_2, où la substitution σ_4 remplit également la condition requise.*

$$\begin{cases} \sigma_3 = \{x_4 \mapsto d, x_5 \mapsto b, x_6 \mapsto \epsilon\} & t_3 = (x_4.e.x_5.x_6)\sigma_3 = d.e.b \\ \sigma_4 = \{x_4 \mapsto d.b, x_5 \mapsto \epsilon, x_6 \mapsto \epsilon\} & t_4 = (x_4.e.x_5.x_6)\sigma_4 = d.b.e \end{cases}$$

3. *Finalement, la dernière partie de la formule (tt) doit être évaluée par rapport aux traces t_3 et t_4. Ces deux traces satisfont cette partie de la formule, étant donné que la formule tt est satisfaite par toutes les traces. La trace t satisfait par conséquent la formule ϕ.*

Il est à noter que le pattern $x_4.e.x_5.x_6$ est superflu dans la formule ϕ et peut être remplacé par ϵ. Il a été rajouté dans le seul but d'expliquer un peu plus comment il est possible d'agir dynamiquement sur la trace analysée.

La logique ADM permet également la spécification de formules récursives. Cette caractéristique est d'autant plus intéressante lorsqu'elle est couplée à la linéarité. En effet, ces deux caractéristiques combinées, dotent la logique ADM de la possibilité de spécifier des propriétés ayant recours au comptage d'actions dans la trace. Cette faculté semble cruciale dans la spécification des propriétés des protocoles de vote électronique. À titre d'exemple, dans la propriété d'exactitude, nous avons besoin de vérifier si le nombre d'occurrences d'une action a ($cast(v)$) est égal au nombre d'occurrences d'une action b ($publish(v)$), avec la contrainte supplémentaire que l'action de vote précède celle de sa publication. Ceci peut être formalisée comme donné par l'exemple 2.3.6.

Exemple 2.3.6 (*Récursivité*)

Soient la trace t et la formule ϕ suivantes :

$$\begin{cases} t & = & d.a.c.b.d.b \\ \phi & = & \nu X.((\langle x_1.a.x_2 \looparrowright \epsilon\rangle \vee \langle x_1.b.x_2 \looparrowright \epsilon\rangle) \rightarrow \langle x_1.a.x_2.b.x_3 \looparrowright x_1.x_2.x_3\rangle X) \end{cases}$$

La partie gauche de la formule $(\langle x_1.a.x_2 \not\mapsto \epsilon\rangle \vee \langle x_1.b.x_2 \not\mapsto \epsilon\rangle)$ *permet de vérifier si la trace contient l'une des actions* a *ou* b*. Si cette condition est satisfaite, alors la partie droite de la formule permet de vérifier si la trace contient une action* a*, suivie, non nécessairement immédiatement, par une action* b*. Si cette condition est également satisfaite, alors la formule doit être vérifiée de nouveau par rapport aux traces résultantes.*

Il est clair ici que la trace t *ne satisfait pas la propriété exprimée par* ϕ*. En effet, si on enlève une occurrence de l'action* a *et une occurrence de l'action* b*, aucune des traces obtenues* ($t_1 = d.c.d.b$, $t_2 = d.c.b.d$) *ne satisfait la formule* ϕ*.*

2.3.3 Application aux protocoles cryptographiques

Un des avantages de la logique ADM dans l'analyse des protocoles cryptographiques est qu'elle offre un certain nombre de caractéristiques intéressantes pour l'analyse des traces : modalités existentielles et temporelles, linéarité, récursivité. Les exemples 2.3.4, 2.3.5 et 2.3.6 témoignent de son pouvoir d'expression et de sa faculté à exprimer certains aspects tels que le comptage d'actions, fondamental dans la spécification des propriétés des protocoles de commerce électronique telles que l'atomicité de l'argent où il est requis que le nombre d'occurrences d'une action c ($credit(m)$) dans toute trace valide d'exécution du protocole soit égal au nombre d'occurrences d'une action d ($debit(m)$).

Autre avantage de la logique ADM est qu'elle est accompagnée d'un système de preuve basé sur la méthode des tableaux [GLM97] permettant une implémentation efficace d'un outil automatisant la tâche de la vérification des propriétés de sécurité. Ce système sera détaillé en chapitre 5 où nous présentons notre outil ADM-Checker.

La logique ADM a été utilisée afin d'exprimer à la fois les propriétés classiques des protocoles cryptographiques (authentification, confidentialité) et les propriétés de sécurité spécifiques aux protocoles de commerce électronique (atomicité de l'argent, atomicité des biens). Le tableau 2.13 récapitule les principales propriétés spécifiées. Cependant, nous notons ici que seule la propriété d'authentification a été évaluée par rapport à des exemples de protocole connus (Woo-Lam [WL94], Needham-Schroeder), où l'analyse a permis de mettre en évidence certaines des attaques possibles sur ces protocoles. Les autres propriétés n'ont été illustrées que sur des exemples simplistes de protocoles, et bien que la logique soit dotée d'un système de preuve permettant la vérification automatique des propriétés spécifiées, ce dernier n' a pas été implémenté.

TABLE 2.13 – Propriétés de sécurité spécifiées au moyen de la logique ADM

Propriété	**Protocole**
Authentification	Needham-Schroeder[1]
	Woo-Lam
Confidentialité	-
Atomicité de l'argent	Exemple[2]
Atomicité des biens	Exemple[2]

[1] Version simplifiée du protocole.

[2] Protocole simpliste de commerce électronique.

2.4 Difficulté de la vérification des protocoles cryptographiques

La vérification des protocoles cryptographiques est connue pour être un problème difficile. Cette difficulté émane essentiellement du caractère non borné des paramètres du protocole à analyser (nombre de participants dans le protocole, nombre de sessions parallèles), et à l'environnement hostile dans lequel l'intrus est capable de construire, puis d'insérer arbitrairement dans le réseau, une infinité de messages. Il en résulte que la vérification des protocoles cryptographiques est généralement indécidable [CDL+99, CS02], et l'automatisation du processus d'analyse est de ce fait souvent problématique. Dans [Cor05], Cortier propose une synthèse des principaux résultats de décidabilité pour les propriétés classiques de sécurité (confidentialité, authentification). Ces résultats sont discutés en tenant compte de diverses restrictions telles que la limitation du nombre de sessions ou bien de la profondeur des messages constructibles par l'intrus.

Pour autant, nous considérons que l'utilisation de méthodes formelles pour la vérification des protocoles cryptographiques devrait être incontournable. Face à cette problématique, deux approches complémentaires sont envisageables. La première est celle adoptée par Genet et Klay dans [GK00] qui proposent d'analyser les protocoles cryptographiques en considérant une sur-approximation des états atteignables par le protocole. Dans leurs travaux, les états initiaux du protocole sont représentés par un automate d'arbres $\mathscr{A}_0$. Les étapes du protocole ainsi que le pouvoir d'analyse de l'intrus sont quant à eux décrits par un système de réécriture $\mathscr{R}$ opérant sur des termes reconnus par l'automate initial $\mathscr{A}_0$. La sur-approximation est donnée par l'automate d'arbres final $\mathscr{A}_f$ calculée à partir de l'automate initial $\mathscr{A}_0$, des règles de réécriture du système $\mathscr{R}$, et d'une fonction d'approximation γ. À signaler ici que l'automate $\mathscr{A}_f$ englobe $\mathscr{A}_0$ et est complet par rapport à $\mathscr{R}$. Le protocole est prouvé comme sûr vis à vis d'une propriété de sécurité p s'il n'y a pas d'intersection entre l'automate final $\mathscr{A}_f$ et l'automate $\mathscr{A}_p$ modélisant une violation de la propriété p. Cette approche, utilisée dans l'outil AVISPA [ABB+05], permet ainsi de confirmer la sécurité d'un protocole lorsque ce dernier est réellement sûr. Cependant, l'approche ne permet pas de mettre en évidence la faille lorsque celle-ci existe.

Par contraste, nous proposons dans cette thèse de considérer une seconde approche consistant à vérifier les propriétés spécifiées en logique ADM par rapport à des traces valides du protocole. Dû à l'infinité de traces pouvant être générées, il nous est impossible de confirmer la sécurité du protocole analysé. Cependant, cette approche permet de mettre en évidence la faille lorsque celle-ci existe, et de disposer de la preuve de la vulnérabilité du protocole. Le scénario d'attaque est illustré dans ce cas par la trace t qui viole la propriété de sécurité exprimée par ϕ.

2.5 Positionnement des travaux de thèse

La vérification des protocoles de vote électronique peut être classée en deux approches : les algèbres de processus [KR05, DKR06, COPD06, MVdV07, BHM08] et les logiques modales [JP06, EO07, BRS07]. La table 2.6 récapitule, en fonction de l'approche et du formalisme utilisé, les propriétés spécifiées et les protocoles analysés. Comme remarqué, les deux approches ne sont pas totalement indissociables, et la technique de vérification se résume le plus souvent à du *model-checking*.

Par ailleurs, Adi et al. ont montré que leur logique (ADM) est bien adaptée à la vérification des protocoles cryptographiques en général et de commerce électronique en particulier (voir

Table 2.13). Dans cette thèse, nous proposons d'étendre l'usage de la logique ADM à l'analyse des protocoles de vote électronique. Comme discuté en section 2.3.3, les motivations pour un tel choix sont multiples, dont notamment la faculté à exprimer des aspects tels que le comptage d'actions (voir exemple 2.3.6), crucial dans la spécification de certaines propriétés telles que la démocratie ou l'exactitude. Autre avantage de la logique est la possibilité de tirer parti de son système de preuve afin de permettre la vérification automatique des propriétés spécifiées.

De manière similaire aux travaux présentés au niveau de ce chapitre, l'approche que nous adoptons pour l'analyse des protocoles de vote électronique repose aussi sur du *model-checking* consistant en la vérification de la relation de satisfaction $\mathscr{M} \models \phi$, où $\mathscr{M}$ désigne un modèle à base de traces, et ϕ une propriété de sécurité exprimée au moyen de la logique modale ADM. Plus précisément, notre contribution dans cette thèse est triple :

1. proposer tout d'abord un ensemble de notations pour la spécification et la modélisation des protocoles de vote ($\mathscr{M}$),
2. spécifier ensuite les principales propriétés de sécurité (ϕ),
3. implémenter finalement un outil pour la vérification automatique des propriétés spécifiées par rapport au modèle défini ($\mathscr{M} \models \phi$).

Chacune de ces étapes est validée par rapport à une étude de cas portant sur le protocole de vote FOO.

Notre première contribution concerne la modélisation des protocoles de vote. Comme décrit au niveau du chapitre précédent, ces derniers introduisent dans leur sillage des primitives cryptographiques complexes (signature en aveugle, chiffrement homomorphe), et font usage de canaux de communication plus élaborés (privés, anonymes) que ceux habituellement impliqués dans les protocoles cryptographiques classiques (authentification, échange de clés de session). L'analyse des protocoles de vote nécessite donc de considérer une représentation des protocoles plus fine que celle usuellement utilisée. Dans cette thèse nous proposons de prendre pour base les notations introduites en section 2.3.1, et de les enrichir en conséquence avec les spécificités introduites par le vote électronique. Cette extension se traduit par une algèbre de messages plus riche que celle se limitant au chiffrement symétrique, par la prise en compte des canaux de communication anonymes et privés, et par la considération d'un intrus plus puissant dont le pouvoir de déduction est ajusté et étendu conformément à l'algèbre des messages définie. Cette première contribution est présentée dans le chapitre 3.

La seconde partie de la thèse est relative à la spécification des propriété de sécurité, et constitue la contribution majeure de notre travail [TMT+08a, TMT+08b, TTB09]. Ce travail consiste en la spécification au moyen de la logique ADM d'une sélection de propriétés de sécurité : droit d'expression, éligibilité, non réutilisabilité, exactitude, équité, anonymat et sans-reçu. Cette partie est traitée en chapitre 4, où il est également question d'une analyse détaillée de la sécurité du protocole FOO.

La troisième partie de la thèse est finalement consacrée à la présentation de notre implémentation de l'outil ADM-Checker, qui permet d'automatiser, conformément à la sémantique de la logique ADM, la vérification des propriétés spécifiées par rapport à des traces d'exécution du protocole analysé. Ce travail est présenté en chapitre 5.

Chapitre 3

Modélisation des protocoles de vote électronique

Comme nous l'avons vu précédemment, notre analyse des protocoles de vote nécessite de définir au préalable le modèle par rapport auquel seront vérifiées les propriétés de sécurité. Le modèle adopté, consistant en l'ensemble des traces d'exécution valides du protocole, doit inclure toutes les informations pertinentes à l'analyse des protocoles et doit tenir compte de toutes les spécificités introduites par le vote électronique.

Dans ce chapitre, nous introduisons en premier lieu un ensemble de notations utilisées pour décrire les protocoles de vote électronique. Ces notations constituent une extension de celles présentées en section 2.3.1. Celles-ci permettent de distinguer les différents types de messages échangés, les divers canaux de communication utilisés, et la nature des actions du protocole impliquées. À l'issue de cette étape, nous serons en mesure de représenter par exemple une opération de communication durant laquelle une entité A envoie à une entité B un message, chiffré au moyen d'un algorithme de chiffrement probabiliste, via un canal privé.

Notre modélisation des protocoles tient compte de la présence d'un intrus actif disposant de certaines facultés (usurpation d'identités, interception et rejeu de messages, etc.) dépendantes de la nature du canal de communication. L'attaquant que nous considérons est celui de Dolev-Yao [DY83] dont nous étendons et ajustons son pouvoir de déduction, dans la seconde partie de ce chapitre, au contexte des protocoles de vote électronique.

Nous rappelons finalement le modèle de protocole adopté tout en décrivant l'approche au moyen de laquelle l'ensemble des traces valides est constitué. Cette approche sera illustrée sur le protocole de vote FOO.

3.1 Notations pour la représentation des protocoles

Comme défini au niveau du chapitre précédent, un protocole de vote est un protocole cryptographique consistant en des opérations de communication (envoi et réception) et de traitements internes (vérification de la validité d'une signature) permettant à des entités communicantes de s'échanger des messages, via des canaux de communication (publiques, anonymes, privés), dans le but d'atteindre un objectif particulier (partage d'un secret, vote). Au vu de cette définition, la

représentation des protocoles doit ainsi comprendre le format des messages échangés, la structure des actions impliquées, et la représentation des canaux de communication.

3.1.1 Algèbre des messages

Cette section introduit le format des messages échangés. Dans le cadre de l'analyse des protocoles cryptographiques classiques, ce format se limite le plus souvent à la définition de constructeurs relatifs à la composition de messages, au chiffrement symétrique, ou bien à la signature numérique [ADM03, HM07, Cer01]. Comme décrit dans le chapitre 1, les protocoles de vote électronique usent de primitives cryptographiques plus complexes (schéma d'engagement, signature en aveugle, chiffrement homomorphe, preuve non interactive à divulgation de connaissance). Par conséquent, afin de décrire au mieux les protocoles de vote, et d'affiner ainsi leur analyse, nous proposons d'enrichir l'algèbre des messages présentée au chapitre précédent (voir section 2.3.1). L'algèbre des messages ainsi obtenue est donnée par les Tables 3.1, 3.2 et 3.3.

TABLE 3.1 – Messages atomiques

a	::=	A	entité
	|	d	datum
	|	n	nonce
	|	k	clé

TABLE 3.2 – Clés de chiffrement

k	::=	k_{AB}	clé partagée
	|	k_A	clé privée
	|	$pub(k_A)$	clé publique

Tout d'abord, nous identifions les messages atomiques dont le format est donné par la grammaire BNF de la Table 3.1. Une lettre majuscule dénote l'entité communicante. À titre d'exemple V_i dénote le votant i, et A_j dénote l'autorité j. Ensuite, d (*datum*) dénote toute information élémentaire telle que le vote v. Les messages atomiques englobent finalement les *nonces* n et les clés de chiffrement k. Selon la Table 3.2, cette dernière peut être symétrique (k_{AB} est une clé partagée entre les entités A et B), privée (k_A est la clé associée à l'entité A), ou publique ($pub(k_A)$).

Les messages sont ensuite construits par application de fonctions (concaténation, chiffrement) sur les messages atomiques. La syntaxe des messages est définie par la grammaire de la Table 3.3.

TABLE 3.3 – Syntaxe des messages

m	::=	a	message atomique
	|	m_1, m_2	concaténation
	|	$\{m\}_{k_{AB}}$	chiffrement symétrique
	|	$\{\{m\}\}_{pub(k_A)}$	chiffrement asymétrique
	|	$\{\{m\}\}^n_{pub(k_A)}$	chiffrement asymétrique probabiliste
	|	$\|m\|_{k_A}$	signature numérique
	|	$\mathscr{B}(m,n)$	masquage / aveuglement
	|	$\mathscr{Z}(m_1 \mid m_2)$	preuve à divulgation nulle de connaissance
	|	$\mathscr{S}(m)$	somme/produit
	|	$\mathscr{P}^{f(ns)}_{pub(k_A)}(m)$	agrégation

Tout d'abord, un message peut être atomique ou une séquence de messages. Ensuite, la grammaire définit des constructeurs permettant de représenter les différentes primitives cryptogra-

phiques introduites en section 1.2.1. Nous distinguons trois algorithmes de chiffrement : symétrique, asymétrique et asymétrique probabiliste. Dans le cas de la dernière possibilité, le chiffrement dépend également d'un *nonce n*.

Ensuite, le terme $\|m\|_{k_A}$ représente la signature numérique du message m au moyen de la clé privée associée à l'entité A. Il est à noter que par $\|m\|_{k_A}$, nous désignons à la fois le message en clair et la signature en elle-même du message.

Nous ne définissons pas de constructeur spécifique à la mise en gage (voir section 1.2.2) étant donné que nous considérons cette primitive similaire au chiffrement symétrique. Bien que l'aveuglement (voir section 1.2.3) soit aussi considéré comme similaire au chiffrement symétrique, nous introduisons un constructeur propre à cette primitive afin de modéliser certaines propriétés relatives à la signature en aveugle telles que la capacité d'extraire une signature sur le message m ($\|m\|_{k_A}$), à partir de celle apposée sur le message aveuglé ($\|\mathscr{B}(m,n)\|_{k_A}$) par un facteur n.

Une preuve non interactive à divulgation nulle de connaissance est abstraite par le terme $\mathscr{Z}(m_1 \mid m_2)$ où les termes m_1 et m_2 identifient respectivement la partie secrète et la partie publique de la preuve.

Finalement, le dernier terme de la grammaire représente une agrégation de messages chiffrés. Plus précisément, le constructeur $\mathscr{P}$ a été introduit afin de représenter l'agrégation des votes dans le cas d'un schéma utilisant le chiffrement homomorphe. À titre d'exemple, le message $\mathscr{P}^{f(n_1,n_2)}_{pub(k_A)}(v_1,v_2)$ représente l'agrégation (produit) des votes chiffrés $\{\{v_1\}\}^{n_1}_{pub(k_A)}$ et $\{\{v_2\}\}^{n_2}_{pub(k_A)}$, et à partir duquel, il est possible de dériver la somme/produit des votes $\mathscr{S}(v_1,v_2)$. Il est à noter que le terme $f(ns)$ dans $\mathscr{P}^{f(ns)}_{pub(k_A)}(m)$ désigne une fonction qui prend en paramètre une séquence de *nonces*. La définition de la fonction $f()$ dépend du système de chiffrement utilisé (Paillier [Pai99], El-Gamal [Gam85]).

3.1.2 Canaux de communication

Dans leur analyse des protocoles de commerce électronique [ADM03], les auteurs ont considéré que les messages transitent uniquement via des canaux publiques. Or, comme discuté en chapitre 1 (section 1.3), outre des canaux publiques, les protocoles de vote peuvent également utiliser des canaux anonymes et/ou privés. Nous rappelons ci-dessous les caractéristiques de ces canaux et définissons les notations permettant de les distinguer :

- **Canaux publiques.** Il s'agit de canaux où l'information en transit est accessible par l'intrus. Nous utilisons les symboles $\triangleright$ et $\triangleleft$ pour désigner respectivement l'envoi et la réception via un canal publique.
- **Canaux anonymes.** Il s'agit de canaux garantissant l'anonymat de l'expéditeur. Cependant, l'information transmise est toujours accessible par l'intrus. Nous réservons les symboles $\trianglerighteq$ et $\trianglelefteq$ pour désigner respectivement l'envoi et la réception via un canal anonyme.
- **Canaux privés.** Il s'agit de canaux où l'information en transit est hors de portée de l'intrus. Nous employons les symboles $\blacktriangleright$ et $\blacktriangleleft$ pour dénoter respectivement l'envoi et la réception via un canal privé.

3.1.3 Actions du protocole

Concernant les actions du protocole, nous reprenons les notations proposées par [ADM03] (voir page 37). Pour mémoire ces notations permettent de distinguer deux types d'action :

- **Action interne.** Il s'agit d'une opération de traitement interne telle que la vérification d'une signature ou d'une preuve, ou bien la vérification de la légitimité des votants. Elle prend la forme suivante $i\text{-}act(A, p_1, \ldots, p_n)$ et dénote l'exécution de l'action act à l'étape i du protocole, par l'entité A, et sur l'ensemble des paramètres $p_1, \ldots, p_n$.
- **Action externe.** Il s'agit d'une opération de communication. Elle se présente selon la forme $i\text{-}A \triangleright B : m$ (respectivement $i\text{-}A \triangleleft B : m$) pour dénoter une opération d'envoi (respectivement de réception) durant laquelle une entité A envoie à (respectivement reçoit de) une entité B un message m à l'étape i du protocole.

À noter ici que le format des actions est allégé du paramètre α désignant la session par rapport à celui donné en section 2.3.1. En effet, nous n'avons pas besoin de considérer de sessions multiples dans le cadre de notre analyse des protocoles de vote électronique.

3.1.4 Représentation générale d'un protocole de vote

En réutilisant la représentation des protocoles cryptographiques adoptée par [ADM03], un protocole de vote électronique peut être représenté, au moyen des notations introduites précédemment, comme une séquence finie de déclarations ayant la forme suivante :

$$\left(\begin{array}{c} \ldots\ldots \\ i - act(A, p_1, \ldots, p_n) \\ \ldots\ldots \\ \\ i - A \triangleright B : m \\ i - B \triangleleft A : m \\ \\ \ldots\ldots \\ i - act(B, q_1, \ldots, q_m) \\ \ldots\ldots \end{array}\right)$$

Selon cette représentation, une étape du protocole (étape i) est représentée par une action externe (envoi et réception) optionnellement précédée et suivie par un ensemble d'actions internes. Un exemple d'une telle représentation est donné en Table 3.4. Plus précisément, nous donnons la représentation de la phase de préparation relative au protocole FOO.

Dans cet exemple, la préparation du message représentant le choix du votant V se traduit par une série d'actions internes : le votant procède d'abord à un choix v $(1 - choose(V, v))$. Il génère ensuite ses paramètres de chiffrement r $(1 - generate(V, r))$ et b $(1 - generate(V, b))$ au moyen desquels il construit une version chiffrée de son bulletin par appel successif aux primitives $commit$ et $blind$. Le bulletin chiffré $(\mathscr{B}(\{v\}_r, b))$ est par la suite signé au moyen de la clé privée k_V du votant V. Le message constituant la requête du votant est finalement formée par la paire $(V, \|\mathscr{B}(v_r, b)\|_{k_V})$ consistant en l'identité du votant V accompagnée de son bulletin signé.

TABLE 3.4 – Spécification de la phase de préparation du protocole FOO

$1\text{-}choose(V, v)$
$1\text{-}generate(V, r)$
$1\text{-}generate(V, b)$
$1\text{-}commit(V, v, r)$
$1\text{-}blind(V, \{v\}_r, b)$
$1\text{-}sign(V, \mathscr{B}(\{v\}_r, b), k_V)$
$1\text{-}concat(V, V, \|\mathscr{B}(v_r, b)\|_{k_V})$
$1\text{-}V \rhd A : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}$
$1\text{-}A \lhd V : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}$
$1\text{-}decompose(A, V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V})$
$1\text{-}check\_legitimacy(A, V)$
$1\text{-}check\_registration(A, V)$
$1\text{-}check\_sign(A, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}, pub(k_V))$

L'ensemble de ces actions internes, opérées par le votant, sont suivies par les actions externes identifiées par $(1 - V \rhd A : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V})$ et $(1 - A \lhd V : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V})$, et représentant respectivement l'envoi du message ainsi préparé par le votant et sa réception par l'administrateur A.

Ce dernier, procède à son tour à certaines vérifications se traduisant, là encore, par une série d'actions internes. Ces opérations consistent en la vérification de la légitimité du votant V $(1 - check\_legitimacy(A, V))$, à la vérification de son enregistrement $(1-check\_registration(A, V))$, et à la vérification de la validité de sa signature $(1 - check\_sign(A, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}, pub(k_V)))$.

3.2 Modèle de l'intrus

Notre modélisation des protocoles de vote tient compte de la présence d'un intrus capable d'agir activement dans le protocole par l'insertion arbitraire de messages constructibles à partir de sa connaissance initiale, et des messages interceptés par écoute du réseau. Comme nous l'avons signalé précédemment (voir section 2.3.1), le modèle de l'intrus communément utilisé est celui de Dolev-Yao, dans lequel l'intrus dispose d'un contrôle total des canaux publiques. Bien que ce modèle ait contribué à l'analyse de plusieurs protocoles cryptographiques, certains ajustements sont nécessaires afin d'adapter ce modèle au contexte du vote électronique impliquant des primitives cryptographiques plus avancées et faisant usage de canaux de communication plus élaborés. Dans ce qui suit, nous exposons en premier lieu les hypothèses relatives au pouvoir de l'intrus face aux divers canaux de communication. Ensuite, nous étendons les règles de déduction conformément à l'algèbre de messages définie en Table 3.3.

Dans notre modélisation, nous supposons que l'intrus dispose du même pouvoir sur les canaux publiques et anonymes, à savoir que tout message émis sur un canal publique ou anonyme enrichit la connaissance de l'intrus [1]. En revanche, les canaux privés sont par définition sûrs, et de ce fait nous les considérons hors de portée de l'intrus.

1. La différence entre canaux publiques et canaux anonymes sera principalement exploitée lors de la spécification des propriétés de sécurité (voir chapitre 4), un canal anonyme empêchant l'intrus de déduire l'identité de l'émetteur.

La Table 3.5 décrit la visibilité des messages par l'intrus en fonction des actions du protocole. Intuitivement, l'ensemble des messages observables par l'intrus I ne concerne que les actions externes ($obs(i-act(X,m))_{\Downarrow_I} = \emptyset$). Plus précisément, cet ensemble englobe les messages transmis via des canaux publiques ($obs(i-X \rhd Y:m)_{\Downarrow_I} = \{m\}$), anonymes ($obs(i-X \trianglerighteq Y:m)_{\Downarrow_I} = \{m\}$), et privés lorsque ces derniers sont adressés à l'intrus même ($obs(i-X \blacktriangleright I:m)_{\Downarrow_I} = \{m\}$).

TABLE 3.5 – Messages observables par l'intrus I en fonction des actions du protocole

$obs(i-act(X,m)_{\Downarrow_I}$	$=$	$\emptyset$	
$obs(i-X \rhd Y:m)_{\Downarrow_I}$	$=$	$\{m\}$	
$obs(i-X \lhd Y:m)_{\Downarrow_I}$	$=$	$\emptyset$	
$obs(i-X \trianglerighteq Y:m)_{\Downarrow_I}$	$=$	$\{m\}$	
$obs(i-X \trianglelefteq Y:m)_{\Downarrow_I}$	$=$	$\emptyset$	
$obs(i-X \blacktriangleright Y:m)_{\Downarrow_I}$	$=$	$\{m\}$	si $Y=I$
		$\emptyset$	sinon
$obs(i-X \blacktriangleleft Y:m)_{\Downarrow_I}$	$=$	$\emptyset$	

Maintenant que nous avons précisé le pouvoir de l'intrus par rapport aux différents canaux de communication, nous allons présenté les règles de déduction. Pour rappel, ces règles sont présentées ici dans un style hérité de [HM07], où la notation $M \vdash_K m$ signifie que l'intrus est capable de dériver le message m à partir de l'ensemble des message M, de sa connaissance initiale K, et de ses règles $\vdash$.

Concaténation. Tout d'abord, notre modèle de l'intrus intègre les règles de déduction $compose$ et $decompose$ (voir Table 2.7, page 38) associées respectivement à la composition et décomposition des messages.

Chiffrement symétrique. De même, notre modèle de l'intrus inclut les règles $symenc$ et $symdec$ (voir Table 2.7, page 38) relatives au chiffrement symétrique.

Chiffrement asymétrique (probabiliste). La Table 3.6 expose les règles relatives au chiffrement asymétrique. Tout d'abord, la règle $asymenc$ signifie que l'intrus peut construire le message $\{\{m\}\}_{pub(k_A)}$ correspondant au chiffrement asymétrique du message m au moyen de la clé publique $pub(k_A)$, que s'il a en sa connaissance les éléments m et $puk(k_A)$. À noter que la règle $asymenc$ est similaire à la règle $symenc$ à l'exception du chiffrement qui est assuré au moyen d'une clé publique $pub(k_A)$ au lieu d'une clé partagée k_{AB}. La règle $asymdec$ signifie que l'intrus peut retrouver le message m à partir du chiffré $\{\{m\}\}_{pub(k_A)}$ que s'il dispose de la clé privée associée k_A. Là encore, la règle $asymdec$ est obtenue à partir de la règle $symdec$ en considérant que l'inverse de la clé chiffrement ($pub(k_A)$) n'est plus l'identité mais la clé privée associée k_A.

TABLE 3.6 – Règles de déduction - Chiffrement asymétrique (probabiliste)

(***asymenc***)	$\frac{M \vdash_K m \quad M \vdash_K pub(k_A)}{M \vdash_K \{\{m\}\}_{pub(k_A)}}$
(***asymdec***)	$\frac{M \vdash_K \{\{m\}\}_{pub(k_A)} \quad M \vdash_K k_A}{M \vdash_K m}$
(***pasymenc***)	$\frac{M \vdash_K m \quad M \vdash_K pub(k_A) \quad M \vdash_K n}{M \vdash_K \{\{m\}\}^n_{pub(k_A)}}$
(***pasymdec***)	$\frac{M \vdash_K \{\{m\}\}^n_{pub(k_A)} \quad M \vdash_K k_A}{M \vdash_K m}$
(***pasymdec*$_2$**)	$\frac{M \vdash_K \{\{m\}\}^n_{pub(k_A)} \quad M \vdash_K n}{M \vdash_K m}$

Ensuite, le modèle est aisément étendu avec les règles relatives au chiffrement asymétrique probabiliste. La règle $pasymenc$ signifie que la construction du message $\{\{m\}\}^n_{pub(k_A)}$ par l'intrus ne peut se faire que si ce dernier dispose des éléments m, $pub(k_A)$, et n. Ce dernier élément représente le paramètre aléatoire requis dans le chiffrement probabiliste.

Comme pour le cas de la règle $asymdec$, la règle $pasymdec$ signifie que le message en clair m peut être retrouvé à partir du chiffré $\{\{m\}\}^n_{pub(k_A)}$ que si l'intrus dispose de la clé privée k_A.

Finalement, la dernière règle ($pasymdec_2$) est optionnelle ici. Elle permet de retrouver le texte en clair à partir du chiffré et du paramètre aléatoire (*nonce n*) utilisé dans le chiffrement. Cette propriété est vraie pour certains cryptosystèmes tels que El-Gamal.

Signature numérique (en aveugle). Les règles de déduction associées sont données en Table 3.7.

TABLE 3.7 – Règles de déduction - Signature numérique (en aveugle)

(***sign***)	$\frac{M \vdash_K m \quad M \vdash_K k_A}{M \vdash_K \|m\|_{k_A}}$
(***unsign***)	$\frac{M \vdash_K \|m\|_{k_A}}{M \vdash_K m}$
(***blind***)	$\frac{M \vdash_K m \quad M \vdash_K n}{M \vdash_K \mathscr{B}(m,n)}$
(***unblind***)	$\frac{M \vdash_K \mathscr{B}(m,n) \quad M \vdash_K n}{M \vdash_K m}$
(***unblindsig***)	$\frac{M \vdash_K \|\mathscr{B}(m,n)\|_{pub(k_A)} \quad M \vdash_K n}{M \vdash_K \|m\|_{pub(k_A)}}$

La règle $sign$ est relative à la signature numérique classique. La règle $unsign$ permet de retrouver le texte m à partir de sa signature étant donné que nous supposons que la signature d'un message $\|m\|_{k_A}$ englobe en elle-même le message en clair m.

L'aveuglement est assimilé au chiffrement symétrique et de ce fait les règles $blind$ et $unblind$ se dérivent directement à partir, respectivement, des règles $symenc$ et $symdec$ de la Table 2.7.

Finalement, la règle $unblindsig$ permet d'extraire le message signé à partir de la signature produite sur le message aveuglé.

Preuve à divulgation nulle de connaissance. La Table 3.8 présente les règles de déduction associées aux preuves à divulgation nulle de connaissance.

TABLE 3.8 – Règles de déduction - Preuve à divulgation nulle de connaissance

$(\boldsymbol{zerok_1})$	$\frac{M \vdash_K m_1 \quad M \vdash_K m_2}{M \vdash_K \mathcal{Z}(m_1 \mid m_2)}$
$(\boldsymbol{zerok_2})$	$\frac{M \vdash_K \mathcal{Z}(m_1 \mid m_2)}{M \vdash_K m_2}$

Comme énoncé plus haut, une preuve non interactive à divulgation nulle de connaissance est abstraite par $\mathcal{Z}(m_1 \mid m_2)$ où m_1 dénote la partie secrète de la preuve et m_2 la partie publique permettant sa vérification. La règle $zerok_1$ signifie que l'intrus peut construire la preuve que s'il a en sa possession les messages m_1 et m_2. La règle, $zerok_2$ signifie que l'intrus ne peut déduire que la partie publique de la preuve (message m_2) à partir du message $\mathcal{Z}(m_1 \mid m_2)$.

Chiffrement Homomorphe. Les règles données en Table 3.9 formalisent les propriétés relatives au chiffrement homomorphe.

TABLE 3.9 – Règles de déduction - Chiffrement homomorphe

$(\boldsymbol{aggbase})$	$\frac{M \vdash_K \{\{m\}\}^n_{pub(k_A)}}{M \vdash_K \mathscr{P}^{f(n)}_{pub(k_A)}(m)}$
$(\boldsymbol{aggregate})$	$\frac{M \vdash_K \mathscr{P}^{f(n_s)}_{pub(k_A)}(m_1) \quad M \vdash_K \{\{m_2\}\}^n_{pub(k_A)}}{M \vdash_K \mathscr{P}^{f(n_s,n)}_{pub(k_A)}(m_1, m_2)}$
$(\boldsymbol{homoeq})$	$\frac{M \vdash_K \mathscr{P}^{f(n_s)}_{pub(k_A)}(m)}{M \vdash_K \{\{\mathscr{S}(m)\}\}^{f(n_s)}_{pub(k_A)}}$
$(\boldsymbol{talbase})$	$\frac{M \vdash_K m}{M \vdash_K \mathscr{S}(m)}$
$(\boldsymbol{tally})$	$\frac{M \vdash_K \mathscr{S}(m_1) \quad M \vdash_K m_2}{M \vdash_K \mathscr{S}(m_1, m_2)}$
$(\boldsymbol{fnonce_1})$	$\frac{M \vdash_K n}{M \vdash_K f(n)}$
$(\boldsymbol{fnonce_2})$	$\frac{M \vdash_K f(n_s) \quad M \vdash_K n}{M \vdash_K f(n_s, n)}$

Tout d'abord, la règle $aggbase$ permet une simple translation de format. La règle suivante ($aggregate$) permet de rajouter un message chiffré à un ensemble de messages agrégés.

La règle $homoeq$ modélise les équations (1.1) et (1.2) de la section 1.2.4. Plus précisément, elle permet de retrouver le chiffrement de la somme (ou produit) des messages en clair à partir de l'agrégation (produit) des messages chiffrés.

Les règles $talbase$ et $tally$ permettent de construire des messages de la forme $\mathscr{S}(m_1, m_2)$ (somme des votes).

Finalement, les règles $fnonce_1$ et $fnonce_2$ modélisent certaines opérations sur les *nonces*.

L'exemple 3.2.1 illustre une application de certaines des règles de déduction relatives au chiffrement homomorphe.

Exemple 3.2.1

Soient les ensembles K et M suivants :

$$\begin{aligned} K &= \emptyset \\ M &= \{\ \{\{v_1\}\}^{n_1}_{pub(k_A)}, \{\{v_2\}\}^{n_2}_{pub(k_A)}\ \} \end{aligned}$$

L'ensemble K dénote la connaissance initiale de l'intrus. Elle est supposée nulle dans cet exemple. L'ensemble M représente quant à lui la connaissance apprise par écoute du trafic, et consiste en deux votes v_1 et v_2 chiffrés au moyen de clé publique $pub(k_A)$. Nous supposons ici que le schéma de chiffrement est homomorphe.

À partir des ensembles M et K, et des règles de déduction $\vdash$ de la Table 3.9, l'intrus est maintenant en mesure de déduire le message $\{\{\mathscr{S}(v_1, v_2)\}\}^{f(n_1,n_2)}_{pub(k_A)}$ représentant le chiffrement de la somme (ou produit) des votes v_1 et v_2 au moyen de clé publique $pub(k_A)$.

$$(\textbf{aggbase})\quad \frac{M \vdash_K \{\{v_1\}\}^{n_1}_{pub(k_A)}}{M \vdash_K \mathscr{P}^{f(n_1)}_{pub(k_A)}(v_1)}$$

$$(\textbf{aggregate})\quad \frac{M \vdash_K \mathscr{P}^{f(n_1)}_{pub(k_A)}(v_1) \qquad M \vdash_K \{\{v_2\}\}^{n_2}_{pub(k_A)}}{M \vdash_K \mathscr{P}^{f(n_1,n_2)}_{pub(k_A)}(v_1, v_2)}$$

$$(\textbf{homoeq})\quad \frac{M \vdash_K \mathscr{P}^{f(n_1,n_2)}_{pub(k_A)}(v_1, v_2)}{M \vdash_K \{\{\mathscr{S}(v_1, v_2)\}\}^{f(n_1,n_2)}_{pub(k_A)}}$$

Il est à noter que l'écriture d'un jeu complet de règles de déduction est une tâche ardue étant donné que l'on ne connait pas l'ensemble des propriétés associées aux primitives cryptographiques. Il s'ensuit que les propriétés de sécurité que nous spécifierons dans le prochain chapitre ne peuvent être universelles et que le protocole analysé peut être attaqué face à un intrus disposant de capacités supplémentaires. À titre d'exemple, les vulnérabilités liées au protocole d'authentification Bull [BO97] ne sont mises en évidence que si le pouvoir de l'intrus est incrémenté de certaines propriétés algébriques relatives au XOR [RS98].

3.3 Modélisation des protocoles de vote

A l'instar de [ADM03], nous adoptons un modèle de protocole à base de traces où une trace représente une exécution dynamique du protocole. Les motivations pour un tel choix de modèle sont multiples : tout d'abord, les traces représentent le modèle d'exécution le plus simple et le plus naturel considéré dans la littérature. Ensuite, une faille dans un protocole cryptographique est souvent mise en évidence par une trace représentant la violation d'une propriété de sécurité.

Dans ce qui suit, nous décrivons l'approche adoptée pour la modélisation des protocoles de vote. Ensuite, nous illustrons cette approche en prenant pour exemple le protocole FOO.

3.3.1 Approche pour la modélisation des protocoles de vote

Pour rappel, une trace représente une exécution possible du protocole et est définie comme une séquence finie d'actions internes et d'actions externes (voir page 37). Plus précisément, une trace est définie comme suit :

$$t \quad ::= \quad \epsilon \mid t.a$$

La trace vide est représentée par ϵ. Elle est tout simplement éliminée lorsqu'elle est incluse dans une trace non vide. L'opérateur '.' est réservé pour la concaténation d'événements prenant forme selon la grammaire donnée ci-après, où a n'est autre qu'une action du protocole dont le format a été identifié dans la section 3.1.3 :

$$a \quad ::= \quad i - act(A, p_1, \ldots, p_n) \mid i - A \rhd B : m \mid i - A \lhd B : m$$

Afin de faciliter la modélisation des protocoles, nous introduisons les notations auxiliaires suivantes sur les traces :

$$\begin{cases} \bar{t} & \text{ensemble des actions de la trace } t \\ msg(t) & \text{ensemble des messages de la trace } t \end{cases}$$

En suivant la même approche que [ADM03], nous modélisons alors un protocole P par son ensemble de traces valides $\mathscr{P}$. Une trace est dite valide si les entités honnêtes respectent le protocole, et si les messages transmis par l'intrus peuvent être dérivée de sa connaissance cumulée. La connaissance de l'intrus comprend sa connaissance initiale, les messages qui lui sont directement transmis si ce dernier est impliqué dans le protocole, les messages interceptés par écoute des canaux publiques et anonymes, et finalement les messages déduits par application des règles de déduction $\vdash$ données en section 3.2.

L'ensemble des traces valides $\mathscr{P}$ peut être construit en utilisant la même approche que celle adoptée par Paulson dans [Pau98]. Dans ce cas, les traces sont construites dynamiquement par l'intermédiaire d'un ensemble de règles d'inférence ($\frac{\text{prémisse}}{\text{conclusion}}$) s'interprétant de la manière suivante : si les conditions de la prémisse sont satisfaites, alors la conclusion peut être exécutée. L'idée derrière ces règles est de définir pour chaque étape du protocole comment la trace évolue. Cette approche est illustrée dans la section qui suit par l'exemple du protocole FOO.

3.3.2 Modélisation du protocole FOO

Avant de passer à la modélisation du protocole FOO, nous rappelons en Table 3.10, les principales étapes du protocole. Les lignes qui séparent certaines de ces étapes correspondent aux différentes *deadlines* du protocole.

TABLE 3.10 – Étapes du protocole FOO

Étape	Émetteur	Récépteur	Message	Canal
(1) Préparation	V	A	$V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}$	publique
(2) Administration	A	V	$\|\mathscr{B}(\{v\}_r, b)\|_{k_A}$	publique
(3) Vote	V	C	$\|\{v\}_r\|_{k_A}$	anonyme
(4) Notification	C	B	$\|\{v\}_r\|_{k_A}$	publique
(5) Ouverture des bulletins	V	C	r	anonyme
(6) Publication des résultats	C	B	$\|\{v\}_r\|_{k_A}, r, v$	publique

Comme déclaré précédemment, modéliser le protocole FOO revient à définir l'ensemble des règles d'inférence associées au protocole. Afin d'en faciliter la lecture, les termes constants dans les règles seront écrits en gras (collecteur $\boldsymbol{C}$, administrateur $\boldsymbol{A}$, sa clé privée $\boldsymbol{k_A}$, etc.). Les variables quant à elles, sont extraites à partir d'ensembles finis de valeurs. À titre d'exemple, V (votant) est une variable appartenant à l'ensemble des votants légitimes L_v, et v (vote) est une variable appartenant à l'ensemble des choix possibles L_c. Il en est de même pour les paramètres aléatoires r et b pris à partir de l'ensemble fini des *nonces* L_{nc}.

Initialisation. La règle R_{init} permet d'inclure à l'ensemble des traces valides $\mathscr{P}$, la trace vide ϵ.

Règle R_{init}

$$\frac{}{\epsilon \in \mathscr{P}}$$

Réception de messages. La règle R_{rcv} signifie qu'une entité ne peut recevoir un message que s'il lui a été préalablement transmis. Plus précisément, soient X et Y deux variables prises dans l'ensemble des entités du protocole en y incluant l'intrus I. La règle R_{rcv} s'interprète maintenant de la manière suivante : si une trace $t \in \mathscr{P}$ contient une action $(i - X \triangleright Y : m)$ représentant l'envoi d'un message m à une étape i du protocole (première condition de la prémisse), et si ce message n'a pas encore été délivré (seconde condition de la prémisse), alors la trace peut être étendue avec l'action externe $(i - Y \triangleleft X : m)$ représentant la réception du message m par Y de la part de X à l'étape i du protocole (conclusion de la règle). Il est à noter ici que les variables de la conclusion doivent coïncider avec ceux de la prémisse.

Règle R_{rcv}

$$\frac{t \in \mathscr{P} \qquad (i - X \triangleright Y : m) \in \bar{t} \qquad (i - Y \triangleleft X : m) \notin \bar{t}}{t.(i - Y \triangleleft X : m) \in \mathscr{P}}$$

L'exemple qui suit permet de mieux assimiler les spécificités de la règle :

Exemple 3.3.1

Soit t une trace valide propre à un protocole P :

$$t = (1 - A \rhd B : m_1).(1 - B \lhd A : m_1).(2 - B \rhd S : m_2)$$

La trace t décrit une séquence d'événements durant lesquels un message m_1 a d'abord été transmis par une entité A et reçu par une entité B à l'étape 1 du protocole. Ces deux actions sont ensuite suivies par une action durant laquelle l'entité B a transmis à l'entité S le message m_2 à la seconde étape du protocole.

Nous notons ici que la trace t satisfait les conditions de la prémisse de la règle R_{rcv}. En effet, la trace t contient l'action $(2 - B \rhd S : m_2)$ désignant l'envoi du message m_2 non encore délivré à son destinataire S. Par conséquent, la trace t peut évoluer et donner naissance à la trace valide t' obtenue en étendant la trace t par l'action externe $(2 - S \lhd B : m_2)$:

$$t' = (1 - A \rhd B : m_1).(1 - B \lhd A : m_1).(2 - B \rhd S : m_2).(2 - S \lhd B : m_2)$$

La communication dans FOO est assurée également par des canaux anonymes. Nous devons alors définir une règle R_{rcva} relative à la réception des messages émis via un canal anonyme. Cette règle est tout simplement obtenue en remplaçant dans la règle R_{rcv} les symboles $\rhd$ et $\lhd$ respectivement par $\trianglerighteq$ et $\trianglelefteq$.

Règle R_{rcva}

$$\frac{t \in \mathscr{P} \qquad (i - X \trianglerighteq Y : m) \in \bar{t} \qquad (i - Y \trianglelefteq X : m) \notin \bar{t}}{t.(i - Y \trianglelefteq X : m) \in \mathscr{P}}$$

Il est à noter que, telles que sont définies les règles R_{rcv} et R_{rcva}, l'action de réception peut ne pas suivre immédiatement l'action d'émission. L'action de réception peut même être absente de la trace. Ceci modélise implicitement la faculté de l'intrus à retarder/intercepter les messages envoyés via les canaux publiques ou anonymes. Dans le cas d'un canal privé, la communication entière (envoi et réception) doit être atomique (l'action d'envoi est immédiatement suivie de la réception) étant donné que ce canal garantit par définition la sécurité de la transmission.

Génération de *nonces*. La règle R_{fresh} permet de délivrer à chaque entité du protocole sa connaissance initiale de messages frais (*nonces*). Ceci est assuré en considérant une étape virtuelle du protocole (étape 0) où chaque agent envoie à lui-même sa connaissance fraiche. Afin de garantir la confidentialité de l'échange, nous considérons que ces messages sont transmis via un canal privé. La règle R_{fresh} s'interprète de la manière suivante : si le message r n'est pas encore délivré, alors la trace peut être étendue avec les actions externes durant lesquels l'entité X envoie et reçoit d'elle-même le message r. Il est à noter que la condition de la prémisse $(r \notin msg(t))$ permet d'éviter que deux entités partagent un même *nonce*.

Règle R_{fresh}

$$\frac{t \in \mathscr{P} \qquad r \notin msg(t)}{t.(\mathbf{0} - X \blacktriangleright X : r).(\mathbf{0} - X \blacktriangleleft X : r) \in \mathscr{P}}$$

Étape de préparation. La règle $R_{request}$ est relative à la première étape du protocole. Elle s'interprète de la manière suivante : si le votant a généré au préalable ses paramètres de chiffrement (*nonces*), si le votant n'a pas encore émis de requête pour une signature en aveugle, et si la *deadline* marquant la fin de la phase d'administration n'est pas encore atteinte (l'action interne $(2-deadline(A,T_1))$ n'est pas une composante de la trace t), alors le votant peut procéder encore à l'enregistrement : la trace peut être étendue avec l'action représentant la demande de signature en aveugle. À noter que la trace est également étendue avec l'action interne $(1-request(V))$ empêchant le votant d'émettre plusieurs requêtes.

Règle $R_{request}$

$$\frac{\begin{array}{c} t \in \mathscr{P} \quad (\mathbf{0}-V \blacktriangleleft V : r) \in \bar{t} \quad (\mathbf{0}-V \blacktriangleleft V : b) \in \bar{t} \\ (\mathbf{1}-request(V)) \notin \bar{t} \quad (\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1})) \notin \bar{t} \end{array}}{t.(\mathbf{1}-request(V)).(\mathbf{1}-V \rhd \boldsymbol{A} : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}) \in \mathscr{P}} \quad [r \neq b]$$

Étape d'administration. La règle $R_{authorize}$ modélise le rôle de l'administrateur A dans le protocole. Plus précisément, si l'autorité A reçoit une demande de signature en aveugle de la part d'un votant légitime V avant la fin de la phase d'enregistrement, et si la signature associée au votant est valide (condition de bord V_{sign} valide), et que ce dernier ne s'est pas déjà enregistré, alors la trace peut être étendue avec l'action externe $(2-A \rhd V : \|\mathscr{B}(\{v\}_r, b)\|_{k_A})$ représentant l'envoi de la signature apposée par A au votant V. La trace est également étendue avec l'action interne $(2-register(A,V))$ représentant l'enregistrement du votant V.

Règle $R_{authorize}$

$$\frac{\begin{array}{c} t \in \mathscr{P} \quad (\mathbf{1}-\boldsymbol{A} \lhd V : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}) \in \bar{t} \\ (\mathbf{2}-register(\boldsymbol{A},V)) \notin \bar{t} \quad (\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1})) \notin \bar{t} \end{array}}{t.(\mathbf{2}-register(\boldsymbol{A},V)).(\mathbf{2}-\boldsymbol{A} \rhd V : \|\mathscr{B}(\{v\}_r, b)\|_{\boldsymbol{k_A}}) \in \mathscr{P}} \quad V_{sign}$$

Étape de vote. La phase relative au vote est spécifiée par les règles R_{cast} et $R_{collect}$. La première règle considère la procédure de vote du côté du votant (V envoie son vote), tandis que la seconde considère le processus de vote du côté du collecteur (C collecte les votes).

La règle R_{cast} permet d'étendre la trace avec l'envoi du vote chiffré de la part du votant V au collecteur C, si les préconditions qui régissent la prémisse de la règle sont satisfaites. Les deux premières conditions impliquent que le votant V ait déjà opéré à la phase d'enregistrement, en ayant reçu de la part de l'administrateur A une signature valide (condition de bord A_{sign} valide) correspondant à la requête émise à l'étape 1 du protocole. Les deux préconditions qui suivent sont spécifiques au respect des *deadlines*. Finalement, la dernière condition exige que le votant n'a pas encore émis son vote.

Règle R_{cast}

$$\frac{\begin{array}{c} t \in \mathscr{P} \quad (\mathbf{1}-V \rhd \boldsymbol{A} : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V}) \in \bar{t} \\ (\mathbf{2}-V \lhd \boldsymbol{A} : \|\mathscr{B}(\{v\}_r, b)\|_{\boldsymbol{k_A}}) \in \bar{t} \quad (\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1})) \in \bar{t} \\ (\mathbf{3}-deadline(\boldsymbol{C},\boldsymbol{T_2})) \notin \bar{t} \quad (\mathbf{3}-cast(V)) \notin \bar{t} \end{array}}{t.(\mathbf{3}-cast(V)).(\mathbf{3}-V \trianglerighteq \boldsymbol{C} : \|\{v\}_r\|_{\boldsymbol{k_A}}) \in \mathscr{P}} \quad A_{sign}$$

La règle $R_{collect}$ signifie que la trace peut être étendue avec l'action interne représentant l'enregistrement par C du vote reçu, si ce dernier dispose d'une signature valide, n'a pas encore été enregistré, et est réceptionné dans la plage de temps séparant les *deadlines* T_1 et T_2.

Règle $R_{collect}$

$$\frac{\begin{array}{c} t \in \mathscr{P} \quad (\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1})) \in \bar{t} \quad (\mathbf{3}-C \trianglelefteq X : \|\{v\}_r\|_{\boldsymbol{k_A}}) \in \bar{t} \\ (\mathbf{3}-register(\boldsymbol{C},\|\{v\}_r\|_{\boldsymbol{k_A}})) \notin \bar{t} \quad (\mathbf{3}-deadline(\boldsymbol{C},\boldsymbol{T_2})) \notin \bar{t} \end{array}}{t.(\mathbf{3}-register(\boldsymbol{C},\|\{v\}_r\|_{\boldsymbol{k_A}})) \in \mathscr{P}} \; A_{sign}$$

Étape de notification. La règle R_{notify} est relative à la quatrième étape du protocole. Elle s'interprète de la manière suivante : si la *deadline* marquant la fin de la phase de vote est atteinte, et si le collecteur n'a pas déjà opéré à la publication des votes reçus, alors la trace peut être étendue avec les actions correspondant à la publication auprès de l'entité B de tous les votes enregistrés ($\|\{v_i\}_{r_i}\|_{k_A} (i = 1 \ldots n)$). La condition de bord permet d'assurer que le raisonnement soit portée dans la prémisse par rapport à des éléments $m_i = \|\{v_i\}_{r_i}\|_{k_A}$ distincts.

Règle R_{notify}

$$\frac{\begin{array}{c} t \in \mathscr{P} \\ (\mathbf{3}-register(\boldsymbol{C},\|\{v_1\}_{r_1}\|_{\boldsymbol{k_A}})) \in \bar{t} \\ \vdots \\ (\mathbf{3}-register(\boldsymbol{C},\|\{v_n\}_{r_n}\|_{\boldsymbol{k_A}})) \in \bar{t} \\ (\mathbf{3}-deadline(\boldsymbol{C},\boldsymbol{T_2})) \in \bar{t} \quad (\mathbf{4}-notify(\boldsymbol{C})) \notin \bar{t} \end{array}}{t.\begin{pmatrix} (\mathbf{4}-notify(\boldsymbol{C})). \\ (\mathbf{4}-\boldsymbol{C} \triangleright \boldsymbol{B} : \|\{v_1\}_{r_1}\|_{\boldsymbol{k_A}}). \\ \vdots \\ (\mathbf{4}-\boldsymbol{C} \triangleright \boldsymbol{B} : \|\{v_n\}_{r_n}\|_{\boldsymbol{k_A}}) \end{pmatrix} \in \mathscr{P}} \left[\frac{\begin{array}{c} m_i \neq m_j \\ i,j \in \{1,\ldots,n\} \quad i \neq j \end{array}}{m_i = \|\{v_i\}_{r_i}\|_{\boldsymbol{k_A}}} \right]$$

Étape d'ouverture des bulletins. De manière similaire à la phase de vote, l'étape relative à l'ouverture des bulletins est spécifiée par deux règles : R_{key} et R_{open}. Ces règles considèrent la phase d'ouverture des bulletins respectivement du côté du votant et du côté du collecteur.

La règle R_{key} signifie que le votant ne peut émettre la clé permettant l'ouverture de son bulletin que si ce dernier a été émis à la phase de vote, et publié ensuite auprès de l'entité B. La règle est également contrainte par les *deadlines* signifiant que la clé ne peut être transmise que si la phase de vote a été clôturée, et que la phase d'ouverture des bulletins ne s'est pas encore terminée. À noter que la trace est également étendue, en conclusion de la règle, par l'action $(5-key(V))$ empêchant le votant de transmettre à plusieurs reprises sa clé.

Règle R_{key}

$$\frac{\begin{array}{c} t \in \mathscr{P} \quad (\mathbf{3}-V \trianglerighteq \boldsymbol{C} : \|\{v\}_r\|_{\boldsymbol{k_A}}) \in \bar{t} \\ (\mathbf{3}-deadline(\boldsymbol{C},\boldsymbol{T_2})) \in \bar{t} \quad (\mathbf{4}-\boldsymbol{B} \triangleleft \boldsymbol{C} : \|\{v\}_r\|_{\boldsymbol{k_A}}) \in \bar{t} \\ (\mathbf{5}-key(V)) \notin \bar{t} \quad (\mathbf{5}-deadline(\boldsymbol{C},\boldsymbol{T_3})) \notin \bar{t} \end{array}}{t.(\mathbf{5}-key(V)).(\mathbf{5}-V \trianglerighteq \boldsymbol{C} : r) \in \mathscr{P}}$$

Dans la conclusion de la règle R_{open}, l'action interne $(5-register(C,\|\{v\}_r\|_{k_A},r,v))$ représente la sauvegarde par le collecteur C du triplet $(\|\{v\}_r\|_{k_A},r,v)$ dont les éléments identifient respectivement le bulletin chiffré, la clé de déchiffrement, et le vote. Cette action peut être étendue à la trace que si cette dernière contient les actions $(3-register(C,\|\{v\}_r\|_{k_A}))$ et $(5-C\trianglelefteq X:r)$ représentant respectivement l'enregistrement du bulletin $\|\{v\}_r\|_{k_A}$ et la réception de la clé permettant son déchiffrement. À noter que, là encore, l'extension de la trace dans la règle R_{open} est contrainte aussi par les *deadlines* T_2 et T_3.

Règle R_{open}

$$\frac{\begin{array}{c} t\in\mathscr{P} \quad (\mathbf{3}-deadline(\boldsymbol{A},\boldsymbol{T_2}))\in\bar{t} \\ (\mathbf{3}-register(\boldsymbol{C},\|\{v\}_r\|_{\boldsymbol{k_A}}))\in\bar{t} \quad (\mathbf{5}-\boldsymbol{C}\trianglelefteq X:r)\in\bar{t} \\ (\mathbf{5}-register(\boldsymbol{C},\|\{v\}_r\|_{\boldsymbol{k_A}},r,v))\notin\bar{t} \quad (\mathbf{5}-deadline(\boldsymbol{C},\boldsymbol{T_3}))\notin\bar{t}\end{array}}{t.(\mathbf{5}-register(\boldsymbol{C},\|\{v\}_r\|_{\boldsymbol{k_A}},r,v))\in\mathscr{P}}$$

Étape de publication des résultats. La règle $R_{publish}$ est associée à la dernière étape du protocole, et spécifie la phase de la publication du résultat final de l'élection. À noter que comme dans la règle R_{notify}, la condition de bord permet d'assurer que le raisonnement soit portée dans la prémisse par rapport à des éléments $m_i=\|\{v_i\}_{r_i}\|_{k_A},r_i,v_i$ $(i=1\dots n)$ distincts.

Règle $R_{publish}$

$$\frac{\begin{array}{c} t\in\mathscr{P} \\ (\mathbf{5}-register(\boldsymbol{C},\|\{v_1\}_{r_1}\|_{\boldsymbol{k_A}},r_1,v_1))\in\bar{t} \\ \vdots \\ (\mathbf{5}-register(\boldsymbol{C},\|\{v_n\}_{r_n}\|_{\boldsymbol{k_A}},r_n,v_n))\in\bar{t} \\ (\mathbf{5}-deadline(\boldsymbol{C},\boldsymbol{T_3}))\in\bar{t} \quad (\mathbf{6}-publish(\boldsymbol{C}))\notin\bar{t}\end{array}}{t.\left(\begin{array}{c}(\mathbf{6}-publish(\boldsymbol{C})). \\ (\mathbf{6}-\boldsymbol{C}\rhd\boldsymbol{B}:\|\{v_1\}_{r_1}\|_{\boldsymbol{k_A}},r_1,v_1). \\ \vdots \\ (\mathbf{6}-\boldsymbol{C}\rhd\boldsymbol{B}:\|\{v_n\}_{r_n}\|_{\boldsymbol{k_A}},r_n,v_n)\end{array}\right)\in\mathscr{P}} \quad \left[\frac{\begin{array}{c}m_i\neq m_j \\ i,j\in\{1,\dots,n\} \quad i\neq j\end{array}}{m_i=\|\{v_i\}_{r_i}\|_{\boldsymbol{k_A}},r_i,v_i}\right]$$

Deadlines. On distingue trois *deadlines* dans le protocole FOO. La première marque la fin de la phase d'enregistrement. La seconde délimite la fin de la phase de vote. La troisième *deadline* marque finalement la fin de l'étape d'ouverture des bulletins. Ces *deadlines* sont modélisées par les règles $R_{deadline_i}(i=1,2,3)$. Ces règles signifient que si on atteint la limite en temps dénotée par T_i, alors la trace peut être étendue avec l'action interne $(stp_i-deadline(X_i,T_i))$ où stp_i représente l'étape à laquelle la *deadline* a lieu, et X_i l'entité responsable de l'événement.

Règle $R_{deadline_1}$

$$\frac{t\in\mathscr{P} \quad (\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1}))\notin\bar{t} \quad \bar{R}_{authorize}(t)}{t.(\mathbf{2}-deadline(\boldsymbol{A},\boldsymbol{T_1}))\in\mathscr{P}}$$

Règle $R_{deadline_2}$

$$\frac{t \in \mathscr{P} \quad (\mathbf{2} - deadline(\boldsymbol{A}, \boldsymbol{T_1})) \in \bar{t} \quad (\mathbf{3} - deadline(\boldsymbol{C}, \boldsymbol{T_2})) \notin \bar{t} \quad \bar{R}_{collect}(t)}{t.(\mathbf{3} - deadline(\boldsymbol{C}, \boldsymbol{T_2})) \in \mathscr{P}}$$

Règle $R_{deadline_3}$

$$\frac{t \in \mathscr{P} \quad (\mathbf{3} - deadline(\boldsymbol{C}, \boldsymbol{T_2})) \in \bar{t} \quad (\mathbf{5} - deadline(\boldsymbol{C}, \boldsymbol{T_3})) \notin \bar{t} \quad \bar{R}_{open}(t)}{t.(\mathbf{5} - deadline(\boldsymbol{C}, \boldsymbol{T_3})) \in \mathscr{P}}$$

Il est à noter que ces *deadlines* sont sujettes à des conditions supplémentaires. Pa exemple, dans la règle $R_{deadline_1}$, avant d'étendre la trace avec l'action interne $(2 - deadline(A, T_1))$, nous devons nous assurer au préalable que la règle $R_{authorize}$ ne peut être exécutée. Ceci dénote le fait que toutes les requêtes légitimes reçues de demande de signature doivent être acquittées avant la fin de la phase d'enregistrement. Afin de matérialiser ce type de contraintes, nous avons introduit la notation auxiliaire $\bar{R}(t)$ pour signifier que les préconditions de la prémisse de la règle R ne sont pas satisfaites par la trace t.

Cette contrainte se retrouvent également dans la règle $R_{deadline_2}$, où la condition $\bar{R}_{collect}(t)$ empêche la seconde *deadline* d'avoir lieu si tous les bulletins chiffrés n'ont pas encore été sauvegardés. Il en est de même pour la règle $\bar{R}_{collect}(t)$, où $\bar{R}_{open}(t)$ empêche cette fois-ci la troisième *deadline* d'avoir lieu si tous les bulletins chiffrés n'ont pas encore été traités.

Injection de messages. Notre modélisation des protocoles de vote tient compte d'un intrus I capable d'agir activement dans le protocole. Cet intrus, unique mais puissant, peut jouer le rôle d'une entité externe au protocole, d'un votant légitime ($I \in L_v$), ou bien d'un administrateur corrompu ($I = A$). L'identité endossée par l'intrus est fixée à l'avance, et indépendamment de son statut il suit un même dessein : injecter arbitrairement des messages dans le but de provoquer certains agents à réagir, conformément aux règles du protocole, et ainsi les amener à divulguer certaines informations secrètes (clés, *nonces*), ou bien à accepter certains de ses messages (vote). L'intrus que nous considérons est donc un intrus actif, et cette faculté est modélisée par la règle R_{inject} donnée ci-après :

Règle R_{inject}

$$\frac{t \in \mathscr{P} \qquad obs(t)_{\Downarrow_I} \vdash_{K_I} m}{t.(i - X \lhd_I^i Y : m) \in \mathscr{P}}$$

La règle R_{inject} signifie que l'intrus I est capable d'injecter tout message dérivable à partir de sa connaissance cumulée. Celle-ci englobe sa connaissance initiale K_I, les messages transmis directement à l'intrus (si ce dernier est impliqué dans le protocole), les messages interceptées (par écoute des canaux publiques et anonymes), et finalement les messages déduits par l'intrus par application des règles de déduction données en section 3.2.

La détention du message m par l'intrus I est donc donnée par $obs(t)_{\Downarrow_I} \vdash_{K_I} m$, où $obs(t)_{\Downarrow_I}$ désigne les messages observables par l'intrus au niveau de la trace t. Cet ensemble est constructible comme suit :

$$\begin{aligned} obs(\epsilon)_{\Downarrow_I} &= \emptyset \\ obs(t.a)_{\Downarrow_I} &= obs(t)_{\Downarrow_I} \cup obs(a)_{\Downarrow_I} \end{aligned}$$

La règle R_{inject} s'interprète de la manière suivante : si l'intrus a en sa connaissance le message m ($obs(t)_{\Downarrow_I} \vdash_{K_I} m$), alors il est capable de l'injecter à l'intention d'une entité X en se faisant passer pour une entité Y. Ceci se traduit par l'extension de la trace avec l'action externe $(i - X \lhd_I^i Y : m)$ où $\lhd^i$ désigne le canal de réception ($\lhd^i \in \{\lhd, \trianglelefteq\}$) propre à l'étape i. Ce canal est annoté par l'identité ayant injecté le message.

Les règles d'inférence données tout au long de cette section modélisent le protocole FOO et permettent de dériver l'ensemble de ces traces valides. La Table 3.11 illustre une trace du protocole. Cette trace, dérivée par application successive des règles données dans la partie droite de la table, correspond au scénario impliquant deux votants : V_1 et V_2. Plus précisément, elle correspond à une exécution durant laquelle V_1 complète la phase de vote, et V_2 s'abstient de voter après s'être enregistré.

Il est à noter, que nous avons considéré dans notre modélisation du protocole FOO uniquement les informations qui sont pertinentes à l'analyse ultérieure des propriétés de sécurité. Certaines actions internes telles que $(1 - choose(V, v))$ ou bien $(1 - generate(V, r))$, composantes propres à la spécification de la phase de préparation (voir Table 3.4), ont été ignorées.

3.4 Conclusion

Dans ce chapitre, nous avons présenté un ensemble de notations permettant de représenter au mieux un protocole de vote électronique. Ces notations constituent une extension de celles proposées par les concepteurs de la logique ADM dans le cadre de l'analyse des protocoles de commerce électronique. Cette extension se traduit par une algèbre de messages plus riche prenant en compte des primitives plus complexes que celles se limitant au chiffrement symétrique, et par la considération de canaux de communication plus élaborés que les canaux publiques traditionnels.

Nous avons étendu ensuite le modèle de Dolev-Yao dans le contexte des protocoles de vote électronique en définissant le comportement de l'intrus face aux différents canaux de communication, et en incrémentant son pouvoir de déduction conformément à l'algèbre de messages proposée.

Finalement, nous avons rappelé brièvement le modèle de protocole adopté, et présenté l'approche, et présenté l'approche que nous avons utilisé pour la construction de l'ensemble des traces valides. Cette approche a été illustrée sur l'exemple du protocole FOO.

L'analyse de ce dernier revient maintenant à spécifier les propriétés de sécurité, et vérifier ensuite si elles sont satisfaites par les traces propres au protocole FOO. Plus précisément, cela revient à vérifier la relation de satisfaction $t \models \phi$ où t est une trace valide du protocole ($t \in \mathscr{P}$), et ϕ une propriété de sécurité spécifiée au moyen de la logique ADM.

TABLE 3.11 – Une trace valide du protocole FOO

$$
t = \left\{
\begin{array}{ll}
\epsilon & R_{init} \\
(0 - V_1 \blacktriangleright V_1 : r_1).(0 - V_1 \blacktriangleleft V_1 : r_1). & R_{fresh} \\
(0 - V_1 \blacktriangleright V_1 : b_1).(0 - V_1 \blacktriangleleft V_1 : b_1). & R_{fresh} \\
(1 - request(V_1)). & R_{request} \\
(1 - V_1 \rhd A : V_1, \|\mathscr{B}(\{v_1\}_{r_1}, b_1)\|_{k_{V_1}}). & \\
(1 - A \lhd V_1 : V_1, \|\mathscr{B}(\{v_1\}_{r_1}, b_1)\|_{k_{V_1}}). & R_{rcv} \\
(2 - register(A, V_1)). & R_{authorize} \\
(2 - A \rhd V_1 : \|\mathscr{B}(\{v_1\}_{r_1}, b_1)\|_{k_A}). & \\
(2 - V_1 \lhd A : \|\mathscr{B}(\{v_1\}_{r_1}, b_1)\|_{k_A}). & R_{rcv} \\
(0 - V_2 \blacktriangleright V_2 : r_2).(0 - V_2 \blacktriangleleft V_2 : r_2). & R_{fresh} \\
(0 - V_2 \blacktriangleright V_2 : b_2).(0 - V_2 \blacktriangleleft V_2 : b_2). & R_{fresh} \\
(1 - request(V_2)). & R_{request} \\
(1 - V_2 \rhd A : V_2, \|\mathscr{B}(\{v_2\}_{r_2}, b_2)\|_{k_{V_2}}). & \\
(1 - A \lhd V_2 : V_2, \|\mathscr{B}(\{v_2\}_{r_2}, b_2)\|_{k_{V_2}}). & R_{rcv} \\
(2 - register(A, V_2)). & R_{authorize} \\
(2 - A \rhd V_2 : \|\mathscr{B}(\{v_2\}_{r_2}, b_2)\|_{k_A}). & \\
(2 - V_2 \lhd A : \|\mathscr{B}(\{v_2\}_{r_2}, b_2)\|_{k_A}). & R_{rcv} \\
(2 - deadline(A, T_1)). & R_{deadline_1} \\
 & \\
(3 - cast(V_1)). & R_{cast} \\
(3 - V_1 \trianglerighteq C : \|\{v_1\}_{r_1}\|_{k_A}). & \\
(3 - C \trianglelefteq V_1 : \|\{v_1\}_{r_1}\|_{k_A})). & R_{rcva} \\
(3 - register(C, \|\{v_1\}_{r_1}\|_{k_A})). & R_{collect} \\
(0 - A \blacktriangleright A : r_a).(0 - A \blacktriangleleft A : r_a). & R_{fresh} \\
(3 - C \trianglelefteq_A V_2 : \|\{v_a\}_{r_a}\|_{k_A}). & R_{inject} \\
(3 - register(C, \|\{v_a\}_{r_a}\|_{k_A})). & R_{collect} \\
(3 - deadline(C, T_2)). & R_{deadline_2} \\
 & \\
(4 - notify(C)). & R_{notify} \\
(4 - C \rhd B : \|\{v_1\}_{r_1}\|_{k_A}). & \\
(4 - C \rhd B : \|\{v_a\}_{r_a}\|_{k_A}). & \\
(4 - B \lhd C : \|\{v_1\}_{r_1}\|_{k_A}). & R_{rcv} \\
(4 - B \lhd C : \|\{v_a\}_{r_a}\|_{k_A}). & R_{rcv} \\
(5 - key(V_1)). & R_{key} \\
(5 - V_1 \trianglerighteq C : r_1). & \\
(5 - C \trianglelefteq V_1 : r_1). & R_{rcva} \\
(5 - register(C, \|\{v_1\}_{r_1}\|_{k_A}, r_1, v_1). & R_{open} \\
(5 - C \trianglelefteq_A V_2 : r_a). & R_{inject} \\
(5 - register(C, \|\{v_a\}_{r_a}\|_{k_A}, r_a, v_a). & R_{open} \\
(5 - deadline(C, T_3)). & R_{deadline_3} \\
 & \\
(6 - publish(C)). & R_{publish} \\
(6 - C \rhd B : \|\{v_1\}_{r_1}\|_{k_A}, r_1, v_1). & \\
(6 - C \rhd B : \|\{v_a\}_{r_a}\|_{k_A}, r_a, v_a). & \\
(6 - B \lhd C : \|\{v_1\}_{r_1}\|_{k_A}, r_1, v_1). & R_{rcv} \\
(6 - B \lhd C : \|\{v_a\}_{r_a}\|_{k_A}, r_a, v_a) & R_{rcv}
\end{array}
\right.
$$

Le chapitre qui suit est dédié à la spécification de ces propriétés, et à l'adaptation ensuite des formules spécifiées au protocole FOO comme étude de cas. Nous montrons particulièrement, que la trace donnée en Table 3.11 viole certaines des propriétés de sécurité, impliquant en conséquence la faillibilité de ce protocole quant à la satisfaction de certains des objectifs du vote électronique.

Chapitre 4

Spécification des propriétés des protocoles de vote électronique

Après avoir présenté dans le chapitre précédent la modélisation des protocoles de vote électronique, nous abordons dans celui-ci la spécification au moyen de la logique ADM des principales propriétés de sécurité : droit d'expression, éligibilité, non-réutilisabilité, exactitude, équité, anonymat et sans-reçu. Notre spécification est adaptée aux protocoles de vote décrits en chapitre 1 (voir section 1.4), et notamment au protocole FOO dont nous proposons une analyse détaillée de sa sécurité. La vérification des propriétés spécifiées par rapport à des traces valides issues de la modélisation du protocole FOO (voir section 3.3.2), nous permet de prouver la faillibilité de ce protocole à garantir certains des objectifs du vote électronique.

Pour une meilleure lisibilité des propriétés spécifiées, les termes constants seront écrits en gras. Les variables quant à elles seront précédées de la lettre x (x_m est une variable "message") ou X (X_v est une variable "entité"). Ceci permettra d'éviter la confusion entre les variables et les constantes dans les formules.

4.1 Démocratie

La propriété de démocratie englobe trois sous-propriétés : éligibilité, droit d'expression, et non réutilisabilité. Dans ce qui suit, nous donnons la spécification de chacune de ces propriétés pour le protocole FOO. Afin de simplifier notre formalisation, nous proposons de donner en premier lieu une version abstraite des formules spécifiées que nous appliquons ensuite au protocole FOO. La version de la formule, dite abstraite, est basée sur un format des actions plus simple que celui donné en section 3.1.3. Plus précisément, la représentation des actions internes et externes se limite à leurs noms et à leurs paramètres comme suit :

- L'action interne $i - act(A, p)$ est abstraite par $act(p)$.
- L'action externe $i - A \rhd B : m$ est abstraite par $\overrightarrow{act}(A, B, m)$.
- L'action externe $i - A \lhd B : m$ est abstraite par $\overleftarrow{act}(A, B, m)$. Celle-ci sera annotée par l'identité de l'intrus si elle est due à une injection de messages.

Ces abstractions sont illustrées par les exemples donnés ci-après. Nous introduisons en particulier, celles jouant un rôle dans la spécification des formules caractérisant la propriété de démocratie.

- L'action $\overrightarrow{req}(V, A, m_{rq})$ représente la demande de signature transmise par un votant V et à destination de l'autorité A. L'action $\overleftarrow{req}(A, V, m_{rq})$ dénote la réception de la requête par A.
- L'action $\overrightarrow{tok}(A, V, m_{tk})$ représente le jeton m_{tk} transmis par l'autorité A au votant V.
- L'action $\overrightarrow{cast}(V, C, m_{vt})$ représente le vote m_{vt} envoyé par une entité V à destination de l'autorité C. L'action $\overleftarrow{cast}(C, V, m_{vt})$ représente la réception du vote par C.
- L'action $reg(V)$ dénote une action interne représentant l'enregistrement (autorisation) du votant V.
- L'action $acc(m_{vt})$ dénote une action interne représentant l'enregistrement (acceptation) du vote m_{vt}.
- Les actions $d(T_1)$ et $d(T_2)$ représentent les *deadlines* délimitant respectivement la fin de la phase d'enregistrement et la fin de la phase de vote.

La formule adaptée au protocole FOO s'obtient ensuite en appliquant les substitutions définies dans la Table 4.1.

TABLE 4.1 – Adaptation de la propriété de démocratie au protocole FOO

$\overrightarrow{req}(V, A, m_{rq})$	=	$(1 - V \rhd A : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V})$
$\overleftarrow{req}(A, V, m_{rq})$	=	$(1 - A \lhd V : V, \|\mathscr{B}(\{v\}_r, b)\|_{k_V})$
$\overrightarrow{tok}(A, V, m_{tk})$	=	$(2 - A \rhd V : \|\mathscr{B}(\{v\}_r, b)\|_{k_A})$
$\overrightarrow{cast}(V, C, m_{vt})$	=	$(3 - V \trianglerighteq C : \|\{v\}_r\|_{k_A})$
$\overleftarrow{cast}(C, V, m_{vt})$	=	$(3 - C \trianglelefteq V : \|\{v\}_r\|_{k_A})$
$reg(V)$	=	$(2 - register(A, V))$
$acc(m_{vt})$	=	$(3 - register(C, \|\{v\}_r\|_{k_A}))$
$d(T_1)$	=	$(2 - deadline(A, T_1))$
$d(T_2)$	=	$(3 - deadline(C, T_2))$

4.1.1 Droit d'expression

Cette propriété exprime le fait que le protocole de vote doit permettre à chaque votant légitime de voter. Dans le protocole FOO, nous devons nous assurer que chaque votant légitime est capable d'obtenir un jeton (propriété relative au droit à l'enregistrement), et chaque votant enregistré a la possibilité d'émettre son vote (propriété relative au droit au vote). La première de ces propriétés exprime donc le fait que si un votant légitime émet une requête de demande de signature en aveugle, alors l'autorité responsable de l'élection doit fournir à ce votant le jeton lui permettant ultérieurement de participer à la phase de vote. Ce jeton doit être émis avant la fin

de la phase d'enregistrement. De manière similaire, la seconde propriété stipule que si un votant légitime et enregistré émet son vote, alors ce vote doit être obligatoirement accepté par l'autorité responsable de la collecte des votes. Là encore, la propriété est contrainte par une *deadline*, correspondant cette fois-ci à la fin de la phase de vote. En réalité, la propriété relative au droit au vote complète celle relative au droit à l'enregistrement. Il s'ensuit que ces deux propriétés peuvent être spécifiées de la même manière comme décrit par la formule Φ_{ab} donnée ci-après :

$$\begin{array}{lcl}
\Phi_{ab}(V) & \equiv & \langle (p_{va}.p_d)^+ \looparrowright \epsilon \rangle tt \rightarrow \langle (p_{va}.p_{vb}.p_d)^+ \looparrowright \epsilon \rangle tt \\
\Phi_{ab}(L_v) & \equiv & \bigwedge_{V \in L_v} \Phi_{ab}(V) \\
\text{Avec} & & \left(\begin{array}{lcl} p_{va} & = & va_1.va_2 \ldots va_n \\ p_{vb} & = & vb_1.vb_2 \ldots vb_m \\ p_d & = & d \end{array} \right)
\end{array}$$

La formule $\Phi_{ab}(V)$ représente la forme générale de la spécification de la propriété de droit d'expression. Elle s'interprète par rapport à une trace de la manière suivante : si la trace contient une séquence d'actions p_{va} consistant en une série d'opérations va_i $(i = 1 \ldots n)$ associées à un votant V, et si ces actions précèdent la *deadline* dénotée par l'action d, alors la trace doit contenir également la séquence d'actions dénotée par le *pattern* p_{vb}. De plus, la séquence des actions décrite par p_{vb}, doit avoir lieu avant la *deadline* marquée par l'action d. La généralisation de la formule $\Phi_{ab}(V)$ à l'ensemble des votants $V \in L_v$ est donnée par $\Phi_{ab}(L_v)$.

La spécification des propriétés de droit à l'enregistrement et de droit au vote peuvent être maintenant dégagées à partir de la formule Φ_{ab}. Tout d'abord, la spécification de la première composante de la propriété de droit d'expression est donnée par la formule $\mathscr{R}_1$:

$$\begin{array}{lcl}
\mathscr{R}_1 & \equiv & \bigwedge_{V \in L_v} (\langle (p_{vrq}.p_{d_1})^+ \looparrowright \epsilon \rangle tt \rightarrow \langle (p_{vrq}.p_{vtk}.p_{d_1})^+ \looparrowright \epsilon \rangle tt) \\
\text{Avec} & & \left\{ \begin{array}{lcl} p_{vrq} & = & \overrightarrow{req}(\boldsymbol{V}, \boldsymbol{A}, x_{rq}).\overleftarrow{req}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}) \\ p_{vtk} & = & \overrightarrow{tok}(\boldsymbol{A}, \boldsymbol{V}, x_{tk}) \\ p_{d_1} & = & d(\boldsymbol{T_1}) \end{array} \right.
\end{array}$$

Le *pattern* p_{vrq} représente une requête émise par un votant légitime V et sa réception par l'autorité A. Le *pattern* p_{vtk} dénote un jeton valide émis par l'autorité A à destination du votant V. Finalement, le *pattern* p_{d_1} représente l'événement "*deadline*" marquant la fin de la phase d'enregistrement. La formule $\mathscr{R}_1$ s'interprète par rapport à une trace de la manière suivante : si la trace analysée contient une requête valide émise par un votant légitime $V \in L_v$ (la trace contient une instance des actions $\overrightarrow{req}(V, A, x_{rq})$ et $\overleftarrow{req}(A, V, x_{rq})$), alors la trace doit contenir également un jeton émis par l'autorité A, et à destination du votant V (la trace contient une instance de l'action $\overrightarrow{tok}(A, V, x_{tk})$).

De manière similaire, la spécification de la propriété relative au droit au vote (formule $\mathscr{R}_2$) est dérivée à partir de la formule Φ_{ab} :

$$\mathscr{R}_2 \quad \equiv \quad \bigwedge_{V \in L_v} \langle (p_{vct}.p_{d_2})^+ \looparrowright \epsilon \rangle tt \rightarrow \langle (p_{vct}.p_{vac}.p_{d_2})^+ \looparrowright \epsilon \rangle tt$$

$$\text{Avec} \quad \left\{ \begin{array}{lcc} p_{vct} & = & \left(\begin{array}{c} \overrightarrow{req}(\boldsymbol{V},\boldsymbol{A},x_{rq}).\overleftarrow{req}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{cast}(\boldsymbol{V},\boldsymbol{C},x_{vt}).\overleftarrow{cast}(\boldsymbol{C},\boldsymbol{V},x_{vt}) \end{array} \right) \\ p_{vac} & = & acc(x_{vt}) \\ p_{d_2} & = & d(\boldsymbol{T_2}) \end{array} \right.$$

Le *pattern* p_{vct} désigne une séquence d'actions représentant un vote x_{vt} émis par un votant légitime et enregistré. Si cette séquence d'événements se produit avant la fin de la phase de vote (les actions de la séquence p_{vct} précèdent l'action $d(T_2)$ dans la trace analysée), alors la trace doit contenir également l'action $acc(x_{vt})$ représentant l'enregistrement du vote x_{vt}.

La spécification de la formule $\mathscr{R}$ caractérisant la propriété du droit d'expression est finalement obtenue par la conjonction des deux propriétés spécifiées ci-dessus :

$$\mathscr{R} \quad \equiv \quad \mathscr{R}_1 \quad \wedge \quad \mathscr{R}_2$$

Les deux propriétés spécifiées jusqu'ici ont été exprimées au moyen d'actions abstraites. La version de la formule adaptée au protocole FOO est obtenue en appliquant les associations données en Table 4.1. Il est à noter que certaines précautions doivent êtres prises lors de la translation du format des actions abstraites à celui adopté par notre modélisation. Plus précisément, une attention particulière doit être accordée aux variables impliquées dans les actions. À titre d'exemple, les actions $\overrightarrow{req}(\boldsymbol{V},\boldsymbol{A},x_{rq})$ et $\overrightarrow{tok}(\boldsymbol{A},\boldsymbol{V},x_{tk})$ dans la formule $\mathscr{R}_1$ seront substituées respectivement par $(\mathbf{1}-\boldsymbol{V} \rhd \boldsymbol{A} : \boldsymbol{V}, \|\mathscr{B}(\{x_v\}_{x_r}, x_b)\|_{\boldsymbol{k_V}})$ et $(\mathbf{2}-\boldsymbol{A} \rhd \boldsymbol{V} : \|\mathscr{B}(\{x_v\}_{x_r}, x_b)\|_{\boldsymbol{k_A}})$. La version de la formule adaptée au protocole FOO est ainsi plus fine que celle basée sur la forme abstraite, et permet d'illustrer un peu plus la correspondance (*pattern matching*) entre les variables impliquées dans les formules. Par exemple, dans la formule $\mathscr{R}_1$ adaptée au protocole FOO, il faut qu'il y ait une correspondance, au niveau de la trace, entre les éléments variables de la requête $(\|\mathscr{B}(\{x_v\}_{x_r}, x_b)\|_{\boldsymbol{k_V}})$, et ceux de la réponse $(\|\mathscr{B}(\{x_v\}_{x_r}, x_b)\|_{\boldsymbol{k_A}})$.

4.1.2 Éligibilité

La propriété d'éligibilité signifie que seuls les votants légitimes sont autorisés à voter. Dans le protocole FOO, nous devons nous assurer que seuls les votants légitimes peuvent obtenir un jeton (formule $\mathscr{E}_1$), et seuls les votants légitimes et enregistrés peuvent participer à la phase de vote (formule $\mathscr{E}_2$). Ces formules peuvent être spécifiées en appliquant le raisonnement inverse aux propriétés précédemment spécifiées. En effet, la formule $\mathscr{E}_1$ peut être définie comme la réciproque de la formule $\mathscr{R}_1$, exprimant ainsi le fait que si la trace analysée contient un jeton émis par A, alors ce jeton est à destination d'un votant légitime et enregistré. Il en est de même pour

la formule $\mathscr{E}_2$ qui peut être définie comme la réciproque de la formule $\mathscr{R}_2$. Par conséquent, la spécification de la forme générale de la propriété d'éligibilité (Φ_{ba}) peut être exprimée à partir de celle caractérisant le droit d'expression (Φ_{ab}) :

$$\Phi_{ba}(L_v) \equiv \nu X.(\langle p^+_{xb} \looparrowright \epsilon \rangle tt \rightarrow \bigvee_{V \in L_v} \langle p_{vab} \looparrowright p^-_{vab} \rangle X)$$

Avec
$$\begin{pmatrix} p_{xb} = xb_1.xb_2 \ldots xb_m \\ p_{vab} = (vab_1.vab_2 \ldots vab_n)^+ \end{pmatrix}$$

La formule Φ_{ba} s'interprète par rapport à une trace de la manière suivante : si la trace analysée contient une séquence d'actions p_{xb} relative à un événement particulier (transmission d'un jeton, enregistrement d'un vote), alors la trace doit contenir une séquence p_{vab} associée à un votant légitime V ayant été à l'origine de cet événement. À noter que la forme récursive de la formule Φ_{ba} permet de vérifier que chaque événement particulier décrit par le *pattern* p_{xb} est légitime.

La spécification des formules $\mathscr{E}_1$ et $\mathscr{E}_2$ se dérivent maintenant directement à partir de la forme générale de la formule Φ_{ba}. Tout d'abord, la formule $\mathscr{E}_1$ exprime le fait que seuls les votants légitimes sont autorisés à disposer d'un jeton. Elle est définie comme suit :

$$\mathscr{E}_1 \equiv \nu X.(\langle p^+_{xtk} \looparrowright \epsilon \rangle tt \rightarrow \bigvee_{V \in L_v} \langle p_{vlt} \looparrowright p^-_{vlt} \rangle X)$$

Avec
$$\begin{cases} p_{xtk} = \overrightarrow{tok}(\boldsymbol{A}, X_v, x_{tk}) \\ p_{vlt} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V}, \boldsymbol{A}, x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{tok}(\boldsymbol{A}, \boldsymbol{V}, x_{tk}) \end{pmatrix}^+ \end{cases}$$

Le pattern p_{xtk} représente un jeton transmis par l'autorité A à une entité X_v, et le pattern p_{vlt} désigne la séquence d'actions qui conduit légitimement à un jeton. Cette séquence est relative à un votant légitime V, et est composée de la succession des actions $\overrightarrow{req}$, $\overleftarrow{req}$, reg, et $\overrightarrow{tok}$. La formule $\mathscr{E}_1$ s'interprète maintenant de la manière suivante : si la trace analysée contient un jeton émis par l'autorité A à une entité X_v, alors l'entité X_v doit correspondre à un votant légitime. Plus précisément, la trace doit contenir la participation d'un votant légitime $V \in L_v$ ayant émis une requête pour ce jeton (la trace contient une instance des actions $\overrightarrow{req}(V, A, x_{rq})$ et $\overleftarrow{req}(A, V, x_{rq})$), et ayant été authentifié comme votant autorisé à voter (l'action $reg(V)$ appartient également à la trace), et si on enlève de la trace une occurrence des actions $\overrightarrow{req}$, $\overleftarrow{req}$, reg, et $\overrightarrow{tok}$, la trace résultante doit satisfaire de nouveau la formule $\mathscr{E}_1$. La forme récursive de la formule permet de vérifier que chaque jeton émis par l'autorité A est destiné à un votant légitime et autorisé.

À noter que lorsque l'attaquant considéré est modélisé comme votant légitime ($I \in L_v$), il est trivial que ce dernier puisse disposer d'un jeton. Or, nous ne considérons pas ce scénario comme

une attaque. À cet effet, afin d'éviter que la propriété d'éligibilité ne soit violée lorsque ce cas de configuration survient, il nous faut remanier la spécification de la formule $\mathscr{E}_1$.

Dans notre modélisation du protocole FOO (voir section 3.3.2), la règle $R_{request}$ permet au votant, dans le respect du protocole, d'émettre sa demande de signature en aveugle. Cependant, lorsque le rôle de l'intrus est joué par un votant légitime, ce dernier peut émettre sa requête grâce à la règle d'injection de messages qui lui est associée (règle R_{inject}). Dans ce cas précis, la propriété exprimée par $\mathscr{E}_1$ n'est plus satisfaite étant donné que la trace contiendra la séquence d'actions $p_{ilt} = \overleftarrow{req}_V(A,V,x_{rq}).reg(V).\overrightarrow{tok}(A,V,x_{tk})$ à la place de la séquence p_{vlt} requise par la formule $\mathscr{E}_1$. Afin de remédier à ce problème, nous rajoutons à la formule $\mathscr{E}_1$ une composante de plus associée à ce cas de configuration.

$$\mathscr{E}'_1 \quad \equiv \quad \nu X.(\langle p^+_{xtk} \rangle \epsilon \rangle tt \rightarrow \bigvee_{V \in L_v} (\langle p_{vlt} \rangle p^-_{vlt} \rangle X \vee \langle p_{ilt} \rangle p^-_{ilt} \rangle X))$$

Avec
$$p_{ilt} = \begin{pmatrix} \overleftarrow{req}_{\boldsymbol{V}}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{tok}(\boldsymbol{A},\boldsymbol{V},x_{tk}) \end{pmatrix}^+$$

Le *pattern* p_{ilt} représente la séquence d'actions où un jeton est émis par l'autorité A à un votant légitime V ayant agit malicieusement dans le protocole. Les *patterns* p_{xtk} et p_{vlt} sont ceux définis précédemment dans la version initiale de la formule $\mathscr{E}_1$.

De manière similaire, la spécification de la formule $\mathscr{E}_2$ est donnée par :

$$\mathscr{E}_2 \quad \equiv \quad \nu X.(\langle p^+_{xac} \rangle \epsilon \rangle tt \rightarrow \bigvee_{V \in L_v} (\langle p_{vlv} \rangle p^-_{vlv} \rangle X \vee \langle p_{ilv_1} \rangle p^-_{ilv_1} \rangle X \vee \langle p_{ilv_2} \rangle p^-_{ilv_2} \rangle X))$$

Avec
$$\begin{cases} p_{xac} = acc(x_{vt}) \\ p_{vlv} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V},\boldsymbol{A},x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{cast}(\boldsymbol{V},\boldsymbol{C},x_{vt}). \\ \overleftarrow{cast}(\boldsymbol{C},\boldsymbol{V},x_{vt}). \\ acc(x_{vt}) \end{pmatrix}^+ \\ p_{ilv_1} = \begin{pmatrix} \overleftarrow{req}_{\boldsymbol{V}}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overleftarrow{cast}_{\boldsymbol{V}}(\boldsymbol{C},\boldsymbol{V},x_{vt}). \\ acc(x_{vt}) \end{pmatrix}^+ \\ p_{ilv_2} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V},\boldsymbol{A},x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overleftarrow{cast}_{\boldsymbol{V}}(\boldsymbol{C},\boldsymbol{V},x_{vt}). \\ acc(x_{vt}) \end{pmatrix}^+ \end{cases}$$

La *pattern* p_{xac} désigne l'action interne représentant l'acceptation d'un vote donné x_{vt}. Le *pattern* p_{vlv} représente la séquence d'actions conduisant, dans le respect du protocole, à l'acceptation du vote x_{vt}. Les *patterns* p_{ilv_1} et p_{ilv_2} caractérisent des suites d'actions conduisant à l'acceptation du vote lorsque ce dernier est émis par un votant jouant le rôle de l'intrus.

La spécification de la propriété d'éligibilité (formule $\mathscr{E}$) est finalement obtenue par la conjonction des formules $\mathscr{E}'_1$ et $\mathscr{E}_2$:

$\mathscr{E}$	$\equiv$	$\mathscr{E}'_1$	$\wedge$	$\mathscr{E}_2$

La version de la formule adaptée au protocole FOO s'obtient maintenant en appliquant les substitutions données en Table 4.1.

Le protocole FOO ne satisfait pas la propriété d'éligibilité en présence d'un administrateur corrompu. En effet, si un votant s'abstient de voter après s'être enregistré, alors l'autorité A peut voter à sa place. Ce scénario est illustré par la trace t de la Table 3.11 (voir page 66) où l'administrateur A parvient à injecter son vote v_a, et à le faire accepter par le collecteur C.

Dans notre approche, la violation de la propriété d'éligibilité se confirme par $t \not\models \mathscr{E}$. Plus précisément, la trace t ne satisfait pas la formule $\mathscr{E}_2$. En effet, après élimination de la trace t des éléments associés au votant V_1, à savoir les actions décrites par le *pattern* p_{vlv}, la trace résultante ne satisfait pas la formule $\mathscr{E}_2$. La trace obtenue contient un vote enregistré, représenté par l'action interne $(3 - register(C, \|\{v_a\}_{r_a}\|_{k_A}))$, auquel ne correspond aucun vote émis par un votant légitime.

4.1.3 Non réutilisabilité

La propriété de non réutilisabilité signifie qu'un votant légitime ne peut voter plus d'une fois. Dans le protocole FOO, un électeur peut voter plusieurs fois s'il obtient plusieurs jetons de la part de l'autorité A, ou en tentant de rejouer son vote. Dans le cas de la première possibilité, la propriété à exprimer doit permettre de vérifier qu'un votant légitime n'est autorisé à obtenir qu'un seul jeton. Face au rejeu, nous devons nous assurer que le protocole rejette les doublons. La propriété de non réutilisabilité sera donc caractérisée par deux formules : $\mathscr{N}_1$ et $\mathscr{N}_2$. Ces formules peuvent être définies de la même manière telle que donnée par la formule Φ_{2a}, et qui permet de vérifier qu'une action particulière a n'apparait pas plus d'une fois dans la trace.

Φ_{2a}	$\equiv$		$\neg\langle p^+_{2a} \looparrowright \epsilon\rangle\, tt$	
Avec		p^+_{2a}	$=$	$(a.a)^+$
			$=$	$x_1.a.x_2.a.x_3$

Tout d'abord, la spécification de la formule $\mathscr{N}_1$ est donnée par :

$\mathscr{N}_1$	$\equiv$		$\neg(\langle p^+_{2tk} \looparrowright \epsilon\rangle\, tt \vee \langle p^+_{2rg} \looparrowright \epsilon\rangle\, tt)$	
Avec	$\begin{cases} p_{2rg} \\ p_{2tk} \end{cases}$		$=$	$reg(X_v).reg(X_v)$
			$=$	$\overrightarrow{tok}(\boldsymbol{A}, X_v, x_{tk_1}).\overrightarrow{tok}(\boldsymbol{A}, X_v, x_{tk_2})$

Les actions $\overrightarrow{tok}(\boldsymbol{A}, X_v, x_{tk_1})$ et $\overrightarrow{tok}(\boldsymbol{A}, X_v, x_{tk_2})$ représentent deux jetons envoyés à une même entité X_v. La première partie de la formule exprime donc le fait que chaque votant légitime n'est autorisé à obtenir qu'un seul jeton. Ensuite, l'action $reg(X_v)$ représente l'enregistrement (autorisation) de l'entité X_v. Par conséquent, la seconde partie de la formule exprime le fait que chaque votant ne peut être autorisé à voter plus d'une fois.

De la même manière, la spécification de la formule $\mathscr{N}_2$ est donnée par :

$\mathscr{N}_2$	$\equiv$		$\neg\langle p^+_{2ac} \looparrowright \epsilon\rangle tt$	
Avec		p_{2ac}	$=$	$acc(x_{vt}).acc(x_{vt})$

L'action $acc(x_{vt})$ dénote une action interne représentant l'enregistrement (acceptation) d'un vote x_{vt}. La formule $\mathscr{N}_2$ permet de vérifier donc la faculté du protocole à éliminer les votes en double.

La spécification de la propriété de non réutilisabilité (formule $\mathscr{N}$) est donnée par la conjonction des formules $\mathscr{N}_1$ et $\mathscr{N}_2$, et la version correspondante au protocole FOO s'obtient en ayant recours aux associations données en Table 4.1.

$\mathscr{N}$	$\equiv$	$\mathscr{N}_1$	$\wedge$	$\mathscr{N}_2$		

Finalement, la spécification de la propriété de démocratie est obtenue par la conjonction des formules caractérisant ses sous-propriétés, à savoir le droit d'expression, l'éligibilité, et la non réutilisabilité.

$\mathscr{D}$	$\equiv$	$\mathscr{R}$	$\wedge$	$\mathscr{E}$	$\wedge$	$\mathscr{N}$

4.2 Exactitude

La propriété d'exactitude signifie que les résultats publiés à l'issue de l'élection doivent correspondre exactement aux choix des votants. Sa portée est assez large et permet de garantir les points suivants :

- Aucun vote n'a été dupliqué.
- Aucun vote n'a été supprimé.
- Aucun vote n'a été altéré.
- Aucun vote n'a été frauduleusement inséré.

Comme pour le cas de la propriété de démocratie, notre formalisation de la propriété d'exactitude est spécifique au protocole FOO. Nous proposons dans un premier temps de formaliser cette propriété au moyen d'actions abstraites. La version de la formule obtenue sera ensuite appliquée au protocole FOO.

La spécification de la propriété d'exactitude sous sa forme abstraite nécessite d'introduire, en complément aux actions de la Table 4.1, les éléments suivants :

- L'action $\overrightarrow{key}(V, C, m_{ky})$ représente l'envoi de la clé m_{ky} du votant V à l'autorité C. L'action $\overleftarrow{key}(C, V, m_{ky})$ dénote la réception de la clé par l'entité C.
- L'action $count(m_{ovt})$ dénote une action interne représentant l'ouverture du bulletin et la comptabilisation du vote m_{ovt}.

– L'action $ann()$ dénote une action interne caractérisant l'annonce des résultats.
– L'action $\overrightarrow{pub}(C,B,m_{ovt})$ représente la publication auprès de l'entité B du vote m_{ovt} par l'entité C.
– L'action $d(T_3)$ représente la *deadline* marquant la fin de la phase d'ouverture des bulletins.

La formule adaptée au protocole FOO s'obtient en appliquant les substitutions données conjointement en Tables 4.1 et 4.2

TABLE 4.2 – Adaptation de la propriété d'exactitude au protocole FOO

$\overrightarrow{key}(V,C,m_{ky})$	=	$(5-V \trianglerighteq C : r)$
$\overleftarrow{key}(C,V,m_{ky})$	=	$(5-C \trianglelefteq V : r)$
$count(m_{ovt})$	=	$(5-register(C,\|\{v\}_r\|_{k_A},r,v))$
$d(T_3)$	=	$(5-deadline(C,T_3))$
$ann()$	=	$(6-publish(C))$
$\overrightarrow{pub}(C,B,m_{ovt})$	=	$(6-C \triangleright B : \|\{v\}_r\|_{k_A},r,v)$

Dans ce qui suit, nous exprimons la propriété d'exactitude comme étant la conjonction de trois sous propriétés. La première de ces propriétés (formule $\mathscr{C}_1$) garantit que chaque vote émis par un votant légitime est correctement comptabilisé et publié par l'entité responsable de la collecte des votes. La formule $\mathscr{C}_1$ permet de vérifier ainsi qu'aucun vote n'a été altéré ou écarté du scrutin. La seconde propriété peut se voir comme la réciproque de la première propriété, et permet de s'assurer que chaque vote publié est un vote émis par un votant légitime et autorisé ($\mathscr{C}_2$). La formule $\mathscr{C}_2$ permet de vérifier qu'aucun vote n'a été frauduleusement inséré ou dupliqué. Finalement la troisième propriété n'est autre que la première composante de la propriété de non réutilisabilité (formule $\mathscr{N}_1$), et garantissant que chaque votant légitime n'est autorisé à obtenir qu'un seul jeton.

La formule $\mathscr{C}_1$ complète en réalité la propriété de droit d'expression. Sa spécification est donc dérivée, à quelques détails près, à partir de la forme générale de la formule $\mathscr{R}$:

$$\mathscr{C}_1 \quad \equiv \quad \langle p^+_{ann} \looparrowright \epsilon \rangle\, tt \rightarrow \Big(\bigwedge_{V \in L_v} \langle (p_{vvt}.p_{d_3})^+ \looparrowright \epsilon \rangle\, tt \rightarrow \langle (p_{vvt}.p_{vct}.p_{d_3}.p_{vpb})^+ \looparrowright \epsilon \rangle\, tt \Big)$$

Avec

$$\begin{cases} p_{vvt} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V},\boldsymbol{A},x_{rq}).\overleftarrow{req}(\boldsymbol{A},\boldsymbol{V},x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{cast}(\boldsymbol{V},\boldsymbol{C},x_{vt}).\overleftarrow{cast}(\boldsymbol{C},\boldsymbol{V},x_{vt}). \\ acc(x_{vt}). \\ \overrightarrow{key}(\boldsymbol{V},\boldsymbol{C},x_{ky}).\overleftarrow{key}(\boldsymbol{C},\boldsymbol{V},x_{ky}) \end{pmatrix} \\ p_{vpb} = \overrightarrow{pub}(\boldsymbol{C},\boldsymbol{B},x_{ovt}) \\ p_{vct} = count(x_{ovt}) \\ p_{ann} = ann() \\ p_{d_3} = d(\boldsymbol{T_3}) \end{cases}$$

La formule $\mathscr{C}_1$ s'interprète par rapport à la trace à analyser de la manière suivante : si la trace contient la participation d'un votant légitime ayant poursuivi l'ensemble des étapes du protocole, à savoir l'enregistrement, le vote, et l'ouverture des bulletins, alors la trace doit contenir les actions représentant la comptabilisation et la publication de son vote. Plus précisément, la première partie de la formule $(\langle p^+_{ann} \leadsto \epsilon \rangle tt)$ permet de s'assurer d'abord, par la vérification de la présence de l'action $ann()$, que les résultats ont été publiés. Ensuite, si la trace contient la séquence d'actions p_{vvt}, représentant l'envoi par un votant légitime et autorisé, de la clé permettant l'ouverture de son bulletin précédemment transmis et accepté, alors la trace doit contenir à la fois les actions décrites par les *patterns* p_{vct} et p_{vpb}, et représentant respectivement la comptabilisation et la publication du vote.

La seconde composante de la propriété d'exactitude se présente comme la réciproque de la propriété précédemment spécifiée, et sa spécification est calquée sur la forme générale de la propriété d'éligibilité (formule $\mathscr{E}$) :

$$\mathscr{C}_2 \quad \equiv \quad \nu X.(\langle p^+_{xpb} \leadsto \epsilon \rangle tt \rightarrow \bigvee_{V \in L_v} (\langle p_{vlv} \leadsto p^-_{vlv} \rangle X \vee \begin{pmatrix} \langle p_{ilv_1} \leadsto p^-_{ilv_1} \rangle X \\ \vee \\ \langle p_{ilv_2} \leadsto p^-_{ilv_2} \rangle X \\ \vee \\ \langle p_{ilv_3} \leadsto p^-_{ilv_3} \rangle X \end{pmatrix}))$$

Avec

$$p_{xpb} = \overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt})$$

$$p_{vlv} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V}, \boldsymbol{A}, x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{cast}(\boldsymbol{V}, \boldsymbol{C}, x_{vt}). \\ \overleftarrow{cast}(\boldsymbol{C}, \boldsymbol{V}, x_{vt}). \\ acc(x_{vt}). \\ \overrightarrow{key}(\boldsymbol{V}, \boldsymbol{C}, x_{ky}). \\ \overleftarrow{key}(\boldsymbol{C}, \boldsymbol{V}, x_{ky}). \\ count(x_{ovt}). \\ \overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt}) \end{pmatrix}^+ \quad | \quad p_{ilv_1} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V}, \boldsymbol{A}, x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}). \\ reg(\boldsymbol{V}). \\ \overrightarrow{cast}(\boldsymbol{V}, \boldsymbol{C}, x_{vt}). \\ \overleftarrow{cast}(\boldsymbol{C}, \boldsymbol{V}, x_{vt}). \\ acc(x_{vt}). \\ \overleftarrow{key}_{\boldsymbol{V}}(\boldsymbol{C}, \boldsymbol{V}, x_{ky}). \\ count(x_{ovt}). \\ \overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt}) \end{pmatrix}^+$$

$$p_{ilv_2} = \begin{pmatrix} \overrightarrow{req}(\boldsymbol{V}, \boldsymbol{A}, x_{rq}). \\ \overleftarrow{req}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}). \\ reg(\boldsymbol{V}). \\ \overleftarrow{cast}_{\boldsymbol{V}}(\boldsymbol{C}, \boldsymbol{V}, x_{vt}). \\ acc(x_{vt}). \\ \overleftarrow{key}_{\boldsymbol{V}}(\boldsymbol{C}, \boldsymbol{V}, x_{ky}). \\ count(x_{ovt}). \\ \overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt}) \end{pmatrix}^+ \quad | \quad p_{ilv_3} = \begin{pmatrix} \overleftarrow{req}_{\boldsymbol{V}}(\boldsymbol{A}, \boldsymbol{V}, x_{rq}). \\ reg(\boldsymbol{V}). \\ \overleftarrow{cast}_{\boldsymbol{V}}(\boldsymbol{C}, \boldsymbol{V}, x_{vt}). \\ acc(x_{vt}). \\ \overleftarrow{key}_{\boldsymbol{V}}(\boldsymbol{C}, \boldsymbol{V}, x_{ky}). \\ count(x_{ovt}). \\ \overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt}) \end{pmatrix}^+$$

La formule $\mathscr{C}_2$ permet de vérifier que chaque vote publié est transmis par un votant légitime et autorisé. Elle s'interprète par rapport à une trace de la manière suivante : si la trace contient un vote publié par une entité C (la trace contient une instance de l'action $\overrightarrow{pub}(\boldsymbol{C}, \boldsymbol{B}, x_{ovt})$), alors la trace doit contenir la séquence d'actions représentant l'envoi de ce vote par un votant légitime ayant participé à l'ensemble des étapes du protocole. La formule est récursive signifiant que cette condition doit être vérifiée pour chaque vote publié. Plus précisément, à chaque fois qu'une séquence d'actions représentant l'envoi et la publication d'un vote légitime est trouvée, elle est supprimée de la trace et la propriété est vérifiée de nouveau par rapport à la trace résultante. La propriété est satisfaite si la trace obtenue ne contient plus d'actions de publication de votes.

La spécification de la propriété d'exactitude (formule $\mathscr{C}$) est finalement donnée par :

$$\mathscr{C} \quad \equiv \quad \mathscr{C}_1 \quad \wedge \quad \mathscr{C}_2 \quad \wedge \quad \mathscr{N}_1$$

À noter que la formule $\mathscr{N}_1$ complète la spécification, et permet de s'assurer qu'aucun votant n'a été autorisé à voter plus d'une fois.

La version de la formule adaptée au protocole FOO s'obtient maintenant par application des associations d'actions données en Tables 4.1 et 4.2.

Le protocole FOO ne garantit pas la propriété d'exactitude, et ce pour les mêmes raisons qui font échouer le protocole à garantir la propriété d'éligibilité. En effet, un administrateur malicieux A est capable de fabriquer par lui même un bulletin, et de le faire accepter par le collecteur C qui le publiera à l'issue de l'élection. Ce scénario est illustrée par la trace t, donnée en Table 3.11, qui ne satisfait pas la formule $\mathscr{C}_2$. En réalité, la trace t contient un vote publié $(6 - C \rhd B : \|\{v_a\}_{r_a}\|_{k_A}, r_a, v_a)$ auquel ne correspond aucun vote émis par un votant légitime et autorisé. La trace ne satisfait pas par conséquent la formule $\mathscr{C}$ ($t \not\models \mathscr{C}$). Nous en concluons donc que le protocole ne garantit pas la propriété d'exactitude.

4.3 Équité

La propriété d'équité signifie que le protocole ne doit pas permettre la publication de résultats partiels. De tels résultats peuvent influencer le choix des votants qui ne se sont pas encore prononcés, et par conséquent peuvent avoir un impact sur le résultat final de l'élection. Pour que cette propriété soit garantie, il suffit que les votes restent secret tant que la phase de vote n'est pas clôturée. Ceci peut être assuré en exigeant que les votants transmettent leurs bulletins chiffrés, et en procédant à l'ouverture des bulletins que lorsque tous les votants se sont exprimés.

Notre spécification de l'équité est basé sur le secret des votes, ce qui signifie que la formule à spécifier doit permettre de vérifier qu'aucun vote $v \in L_c$ ne peut être déduit par un observateur I avant la *deadline* T marquant la fin de la phase de vote. La spécification de la propriété d'équité sera donc représentée par la formule $\mathscr{F}(I, K, L_c, T)$, où K désigne la connaissance initiale de l'observateur I. À titre d'exemple, si l'autorité A joue le rôle de l'observateur I, alors nous devons inclure dans l'ensemble K, la clé privée k_A. Bien que l'ensemble des choix L_c soit publiquement connu par toutes les entités du protocole FOO, on ne l'inclut pas dans l'ensemble K étant donné qu'on cherche à déduire les choix des votants à partir de leurs participations dans le protocole.

Afin de mieux expliquer la formalisation de la propriété d'équité dans la logique ADM, nous considérons en premier lieu un protocole de vote simpliste où tous les messages sont transmis via des canaux publiques, et où le chiffrement symétrique est la seule primitive cryptographique utilisée dans le protocole. Par la suite, nous décrivons comment cette première spécification de la propriété d'équité peut être simplifiée au moyen des systèmes de réécriture. Finalement, nous donnons la version de la formule adaptée au protocole FOO, et discutons de la portabilité de la propriété à d'autres types de protocoles de vote électronique.

Version initiale. Pour des raisons de simplicité, nous proposons d'abord de formuler la propriété d'équité en considérant un protocole de vote naïf (canaux publiques + chiffrement symétrique). Ci-après, nous expliquons les principales étapes aboutissant à la spécification de la formule $\mathscr{F}(I,K,L_c,T)$.

La première étape pour spécifier la propriété d'équité est d'étendre la trace à analyser par la connaissance de l'observateur I. Ceci peut être réalisé en considérant une étape virtuelle du protocole (étape 0) où l'observateur I envoie à lui même l'ensemble des messages K. Soit t_0 le bout de trace représentant cette étape virtuelle :

$$t_0 \quad = \quad (0 - I \rhd I : K)$$

Maintenant, grâce à la linéarité de la logique ADM, la trace t_0 peut être rajoutée à la trace à analyser comme suit :

$$\mathscr{F}(I,K,L_c,T) \quad \equiv \quad [x \curvearrowright t_0.x]\mathscr{S}(L_c,T)$$

Où $\mathscr{S}(L_c,T)$ représente la formule permettant de vérifier que les éléments précédant la *deadline*, ayant lieu à l'instant T, ne permettent de déduire aucun des votes $v \in L_c$. Étant donné que la déduction est basée uniquement sur les événements précédant la *deadline* marquant la fin de la phase de vote, nous devons supprimer de la trace, tous les éléments ayant lieu après cette *deadline* (représentée ici par l'action abstraite $d(T)$). Ceci peut être effectué de manière similaire à l'étape précédente :

$$\mathscr{S}(L_c,T) \quad \equiv \quad [x_1.d(\boldsymbol{T}).x_2 \curvearrowright x_1]\mathscr{S}(L_c)$$

Il est à noter que ces deux étapes peuvent être fusionnées ensemble comme suit :

$$\mathscr{F}(I,K,L_c,T) \quad \equiv \quad [x_1.d(\boldsymbol{T}).x_2 \curvearrowright t_0.x_1]\mathscr{S}(L_c)$$

La formule $\mathscr{S}$ représente la spécification de la propriété de confidentialité. Dans la logique ADM, cette propriété est formalisée comme donné par la Table 4.3 [ADM03].

TABLE 4.3 – Spécification de la propriété de confidentialité

$$
\begin{array}{lll}
\mathscr{S}(v) & \equiv & \neg\ (\nu X.((\langle x_a.(x_i - X_s \rhd X_r : v).x_b \looparrowright \epsilon\rangle tt) \vee \\
 & & \\
 & & \langle x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : \{x_m\}_{x_k}).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_k).x_c \looparrowright \\
 & & x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_m).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_k).x_c\rangle X \vee \\
 & & \\
 & & \langle x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_k).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : \{x_m\}_{x_k}).x_c \looparrowright \\
 & & x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_k).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_m).x_c\rangle X \vee \\
 & & \\
 & & \langle x_a.(x_i - X_s \rhd X_r : x_{m_1}, x_{m_2}).x_b \looparrowright \\
 & & x_a.(x_i - X_s \rhd X_r : x_{m_1}).(x_i - X_s \rhd X_r : x_{m_2}).x_b\rangle X)) \\
 & & \\
\mathscr{S}(L_c) & \equiv & \bigwedge_{v \in L_c} \mathscr{S}(v)
\end{array}
$$

Tout d'abord, la formule $\mathscr{S}(v)$ caractérise le secret de la donnée v (vote). Cette formule permet de vérifier, récursivement, si la trace analysée t ou l'une de ses versions modifiées contient l'action particulière $(x_i - X_s \rhd X_r : v)$. Cette action représente l'envoi, dans le protocole, du vote v via un canal publique, et par conséquent implique que le vote est visible pour I. Les modifications apportées à la trace modélisent le pouvoir de déduction de l'observateur I :

- déchiffrement de messages : si la trace analysée contient une instance de l'action $(x_{i_1} - X_{s_1} \rhd X_{r_1} : \{x_m\}_{x_k})$ (envoi d'un message chiffré $\{x_m\}_{x_k}$) suivie ou précédée d'une instance de l'action $(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_k)$, alors le message $\{x_m\}_{x_k}$ dans la première action sera remplacée par x_m. Ceci exprime la capacité de l'observateur I à déchiffrer le message $\{x_m\}_{x_k}$ s'il a en sa connaissance le message x_k.
- décomposition de messages : si la trace analysée contient un message composée x_{m_1}, x_{m_2}, alors il sera décomposé en messages x_{m_1} et x_{m_2}.

La formule $\mathscr{S}(L_c)$ est ensuite dérivée de la spécification de $\mathscr{S}(v)$. Elle caractérise le secret de tous les éléments de l'ensemble des choix L_c.

Considérons maintenant l'exemple de protocole de vote illustré par la Table 4.4

TABLE 4.4 – Exemple - Protocole de vote naïf P

Étape	**Action**
(1) Vote	$V \longrightarrow A : \{v\}_{k_{AV}}$
Fin de la phase de vote	Deadline T_1
(2) Publication des résultats	$A \longrightarrow B : \ldots, v, \ldots$

Dans le protocole P décrit par la Table 4.4, le votant envoie son vote chiffré au moyen de la clé partagée k_{AV}. Ensuite, à l'issue de la phase de vote dénotée par la *deadline* T_1, l'autorité A publie auprès de l'entité B, l'ensemble des votes reçus. Soit v_1 et v_2 les choix possibles dans le protocole P ($L_c = \{v_1, v_2\}$).

Maintenant, une trace de ce protocole pourrait être la suivante :

$$t = \begin{cases} (1 - V \rhd A : \{v_1\}_{k_{AV}}). \\ (1 - A \lhd V : \{v_1\}_{k_{AV}}). \\ (2 - deadline(A, T_1)). \\ (2 - A \rhd B : v_1). \\ (2 - B \lhd A : v_1) \end{cases}$$

En réalité, la trace t ne satisfait pas la propriété d'équité donnée ci-après. En effet, l'autorité A a en sa connaissance la clé k_{AV}, et de ce fait peut déduire le vote v_1 à partir de $\{v_1\}_{k_{AV}}$ avant la fin de la phase de vote.

$$\mathscr{F}(A, k_{AV}, L_c, T_1) \quad \equiv \quad [x_1.d(\boldsymbol{T}).x_2 \looparrowright t_0.x_1](\mathscr{S}(v_1) \wedge \mathscr{S}(v_2))$$

Démontrer formellement que le protocole P ne garantit pas la propriété d'équité, revient dans notre approche à vérifier que la trace t ne satisfait pas la formule $\mathscr{F}(A, k_{AV}, L_c, T_1)$ ($t \not\models \mathscr{F}(A, k_{AV}, L_c, T_1)$). Le processus de vérification est décrit ci-après.

Tout d'abord, la première partie de la formule ($[-]$) permet de supprimer de la trace les actions qui se sont déroulées après la *deadline* T_1, marquant la fin de la phase de vote. La trace est aussi étendue avec l'action $(0 - A \rhd A : k_{AV})$ où l'entité A communique à elle même sa connaissance initiale (la clé partagée k_{AV}). Ensuite, la seconde partie de la formule $(\mathscr{S}(v_1) \wedge \mathscr{S}(v_2))$ transforme l'action $(1 - V \rhd A : \{v_1\}_{k_{AV}})$ en l'action $(1 - V \rhd A : v_1)$ étant donné que la trace contient désormais l'action $(0 - A \rhd A : k_{AV})$. La trace résultante contient ainsi l'action $(1 - V \rhd A : v_1)$, qui constitue l'élément recherché par la formule récursive $\mathscr{S}(v_1)$. Par conséquent, cette trace ne satisfait pas la formule $\mathscr{S}(v_1)$ étant donné que sa spécification emploie la négation ($\neg$). Les modifications successives apportées à la trace t par la formule $\mathscr{F}(A, k_{AV}, L_c, T_1)$ sont données ci-dessous :

$$t \quad \looparrowright \quad \begin{cases} (0 - A \rhd A : k_{AV}). \\ (1 - V \rhd A : \{v_1\}_{k_{AV}}). \\ (1 - A \lhd V : \{v_1\}_{k_{AV}}) \end{cases} \quad \looparrowright \quad \begin{cases} (0 - A \rhd A : k_{AV}). \\ (1 - V \rhd A : v_1). \\ (1 - A \lhd V : \{v_1\}_{k_{AV}}) \end{cases}$$

La trace t ne satisfait pas la formule $\mathscr{F}$ ($t \not\models \mathscr{F}(A, k_{AV}, L_c, T_1)$). Nous en concluons donc que le protocole P ne garantit pas la propriété d'équité.

Version simplifiée. La spécification des propriétés de sécurité dans la logique ADM est quelque peu compliquée. Dans cette section, nous proposons une méthode permettant de simplifier la tâche de la spécification de propriétés. L'idée est de recourir à un système de réécriture TRS (*Term Rewriting System*) afin d'opérer certaines opérations sur la trace avant sa vérification. Nous dénotons par $t_{\downarrow S}$ la forme normale de la trace obtenue par application successive des règles du système de réécriture S. Par exemple, si le système S est composé de l'unique règle $decompose$ ($S = TRS(decompose)$), alors la forme normale de la trace $t_{\downarrow S}$ est celle où tous les messages composés dans la trace t sont décomposés. Cette règle est très utile dans la spécification des propriétés de sécurité des protocoles, étant donné qu'elle permet de porter le raisonnement sur des messages atomiques.

$$(\boldsymbol{decompose}) \quad x_a.(x_i - X_s \rhd X_r : x_{m_1}, x_{m_2}).x_b \rightarrow x_a.(x_i - X_s \rhd X_r : x_{m_1}).(x_i - X_s \rhd X_r : x_{m_2}).x_b$$

De manière similaire, on définit les règles de réécriture relatives au chiffrement symétrique :

$$(\boldsymbol{symdec_1}) \quad x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : \{x_m\}_{x_k}).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_k).x_c \rightarrow x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_m).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_k).x_c$$

$$(\boldsymbol{symdec_2}) \quad x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_k).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : \{x_m\}_{x_k}).x_c \rightarrow x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_k).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_m).x_c$$

Le système $S = TRS(decompose, symdec_1, symdec_2)$ est terminant et confluant [1]. Par conséquent, le système est convergent, et toute forme normale dérivée à partir de ces règles de réécriture est unique.

L'usage du système de réécriture S simplifie la spécification de la propriété d'équité. En effet, la formule $\mathscr{F}$ se réduit maintenant à la forme suivante :

$$\mathscr{F}(I, K, L_c, T) \quad \equiv \quad \bigwedge_{v \in L_c} (\neg \langle x_a.(x_i - X_s \rhd X_r : \boldsymbol{v}).x_b \curvearrowright \epsilon \rangle\, tt)$$

Une trace t satisfait la propriété d'équité si la formule $\mathscr{F}(I, K, L_c, T)$ est satisfaite par la forme normale de trace $t'_{\downarrow S}$ avec :

$$\begin{cases} t' = (t_0.t)_{\downarrow S'} \\ S' = TRS(deadline) \end{cases}$$

Le système de réécriture S' est celui permettant de supprimer, au moyen de la règle $deadline$, les actions ayant lieu après l'événement $d(T)$:

$$(\boldsymbol{deadline}) \quad x_1.d(\boldsymbol{T}).x_2 \rightarrow x_1$$

Version adaptée au protocole FOO. Adapter la spécification de la propriété d'équité au protocole FOO revient à définir les règles de réécriture associées aux primitives cryptographiques utilisées dans le protocole. Le système de réécriture S, définit précédemment, est étendu avec les règles suivantes :

$$(\boldsymbol{unsign}) \quad x_a.(x_i - X_s \rhd X_r : \|x_m\|_{x_k}).x_b \rightarrow x_a.(x_i - X_s \rhd X_r : x_m).x_b$$

$$(\boldsymbol{unblind}) \quad x_a.(x_i - X_s \rhd X_r : \mathscr{B}(x_m, x_k)).x_b \rightarrow x_a.(x_i - X_s \rhd X_r : \{x_m\}_{x_k}).x_b$$

$$(\boldsymbol{anonymous}) \quad x_a.(x_i - X_s \trianglerighteq X_r : x_m).x_b \rightarrow x_a.(x_i - X_s \rhd X_r : x_m).x_b$$

$$(\boldsymbol{private}) \quad x_a.(x_i - X_s \blacktriangleright I : x_m).x_b \rightarrow x_a.(x_i - X_s \rhd I : x_m).x_b$$

La règle $unsign$ permet de transformer les messages $\|x_m\|_{x_k}$ en messages x_m. En effet, dans notre modélisation, nous faisons l'hypothèse que la signature d'un message intègre en elle-même le message en clair. La règle $unblind$ permet de transformer les messages $\mathscr{B}(x_m, x_k)$ en messages $\{x_m\}_{x_k}$. Il s'agit ici d'une simple translation de format, qui couplée avec les règles $symdec_1$

1. La terminaison et la confluence du système de réécriture S peuvent être démontrées en ayant recours aux techniques de preuves détaillées dans [DJ90].

et $symdec_2$, permet de retrouver le message en clair à partir de son aveuglement. Ensuite, la règle $anonymous$ est également incluse dans le système de réécriture S, afin de permettre la décomposition et le déchiffrement des messages transmis via les canaux anonymes. Finalement, nous avons rajouté la règle $private$ étant donné que nous avons modélisé la génération de *nonces* dans le protocole FOO par l'intermédiaire de messages transitant via des canaux privés. Par conséquent, si la propriété d'équité est définie face à un observateur I, alors il faut prendre en considération les messages adressés à ce dernier via les canaux privés. La règle $private$ est ainsi incluse dans le système S afin de permettre aussi la décomposition et le déchiffrement des messages transmis à l'observateur I via les canaux privés.

Il est à noter que nous devons également adapter la règle $deadline$ au protocole FOO. Pour cela, il suffit de substituer l'action abstraite $d(T)$ par l'action $(3 - deadline(C, T_2))$ étant donné que cette dernière dénote la fin de la phase de vote dans le protocole FOO.

Version générale. La forme générale de la spécification de la propriété d'équité peut être obtenue en étendant le système de réécriture S avec les règles modélisant le pouvoir d'analyse de l'intrus (voir section 3.2). Par exemple, le système S peut être étendu avec les règles relatives aux chiffrement asymétrique probabiliste. Cependant, une attention particulière doit être accordée quant à la convergence du système de réécriture. Autre point à considérer dans la généralisation de la propriété, est le passage d'un simple adversaire I à une coalition d'entités malicieuses T. La spécification de la propriété d'équité face à un ensemble d'adversaires T peut être obtenue en intégrant dans l'ensemble K la connaissance initiale de chaque adversaire $I \in T$, et en dupliquant la règle de réécriture $private$ pour chacune de ces entités.

Notre formalisation de la propriété d'équité repose sur la confidentialité des votes soumis, c'est à dire aucun vote ne peut être déduit avant la fin de la phase de vote. Notre spécification rejoint en ce sens celle proposée par Kremer et Ryan dans [KR05], et définie comme une propriété d'accessibilité dans l'algèbre du π-calcul appliqué. Cependant, comme remarqué par ces mêmes auteurs, un vote peut être déduit sans avoir recours au déchiffrement. Si on prend pour exemple le protocole de vote naïf P, en supposant cette fois-ci que les votes sont chiffrés au moyen de la clé publique de l'autorité A ($pub(k_A)$) au lieu de la clé partagée k_{AV}, alors même une entité externe I serait capable d'en déduire le vote v à partir de $\{\{v\}\}_{pub(k_A)}$, et ce, sans avoir recours au déchiffrement. En effet, si les choix possibles v_1 et v_2 sont connus de l'observateur I, et si ce dernier détient également en sa connaissance la clé publique $pub(k_A)$, alors I serait capable de deviner la valeur du vote émis par le votant. Pour ce faire, l'observateur I construit à son tour un bulletin (par exemple v_1) chiffré au moyen de la clé $pub(k_A)$, et le compare ensuite au vote observé. Si les deux bulletins coïncident, alors le choix de I est le même que celui observé. Dans la cas contraire, il s'agit de l'autre possibilité de choix (v_2). Une formalisation des attaques par dictionnaire a été proposée par Delaune et Jaquemard dans [DJ04] où le modèle de Dolev-Yao est étendu par des règles relatives à la prédiction et l'évaluation des termes. Dans notre formalisation de la propriété d'équité, les règles de réécriture modélisent le pouvoir d'analyse de l'observateur I, et par conséquent, il serait possible d'étendre ces règles en s'inspirant de celles proposées dans [DJ04].

4.4 Anonymat

La propriété d'anonymat signifie que le lien entre le votant V et son $v \in L_c$ ne peut être établi par une tierce personne I ayant comme connaissance initiale l'ensemble des messages K. En partant de cette définition, la propriété d'anonymat est représentée par $\mathscr{A}(V,I,K,L_c)$, et sa spécification dans la logique ADM est donnée par :

$$\mathscr{A}(V,I,K,L_c) \equiv \bigwedge_{\boldsymbol{v}\in L_c} (\langle x_a.(x_i - \boldsymbol{V} \rhd X_r : \boldsymbol{v}).x_b \looparrowright \epsilon\rangle\, tt \rightarrow \bigvee_{v'\in L_c\setminus\{v\}} \langle x_a.(x_i - \boldsymbol{V} \rhd X_r : \boldsymbol{v'}).x_b \looparrowright \epsilon\rangle\, tt)$$

Une instance de l'action $(x_i - V \rhd X_r : v)$ dans la trace à analyser, représente l'envoi en clair du vote v par le votant V. Maintenant, une trace t satisfait la formule $\mathscr{A}(V,I,K,L_c)$ si la trace ne contient pas une action $(x_i - V \rhd X_r : v)$, ou bien lorsque celle-ci est présente, il faut qu'il y ait une seconde action $(x_i - V \rhd X_r : v')$ permettant de confondre le choix de v par un vote $v' \neq v$, ou vice versa.

Comme paramétrée par la formule $\mathscr{A}(V,I,K,L_c)$, la propriété d'anonymat est définie face à un adversaire I dont le rôle est de faire jouer son pouvoir de déduction, et vérifier ensuite s'il est capable d'atteindre la conclusion unique $(x_i - V \rhd X_r : v)$. La formule doit être donc vérifiée, comme pour le cas de la propriété d'équité, par rapport à une forme normale de la trace incluant la connaissance initiale de l'observateur I, et dérivée à partir d'un système de réécriture S modélisant les capacités de déduction de I. Par exemple, le système doit inclure la règle $decompose$ étant donné que notre spécification de l'anonymat porte le raisonnement sur l'envoi de messages atomiques $v \in L_c$. De manière générale, les règles de réécriture à définir dépendent du schéma de vote, et plus particulièrement des primitives cryptographiques utilisées dans le protocole. Dans ce qui suit, nous discutons des règles à définir pour les schémas de vote utilisant la signature en aveugle et ceux utilisant le chiffrement homomorphe, et appliquons ensuite notre spécification aux protocoles de vote A et B présentés en section 1.4.2 (voir page 17).

Dans les schémas utilisant la signature en aveugle, le vote est le plus souvent transmis par l'intermédiaire d'un canal anonyme. Dans sa forme la plus simple, l'action de vote se présente dans la trace par $(i - V \trianglerighteq C : v)$ représentant l'envoi par V du vote v à l'entité C à l'étape i du protocole. Au vu de cet événement, l'observateur en déduit la valeur du vote, mais n'est pas apte à remonter à l'identité de l'émetteur. Cependant, si le protocole est mal conçu, l'information transmise peut renseigner sur l'identité de l'émetteur, et le canal anonyme ne constitue plus de barrière de sécurité dans ce cas. À titre d'exemple, dans le protocole FOO, si on suppose que la phase d'ouverture des bulletins n'est plus assurée par un canal anonyme mais par un canal publique, alors l'observateur I serait capable d'en déduire le vote associé à un votant V malgré que le vote soit transmis via un canal anonyme. En effet, si le votant transmet sa clé r de manière publique lors de la phase d'ouverture des bulletins, alors l'observateur I serait capable d'associer le message $\|\{v\}_r\|_{k_A}$ (et par conséquent, le vote v) au votant V étant donné qu'il est le seul a pouvoir construire le message $\{v\}_r$.

Adapter la spécification de la propriété d'anonymat aux schémas utilisant la signature en aveugle, se résume donc à transformer dans la trace les actions de type $(x_i - X_s \trianglerighteq X_r : x_m)$ en $(x_i - X_s \rhd X_r : x_m)$, et vérifier ensuite si ces transformations, couplées à celles relatives à la décomposition et au déchiffrement des messages, conduit à la trace mettant en évidence de manière unique le choix émis par V (la trace contient une et une seule action $(x_i - V \rhd X_r : v), v \in L_c$).

Une trace t satisfait la spécification de la propriété d'anonymat si la formule $\mathscr{A}(V,I,K,L_c)$ est satisfaite par la forme normale de la trace $t' = (t_0.t)_{\downarrow S}$ où t_0 désigne le bout de trace incluant la connaissance initiale de l'adversaire I, et S le système de réécriture qui intègre l'ensemble des règles de modification de la trace caractérisant le pouvoir de déduction de l'intrus.

Il est bien difficile de construire le système S pour les protocoles utilisant la signature en aveugle, étant donné que l'on ne connait pas l'ensemble des règles de déduction permettant d'associer un message anonyme à son initiateur. Cependant, cette tâche est quelque peu "simplifiée" dans le cas des schémas utilisant le chiffrement homomorphe. Dans ce cas, l'identité du votant est connu, et son vote est le plus souvent protégé par le système de chiffrement employé. Là encore, l'anonymat des votants pourrait être violé si l'adversaire I est capable de dériver le vote v à partir de sa version chiffrée. Dans ce qui suit, nous analysons et détaillons la spécification de l'anonymat pour chacun des protocoles A et B introduits en section 1.4.2.

Anonymat dans le protocole A. Dans le protocole A, les votes sont chiffrés au moyen d'un algorithme asymétrique probabiliste. Le message $\{\{v\}\}^k_{pub(k_G)}$ représente donc le format des bulletins émis, où k désigne le paramètre de sécurité aléatoire, et $pub(k_G)$ la clé publique, dont la clé privée associée k_G est partagée par l'ensemble des autorités $G = \{A_1,\ldots,A_n\}$. Les bulletins sont transmis via un canal publique, et les autorités coopèrent ensuite, en exploitant les propriétés du chiffrement homomorphe, afin de calculer le résultat de l'élection.

Comme expliqué précédemment, l'anonymat d'un votant V serait violé si l'adversaire est capable de déduire le vote v à partir de sa version chiffrée $\{\{v\}\}^k_{pub(k_G)}$. Le système de réécriture à définir, doit donc intégrer les règles relatives au chiffrement asymétrique probabiliste que nous définissons ci-après :

$$(\boldsymbol{asymdec_1}) \quad x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : \{\{x_m\}\}^{x_n}_{x_k}).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_n).x_c \rightarrow x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_m).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_n).x_c$$

$$(\boldsymbol{asymdec_2}) \quad x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_n).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : \{\{x_m\}\}^{x_n}_{x_k}).x_c \rightarrow x_a.(x_{i_1} - X_{s_1} \rhd X_{r_1} : x_n).x_b.(x_{i_2} - X_{s_2} \rhd X_{r_2} : x_m).x_c$$

Le système S est complété ensuite par les règles $decompose$ et $private$ définies précédemment. Une trace t satisfait maintenant la propriété d'anonymat, si la formule $\mathscr{A}(V,I,K,L_c)$ est satisfaite par la forme normale de la trace t' telle que :

$$\begin{cases} t' = (t_0.t)_{\downarrow S} \\ S = TRS(decompose, private, asymdec_1, asymdec_2) \end{cases}$$

Il est à noter que dans le protocole A, tous les votes sont chiffrés par la même clé publique $pub(k_G)$. Si la clé privée associée (k_G) venait à être divulguée, alors l'anonymat de l'ensemble des votants serait compromis. Par conséquent, il faudrait compléter notre spécification de la propriété d'anonymat par une propriété caractérisant la confidentialité de la clé privée k_G. Cette propriété peut être exprimée de manière similaire à celle de l'équité :

$$\mathscr{A}(L_v,I,K,k_G) \quad \equiv \quad [x \not\mapsto t_0.x]\mathscr{S}(k_G)$$

Où $\mathscr{S}$ représente la spécification de la propriété de confidentialité (voir Table 4.3). À noter que celle-ci doit être adaptée aux règles de réécriture définies précédemment.

Rappelons cependant, que la clé k_G est partagée en n entités. Chaque autorité A_j dispose uniquement d'un fragment s_j de la clé k_G. Selon le schéma de partage de secret (t,n), si t fragments viendraient à être connus, la clé k_G pourrait alors être reconstruite. Afin de formaliser cet aspect, nous devons raffiner la propriété $\mathscr{S}(k_G)$. Soit A_{sh} un ensemble contenant C_n^t listes distinctes, chacune composée de t fragments du secret, et soit L_{sh} un ensemble de t fragments du secret ($L_{sh} \in A_{sh}$), Maintenant, la formule caractérisant la confidentialité de la clé privée k_G peut être raffinée comme suit :

$$\begin{array}{lcl} \mathscr{S}(k_G) & \equiv & \bigwedge\limits_{L_{sh} \in A_{sh}} \mathscr{S}(L_{sh}) \\ \mathscr{S}(L_{sh}) & \equiv & \bigvee\limits_{s \in L_{sh}} \mathscr{S}(s) \end{array}$$

Anonymat dans le protocole B. Dans le protocole B, le votant fait appel à un schéma de partage de secret (t,n) afin de partager son vote v parmi n autorités. Si n fragments du vote venaient à être révélés, l'anonymat du votant serait alors violé. Par conséquent, la propriété d'anonymat se ramène là encore à une propriété caractérisant le secret des fragments constituant le vote. La spécification de la propriété d'anonymat dans le protocole B est donc similaire à celle définie précédemment et relative à la confidentialité de la clé privée k_G.

Notre formalisation de la propriété repose principalement sur des règles de réécriture modélisant le pouvoir d'analyse de l'adversaire I face auquel est définie la propriété d'anonymat. Plus précisément, ces règles de réécriture permettent de transformer la trace à analyser en fonction des déductions faites par l'observateur I. L'objectif étant de vérifier par la suite, si la trace résultante contient l'action mettant en évidence le choix effectué par un votant V. Cependant, il est souvent bien difficile d'établir un jeu complet de règles de réécriture permettant d'aboutir à l'association liant le votant V à son choix v lorsque l'anonymat du votant V est compromis. À ce titre, notre spécification de l'anonymat ne peut être universelle, et sa portée est par conséquent moins large que celle exprimée en terme de relations d'équivalence (équivalences observationnelles).

La propriété d'anonymat a été définie tout au long de cette section face à un simple observateur I. Il est toutefois possible, comme pour le cas de la propriété d'équité, de généraliser la spécification à une coalition d'entités malicieuses.

4.5 Sans-Reçu

La propriété sans-reçu signifie que le votant ne peut disposer d'un reçu permettant de prouver à une tierce personne le contenu de son bulletin, même si le votant coopère en divulguant les paramètres secrets (clés, *nonces*) sur lesquels reposent la confidentialité de son vote. La propriété sans-reçu peut être vue ainsi comme si le votant était anonyme face à lui même.

À titre d'exemple, si la spécification de la propriété d'anonymat dans le protocole FOO est donnée par la formule $\mathscr{A}(V,I,K,L_c)$, alors la spécification de la propriété sans-reçu s'obtient par la formule $\mathscr{A}(V,V,K_V,L_c)$, où K_V désigne la connaissance initiale propre au votant V :

$$\mathscr{X} \equiv \mathscr{A}(V, V, K_V, L_c)$$

Il est clair que le protocole FOO ne satisfait pas la propriété sans-reçu. En effet, les *nonces* r et b étant désormais associés au votant V, il s'ensuit que le vote émis anonymement à l'étape 3, ne peut être attribué qu'au votant V. Ceci se traduit par la transformation dans la trace analysée de l'action $(3-V \trianglerighteq C : \|\{v\}_r\|_{k_A})$ en $(3-V \rhd C : \|\{v\}_r\|_{k_A})$, qui évoluera ensuite en $(3-V \rhd C : v)$ étant donné que le paramètre r est connu par le votant V. Il s'agit là de l'action mettant en évidence le choix effectué par le votant V. Par conséquent, la propriété sans-reçu est non satisfaite.

4.6 Conclusion

Dans ce chapitre, nous avons montré que la logique ADM constitue un élément de choix pour la spécification des propriétés des protocoles de vote électronique. En effet, la logique est dotée de modalités temporelles et existentielles, et permet la spécification de formules récursives. Autre avantage de la logique est sa linéarité, qui couplée avec la récursivité, offre la possibilité de spécifier des formules difficilement exprimables, voire impossibles dans d'autres formalismes. À titre d'exemple, les formules ayant trait au comptage des actions dans la trace ne sont pas exprimables dans une logique telle que le μ-calcul. Or, cette faculté constitue un élément clé pour la spécification de plusieurs propriétés de protocoles de vote électronique (démocratie, exactitude).

Notre analyse des protocoles de vote se traduit par la spécification de propriétés au moyen de la logique ADM, et de la vérification ensuite des formules exprimées par rapport à des traces du protocole. Dans notre modélisation, l'ensemble des traces valides à vérifier est infini. Ceci est dû à un intrus puissant capable de construire et d'insérer arbitrairement une infinité de messages. À cet effet, il nous est impossible d'affirmer la sécurité d'un protocole lorsqu'une propriété est réellement garantie. Cependant, lorsque la faille existe, elle est mise en évidence dans notre approche par une trace qui ne satisfait pas une propriété de sécurité donnée ($t \not\models \phi$). C'est le cas de la trace t de la Table 3.11 qui ne satisfait pas la propriété d'éligibilité ($t \not\models \mathscr{E}$). Il en résulte que le protocole FOO ne garantit pas cette propriété de sécurité. La trace t met en évidence un administrateur malicieux qui a été capable d'injecter un bulletin frauduleux. De ce fait, la trace t constitue également une violation de la propriété d'exactitude. Ceci peut être vérifiée par $t \not\models \mathscr{C}$.

Bien que notre spécification des propriétés d'anonymat et de sans-reçu n'a pas été appliquée au protocole FOO, cette formalisation à aider à en déduire que ce protocole ne garantit pas la seconde propriété. Il s'ensuit que le protocole ne garantit pas également la propriété de résistance à la coercition, étant donné que cette dernière constitue la version robuste de la propriété sans-reçu.

Finalement, bien que la spécification des propriétés relatives à la vérifiabilité n'a pas été abordée dans ce chapitre, il nous est possible d'affirmer indirectement que le protocole FOO ne garantit pas la propriété de vérifiabilité universelle. En effet, cette propriété implique que les résultats publiés doivent correspondre exactement aux choix des électeurs (la propriété d'exactitude doit être satisfaite), et que ce résultat doit être vérifiable par chaque votant, au moyen des éléments exposés publiquement par le protocole à son environnement.

Tout au long de ce chapitre, nous avons présenté une formalisation dans logique ADM d'une sélection de propriétés des protocoles de vote électronique. Notre formalisation des propriétés a été orientée particulièrement sur le protocole FOO. Ceci a permis de dégager une analyse de ce protocole présenté en Table 4.5, et montrer que ce dernier ne garantit pas certaines des propriétés des protocoles de vote. La violation d'une propriété de sécurité (exprimée par une formule ϕ) se confirme dans ce cas par l'existence d'au moins une trace t du protocole analysé qui ne satisfait pas la formule ϕ ($t \not\models \phi$). Dans le chapitre qui suit, nous proposons de développer un outil permettant d'automatiser la tâche de la vérification des propriétés spécifiées par rapport à des traces du protocole analysé. L'outil développé permettra notamment de confirmer les résultats présentés en Table 4.5.

TABLE 4.5 – Sécurité du protocole FOO

Propriété	**Spécification**	**Résultat**
Droit d'expression	$\mathscr{R}$	-
Éligibilité	$\mathscr{E}$	non satisfaite[1]
Non réutilisabilité	$\mathscr{N}$	-
Exactitude	$\mathscr{C}$	non satisfaite[1]
Équité	$\mathscr{F}$	-
Anonymat	$\mathscr{A}$	-
Sans-reçu	$\mathscr{X}$	non satisfaite
Résistance à la coercition	-	non satisfaite
Vérifiabilité individuelle	-	-
Vérifiabilité universelle	-	non satisfaite

[1] En présence d'un administrateur corrompu.

Chapitre 5

Vérification automatique des propriétés des protocoles de vote électronique

Il existe une variété d'outils pour la vérification des protocoles cryptographiques. Nous pouvons citer entre autres ProVerif [Bla01], AVISPA [ABB+05], FDR [For05, Ros94], Murφ [DDHY92] et NPA [Mea96]. Dans ce chapitre, nous proposons à notre tour un outil permettant la vérification automatique des protocoles cryptographiques en général, et des protocoles de vote électronique en particulier. Cet outil aura pour tâche de permettre à l'utilisateur final de spécifier ses propriétés et de les vérifier ensuite par rapport à une trace ou un ensemble de traces. L'implémentation d'un tel outil est quelque peu simplifié dans l'approche que nous avons adoptée. En effet, la logique ADM est dotée d'un système de preuve basé sur la méthode des tableaux permettant de dériver, par l'intermédiaire d'un ensemble de règles d'inférence, la preuve associée à la vérification de la relation de satisfaction $t \models \phi$. Automatiser le processus de vérification des formules, revient ainsi à implémenter le jeu de règles constituant le système de preuve.

Ce chapitre présente l'outil ADM-Checker consistant en l'implémentation de ces règles. Cet outil a été conçu de manière à permettre à l'utilisateur final de spécifier ses propriétés de sécurité, et de les vérifier ensuite conformément à la syntaxe et à la sémantique de la logique ADM. L'outil ADM-Checker est muni d'une interface graphique et d'assistants permettant de guider l'utilisateur dans la spécification de ses formules. Il est également doté d'un analyseur syntaxique permettant de notifier l'utilisateur des éventuels erreurs de spécification. Finalement, ADM-Checker dispose d'un vérificateur formel qui opère à la vérification des propriétés de sécurité selon les règles du système de preuve.

Dans ce qui suit, nous donnons en premier lieu un aperçu du système de preuve de la logique ADM. Les règles de ce système seront mises en œuvre sur deux exemples illustratifs permettant ainsi de mieux assimiler le mode opératoire du mécanisme de vérification. Nous introduisons ensuite l'outil ADM-Checker. Nous commençons par donner une vue globale de l'outil développé avant de passer à son architecture et aux détails techniques liés à ses différentes composantes.

5.1 Système de preuve basé sur les tableaux

5.1.1 Présentation

La logique ADM est accompagnée d'un système de preuve basé sur la méthode des tableaux [GLM97] permettant de vérifier si une trace t satisfait ou non une formule ϕ. Ce système, présenté en Table 5.1, est construit sur la base d'un ensemble de règles d'inférence opérant sur des séquents de la forme $H, b, e, \sigma \vdash t \in \phi$. L'élément H est une table qui associe à chaque variable propositionnelle X, un ensemble de traces. Il joue un rôle dans le traitement des formules récursives. Ensuite, le paramètre b est une variable prise dans l'ensemble $\{\epsilon, \neg\}$. Son état indique si l'on a affaire à une formule employant la négation ou pas. Par la suite, le paramètre e désigne un environnement, et est introduit afin d'associer une sémantique à la variable propositionnelle X. Comme pour la cas de H, il consiste en une table qui fait correspondre à la variable propositionnelle X, un ensemble de traces. Finalement, σ représente une substitution, t une trace, et ϕ une formule.

TABLE 5.1 – Système de preuve basé sur les tableaux

$R_\neg$	$\dfrac{H, b, e, \sigma \vdash t \in \neg\phi}{H, \neg b, e, \sigma \vdash t \in \phi}$	
$R_\wedge$	$\dfrac{H, b, e, \sigma \vdash t \in (\phi_1 \wedge \phi_2)}{H, b_1, e, \sigma \vdash t \in \phi_1 \qquad H, b_2, e, \sigma \vdash t \in \phi_2}$	$b_1 \times b_2 = b$
R_ν	$\dfrac{H, b, e, \sigma \vdash t \in \nu X.\phi}{H[X \mapsto H(X) \cup t], b, e, \sigma \vdash t \in \phi[\nu X.\phi / X]}$	$t \notin H(X)$
$R_{[-]}$	$\dfrac{H, b, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi}{H, b_1, e, \sigma_1 \circ \sigma \vdash p_2\sigma\sigma_1 \in \phi \quad \ldots \quad H, b_n, e, \sigma_n \circ \sigma \vdash p_2\sigma\sigma_n \in \phi}$	C

Avec $$C = \begin{pmatrix} \{\sigma_1, \ldots, \sigma_n\} = \{\sigma' \mid p_1\sigma\sigma' = t\} \neq \emptyset \\ et \\ b_1 \times \ldots\ldots\ldots\ldots \times b_n = b, n > 0 \end{pmatrix}$$

La vérification de la relation de satisfaction $t \models \phi$ s'effectue par application des règles de la Table 5.1 en prenant pour départ le séquent initial $\theta_0 = (\emptyset, \epsilon, \emptyset, \emptyset \vdash t \in \phi)$. Le séquent est ensuite transformé en fonction des opérateurs rencontrés dans la formule ϕ, donnant possiblement lieu à plusieurs branchements. La structure en arbre résultante est dite tableau, et un séquent feuille de l'arbre est dit à succès (*successful leaf*) s'il remplit l'une des conditions de la partie gauche de la Table 5.2. Inversement, le séquent est à échec (*unsuccessfull leaf*) s'il satisfait l'une des conditions de la partie droite de cette même table. Maintenant, la trace t satisfait la formule ϕ, s'il est possible de dériver à partir du séquent racine θ_0, un tableau où tous les séquents feuille sont à succès. Dans ce cas, le tableau généré est dit à succès (*successful tableau*).

TABLE 5.2 – Système de preuve - Conditions d'arrêts

Succès	Échec
$\theta = (H, \epsilon, e, \sigma \vdash t \in X)$ et $t \in e(X)$	$\theta = (H, \epsilon, e, \sigma \vdash t \in X)$ et $t \notin e(X)$
$\theta = (H, \neg, e, \sigma \vdash t \in X)$ et $t \notin e(X)$	$\theta = (H, \neg, e, \sigma \vdash t \in X)$ et $t \in e(X)$
$\theta = (H, \epsilon, e, \sigma \vdash t \in \nu X.\phi)$ et $t \in H(X)$	$\theta = (H, \neg, e, \sigma \vdash t \in \nu X.\phi)$ et $t \in H(X)$
$\theta = (H, \epsilon, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi)$ et $\{\sigma' \mid p_1\sigma\sigma' = t\} = \emptyset$	$\theta = (H, \neg, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi)$ et $\{\sigma' \mid p_1\sigma\sigma' = t\} = \emptyset$

Le système de preuve définit une règle pour chaque opérateur de base de la logique ADM. Il s'agit de règles d'inférence qui s'interprètent de la manière suivante :

– La règle $R_\neg$ permet la vérification des formules qui sont précédées de la négation ($\neg$). Cette règle repose sur l'idée simple consistant à vérifier la formule $\neg\phi$ en répercutant le raisonnement par rapport à ϕ. En effet, vérifier si une trace t satisfait ou non la formule $\neg\phi$, revient à vérifier d'abord si la trace t satisfait la formule ϕ, et conclure ensuite par rapport à la formule $\neg\phi$. Plus précisément, vérifier si le séquent $H, b, e, \sigma \vdash t \in \neg\phi$ aboutit à un tableau à succès, se réduit à vérifier si le séquent $H, \neg b, e, \sigma \vdash t \in \phi$ conduit à son tour à un tableau à succès. À noter que la vérification des formules $\neg\phi$ est assurée particulièrement par le paramètre b dont le rôle est de mémoriser si l'on a affaire à la négation ou pas. L'opération $\neg b$ est définie de la manière suivante :

b	$\neg b$
ϵ	$\neg$
$\neg$	ϵ

– La règle $R_\wedge$ est relative à l'opérateur de conjonction. Prouver que le séquent $H, b, e, \sigma \vdash t \in (\phi_1 \wedge \phi_2)$ aboutit à un tableau à succès, revient à prouver que les séquents $H, b_1, e, \sigma \vdash t \in \phi_1$ et $H, b_2, e, \sigma \vdash t \in \phi_2$ mènent tous deux à un tableau à succès, pour des valeurs de b_1 et b_2 vérifiant $b = b_1 \times b_2$. L'opération $b_1 \times b_2$ est définie comme suit :

b_1	b_2	$b_1 \times b_2$
ϵ	ϵ	ϵ
ϵ	$\neg$	$\neg$
$\neg$	ϵ	$\neg$
$\neg$	$\neg$	$\neg$

– La règle R_ν est relative à l'opérateur de récursivité ν, et n'est applicable que si la condition de bord $t \notin H(X)$ est non satisfaite. En effet, dans le cas contraire, il s'agit d'un séquent feuille, et la formule $\nu X.\phi$ est satisfaite par la trace t que si $b = \epsilon$. Les transformations opérées par la règle R_ν sur le séquent $H, b, e, \sigma \vdash t \in \nu X.\phi$ se traduisent par la mise à jour de la

table $H(X)$ avec la trace t, et la transformation de la formule $\nu X.\phi$ en $\phi[\nu X.\phi/X]$. Celle-ci correspond à la formule ϕ où toutes les occurrences de la variable propositionnelle X sont remplacées par $\nu X.\phi$. Ci-après est donné un exemple d'application de la règle R_ν :

$$R_\nu \quad \frac{\emptyset, \epsilon, \emptyset, \emptyset \vdash t \in \nu X.[p_1 \looparrowright p_2]X}{[X \mapsto t], \epsilon, \emptyset, \emptyset \vdash t \in [p_1 \looparrowright p_2]\nu X.[p_1 \looparrowright p_2]X}$$

– La règle $R_{[-]}$ est associée finalement à l'opérateur modal $[-]$, et n'est applicable sur le séquent $\theta = (H, b, e, \sigma \vdash t \in [p_1 \looparrowright p_2]\phi)$ que s'il existe au moins une substitution σ' telle que $p_1\sigma\sigma' = t$. Dans le cas contraire, le séquent θ est un séquent feuille, et la formule $[p_1 \looparrowright p_2]\phi$ est satisfaite par la trace t que si $b = \epsilon$.

5.1.2 Exemples

Afin de mieux assimiler le mode opératoire du système de preuve présenté dans la section précédente, nous proposons de dérouler ses règles sur deux exemples de formule.

Exemple 5.1.1

L'objectif est de vérifier, au moyen du système de preuve, si la trace t satisfait ou non la formule ϕ_{2a} , où t et ϕ_{2a} sont définies comme suit :

$$\begin{cases} t & = & b.a.d.b.a.c \\ \phi_{2a} & = & \neg\langle x_1.a.x_2.a.x_3 \looparrowright \epsilon\rangle tt \end{cases}$$

À noter que la formule ϕ_{2a} représente la forme générale de la spécification de la propriété de non réutilisabilité, et permet de vérifier si une action particulière a n'apparaît pas plus d'une fois dans la trace à analyser.

Le système de preuve définit uniquement des règles pour les opérateurs de base. La vérification de la formule ϕ_{2a} par le système de preuve, nécessite donc de la transformer au préalable au moyen des abréviations de la Table 2.11 (voir page 39) :

$$\neg\langle x_1.a.x_2.a.x_3 \looparrowright \epsilon\rangle tt \quad \equiv \quad [x_1.a.x_2.a.x_3 \looparrowright \epsilon]\neg\nu X.X$$

La preuve débute par le séquent initial $\theta_0 = (\emptyset, \epsilon, \emptyset, \emptyset \vdash t \in \phi_{2a})$, et aboutit à l'arbre de preuve suivant :

$$\cfrac{\cfrac{\cfrac{}{\emptyset, \epsilon, \emptyset, \emptyset \vdash b.a.d.b.a.c \in [x_1.a.x_2.a.x_3 \looparrowright \epsilon]\neg\nu X.X}\, R_{[-]}}{\emptyset, \epsilon, \emptyset, \{x_1 \mapsto b, x_2 \mapsto d.b, x_3 \mapsto c\} \vdash \epsilon \in \neg\nu X.X}\, R_\neg}{[X \mapsto \epsilon], \neg, \emptyset, \{x_1 \mapsto b, x_2 \mapsto d.b, x_3 \mapsto c\} \vdash \epsilon \in \nu X.X}\, R_\nu$$

Le séquent dérivé $\theta_f = ([X \mapsto \epsilon], \neg, \emptyset, \{x_1 \mapsto b, x_2 \mapsto d.b, x_3 \mapsto c\} \vdash \epsilon \in \nu X.X)$ est un séquent feuille à échec étant donné qu'il satisfait les 2 conditions suivantes :

$$\begin{cases} \theta = (H, \neg, e, \sigma \vdash t \in \nu X.\phi) \\ t \in H(X) \end{cases}$$

Nous en concluons donc que la trace $t = b.a.d.b.a.c$ ne satisfait pas la formule ϕ_{2a}.

L'exemple 5.1.1 converge rapidement, et n'illustre pas l'ensemble des règles du système de preuve. À cet effet, nous proposons de décrire le fonctionnement du processus de vérification à travers un second exemple.

Exemple 5.1.2

Soient la trace t et la formule ϕ_{ab} suivantes :

$$\begin{cases} t & = & d.a.b \\ \phi_{ab} & = & \neg\nu X.(\langle x_1.a.x_2 \looparrowright \epsilon\rangle tt \longrightarrow \langle y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3\rangle X) \end{cases}$$

La formule ϕ_{ab} est en premier lieu transformée conformément aux abréviations de la logique ADM :

$$\phi_{ab} \quad \equiv \quad \neg\nu X.\neg(\neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \wedge [y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg X)$$

L'arbre de preuve associé au processus de vérification de la formule ϕ_{ab} par rapport à la trace t est donné en Table 5.3 et complété en Table 5.4. Nous en concluons à travers cette preuve que la trace t ne satisfait pas la formule ϕ_{ab}. En effet, la branche de droite, engendrée par la règle $R_\wedge$ dans la Table 5.3, conduit en Table 5.4 à deux séquents feuille dont l'un est à échec.

5.2 L'outil ADM-Checker

5.2.1 Présentation

Dans le but de permettre la vérification automatique des propriétés des protocoles de vote électronique, nous avons conçu puis développé l'outil ADM-Checker. Il s'agit d'une application Java permettant à l'utilisateur final de spécifier ses propriétés et de les vérifier ensuite par rapport à une trace du protocole analysé. L'outil ADM-Checker est doté d'une interface graphique permettant d'assister l'utilisateur dans la spécification des propriétés et de leurs paramètres associés. Comme tout EDI (Environnement de Développement Interne), ADM-Checker est muni d'un compilateur permettant de vérifier, conformément à la grammaire de la logique ADM, la syntaxe des propriétés spécifiées par l'utilisateur. Si ces formules sont exemptes d'erreurs, elles sont transformées au moyen des abréviations qui accompagnent la forme minimale de la logique ADM. Cette étape prépare à la vérification formelle des formules conformément aux règles du système de preuve présenté dans le section précédente.

L'interface graphique de l'outil ADM-Checker est illustrée par la figure 5.1. Elle dispose d'une barre d'outils et de deux panneaux d'édition. Le panneau central est dédié à la spécification des propriétés qui s'opère par l'intermédiaire d'un ensemble de clauses prédéfinies. Les propriétés spécifiées peuvent ensuite être enregistrées sur le disque comme fichier portant l'extension ".logic". La figure 5.1 illustre le fichier de spécification relatif à la propriété de non réutilisabilité. Le panneau du bas sert quant à lui à l'affichage des informations relatives aux résultats de la vérification syntaxique et formelle des propriétés.

TABLE 5.3 – Arbre de preuve associé à l'exemple 5.1.2 - Partie 1

$$\emptyset,\epsilon,\emptyset,\emptyset \vdash d.a.b \in \neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}$$

$R_\neg$

$$\emptyset,\neg,\emptyset,\emptyset \vdash d.a.b \in \nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}$$

R_ν

$$[X\mapsto\{d.a.b\}],\neg,\emptyset,\emptyset \vdash d.a.b \in \neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}\end{pmatrix}$$

$R_\neg$

$$[X\mapsto\{d.a.b\}],\epsilon,\emptyset,\emptyset \vdash d.a.b \in \begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}\end{pmatrix}$$

$R_\wedge$

Left branch:

$$[X\mapsto\{d.a.b\}],\epsilon,\emptyset,\emptyset \vdash d.a.b \in \neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y$$

$R_\neg$

$$[X\mapsto\{d.a.b\}],\neg,\emptyset,\emptyset \vdash d.a.b \in [x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y$$

$R_{[-]}$

$$[X\mapsto\{d.a.b\}],\neg,\emptyset,\{x_1\mapsto d, x_2\mapsto b\} \vdash \epsilon \in \neg\nu Y.Y$$

$R_\neg$

$$[X\mapsto\{d.a.b\}],\epsilon,\emptyset,\{x_1\mapsto d, x_2\mapsto b\} \vdash \epsilon \in \nu Y.Y$$

R_ν

$$[X\mapsto\{d.a.b\}, Y\mapsto\{\epsilon\}],\epsilon,\emptyset,\{x_1\mapsto d, x_2\mapsto b\} \vdash \epsilon \in \nu Y.Y$$

Right branch:

$$[X\mapsto\{d.a.b\}],\epsilon,\emptyset,\emptyset \vdash d.a.b \in [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}$$

$R_{[-]}$

$$[X\mapsto\{d.a.b\}],\epsilon,\{y_1\mapsto d, y_2\mapsto\epsilon, y_3\mapsto\epsilon\},\emptyset \vdash d \in \neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}$$

$R_\neg$

$$[X\mapsto\{d.a.b\}],\neg,\{y_1\mapsto d, y_2\mapsto\epsilon, y_3\mapsto\epsilon\},\emptyset \vdash d \in \nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}$$

R_ν

$$[X\mapsto\{d.a.b,d\}],\neg,\{y_1\mapsto d, y_2\mapsto\epsilon, y_3\mapsto\epsilon\},\emptyset \vdash d \in \neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}\end{pmatrix}$$

$R_\neg$

$$[X\mapsto\{d.a.b,d\}],\epsilon,\{y_1\mapsto d, y_2\mapsto\epsilon, y_3\mapsto\epsilon\},\emptyset \vdash d \in \begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg\nu X.\neg\begin{pmatrix}\neg[x_1.a.x_2 \nrightarrow \epsilon]\neg\nu Y.Y\\ \wedge \\ [y_1.a.y_2.b.y_3 \nrightarrow y_1.y_2.y_3]\neg X\end{pmatrix}\end{pmatrix}$$

$\vdots$

TABLE 5.4 – Arbre de preuve associé à l'exemple 5.1.2 - Suite

$$
\cdots \quad \cdots \quad \cdots \quad \cdots
$$

$$
R_{\wedge}\ \frac{\left(\begin{array}{c}[X \mapsto \{d.a.b,d\}],\epsilon,\{y_1 \mapsto d, y_2 \mapsto \epsilon, y_3 \mapsto \epsilon\},\emptyset \vdash d \in \\ [y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg\nu X.\neg\end{array}\left(\begin{array}{c}\neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ \neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ [y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg X\end{array}\right)\right)}{\left\{\begin{array}{c}[y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg\nu X. \\ \neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ \neg\left(\begin{array}{c}\neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ [y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg X\end{array}\right)\end{array}\right\}}
$$

$$
R_{\neg}\ \frac{[X \mapsto \{d.a.b,d\}],\epsilon,\{y_1 \mapsto d, y_2 \mapsto \epsilon, y_3 \mapsto \epsilon\},\emptyset \vdash d \in \neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y}{X \mapsto \{d.a.b,d\}],\neg.\{y_1 \mapsto d, y_2 \mapsto \epsilon, y_3 \mapsto \epsilon\},\emptyset \vdash d \in [x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y} \qquad [X \mapsto \{d.a.b,d\}],\epsilon,\{y_1 \mapsto d, y_2 \mapsto \epsilon, y_3 \mapsto \epsilon\},\emptyset \vdash d \in \left\{\begin{array}{c}[y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg\nu X. \\ \neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ \neg\left(\begin{array}{c}\neg[x_1.a.x_2 \looparrowright \epsilon]\neg\nu Y.Y \\ \wedge \\ [y_1.a.y_2.b.y_3 \looparrowright y_1.y_2.y_3]\neg X\end{array}\right)\end{array}\right\}
$$

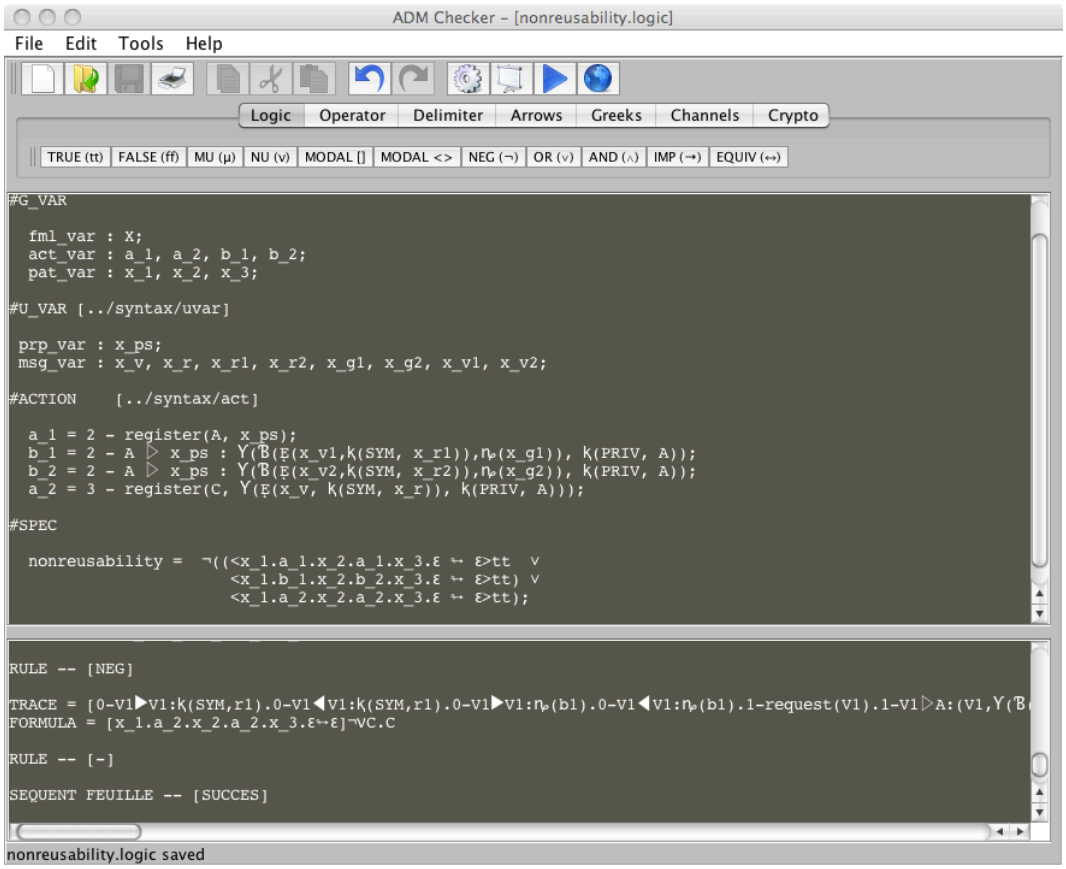


FIGURE 5.1 – ADM-Checker

5.2.2 Architecture

L'architecture de l'outil ADM-Checker est illustrée par la figure 5.2. Elle décrit le processus de fonctionnement de l'outil ADM-Checker partant de la spécification des propriétés jusqu'à leurs vérifications par rapport à des traces. Tout d'abord, l'utilisateur commence par spécifier ses propriétés de sécurité donnant ainsi lieu à un fichier de spécification ".logic". Le vérificateur syntaxique permet ensuite de s'assurer que les propriétés définies par l'utilisateur respectent la grammaire de la logique ADM. Cette étape est assurée par l'appel à des analyseurs construits au moyen des outils Flex et Bison [fle]. Si les propriétés sont exemptes d'erreurs, elles seront transformées par la suite conformément aux abréviations de formules données par la Table 2.11 (voir page 39). Le vérificateur formel prend ensuite le relais et permet de vérifier si le modèle construit à partir d'un fichier de trace (fichier ".trace") satisfait où non les propriétés spécifiées par l'utilisateur. Cette étape est assurée par l'application des règles du système de preuve de la Table 5.1 et produit pour chaque propriété analysée deux résultats possibles : "true" ou "false".

Dans ce qui suit nous nous attardons sur les détails techniques des différentes composantes de l'architecture de l'outil ADM-Checker.

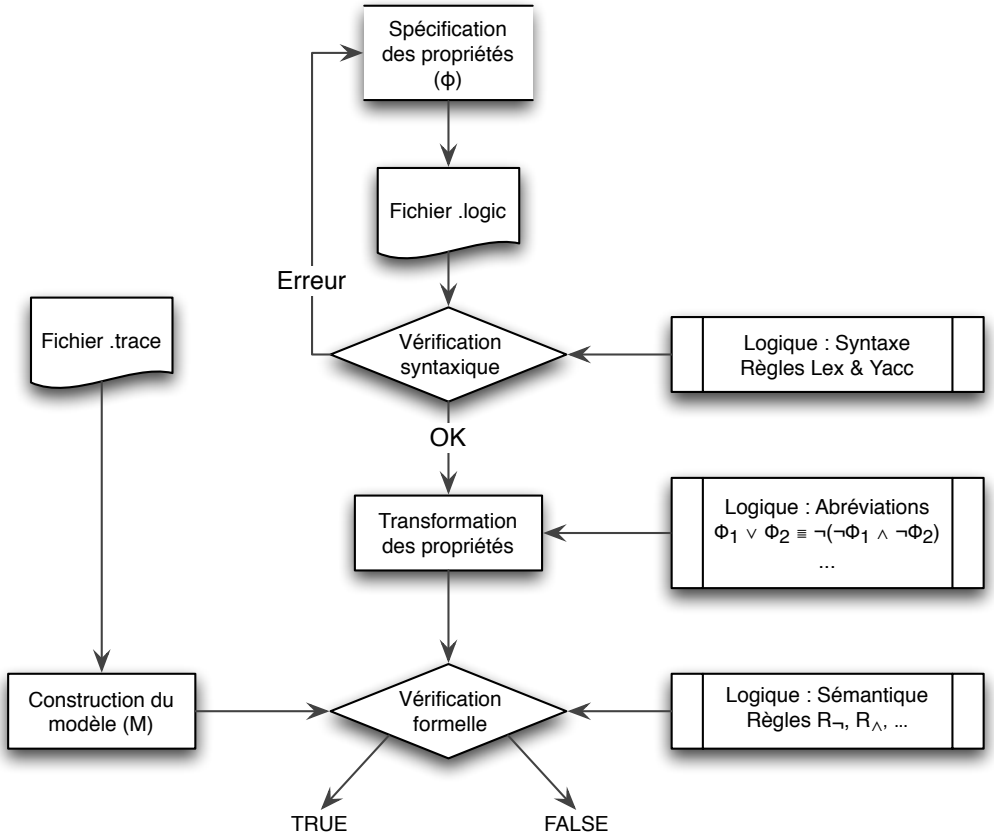


FIGURE 5.2 – Architecture de l'outil ADM-Checker

5.2.3 Spécification des propriétés

L'outil ADM-Checker offre un environnement permettant d'assister l'utilisateur final dans la spécification des propriétés de sécurité et de leurs paramètres associés. Un fichier de spécification porte l'extension ".logic" et doit obéir à la structure de programme donnée par la Table 5.5 qui réglemente la spécification des propriétés par l'intermédiaire de clauses prédéfinies :

TABLE 5.5 – Structure d'un fichier de propriétés ".logic"

#INCLUDE [chemin analyseur syntaxique]

#G_VAR

$$\begin{bmatrix} g\_type_1 : g\_var_1; \\ \vdots \\ g\_type_n : g\_var_n; \end{bmatrix}$$

#U_VAR [chemin analyseur syntaxique]

$$\begin{bmatrix} u\_type_1 : u\_var_1; \\ \vdots \\ u\_type_n : u\_var_n; \end{bmatrix}$$

#ACTION [chemin analyseur syntaxique]

$$\begin{bmatrix} act\_var_1 = action_1; \\ \vdots \\ act\_var_n = action_n; \end{bmatrix}$$

#SPEC

$$\begin{bmatrix} fml\_spec_1 : nom_1; \\ \vdots \\ fml\_spec_n : nom_n; \end{bmatrix}$$

La clause G_VAR. La clause *G_VAR* est utilisée pour définir les variables propres à la logique ADM. Nous distinguons trois types de variables :

- *fml_var* : le type *fml_var* est associé aux variables propositionnelles utilisées dans les formules récursives. À titre d'exemple, si X est une variable de type *fml_var* , alors la formule $\phi = \nu X.X$ est une formule qui respecte la syntaxe de la logique ADM.
- *act_var* : le type *act_var* est associé aux actions incluses dans les *patterns*.
- *pat_var* : le type *pat_var* est associé aux variables "pattern". Par exemple, si x et a désignent respectivement des variables de type *pat_var* et *act_var*, alors le *pattern* $p = x.a$ respecte la syntaxe associée aux *patterns* donnée en Table 2.8 (voir page 38), et peut être employé dans les formules modales $\langle - \rangle$ et $[-]$.

La clause U_VAR. La clause *U_VAR* est utilisée dans la déclaration des variables contenues dans les actions. Ces variables sont dissociées de celles introduites par la clause *G_VAR* dans le but de permettre à l'utilisateur de définir son propre format d'action. Ceci rend l'outil ADM-Checker ouvert à d'autres types d'applications (commerce électronique [ADM03], détection d'intrusions [GTM07, TTM08]), et non spécifique uniquement à l'analyse des protocoles de vote électronique.

Dans le modèle défini dans le chapitre 3, nous avons introduit le format associé aux actions du protocole. Ce format inclut trois variables dont les types associés sont donnés ci-après :

- *stp_var* : le type *stp_var* est utilisé afin de définir des variables "étape".
- *prp_var* : le type *prp_var* est réservé à la déclaration des variables "entité".
- *msg_var* : le type *msg_var* est associé aux variables "message".

À noter que la clause *U_VAR* doit être paramétrée par le chemin vers l'analyseur permettant de s'assurer de la bonne syntaxe des variables déclarées.

La clause ACTION. La clause *ACTION* permet d'attribuer une affectation aux variables de type *act_var*. Par exemple, la déclaration $a_1 = (x_i - X \rhd Y : x_m)$ aura pour effet de substituer toutes occurrences de l'action a_1 dans les formules spécifiées par la valeur $(x_i - X \rhd Y : x_m)$.

Il est à noter que, là encore, la clause doit être paramétrée par le chemin vers l'analyseur syntaxique permettant de s'assurer du respect des éléments définis dans la clause *ACTION*.

La clause SPEC. La clause *SPEC* est réservée pour la spécification des propriétés. À noter qu'un fichier ".logic" peut comporter plusieurs formules. Afin de se référer par rapport à chacune d'elles, un nom significatif est attribuée à chaque formule définie.

La clause INCLUDE. Finalement, le paramètre de la clause *INCLUDE* est optionnelle, et est utilisée par le vérificateur formel afin de s'assurer de la validité des substitutions trouvées.

5.2.4 Vérification syntaxique des propriétés

L'outil ADM-Checker est doté d'un analyseur syntaxique permettant de s'assurer du respect de la structure donnée par la Table 5.5, et de la syntaxe des déclarations opérées au niveau de chaque clause. Cet analyseur est conçu au moyen des outils Flex et Bison. Flex effectue une vérification lexicale et prépare le terrain à Bison dont la tâche principale est la génération d'un *parseur* respectant les règles de production qui lui ont été spécifiées. Des règles ont été définies pour chacune des clauses constituant un fichier ".logic".

L'analyseur syntaxique permet de signaler les erreurs syntaxiques trouvées. Lorsque celles-ci sont absentes, les formules spécifiées sont transformées au moyen des abréviations de la Table 2.11 (voir page 39). Ceci permet de préparer le terrain au vérificateur formel qui ne définit des règles de preuve que pour les opérateurs de base de la logique ADM.

5.2.5 Vérification formelle des propriétés

L'outil ADM-Checker est accompagné d'un vérificateur formel implémentant les règles du système de preuve présenté en section 5.1. Cet analyseur formel opère à la vérification des propriétés, prises une à une, par rapport au modèle construit à partir d'un fichier de trace ".trace". La figure 5.3 illustre le chargement d'une trace valide du protocole FOO. À l'issue de cette étape de vérification, l'outil nous informe du résultat de vérification : un résultat "true" pour une formule donnée ϕ indique que la formule est satisfaite par la trace analysée. Un résultat "false" indique que la formule est non satisfaite par la trace, montrant ainsi une violation de la propriété de sécurité décrite par la formule ϕ. Ce résultat est complété par des informations complémentaires telles que les règles du système de preuve qui ont été exécutées, les substitutions trouvées, etc.

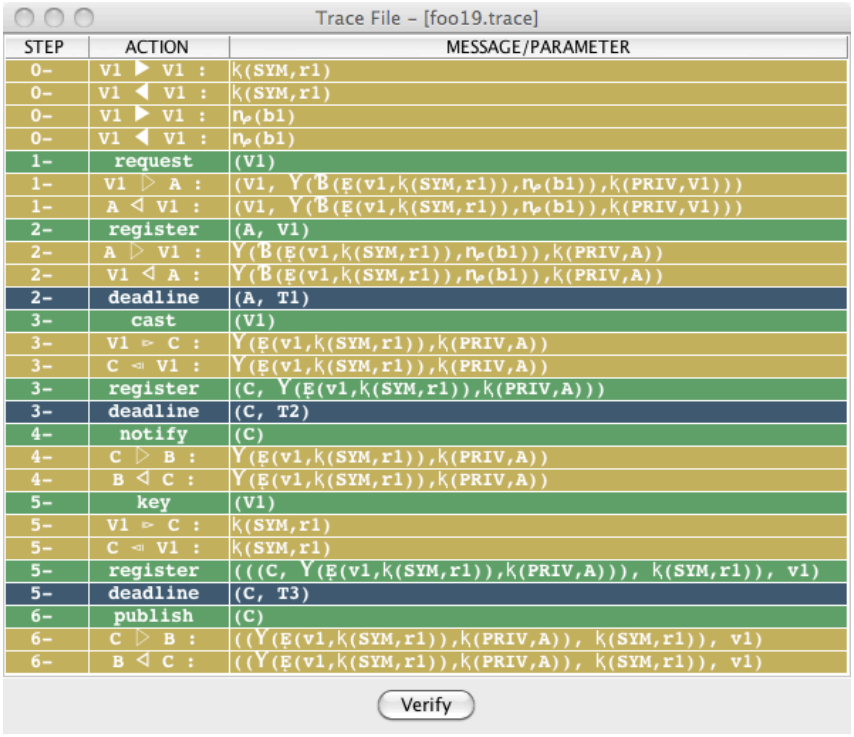


FIGURE 5.3 – Visualisation d'un fichier ".trace" dans l'outil ADM-Checker

5.3 Conclusion

Dans ce chapitre, nous avons montré que l'approche que nous avons adoptée se prête bien à l'automatisation. En effet, le système de preuve a rendu possible l'implémentation d'un outil efficace pour la vérification des propriétés spécifiées au moyen de la logique ADM. L'outil développé, dénommé ADM-Checker, est doté d'un analyseur syntaxique permettant de s'assurer que les formules spécifiées respectent la grammaire de la logique ADM. ADM-Checker dispose également d'un vérificateur formel qui prend en entrée un fichier de propriétés et un fichier de trace, et génère en sortie les résultats de vérification de la relation de satisfaction $t \models \phi$ pour chaque formule ϕ du fichier de propriétés.

L'ensemble des propriétés spécifiées au niveau du chapitre précédent et appliquées au protocole FOO, ont été transcrites dans des fichiers ".logic" puis vérifiées au moyen de l'outil ADM-Checker par rapport à un ensemble de fichiers de trace ".trace". Ces traces concernent le protocole FOO, et ont été construites manuellement par application des règles qui modélisent ce protocole (voir section 3.3.2, page 59). Parmi les traces analysées, figure celle donnée en exemple dans la Table 3.11 (voir page 66). Cette dernière a permis de confirmer certains des résultats présentés dans le chapitre précédent (voir Table 4.5, page 89), et notamment, le non respect par le protocole des propriétés d'éligibilité et d'exactitude dans le cas particulier d'un administrateur corrompu.

Conclusion

L'objectif de cette thèse a été la spécification formelle, et la vérification automatique des principales propriétés des protocoles de vote électronique au moyen de la logique ADM. Nous récapitulons dans ce qui suit nos principales contributions, et discutons de notre apport par rapport aux travaux connexes. Nous concluons ensuite par quelques perspectives.

Bilan

Durant cette thèse nous avons adopté une approche pour l'analyse des protocoles de vote électronique qui s'articule en trois phases : la modélisation des protocoles, la spécification des propriétés, et la vérification automatique des propriétés spécifiées par rapport au modèle du protocole.

Modélisation des protocoles. Nous avons défini en chapitre 3 le modèle par rapport auquel sont vérifiées les propriétés de sécurité, et présenté l'approche par laquelle l'ensemble des traces valides constituant ce modèle peut être construit. Notre modélisation tient compte d'un intrus puissant, dont la capacité de déduction a été ajustée conformément à l'algèbre de messages que nous avons définie. Le pouvoir de l'intrus a été également adapté en fonction des canaux de communication employés par le protocole. Les protocoles de vote électronique se distinguent des protocoles cryptographiques classiques par l'emploi de primitives plus avancées (signature en aveugle, chiffrement homomorphe), et l'usage de canaux de communication plus élaborés (anonymes, privés). La proposition d'une algèbre de messages riche, et la distinction entre les différents types de canaux de communication, nous ont permis de mieux représenter les spécificités des protocoles de vote, et d'affiner par conséquent notre analyse.

Spécification des propriétés. Nous avons montré en chapitre 4 que la logique ADM constitue un excellent candidat pour la spécification des propriétés des protocoles de vote électronique. En effet, le pouvoir d'expression de la logique, dû à ses modalités temporelles et existentielles, à sa linéarité, et à la possibilité de spécifier des formules récursives, nous a permis d'exprimer une large variété de propriétés de sécurité : équité, droit d'expression, éligibilité, non réutilisabilité, exactitude, anonymat et sans-reçu. D'autre part, l'introduction des systèmes de réécriture, nous a permis de simplifier considérablement la spécification de certaines de ces propriétés (équité, anonymat).

TABLE 5.6 – Vérification des protocoles de vote électronique

		Algèbre de processus					**Logique modale**			
		[KR05]	[DKR06]	[COPD06]	[MVdV07]	[BHM08]	[JP06]	[EO07]	[BRS07]	**[Notre approche]**
	Formalisme	π-calcul appliqué	π-calcul appliqué	μ-CRL	ACP	π-calcul appliqué		DEL		ADM
Propriétés	Équité	×							×	×
	Droit d'expression									×
	Éligibilité	×				×				×
	Non réutilisabilité					×				×
	Exactitude									×
	Vérifiabilité individuelle								×	
	Vérifiabilité universelle									
	Anonymat	×	×	×	×			×	×	×[c]
	Sans-Reçu		×			×	×		×	×[c]
	Résistance à la coercition		×			×				
	Autre					Inaltérabilité				
	Protocole	FOO	LBDKYY [LBD+03][a]	FOO	FOO	JCJY [JCJ05]	-	FOO[a]	FOO[a]	FOO
	Automatisation	Partielle[b]	Partielle[b]	Oui	Non	Oui	Non	Oui	Non	Oui
	(**Outils**)	ProVerif	ProVerif	Outils μ-CRL		ProVerif		CADP LYS		ADM-Checker

a. L'analyse est portée sur une version simplifiée du protocole.
b. Certaines propriétés sont prouvées manuellement.
c. Spécification partielle de la propriété.

Vérification automatique des propriétés. Finalement, nous avons exploité en chapitre 5, le système de preuve de la logique ADM, afin de disposer d'un outil automatisant la tâche de la vérification formelle des propriétés. Cet outil, que nous avons dénommé ADM-Checker, offre à l'utilisateur la possibilité d'exprimer ses propriétés, et de les vérifier ensuite par rapport à une trace ou un ensemble de traces. L'outil ADM-Checker est doté d'un analyseur syntaxique permettant d'assister l'utilisateur final dans la spécification, dans le respect de la grammaire de la logique ADM, de ses propriétés et de leurs paramètres associés. Notre implémentation du vérificateur formel est fidèle au système de preuve de la logique ADM, et permet de dériver automatiquement le résultat associé à la vérification de la relation de satisfaction $t \models \phi$, où t et ϕ désignent respectivement une trace et une formule.

Chacune de ces trois étapes a été appuyée par notre étude de cas portant sur le protocole FOO. Notre analyse de ce protocole nous a permis de montrer que ce dernier ne garantit pas les propriétés d'éligibilité et d'exactitude face à un administrateur malhonnête. De ces résultats, découle la faillibilité du protocole à satisfaire la propriété de vérifiabilité universelle, du fait de son lien étroit avec la propriété d'exactitude. Finalement, notre formalisation des propriétés d'anonymat et de sans reçu, quoique partielle, nous a permis d'établir que ce protocole ne garantit pas la seconde propriété, et par conséquent, qu'il ne garantit pas non plus la propriété de résistance à la coercition.

Le tableau 5.6 récapitule enfin les principales propriétés que nous avons spécifiées tout en positionnant notre approche par rapport aux travaux connexes. Tout d'abord, par comparaison aux logiques modales utilisées dans [JP06, EO07, BRS07], la logique ADM nous a permis de spécifier plus de propriétés. En effet, la spécification des propriétés telles que le droit d'expression, l'éligibilité, la non réutilisabilité ou bien l'exactitude, n'a pas été explorée dans ces travaux. De plus, grâce au système de preuve de la logique ADM, il a été possible d'automatiser, par l'intermédiaire de l'outil ADM-Checker, le processus de vérification des propriétés. Ce point n'a pas été abordé dans les travaux [JP06, BRS07]. La logique DEL, utilisée quant à elle dans [EO07], a servi à la fois à la spécification des propriétés et du protocole. L'inconvénient d'une telle approche est que la logique pourrait être expressive pour la spécification des propriétés, mais pourrait l'être moins pour la spécification du protocole, et vice-versa. Finalement, notre analyse s'est portée sur une version du protocole FOO qui est plus fidèle au protocole original, que celle considérée dans les travaux [EO07, BRS07].

D'un autre côté, les algèbres de processus, et en particulier le π-calcul appliqué, sont plus appropriées pour la spécification des propriétés d'anonymat et sans-reçu, où leur formalisation est donnée en terme d'équivalences observationnelles. Cependant, la spécification de certaines propriétés, dites de "trace" (démocratie, exactitude), semble plus naturelle à capturer dans la logique ADM. En effet, celle-ci a été spécialement conçue dans le but d'analyser des traces d'exécution de protocoles cryptographiques, et a été dotée d'opérateurs permettant d'apporter des modifications dynamiques à la trace analysée. Cette caractéristique (linéarité + récursivité) est essentielle dans la spécification des propriétés des protocoles de vote électronique, où l'on a à vérifier si certaines actions du protocole vont de paire (la demande de signature et le jeton, le vote et sa publication, etc.).

Perspectives

Afin de compléter le travail réalisé durant cette thèse, nous envisageons les perspectives discutées ci-après.

Spécification des propriétés. Nous proposons tout d'abord, de compléter notre travail par la spécification des propriétés de vérifiabilité. La vérifiabilité individuelle signifie que le protocole doit permettre à l'électeur la possibilité de vérifier la prise en compte de son vote. La vérifiabilité universelle exprime quant à elle le fait que les résultats de l'élection doivent être vérifiables par tout votant. Il serait intéressant d'étudier la spécification de ces deux propriétés dans la logique ADM.

Générateur de traces. Dans notre analyse de la sécurité du protocole FOO, nous avons vérifié les propriétés spécifiées par rapport à des traces construites manuellement. Il serait dès lors intéressant de recourir à un générateur automatique de traces valides. Celui-ci pourra être développé par l'implémentation des règles qui modélisent les étapes du protocole, et celles qui caractérisent le pouvoir de déduction de l'intrus. À titre d'exemple, le générateur de traces associé au protocole FOO devra implémenter le jeu de règles données en section 3.3.2, et tenir compte des règles de déduction relatives aux primitives cryptographiques employées dans ce protocole (voir section 3.2). Cependant, le nombre de traces valides à générer est infini. Ceci est dû à l'intrus, capable de fabriquer puis d'insérer une infinité de messages dans le protocole. Afin de remédier à ce problème, une solution serait la génération de traces orientées, obtenues par la définition d'heuristiques favorisant l'exécution de certaines règles du modèle par rapport à d'autres. Autre aspect à considérer dans ce générateur de traces, est la taille des messages constructibles par l'intrus qui pourrait être limitée à une profondeur donnée.

Études de cas. Dans cette thèse, nous avons validé notre travail par une étude de cas, portant sur l'analyse d'un protocole de vote électronique utilisant la signature en aveugle, à savoir le protocole FOO. Il serait intéressant de réutiliser l'approche dans le but de porter l'analyse à d'autres protocoles, et notamment à ceux utilisant le chiffrement homomorphe. Ce travail a déjà été entamé par la modélisation, dans le cadre de notre participation au projet SAVE (Sécurité et Audit du Vote Électronique), du protocole conçu par les partenaires associés [1] au projet. Il serait intéressant maintenant de poursuivre ce travail par l'adaptation des propriétés spécifiées au protocole SAVE. Le recours à l'outil ADM-Checker permettra ensuite de dégager certains résultats, issus de la vérification de ces propriétés par rapport au modèle du protocole ainsi défini.

L'outil ADM-Checker. Nous proposons finalement de doter notre outil de fonctionnalités supplémentaires. ADM-Checker a été conçu de manière à permettre la vérification de toutes formules respectant la grammaire de la logique ADM, et ceux indépendamment de la structure des actions utilisées dans ces formules. Cette caractéristique ouvre le champ à d'autres types d'applications telles que celui de la détection d'intrusions [GTM07, TTM08], où l'outil ADM-Checker peut être utilisé dans le but de vérifier des formules caractérisant des scénarios d'attaque par

1. Cryptolog, la Direction Centrale de la Sécurité des Systèmes d'Information (DCSSI), l'École Normale Supérieure de Cachan (ENS), France Télécom, l'Institut National des Télécommunications (INT), et l'École Supérieure d'Électricité de Rennes (Supélec)

rapport à des traces recueillies à partir des données de l'audit. Ceci est rendu possible en laissant à l'utilisateur final le soin de définir son propre format d'action. Cependant, l'utilisateur est contraint à fournir par lui même un analyseur syntaxique permettant de s'assurer du respect du format imposé. Cette tâche peut s'avérer fastidieuse pour un simple utilisateur, et il serait intéressant de doter l'outil ADM-Chcker d'assistants permettant la génération automatique de l'analyseur syntaxique.

Nous proposons également d'améliorer dans une prochaine version de l'outil ADM-Checker, le rendu des résultats de vérification. En effet, il serait intéressant de dégager des résultats graphiques à la manière de ceux présentés en Tables 5.3 et 5.4. En effet, une telle disposition des résultats permettrait à l'utilisateur final de mieux assimiler la preuve associée à la vérification de la relation de satisfaction $t \models \phi$.

Annexe A

π-calcul appliqué

Le présent annexe est un complément sur le π-calcul appliqué introduit au niveau du chapitre 2 (section 2.1.2). Nous donnons dans ce qui suit la sémantique opérationnelle de l'algèbre, ainsi que la modélisation du protocole FOO dans ce formalisme.

A.1 Sémantique opérationnelle

La sémantique opérationnelle du π-calcul appliqué est définie par l'intermédiaire de règles de congruence structurelle ($\equiv$) et de règles de réduction ($\rightarrow$).

A.1.1 Congruence structurelle

Dans cette section, nous donnons la définition ainsi que les règles de la relation de congruence structurelle.

Définition A.1.1 (*Relation de congruence structurelle*)

La congruence structurelle (notée $\equiv$) est la plus petite relation d'équivalence sur les processus étendus close par α-conversion à la fois sur les variables et les noms, et par application de contextes d'évaluation, et satisfaisant les règles structurelles de la Table A.1 telles que l'associativité (PAR-A), ou bien la commutativité (PAR-C) de la composition parallèle $|$.

Dans les règles de congruence structurelle données en Table A.1, A, B et C désignent des processus étendus, P est un processus simple, M et N sont des termes de l'algèbre, E est une théorie équationnelle, n est un nom ($n \in \mathcal{N}$), x est une variable ($x \in \mathcal{V}$), et u et v sont des méta-variables ($u, v \in \mathcal{N} \cup \mathcal{V}$). Les notations $f_n(A)$ et $f_v(A)$ représentent respectivement l'ensemble des noms et des variables libres (non liés) dans le processus A.

TABLE A.1 – Règles de congruence structurelle

PAR-0	A	$\equiv$	$A \mid 0$	
PAR-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$	
PAR-C	$A \mid B$	$\equiv$	$B \mid A$	
REPL	$!P$	$\equiv$	$!P \mid P$	
NEW-0	$\nu n.0$	$\equiv$	0	
NEW-C	$\nu u.\nu v.A$	$\equiv$	$\nu v.\nu u.A$	
NEW-PAR	$A \mid \nu u.B$	$\equiv$	$\nu u.(A \mid B)$	si $u \notin fv(A) \cup fn(A)$
ALIAS	$\nu x.\{^M/_x\}$	$\equiv$	0	
SUBST	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A\{^M/_x\}$	
REWRITE	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$	si $E \vdash M = N$

A.1.2 Relation de réduction

Dans cette section, nous donnons la définition ainsi que les règles de la relation de réduction.

Définition A.1.2 (*Relation de réduction*)

La relation de réduction (notée $\rightarrow$) est la plus petite relation sur les processus étendus close par congruence structurelle et par application de contextes d'évaluation, et satisfaisant les règles de la Table A.2.

TABLE A.2 – Règles de réduction

COMM	$\bar{a}\langle x\rangle.P \mid a(x).Q$	$\rightarrow$	$P \mid Q$	
THEN	*if* $M = M$ *then* P *else* Q	$\rightarrow$	P	
ELSE	*if* $M = N$ *then* P *else* Q	$\rightarrow$	Q	si $E \not\vdash M = N$

Dans les règles de réduction données en Table A.2, P et Q sont des processus simples, E est une théorie équationnelle, et M et N sont des termes clos ($f_v(M) = f_v(N) = \emptyset$; $f_v(T)$ représente l'ensemble des variables libres dans le terme T). Finalement, a désigne un nom de canal, et x représente une variable.

A.2 Modélisation du protocole FOO

La spécification des processus donnée en Table A.3 est issue de [DKR09]. Elle correspond à une configuration minimale faisant intervenir deux votants honnêtes, requis pour la formalisation de la propriété de l'anonymat. La spécification des processus est donnée dans un style proche de ProVerif [Bla01] dont il hérite certaines de ces déclarations telles que la commande de séparation de phase $phase$ permettant de modéliser les *deadlines* du protocole.

TABLE A.3 – Spécification du protocole FOO dans l'algèbre du π-calcul appliqué

Processus principal

process
$vprivch.\ vpkach_1.\ vpkach_2.\ vskach.\ vskvach.\ vskvbch.$
$(processK \mid processA \mid processA \mid processC \mid processC \mid$
(**let** $skvch = skvach$ **in** **let** $v = a$ **in** $processV$) |
(**let** $skvch = skvbch$ **in** **let** $v = b$ **in** $processV$))

Processus PKI K

let $processK =$
$vska.\ vskva.\ vskvb.$
let $pka = pk(ska)$ **in** **let** $hosta = host(pka)$ **in**
let $pkva = pk(skva)$ **in** **let** $hostva = host(pkva)$ **in**
let $pkvb = pk(skvb)$ **in** **let** $hostvb = host(pkvb)$ **in**
out$(ch, hosta)$. **out**$(ch, hostva)$. **out**$(ch, hostvb)$.
out(ch, pka). **out**$(ch, pkva)$. **out**$(ch, pkvb)$.
(**out**$(privch, pkva)$ | **out**$(privch, pkvb)$ |
out$(pkach_1, pka)$ | **out**$(pkach_1, pka)$ | **out**$(pkach_2, pka)$ | **out**$(pkach_2, pka)$ |
out$(skach, ska)$ | $out(skach, ska)$ | **out**$(skvach, skva)$ | **out**$(skvbch, skvb)$)

Processus votant V

let $processV =$
in$(skvch, skv)$.
let $hostv = host(pk(skv))$ **in**
in$(pkach_1, pubka)$.
$vb.\ vr.$
let $cv = commit(v, r)$ **in**
let $bcv = blind(cv, b)$ **in**
out$(ch, (hostv, sign(bcv, skv)))$.
in(ch, m_2).
let $res = checksign(m_2, pubka)$ **in**
if $res = bcv$ **then**
let $scv = unblind(m_2, b)$ **in**
$phase\,1$.
out$(ch, (cv, scv))$.
in$(ch, (l, = (cv, scv)))$.
$phase\,2$.
out$(ch, (l, r))$

Processus administrateur A	Processus collecteur C
let $processA =$	**let** $processC =$
in$(skach, skadm)$.	**in**$(pkach_2, pkadm)$.
in$(privch, pubkv)$.	$phase\,1$.
in(ch, m_1).	**in**$(ch, (l, (m_3, m_4)))$.
let $(hv, sig) = m_1$ **in**	**if** $checksign(m_4, pkadm) = m_3$ **then**
let $pubkeyv = getpk(hv)$	$vl.$ **out**$(ch, (l, (m_3, m_4)))$.
if $pubkeyv = pubkv$ **then**	$phase\,2$.
out$(ch, sign(checksign(sig, pubkeyv), skadm))$	**in**$(ch, (= l, rand))$.
	let $vote = open(m_4, rand)$ **in**
	out$(ch, vote)$

Processus principal. Le processus principal P modélise l'environnement et spécifie comment les divers processus sont combinés en parallèle. La spécification débute par la génération d'une série de canaux privés servant notamment à distribuer les clés publiques et privés aux entités concernées. Le processus principal consiste ensuite en la combinaison parallèle des processus représentant les différentes entités du protocole, à savoir les processus $processK$, $processV$, $processA$ et $processC$. Ces deux derniers sont dupliqués afin de faire correspondre à chaque processus votant, un processus administrateur et un processus collecteur.

Processus PKI (K). Dans leur modélisation du protocole FOO, Delaune et al. supposent une entité supplémentaire K faisant office de PKI (*Public Key Infrastructure*), et dont le rôle est la distribution des clés cryptographiques. Le processus $processK$ modélise le rôle de cette PKI. La spécification débute par la génération des clés privés ska, $skva$ et $skvb$ associées respectivement à l'administrateur A, et aux deux votants impliqués dans cette spécification. Les lignes qui suivent permettent d'extraire les clés publiques associées ainsi que les identités des détenteurs de ces clés, et de les diffuser sur le canal ch. Les clés privées sont quant à elles transmises via des canaux privés partagés entre la PKI, et les entités concernées. À noter que l'entité K est responsable également de la mise en place de la liste des votants légitimes, matérialisée par la transmission de leurs clés publiques sur le canal privé $privch$.

Processus votants. Le processus $processV$ modélise le rôle des votants dans le protocole. La spécification débute par une phase d'initialisation durant laquelle le votant reçoit sa clé privée skv (via le canal restreint $skvch$), dont il extrait son identité $hostv$ ($hostv = host(pk(skv))$). Le votant reçoit également, durant cette phase d'initialisation, la clé publique de l'administrateur ($in(pkach_1, pubka)$). La spécification se poursuit par la génération des paramètres aléatoires b et r servant respectivement à l'aveuglement et à la mise en gage. Le votant construit ensuite une version signée de son vote, par appel successif aux primitives $commit$, $blind$ et $sign$, qu'il transmet accompagnée de son identité $hostv$ à travers le canal ch. Le votant reçoit sur ce même canal la signature produite par l'administrateur dont il vérifie la validité, et à partir de laquelle il extrait une signature sur son vote gagé. Les lignes qui suivent décrivent la transmission du vote $out(ch, (cv, scv))$ et la réception du message l identifiant le bulletin transmis (la déclaration $in(ch, (m_1, = m_2))$ se traduit comme le processus qui lit sur le canal ch une paire de messages dont le second argument est m_2). Finalement, la spécification se termine par la transmission de la clé r permettant l'ouverture du bulletin identifié par l. À noter ici qu'un canal est par défaut anonyme en π-calcul appliqué, et de ce fait le bulletin ainsi que la clé permettant son ouverture sont transmis de manière anonyme. À noter également que le votant intervient dans le protocole FOO en trois étapes distinctes. Celles-ci sont séparées entre elles par le recours à la commande $phase$. La déclaration $phase\ 1$ permet de délimiter les instructions de la phase d'enregistrement de la phase de vote. De même, pour $phase\ 2$ qui délimite la phase de vote de celle d'ouverture des bulletins.

Processus administrateur. Le processus $processA$ modélise le rôle joué par l'administrateur A dans le protocole. L'administrateur commence par recevoir sa clé privée sur le canal restreint $skach$. Il reçoit également, sur le canal privé $privch$, la clé publique du votant. Celle-ci est ensuite comparée à celle correspondant à l'entité ayant communiqué sa demande d'enregistrement

sur le canal ch. Cette comparaison permet de s'assurer de la légitimité du votant. La spécification se poursuit par la transmission sur le même canal ch de la signature requise par le votant.

Processus collecteur. Le processus $processC$ modélise le rôle joué par le collecteur C. Le collecteur reçoit en premier lieu la clé publique de l'administrateur A. La suite des instructions du processus n'est enclenchée que lorsque la phase de vote ait été atteinte (spécifiée par $phase\,1$). Durant cette phase du protocole, le collecteur reçoit le vote sur le canal ch, vérifie sa signature, génère un identifiant l, l'associe au vote, et retransmet le tout sur le canal ch. Durant, la seconde phase (spécifiée par $phase\,2$), le collecteur reçoit la clé $rand$ permettant de déchiffrer le bulletin dont l'identifiant est l. Le vote ainsi déchiffré ($vote$) est finalement publié sur le canal ch.

Bibliographie

[19701] FIPS PUBS 197. Advanced Encryption Standard (AES). Technical report, National Institute of Standards and Technology, 2001.

[46-99] FIPS PUBS 46-3. Data Encryption Standard. Technical report, National Institute of Standards and Technology, 1999.

[ABB+05] Alessandro Armando, David A. Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, Paul Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, Sebastian Mödersheim, David von Oheimb, Michaël Rusinowitch, Judson Santiago, Mathieu Turuani, Luca Viganò, and Laurent Vigneron. The AVISPA tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of the 17th International Conference on Computer Aided Verification - CAV'05*, Lecture Notes in Computer Science, pages 281–285. Springer, 2005.

[ADM03] Kamel Adi, Mourad Debbabi, and Mohamed Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 291(3) :223–283, 2003.

[AF01] Martin Abadi and Cédric Fournet. Mobile values, new names, and secure communication. *ACM SIGPLAN Notices*, 36(3) :104–115, 2001.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1) :18–36, 1990.

[Ben87] Josh C. Benaloh. *Verifiable secret-ballot elections*. PhD thesis, Yale University, 1987.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. *Theory of Cryptography*, pages 325–341, 2005.

[BHM08] Michael Backes, Catalin Hritcu, and Matteo Maffei. Automated verification of remote electronic voting protocols in the applied pi-calculus. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium - CSF'08*, pages 195–209. IEEE Computer Society, 2008.

[BK85] Jan A. Bergstra and Jan Willem Klop. Algebra of communicating processes with abstraction. *Theoritical Computer Science*, 37 :77–121, 1985.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings of the 14th IEEE Computer Security Foundations Workshop (CSFW-14 Proceedings of the 14th IEEE Computer Security Foundations Workshop - CSFW'01*, IEEE Computer Society, pages 82–96, 2001.

[BO97] John Bull and Dave Otway. The authentication protocol. Technical Report DRA/CIS3/PROJ/CORBA/SC/1/CSM/436-04/03, Defence Research Agency, 1997.

[Boy89] Colin Boyd. A new multiple key cipher and an improved voting scheme. In *Proceedings of Advances in Cryptology - EUROCRYPT'89*, pages 617–625, 1989.

[BRS07] A. Baskar, Ramaswamy Ramanujam, and S. P. Suresh. Knowledge-based modelling of voting protocols. In *Proceedings of the 11th Conference on Theoretical Aspects of Rationality and Knowledge - TARK'07*, pages 62–71. ACM, 2007.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proceedings of the 26th annual ACM Symposium on Theory of Computing - STOC '94*, pages 544–553. ACM, 1994.

[CC97] Lorrie Faith Cranor and Ron Cytron. Sensus : A security-conscious electronic polling system for the internet. In *Proceedings of the 30th Annual Hawaii International Conference on System Sciences HICSS'97*, volume 3, pages 561–570, 1997.

[CDL+99] Iliano Cervesato, Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. A meta-notation for protocol analysis. In *Proceedings of the 12th IEEE Workshop on Computer Security Foundations - CSFW'99*, pages 55–69. IEEE Computer Society, 1999.

[Cer01] Iliano Cervesato. Typed MSR : Syntax and examples. In *Proceedings of the 1st International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security - MMM'01*, Lecture Notes in Computer Science, pages 159–177. Springer, 2001.

[Cet08] Orhan Cetinkaya. Analysis of security requirements for cryptographic voting protocols (extended abstract). In *Proceedings of the 3rd International Conference on Availability, Reliability and Security, ARES'08*, pages 1451–1456. IEEE Computer Society, 2008.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Proceedings of Advances in Cryptology - EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 1997.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2) :84–90, 1981.

[Cha88] David Chaum. The dining cryptographers problem : unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1) :65–75, 1988.

[CJ97] John Clark and Jeremy Jacob. A survey of authentication protocol literature : Version 1.0, 1997.

[Com09] Estonian National Election Committee. Internet voting in Estonia. http ://www.vvk.ee/, 2009.

[COPD06] Tom Chothia, Simona Orzan, Jun Pang, and Muhammad Torabi Dashti. A framework for automatically checking anonymity with μ-CRL. In *The 2nd Symposium on Trustworthy Global Computing - TGC'06*, pages 301–318, 2006.

[Cor05] Véronique Cortier. Vérifier les protocoles cryptographiques. *Technique et Science Informatiques*, 24(1) :115–140, 2005.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Proceedings of Advances in Cryptology - CRYPTO'92*, pages 89–105, 1992.

[CS02] Hubert Comon and Vitaly Shmatikov. It is possible to decide whether a cryptographic protocol is secure or not? *Journal of Telecommunications and Information Technology*, 2002.

[DDHY92] David L. Dill, Andreas J. Drexler, Alan J. Hu, and C. Han Yang. Protocol verification as a hardware design aid. In *Proceedings of the IEEE International Conference on Computer Design on VLSI in Computer & Processors - ICCD '92*, pages 522–525. IEEE Computer Society, 1992.

[DEK82] Danny Dolev, Shimon Even, and Richard M. Karp. On the security of ping-pong protocols. In *Proceedings of Advances in Cryptology - CRYPTO'82*, pages 177–186, 1982.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite Systems. In *Handbook of Theoretical Computer Science*, volume B : Formal Models and Semantics, chapter 6, pages 243–320. North-Holland, 1990.

[DJ04] Stéphanie Delaune and Florent Jacquemard. A theory of dictionary attacks and its complexity. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop - CSFW'04*, pages 2–15, 2004.

[DKR06] Stephanie Delaune, Steve Kremer, and Mark Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop - CSFW'06*, pages 28–42. IEEE Computer Society, 2006.

[DKR09] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security*, 17(4) :435–487, 2009.

[DY83] Danny Dolev and Andrew Chi-Chih Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–207, 1983.

[EO07] Jan Van Eijck and Simona Orzan. Epistemic verification of anonymity. *Electronic Notes in Theoretical Computer Science*, 168 :159–174, 2007.

[fle] The lex & yacc page. http ://dinosaur.compilertools.net/.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In *Proceedings of Advances in Cryptology - AUSCRYPT '92*, Lecture Notes in Computer Science, pages 244–251. Springer, 1992.

[For05] Formal Systems (Europe) Ltd. *Failures-Divergence Refinement - FDR2 User Manual*, 2005.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself : Practical solutions to identification and signature problems. In *Proceedings of Advances in Cryptology - CRYPTO'86*, Lecture Notes in Computer Science, pages 186–194. Springer, 1986.

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4) :469–472, 1985.

[GK00] Thomas Genet and Francis Klay. Rewriting for cryptographic protocol verification. In *Proceedings of the 17th International Conference on Automated Deduction - CADE'00*, pages 271–290, 2000.

[GLM97] Jean Goubault-Larrecq and Ian Mackie. *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer, 1997. ISBN 0-7923-4593-2.

[GR01] J. F. Groote and M. A. Reniers. Algebraic process verification. In J. A. Bergstra, A. Ponse, and S. A. Smolka, editors, *Handbook of Process Algebra*, pages 1151–1208. North-Holland, 2001.

[Gri02] Dimitris Gritzalis. Principles and requirements for a secure e-voting system. *Computers & Security*, 21(6) :539–556, 2002.

[GTM07] Meriam Ben Ghorbel, Mehdi Talbi, and Mohamed Mejri. Specification and detection of TCP/IP based attacks using the ADM logic. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security - ARES'07*, pages 206–212. IEEE Computer Society, 2007.

[HM07] Hanane Houmani and Mohamed Mejri. Secrecy by interpretation functions. *Knowledge-Based Systems*, 20(7) :617–635, 2007.

[Hoa85] Charles A.R. Hoare. *Communicating sequential processes*. Prentice-Hall, Inc., 1985.

[HR00] M. Huth and M. Ryan. *Logic in Computer Science : Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England, 2000.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Proceedings of Advances in Cryptology - EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 539–556. Springer, 2000.

[JCJ05] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society - WPES'05*, pages 61–70. ACM, 2005.

[JL97] Wen-Shenq Juang and Chin-Laung Lei. A secure and practical electronic voting scheme for real world environments. *TIEICE : IEICE Transactions on Communications/Electronics/Information and Systems*, 1997.

[JP06] Hugo Jonker and Wolter Pieters. Receipt-freeness as a special case of anonymity in epistemic logic, 2006. IAVoSS Workshop On Trustworthy Elections - WOTE'06.

[KR05] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied Pi calculus. In *Programming Languages and Systems - Proceedings of the 14th European Symposium on Programming - ESOP'05*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200, 2005.

[KSRW04] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. Analysis of an electronic voting system. In *IEEE Symposium on Security and Privacy - S&P'04*, pages 27–. IEEE Computer Society, 2004.

[LBD+03] Byoungcheon Lee, Colin Boyd, Ed Dawson, Kwangjo Kim, Jeongmo Yang, and Seungjae Yoo. Providing receipt-freeness in mixnet-based voting protocols. In *Proceedings of the 6th International Conference on Information Security and Cryptography - ICISC'03*, Lecture Notes in Computer Science, pages 245–258. Springer, 2003.

[Low95] Gavin Lowe. An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, 56(3) :131–133, 1995.

[Low96] Gavin Lowe. Splice-as : A case study in using csp to detect errors in security protocols. Technical report, Programming research Group, Oxford, 1996.

[Mea96] Catherine Meadows. The NRL Protocol Analyzer : An overview. *Journal of Logic Programming*, 26 :113–131, 1996.

[Mea03] Catherine Meadows. Formal methods for cryptographic protocol analysis : emerging issues and trends. *IEEE Journal on Selected Areas in Communications*, 21(1) :44–54, 2003.

[Mil82] Robin Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., 1982.

[Mil99] Robin Milner. *Communicating and mobile systems : the π-calculus*. Cambridge University Press, New York, NY, USA, 1999.

[MVdV07] Sjouke Mauw, Jan Verschuren, and Erik P. de Vink. Data anonymity in the FOO voting scheme. *Electronic Notes in Theoretical Computer Science*, 168 :5–28, 2007.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12) :993–999, 1978.

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *IFIP World Conference on IT Tools*, pages 21–30, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. In *Proceedings of the 5th International Workshop on Security Protocols - IWSP'97*, Lecture Notes in Computer Science, pages 25–35. Springer, 1997.

[PAB+04] Kun Peng, Riza Aditya, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Multiplicative homomorphic e-voting. In *Progress in Cryptology - INDOCRYPT'04*, Lecture Notes in Computer Science, pages 61–72. Springer, 2004.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology - EUROCRYPT'99*, Lecture Notes in Computer Science, pages 223–238. Springer, 1999.

[Pau98] Lawrence C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2) :85–128, 1998.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, University of Aarhus, 1981.

[Rja02] Zuzana Rjaskova. *Electronic Voting Schemes*. PhD thesis, Comenius University, 2002.

[Ros94] A. W. Roscoe. *Model-checking CSP*. Prentice Hall International (UK) Ltd., 1994.

[Ros95] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *Proceedings of the 8th IEEE Computer Security Foundations Workshop - CSFW'95*, pages 98–107. IEEE Computer Society, 1995.

[RS98] Peter Ryan and Steve Schneider. An attack on a recursive authentication protocol. a cautionary tale. *Information Processing Letters*, 65(1) :7–10, 1998.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[Saf00] Safevote. Internet voting requirements. *The Bell*, 1(7) :3–4, 2000.

[Sch99] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic voting. In *Proceedings of Advances in Cryptology - CRYPTO'99*, Lecture Notes in Computer Science, pages 148–164. Springer, 1999.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11) :612–613, 1979.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme - a practical solution to the implementation of a voting booth. In *Proceedings of Advances in Cryptology - EUROCRYPT'95*, Lecture Notes in Computer Science, pages 393–403. Springer, 1995.

[SS96] Steve Schneider and Abraham Sidiropoulos. CSP and anonymity. In *Proceedings of the 4th European Symposium On Research In Computer Security - ESORICS'96*, Lecture Notes in Computer Science, pages 198–218. Springer, 1996.

[Sti96] Colin Stirling. Modal and temporal logics for processes. In *Proceedings of the VIII Banff Higher order workshop conference on Logics for concurrency : structure versus automata*, pages 149–237. Springer, 1996.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2) :285–309, 1955.

[TMT+08a] Mehdi Talbi, Benjamin Morin, Valérie Viet Triem Tong, Adel Bouhoula, and Mohamed Mejri. Spécification formelle des propriétés des protocoles de vote électronique au moyen de la logique ADM. In *Proceedings of the 3rd Conference on Security in Network Architecture and Information Systems - SARSSI'08*, pages 151–165. Publibook, 2008.

[TMT+08b] Mehdi Talbi, Benjamin Morin, Valérie Viet Triem Tong, Adel Bouhoula, and Mohamed Mejri. Specification of electronic voting protocol properties using ADM logic : FOO case study. In *Proceedings of the 10th International Conference on Information and Communications Security - ICICS'08*, Lecture Notes in Computer Science, pages 403–418. Springer, 2008.

[TTB09] Mehdi Talbi, Valérie Viet Triem Tong, and Adel Bouhoula. Specification of anonymity as a secrecy property in the ADM logic - homomorphic-based voting protocols. In *Proceedings of the 4th International Conference on Availability, Reliability and Security - ARES'09*, pages 281–288. IEEE Computer Society, 2009.

[TTM08] Mehdi Talbi, Meriam Ben Ghorbel Talbi, and Mohamed Mejri. Intusion detection prototype based on ADM logic. *Journal of Software*, 3(2) :15–22, 2008.

[WL94] Thomas Y. C. Woo and Simon S. Lam. A lesson on authentication protocol design. *ACM SIGOPS Operating Systems Review*, 28(3) :24–37, 1994.

VU :
Le Directeur de Thèse

VU :
Le Responsable de l'École Doctorale

VU pour autorisation de soutenance
Rennes, le

Le Président de l'Université de Rennes 1

Monsieur Guy Cathelineau

VU après soutenance pour autorisation de publication :
Le Président du Jury,

Résumé

Les systèmes de vote électronique sont basés sur des protocoles cryptographiques faisant usage de primitives avancées (chiffrement homomorphe, signature en aveugle, etc.) et impliquant des canaux de communication élaborés (anonymes, privés) dans le but de garantir certaines propriétés de sécurité : éligibilité, équité, anonymat, etc. Pour les systèmes critiques tels que celui du vote électronique, l'usage des méthodes formelles, pour prouver que les propriétés prétendues satisfaites par un protocole sont réellement garanties, devrait être incontournable. Dans cette thèse, nous proposons d'utiliser la logique modale ADM afin de spécifier une sélection de propriétés de sécurité. Ces propriétés sont vérifiables par rapport à un modèle à base de traces représentant des exécutions valides du protocole. Notre modélisation des protocoles tient en compte de la présence d'un intrus actif dont le pouvoir de déduction est ajusté en fonction des spécificités introduites par le vote électronique. La modélisation des protocoles et la spécification des propriétés sont appuyées par une étude de cas portant sur le protocole de vote FOO dont nous proposons une analyse de sa sécurité. En complément, nous proposons d'exploiter le système de preuve qui accompagne la logique ADM, afin d'implémenter un outil permettant d'automatiser le processus de la vérification formelle des propriétés spécifiées par rapport aux traces du protocole analysé.

Abstract

It is a well known fact that only formal methods can provide a proof that a given system meets its requirements. Their use should be mandatory for critical systems such as electronic voting. In this thesis, we propose the use of the modal and linear ADM logic in order to specify a set of security properties that a voting protocol is required to satisfy : eligibility, non-reusability, accuracy, fairness, anonymity, receipt-freeness, etc. Our goal is to check these properties against a trace-based model where a trace represents a valid run of the protocol. Protocol executions take place in a hostile environment controlled by an active intruder whose deduction power is incremented according to electronic voting cryptographic primitives (blind signature, bit-commitment, homomorphic encryption, zero-knowledge proof, etc.). Protocol modeling and properties specification are applied to the FOO protocol as a case study. Additionally, we propose to exploit the tableau-based proof system of the ADM logic, in order to develop a tool enabling the automatic verification of security properties against selected traces of the analyzed protocol.