THÈSE

présentée à l'École Normale Supérieure de Cachan

pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par : Antoine Mercier

Spécialité : Informatique

Contributions à l'Analyse Automatique des Protocoles Cryptographiques en Présence de Propriétés Algébriques : Protocoles de Groupe, Équivalence Statique

Soutenue le 04/12/2009, devant un jury composé de :

– Bruno Blanchet	rapporteur
– Hubert Comon-Lundh	examinateur
– Steve Kremer	directeur de thèse
– Denis Lugiez	examinateur
– Michaël Rusinowitch	rapporteur
– Ralf Treinen	directeur de thèse

Résumé

Les protocoles de sécurité sont des petits programmes dont la fonction est d'assurer certaines propriétés comme la confidentialité d'un message, l'authentification ou encore l'anonymat des parties qui exécutent ce programme. Dans cette thèse nous nous sommes intéressés à la vérification automatique de ces programmes, c'est à dire la vérification automatique du fait qu'ils remplissent bien leur fonction. Dans ce cadre, nous avons étudié deux questions distinctes.

Dans la première partie, nous nous intéressons à la question de la vérification d'un type particulier de protocoles, les protocoles de groupe, en présence d'un intrus passif. La spécificité des protocoles de groupe est le fait qu'ils sont spécifiés pour un nombre non borné de participants. Notre approche a consisté a représenter un ensemble infini de messages échangés durant un ensemble infini de sessions, une pour chaque nombre de participants. Nous utilisons une catégorie spéciale d'automates d'arbres représentant des sur-approximations des ensembles de messages secrets et des ensembles de messages échangés. Nous identifions des restrictions sur la spécification des protocoles qui nous permettent de réduire les capacités de l'intrus de manière à ce que la classe d'automates que nous avons évoquée soit close sous des règles restreintes de l'intrus. Nous obtenons ainsi la décidabilité d'une approximation du problème du secret pour une classe de protocoles de groupe.

Dans la seconde partie nous nous intéressons au problème de l'équivalence statique qui est une notion répandue d'indistinguabilité de séquences de termes, utile pour représenter certaines propriétés des protocoles cryptographiques. L'équivalence statique modulo une théorie équationnelle permet une représentation plus précise de certaines primitives cryptographiques en représentant les propriétés des opérateurs par des axiomes équationnels. Nous développons des méthodes qui, dans certains cas, nous permettent de simplifier la tâche de la décision de l'équivalence statique en retirant certains symboles de la signature et en réduisant la théorie équationnelle à des théories plus simples. Nous illustrons notre technique avec l'exemple du couplage bilinéaire.

Mots-clés :

Vérification formelle, protocoles cryptographiques, protocoles de groupe, automates d'arbre, équivalence statique, théories équationnelles.

Remerciements

En premier lieu, mes remerciements vont aux membres de mon jury. Je remercie Denis Lugiez de m'avoir fait l'honneur de bien vouloir présider ce jury. Je remercie Michaël Rusinowitch et Bruno Blanchet d'avoir accepté d'être les rapporteurs de ma thèse. Je tiens en particulier à remercier Bruno Blanchet pour la précision et le soin extrême qu'il a apporté à la relecture de mon manuscrit, allant jusqu'à me suggérer certaines améliorations au coeur de longues preuves techniques. Je remercie Hubert Comon-Lundh de m'avoir lui aussi fait l'honneur de participer à ce jury en tant qu'examinateur. Enfin et surtout je remercie Ralf et Steve, mes directeurs de thèse. Il me semble qu'ils ont su parfaitement équilibrer le dosage entre encadrement et liberté. J'ai ainsi pu à la fois expérimenter la difficulté de la recherche sans être bloqué trop longtemps face à des questions insolubles. Ils ont toujours été présents lorsque j'en avais besoin. De mon humble point de vue de doctorant, je peux dire aujourd'hui que je leur dois beaucoup, aussi bien pour mon parcours personnel que pour une chose essentielle qu'ils m'ont transmise et qui est utile dans bien des situtations, l'attitude à tenir face à un problème complexe : calme et rigueur.

Je tiens à remercier l'ensemble des membres du LSV. Mes remerciements vont en particulier à Philippe Schnoebelen, Alain Finkel et Stéphane Demri, qui ont toujours eu un souci réel et sincère de l'ensemble des doctorants et de leur devenir. Je remercie aussi l'ensemble des membres de l'axe SECSI. Les groupes de travail hebdomadaires auront été un véritable stimulant tout au long de mon doctorat. Là encore j'ai pu apprendre quelque chose d'important, répondre à un feu croisé de questions. L'exigence de rigueur et de précision qui régnait au sein de ce groupe de travail aura été plus que bénéfique pour mener a bien ma thèse mais aussi pour apprendre à ne pas craindre des auditoires exigeants et toujours à juste titre.

Je tiens par ailleurs à remercier la joyeuse bande avec qui j'ai eu la chance de cohabiter en salle béniteau. Chronologiquement, je remercie Régis pour son calme, sa délicatesse et sa sérénité légendaire. Je le remercie de m'avoir appris qu'il fallait toujours mettre sa cagoule. Je remercie Tali pour ses imitations de Betsalel, et surtout pour les histoires cocasses qui allaient avec. Je la remercie aussi pour les discussions passionnantes qu'on a pu avoir sur des sujets très divers et très variés. Arnaud, ... il y a beaucoup de choses à dire sur Arnaud, en tous cas je le remercie d'être devenu mon pote, enfin j'espère. Enfin je remercie Najla, pour son calme aussi légendaire que celui de Régis. Je la remercie surtout d'avoir toujours gardé sa gaité naturelle. Globalement je tiens à les remercier tous pour le soutien que l'on s'est apporté mutuellement et la bonne humeur qui a régné dans ce bureau, même pendant les phases de rédaction intenses des uns et des autres.

Je remercie enfin ma famille à commencer par ma femme Marianne, qui m'a supporté pendant ces trois années de thèse et mes parents dont l'ouverture d'esprit et le soutien aussi bien moral que matériel, m'a finalement permis par des détours inattendus d'arriver au LSV.

Table des matières

Introduction

Chapitre 1

Introduction

Sommaire

Nous allons commencer par décrire quelques enjeux des recherches que nous avons menées (section 1.1). Puis nous passerons en revue les objectifs de la vérification des protocoles cryptographiques (section 1.2). Ensuite nous décrirons notre manière de modéliser ces protocoles (section 1.3), puis certaines propriétés de sécurité (section 1.4). Enfin nous présenterons brièvement nos contributions ainsi que la manière dont la suite est organisée (section 1.5).

1.1 Motivations pour le grand public

1.1.1 Survol sociologico-médiatique du développement des transactions en ligne

Seulement dix ans mais que de distance avec cet extrait [Ina99] du journal de 20 h où Béatrice Schönberg évoquait l'Internet comme une « révolution qui touche tous les domaines,

y compris celui du commerce ». À l'époque, parmi les sites les plus fréquentés figuraient ceux des vendeurs de vin, faut-il y voir un signe ? Malheureusement comme l'indique le reportage, « Revers de la médaille : le risque de piratage », que le gouvernement de l'époque avait tenté de contrer en « autorisant le cryptage pour des paiements sécurisés ». La directrice marketing de MSN expliquait au grand public que le paiement sécurisé était un « transfert d'information confidentiel entre l'ordinateur [...] de l'internaute et le site marchand ».

Nous pouvons aussi citer cet autre reportage [Ina98], plus ancien d'une année, dans lequel il est clair qu'on ne peut évoquer le commerce électronique sans évoquer « les sérieux risques de fraudes liés au mode de paiement ». On prévoyait 200 000 acheteurs pour Noël et donc 200 millions de francs de chiffre d'affaires, c'est-à-dire 1 % des achats de Noël.

Ce n'est qu'hier mais c'est déjà si loin. Si nous en croyons les chiffres de la Fevad, la « Fédération du e-commerce et de la vente à distance » [Fev], 78 % des français achètent à distance, en 2008, il y a eu plus de 200 millions de colis pour les achats de la vente à distance, et 80 % de la vente à distance s'effectue sur Internet. Si nous revenons à nos petits reportages du 20 h sur le commerce électronique pour les cadeaux de Noël, nous apprenons (en 2007 [Ina07]) que « dès 10 heures [le lendemain de Noël] des centaines de milliers d'articles sont en vente aux enchères ». Le commerce électronique n'est plus réservé à des sites spécialisés. Tout un chacun peut acheter et vendre sur Internet, et on n'évoque plus les risques de fraudes à chaque fois que l'on parle des achats des cadeaux de Noël par ce moyen.

Nous pourrions donc croire que tous les problèmes de sécurité liés aux échanges sur Internet sont réglés, puisque tout le monde s'en sert. Et pourtant d'après le ministère de l'intérieur [Min09], « s'agissant de la fraude à la carte bancaire recensée par les banques, elle oscillerait pour l'ensemble de la vente à distance entre 0,04 et 0,15% du chiffre d'affaires. Ces chiffres sont supérieurs au taux de fraude constaté pour les transactions de proximité, de l'ordre de 0,026% ».

En termes de sécurité, les échanges numériques ne sont toujours pas parvenus au niveau des échanges traditionnels. Les échanges dans le monde virtuel demeurent moins sûrs que les échanges dans le monde réel. Le travail visant à améliorer la sécurité de ces échanges est donc loin d'être terminé, et plus les échanges seront nombreux, plus ce pourcentage de fraude représentera une perte importante pour la société.

Nous avons évoqué le commerce électronique comme domaine nécessitant une sécurisation accrue, mais de nombreux autres domaines où les applications numériques sont centrales, ont ce même besoin de sécurité. Entre autres, nous pouvons évoquer la téléphonie mobile et sur IP, la banque en ligne, les distributeurs de billets ainsi que le paiement par carte bancaire ...

Afin de sécuriser ces échanges, différents moyens sont mis en œuvre parmi lesquels les algorithmes et les protocoles cryptographiques.

1.1.2 Les protocoles cryptographiques

Traditionnellement, si nous voulons envoyer un message de manière discrète, nous le mettons dans une enveloppe et nous nous assurons que le rabat qui est censé la refermer colle bien. Si nous voulons assurer une tierce personne que nous sommes bien celui que nous prétendons être et que nous assumons une déclaration, nous signons. Si nous voulons être sûr qu'un message arrive à une destination, nous utilisons un recommandé ou nous confions notre message à un porteur en qui nous avons confiance. Nous évoquerons aussi l'exemple d'un scrutin dans le paragraphe sur l'indistinguabilité de la section 1.4. En bref, nous sommes habitués à un certain nombre d'opérations élémentaires qui nous permettent d'effectuer l'essentiel de nos

échanges et qui nous garantissent certaines propriétés de confidentialité, d'authenticité...

Nous pouvons distinguer deux approches complémentaires pour atteindre cette sécurité : celle qui est centrée sur les messages comme le scellage ou la signature, dont le correspondant numérique est la cryptographie, et celle qui est centrée sur les échanges comme le recours à la poste ou au mandat, dont le pendant numérique est l'établissement de protocoles.

Plusieurs ouvrages introduisent en détails la cryptographie et les protocoles cryptographiques tout en donnant des références complètes concernant les articles scientifiques où ils ont été publiés. Entre autres nous pouvons citer [Sch94] et [MvOV96].

La cryptographie

Signer, enfermer dans une enveloppe, ces opérations ne vont pas d'elles-mêmes dans le monde virtuel, et les simuler est devenu l'un des objets essentiels de la cryptographie. Ainsi mettre un message dans une enveloppe revient pour l'envoyeur à le chiffrer et ouvrir l'enveloppe revient pour le récepteur à le déchiffrer. Signer pourrait consister à effectuer un calcul dont la personne à qui nous voulons prouver notre identité sait que nous sommes très probablement les seuls capables d'effectuer un tel calcul. Ce calcul sera d'ailleurs appelé signature numérique.

L'histoire de ces outils est déjà dense. De nombreux scientifiques y ont pris part et les tourments de l'Histoire aussi. Au lieu de détailler cette longue histoire, nous préférons nous en remettre à un expert en la matière et indiquer l'ouvrage de Jacques Stern : « La Science du Secret » [Ste98].

En résumé, grâce à ces outils mathématiques, il devient en quelque sorte possible de simuler les outils habituels de sécurité comme le scellage ou la signature, tout en bénéficiant des avantages du monde virtuel, rapidité (voire immédiateté), reproductibilité, gain de place. Attaquer ces outils reviendrait à ouvrir une enveloppe et à la transmettre sans que cela ne se remarque, ou à imiter une signature. Ceci est possible dans la réalité, le but est que cela soit au moins aussi difficile à réaliser dans le monde numérique.

Les protocoles

Néanmoins, quelle que soit la qualité du travail des cryptographes, la tâche pour sécuriser les échanges numériques ne s'arrête pas là. En effet dans la réalité, pour effectuer des échanges sûrs, nous disposons de certaines institutions de confiance comme les banques (les agences), la Poste, ou les messageries qui donnent une garantie que l'envoi se fait comme convenu. C'est pour obtenir un correspondant à cela dans le monde numérique que nous utilisons, souvent sans le savoir, les protocoles cryptographiques.

Lorsque plusieurs unités communiquent grâce à un réseau informatique, elles le font en respectant certaines conventions de manière à assurer la transmission correcte des messages. Les descriptions de ces conventions s'appellent des *protocoles*. Concrètement, il s'agit de la description de suites d'étapes réalisées par plusieurs participants. Ces étapes peuvent être de natures diverses : réception d'un message, envoi d'un message, calcul... Et si chacune des étapes s'est déroulée comme cela est prévu, alors une propriété est censée être réalisée. On peut donc voir un protocole comme un programme distribué, dont l'exécution permet de faire changer le système d'état en rendant vraies certaines propriétés.

Ces échanges peuvent être analysés à plusieurs niveaux. On peut analyser des séquences de signaux électriques, électromagnétiques, lumineux, il s'agit là du niveau physique, le plus

concret. On peut analyser ces échanges de manière de plus en plus abstraite, et arriver au niveau des applications informatiques, tout en sachant que la description à un niveau abstrait correspond à des échanges concrets.

Nous allons analyser les protocoles au niveau des applications, et nous allons nous concentrer sur les protocoles visant à assurer une sécurité des échanges. La spécification de tels protocoles imposant en général que des moyens cryptographiques soient mis en oeuvre, ils sont appelés *protocoles cryptographiques*. De nombreux ouvrages et recueils existent parmi lesquels [Spo, CJ97], et il y a des ouvrages de référence comme [Sch94] qui brossent un portrait des différents types de tels protocoles.

A titre d'exemple de propriété, nous pouvons décrire la confidentialité (ou secret) d'un échange de la manière suivante. La confidentialité d'un message s échangé entre deux participants A et B est réalisée si à l'issue de l'exécution du protocole seuls A et B connaissent s. On s'attend à ce que l'on passe d'un état où au moins un des deux participants ne connaît pas s à un état où les deux participants connaissent s.

1.2 Vérification des protocoles cryptographiques

La vérification de protocoles consiste à développer des méthodes permettant de décider si un protocole cryptographique remplit bien les fonctions qu'il est censé remplir. En effet, il n'est plus suffisant d'affirmer qu'un protocole ou un algorithme cryptographique est sûr pour obtenir la confiance des personnes qui les utilisent. L'histoire scientifique de la cryptographie et des protocoles est à certains égards une succession de publications de nouveaux protocoles et de nouveaux algorithmes auxquels répondent des publications présentant des failles dans ces innovations. Cette dialectique du concepteur et de l'attaquant a ouvert la voie à un nouveau champ de recherche : celui qui consiste en la mise au point de méthodes permettant soit d'invalider les protocoles comportant des failles (sous-section 1.2.1), soit de valider ceux qui sous certaines hypothèses n'en comportent pas (sous-section 1.2.2).

Ces méthodes sont de différentes natures. Certaines consistent à définir des modèles très réalistes qui nécessitent alors l'intervention humaine en vue d'obtenir *une preuve de sécurité*. Une preuve de sécurité est une preuve mathématique que le protocole réalise bien ce qui est attendu. D'autres méthodes consistent à tendre davantage à l'abstraction afin d'obtenir des modèles plus simples, dont on espère la possibilité d'analyse par des ordinateurs. C'est sur ces modèles logiques (section 1.3) que nous allons nous concentrer.

Malheureusement, même dans un cadre déjà abstrait, la vérification formelle des protocoles n'est pas chose aisée. En effet dans le cas général, c'est-à-dire sans imposer aucune restriction sur les protocoles, de nombreux résultats d'indécidabilité ont été obtenus parmi lesquels on compte [DLMS99, CC05]. Néanmoins, certaines restrictions permettent d'obtenir des résultats de décidabilité concernant ces protocoles, que cela concerne la recherche d'attaques ou la preuve automatique.

1.2.1 Recherche d'attaques

Étant donnée une propriété dont on prétend qu'un protocole la réalise, une *attaque* est une exécution de ce protocole rendant cette propriété fausse. La recherche d'attaques consiste donc à créer un outil dont la fonction est d'élaborer une exécution possible d'un protocole constituant une attaque. Certains outils permettent de garantir cette détection sans garantir l'obtention d'une preuve dans le cas où aucune attaque n'est possible. Nous reparlerons de

cette possibilité dans la section 3.3. De nombreux paramètres concernant l'environnement dans lequel ce protocole est exécuté peuvent être pris en compte. Par exemple, le protocole peut être exécuté de manière isolée ou en présence d'autres exécutions de protocoles. Le nombre d'exécutions d'un même protocole peut varier. De nombreux autres paramètres peuvent être pris en compte.

Parmi ces approches, nous pouvons évoquer [RT01] qui est centré sur le problème de l'insécurité d'un protocole, c'est-à-dire sur l'existence d'une attaque et l'outil AVISPA [Avi, ABB+02, ABB+05]. Notons que les outils de vérification évoqués dans la section suivante permettent parfois d'extraire une attaque dans le cas où la vérification formelle échoue.

1.2.2 Preuve automatique

La seconde direction de recherche est de mettre au point des outils permettant de prouver la correction des protocoles. L'objectif est que l'outil réponde par l'affirmative à chaque fois que le protocole réalise bien une propriété qu'il est censé réaliser. De nombreux outils ont été développés en ce sens. Nous pouvons citer entre autres la plateforme de validation AVANTSSAR [Ava] qui est en particulier une continuation de l'outil AVISPA [Avi], les outils Proverif [Bla01, Blab] et Cryptoverif [Bla06, Blaa], l'outil Maude-NPA [EMM06], l'outil Scyther [Cre06]. Nous pouvons remarquer que lorsque ces outils fournissent une réponse négative, on peut parfois extraire une attaque de leurs sorties et il arrive que l'attaque ne soit pas réelle, en ce sens qu'elle provient de certaines approximations faites au cours de la modélisation.

1.3 Modélisation et abstraction

1.3.1 Participants et intrus

Jusqu'ici nous avons plusieurs fois utilisé le terme de participant sans véritablement l'expliquer. Ce que nous nommons un participant à un protocole est simplement l'une des parties qui exécute le protocole. L'intrus représente la capacité d'une partie mal intentionnée. Nous pouvons considérer deux types d'intrus contre les protocoles : les intrus actifs et les intrus passifs.

Intrus passif

Les capacités d'un intrus passif sont d'écouter, d'enregistrer et de calculer. Un intrus passif ne peut pas empêcher les messages d'arriver à destination. Il ne peut pas non plus émettre de message à destination des participants. Son objectif est donc d'enregistrer des messages émis au cours d'exécutions d'un protocole et de déduire des informations qu'il n'est pas censé obtenir.

Exemple 1 Considérons l'exemple du protocole de Shamir permettant d'échanger un message sans échanger de clé préalablement. Ce protocole n'a jamais été publié mais on peut en trouver une description dans [Sch94]. Il requiert un chiffrement commutatif. Si $\{m\}_K$ représente un message m chiffré par une clé K alors un chiffrement est commutatif si pour deux clés K_1 et K_2 $\{\{m\}_{K_1}\}_{K_2} = \{\{m\}_{K_2}\}_{K_1}$.

On peut donc utiliser le ou exclusif (sous-section 2.3.2) noté $\oplus$ même si nous allons constater que cela permet une attaque.

Soient deux participants A et B voulant partager un message représenté par s. A (resp. B) dispose d'une clé K_A (resp. K_B) que lui seul connaît. On représente l'envoi d'un message de A à B par $A \to B$.

Envois	Message
$A \to B$	$s \oplus K_A$
$B \to A$	$s \oplus K_A \oplus K_B$
$A \to B$	$s \oplus K_B$

En considérant les caractéristiques de l'opérateur $\oplus$ (définies formellement sous-section 2.3.2) on constate qu'en possession de l'ensemble des messages émis on peut obtenir s :

$$(s \oplus K_A) \oplus (s \oplus K_A \oplus K_B) = K_B \qquad \text{et} \qquad (s \oplus K_B) \oplus K_B = s$$

Intrus actif

Un intrus actif est en revanche un intrus qui a non seulement toutes les capacités d'un intrus passif, mais qui est en plus capable d'interrompre la circulation des messages, et d'envoyer des messages aux participants en prenant la place d'autres participants. Dans cette situation, on dit souvent que l'intrus est le réseau puisque l'on peut alors considérer que tous les messages passent par lui. Il peut être un participant à un protocole. Dans cette thèse, nous nous concentrerons sur des intrus passifs.

Exemple 2 Le protocole de Diffie-Hellman que nous présentons en utilisant les mêmes conventions que dans l'exemple 1 a pour but d'établir une valeur secrète entre deux participants A et B sans qu'il n'y ait d'échange sécurisé préalable. Dans ce protocole, on suppose que les participants s'entendent publiquement sur un certain entier p et un entier α et que connaissant deux valeurs $\alpha^{\mathcal{N}_1}$ (mod p) et $\alpha^{\mathcal{N}_2}$ (mod p), il est difficile de distinguer $\alpha^{\mathcal{N}_1\mathcal{N}_2}$ (mod p) d'une valeur aléatoire. $\mathcal{N}_A$, $\mathcal{N}_B$ et plus tard $\mathcal{N}_I$ représentent des valeurs secrètes fraîchement générées.

Envois	Message
$A \to B$	$\alpha^{\mathcal{N}_A}$ (mod p)
$B \to A$	$\alpha^{\mathcal{N}_B}$ (mod p)

Le message partagé par les deux participants est $\alpha^{\mathcal{N}_A\mathcal{N}_B}$. Ainsi même en présence d'un intrus passif, ce protocole permet d'établir un message secret commun.

En revanche, considérons la séquence de messages suivante où $I(A)$ désigne l'intrus prenant la place de A et $I(B)$ l'intrus prenant la place de B.

Envois	Message
$A \to I(B)$	$\alpha^{\mathcal{N}_A}$ (mod p)
$I(A) \to B$	$\alpha^{\mathcal{N}_I}$ (mod p)
$B \to I(A)$	$\alpha^{\mathcal{N}_B}$ (mod p)
$I(B) \to A$	$\alpha^{\mathcal{N}_I}$ (mod p)

Ici non seulement l'intrus peut calculer le message $\alpha^{\mathcal{N}_A\mathcal{N}_I}$ que A croit partager avec B, le message $\alpha^{\mathcal{N}_B\mathcal{N}_I}$ que B croit partager avec A et en plus A et B ne partagent finalement aucune valeur. On dit que cette attaque est de type « homme au milieu ».

Notons dès à présent que dans cette thèse nous analyserons la sécurité des protocoles en présence d'un intrus passif.

1.3.2 Canaux de communication

On peut distinguer deux types de canaux de communication. Les canaux de communication publics et les canaux de communication privés.

Les canaux publics

Un canal de communication est public si tous les messages transitant par ce canal sont accessibles à tous, participants honnêtes ou malhonnêtes et intrus.

Les canaux privés

Un canal de communication est privé lorsqu'il est défini pour un ensemble G de participants et qu'il est impossible à tout participant p n'appartenant pas à G d'accéder à ce canal en lecture ou en écriture.

Dans certaines situations, on pourrait considérer un modèle comportant des canaux privés et publics. Le π-calcul [Mil99] permet par exemple de représenter de tels canaux.

De nombreuses modélisations ne considèrent que des canaux publics. Cela permet d'accorder un maximum de pouvoir à l'intrus. Les preuves de sécurité valables dans un tel cadre vaudront aussi en présence de canaux privés. Pour notre part nous considérons que les canaux sont publics.

1.3.3 Nombre de sessions, nombre de participants et taille des messages

Habituellement le nombre de participants à un protocole est fixé. Dans les exemples 1 et 2, ce nombre est 2. Dans la partie I, nous étudierons la vérification de protocoles dont la spécification autorise un nombre non borné de participants. Ces protocoles sont appelés *protocoles de groupe*.

Une session d'un protocole est l'exécution de ce protocole par un ensemble donné de participants. A priori le nombre de sessions d'un protocole est non-borné. Dans la vérification, une approximation courante est de borner le nombre de sessions. Cependant dans la partie I, nous verrons que nous considérons un nombre infini de sessions.

Un troisième aspect qui peut varier d'un protocole à un autre est le fait que la taille des messages est bornée ou non. En effet dans certains protocoles, c'est souvent le cas des protocoles de groupe, des opérations récursives sur les messages sont utilisées. Ceci permet entre autres des exécutions dont les messages ont une taille non bornée.

1.3.4 Hypothèse du chiffrement parfait et théories équationnelles

Hypothèse du chiffrement parfait

Cette hypothèse consiste à supposer que l'intrus n'a aucun moyen de mener une attaque contre les algorithmes cryptographiques. Le but d'une telle hypothèse n'est évidemment pas

de considérer que les cryptographes font ou feront un travail plus parfait que la réalité mathématique ne le permet. Le but d'une telle hypothèse est de s'abstraire des attaques prenant pour cible les algorithmes de chiffrement afin de mieux se focaliser sur une faiblesse potentielle des protocoles. Lorsqu'une attaque est trouvée alors que nous avons fait cette hypothèse, on parle habituellement *d'attaque logique*. La possibilité d'une attaque logique met en évidence l'existence d'une *faille logique*. Une faille logique est une possibilité d'exploiter le déroulement du protocole en enregistrant les messages émis (exploit d'un intrus passif) ou en interagissant avec les participants (exploit d'un intrus actif), sans recourir à un travail de cryptanalyse sur les messages.

Cette hypothèse a donné lieu à la représentation des primitives cryptographiques d'une manière abstraite. Les premiers travaux allant dans ce sens sont ceux de Dolev et Yao [DY83]. C'est en suivant cette même idée que nous représenterons les messages par des termes (définis formellement au chapitre 2). L'égalité des messages sera ainsi représentée par l'égalité syntaxique entre les termes qui les représentent.

Cette approche logique trouve en outre une justification grâce à de nombreux travaux visant à établir des liens entre modèles symboliques et modèles concrets en montrant l'adéquation des premiers. L'un des premiers fût celui d'Abadi et Rogaway [AR00]. Ces travaux visent à montrer qu'une preuve dans un modèle symbolique implique l'existence d'une preuve dans un modèle plus proche de la réalité, sous réserve que les primitives cryptographiques vérifient des hypothèses standards, parfois fortes.

Théories équationnelles

Nous affaiblirons parfois l'hypothèse du chiffrement parfait afin de mieux représenter certains rapports entre les termes. En effet, cette hypothèse est parfois irréaliste, en ce sens qu'elle ne permet pas de trouver certaines attaques qui n'ont pas recours à un algorithme de cryptanalyse. Nous prenons donc en compte certaines propriétés des opérations effectuées sur les messages grâce à une théorie équationnelle (définie formellement au chapitre 2). Un recueil de ces propriétés est disponible [CDL06].

1.3.5 Nonce

Il est fréquent que les protocoles cryptographiques utilisent des nombres aléatoires supposés frais dans leurs spécifications. Par fraîcheur, nous entendons le fait qu'ils sont propres à une exécution du protocole. Ces nombres frais sont appelés nonces. « Nonce » signifie simplement nombre utilisé une fois et est l'abréviation de « number used once ». Ces nombres jouent parfois un rôle essentiel dans les protocoles. Ils sont par exemple une partie centrale des clés du protocole de Diffie-Hellman présenté à l'exemple 2. On s'attend souvent à ce que l'intrus, extérieur au protocole, n'y ait pas accès à moins que l'un des participants ne décide de le révéler.

Il existe plusieurs manières de représenter la fraîcheur de certains éléments en vue de la vérification des protocoles. Dans la partie I, nous utiliserons des indices sur certains éléments du modèle associant l'élément représentant le nonce à une exécution du protocole. Dans la partie II, la possibilité de représenter la fraîcheur d'une valeur appartient au formalisme lui-même.

1.4 Propriétés à vérifier

Nous passons en revue certaines propriétés de sécurité souvent analysées.

1.4.1 Les propriétés de secret

Atteignabilité

Nous considérerons qu'un message s est secret s'il n'est pas *atteignable* par l'intrus. Dans le cas d'un intrus passif, cela signifie simplement que l'intrus n'a aucune possibilité de combiner les messages émis au cours de l'exécution du protocole en vue d'obtenir s.

Cette notion de secret peut être utile pour caractériser la sécurité de la transmission d'un message. Cela permet de vérifier que des combinaisons inattendues de messages ne permettent pas d'obtenir ce message secret. Dans le cas d'un protocole d'établissement de clé, qui vise à établir une nouvelle valeur secrète entre les participants, cette propriété garantit qu'un intrus ne peut pas obtenir la clé calculée.

Notons que contrairement à la notion de secret fort défini en termes d'indistinguabilité, le secret défini comme non-atteignabilité permet la divulgation partielle d'un secret.

Dans la partie I, nous nous sommes concentrés sur cette notion de secret. Nous en proposons une définition formelle dans la section 4.5 du chapitre 4.

Indistinguabilité

On peut aussi définir la notion de secret par une notion d'équivalence. La notion de secret que nous allons analyser dans la partie I est tout-à-fait adéquate lorsqu'il s'agit de savoir si un protocole d'échange de clés a une faille qui permettrait d'obtenir une des clés calculées. En revanche, évoquons ici l'exemple classique des protocoles de vote. Le but d'un tel protocole est de garantir un niveau de sécurité au moins identique à celui d'un scrutin réel (avec papier, enveloppes, urnes, assesseurs...). Cette sécurité se définit par rapport à diverses propriétés. Observons de plus près la propriété d'anonymat, qui peut se définir simplement comme le fait de ne pas pouvoir déterminer qui a voté pour qui. Il s'agit bien là d'une forme de secret, puisque l'application qui associe à chaque votant son vote doit rester secrète. Ce secret a un sens bien particulier puisque l'identité du votant ou le bulletin électronique ne peuvent être considérés comme secrets au sens de l'atteignabilité. Les votants ainsi que les bulletins doivent être connus de tous, et évidemment de l'intrus lui-même. La publicité nécessaire de la plupart des éléments du vote fait de la définition de l'anonymat par la notion d'atteignabilité une gageure.

Dans le cas du vote électronique, les éléments du vote, prendre une enveloppe, un bulletin, se cacher dans l'isoloir, présenter ses papiers aux assesseurs, glisser l'enveloppe dans l'urne, etc... sont réalisés par des échanges de messages. Ainsi on peut considérer qu'un vote électronique est anonyme (à bulletin secret), s'il est impossible de distinguer la séquence de messages envoyés pour voter pour A de la séquence de messages envoyés pour voter pour B. Il semble alors plus aisé de caractériser l'anonymat par une notion d'équivalence entre plusieurs séquences que par la capacité d'en obtenir certains éléments. On sent poindre l'adéquation d'une notion d'équivalence pour caractériser ce type de secret, reste désormais sa définition formelle qui est donnée dans la sous-section 7.1.3.

Cette notion d'indistinguabilité permet en particulier de définir comme dans [Bla04] une notion de secret appelée « secret fort », que l'on peut décrire intuitivement comme l'incapacité

d'un intrus de distinguer deux séquences de messages dans lesquels seule la valeur du secret a changé. Dans [CRZ07] V. Cortier, M. Rusinowitch et E. Zălinescu ont effectué une analyse systématique des rapports entre ces deux notions de secret.

1.4.2 D'autres propriétés

Authentification

La propriété d'authentification consiste à être assuré de l'identité du participant avec lequel nous communiquons. On dit souvent qu'un participant A s'est authentifié auprès d'un participant B, si B a un élément probant lui assurant qu'il communique bien avec A. À cette fin nous utilisons parfois des mots de passe. Par exemple, lorsque nous utilisons une carte bancaire pour assurer le distributeur de billet que nous sommes bien le propriétaire de la carte, nous saisissons un mot de passe, ce qui nous permet de nous authentifier auprès de ce distributeur. Il existe plusieurs manières de caractériser formellement cette propriété [Low97], [Sch98].

Anonymat

Un protocole assure l'anonymat à un de ses participants, s'il garantit qu'un intrus ne peut pas déterminer son identité. La propriété d'indistinguabilité que nous avons présentée dans la sous-section 1.4.1 peut être utile pour caractériser la propriété d'anonymat. Comme nous l'avons expliqué, elle permet de définir la propriété d'anonymat d'un votant.

Équité

Cette propriété peut être attendue dans certains cas de signature de contrat. Il y a souvent un intérêt à signer en dernier un contrat. Si la propriété d'équité est vraie pour un protocole, cela signifie qu'aucune des parties n'est désavantagée à moins qu'elle n'y consente. Différentes formalisations existent [KR02], [MGK03], [CMSS03], [KKW06].

1.5 Contributions et contenu de la thèse

Nous avons tenté de contribuer à l'élargissement de la possibilité d'analyse automatique de ces protocoles dans deux directions distinctes. Ainsi après avoir introduit le cadre formel commun dans le chapitre 2, nous présenterons tour à tour un travail concernant un type particulier de protocoles, les protocoles de groupe, puis un travail concernant la simplification de certains éléments formels de la vérification.

1.5.1 Les protocoles de groupe en présence d'un intrus passif

Dans la partie I, nous nous concentrons sur l'analyse des protocoles de groupe. L'objet de cette première partie est donc de contribuer à l'élargissement des capacités d'analyse des modèles habituels, dans la plupart desquels le nombre de participants est borné. Notons que le nombre de sessions lui-aussi est souvent borné. Nous y sommes parvenus dans l'analyse des protocoles pour le cas d'un intrus passif tout en représentant certaines propriétés des

primitives cryptographiques, et en analysant la propriété d'atteignabilité. Notre modèle permet de représenter des protocoles à nombre non borné de participants, de sessions, et où les participants effectuent des calculs récursifs. Ce premier résultat a été publié dans [KMT08].

Plan

Nous introduirons en détails le problème analysé, ses enjeux, certains protocoles de ce type et les travaux qui y sont liés dans le chapitre 3. Nous présenterons notre manière de modéliser ces protocoles et les capacités d'un intrus passif dans le chapitre 4. Dans le chapitre 5 nous identifierons des conditions sur la spécification de certains protocoles nous permettant de ne considérer qu'un intrus plus faible qu'attendu contre ces protocoles. Nous prouverons bien entendu que cela est possible. Enfin dans le chapitre 6 nous décrirons un modèle d'automates d'arbres à mémoire qui nous permet de décrire une approximation des exécutions de certains protocoles de groupe auxquels nous nous intéressons. Grâce à certaines propriétés de ces automates nous prouverons la décidabilité d'une approximation du problème du secret.

1.5.2 Simplification de théories équationnelles

Dans la partie II, nous nous intéressons à l'établissement de conditions permettant de combiner des théories équationnelles tout en préservant la décidabilité d'une propriété d'indistinguabilité dans le cadre d'un modèle plus récent. Nous utilisons pour ce faire un typage des éléments représentant les messages échangés, c'est-à-dire une catégorisation de ces éléments. Grâce à ce typage, formellement défini dans le chapitre 2, nous identifions certaines opérations, représentées par certains symboles, dont on peut éviter la considération, si l'on a préalablement effectué certains calculs. Nous établirons de cette manière un résultat de décidabilité pour une théorie particulière représentant le couplage bilinéaire. Ce second résultat a été publié dans [KMT09a].

Plan

Dans le chapitre 7, nous introduirons le modèle, le problème de l'équivalence statique, les notions de combinaison et nous évoquerons les autres résultats de combinaison dans le domaine de la vérification des protocoles. Dans le chapitre 8, nous établirons les critères sur les théories équationnelles nous permettant de les simplifier. Enfin dans le chapitre 9, nous présenterons les preuves du résultat en détail et nous prouverons la décidabilité de la théorie du couplage bilinéaire.

Chapitre 2

Préliminaires techniques

Sommaire

Dans ce chapitre, nous proposons un bref rappel de notions qui ont trait aux algèbres de termes (section 2.1) munies de théories équationnelles (section 2.2), parfois exprimées sous forme de systèmes de réécriture. De nombreuses références existent parmi lesquelles on trouve [DJ90] et [BN98]. Nous présentons formellement les propriétés de certaines primitives cryptographiques (section 2.3).

Ces notions sont communes aux parties I et II.

2.1 Les algèbres de termes

2.1.1 Signatures

Signatures typées

Une *signature typée* $(\mathcal{S}, \mathcal{F})$ est définie par un ensemble de *types* $\mathcal{S} = \{s, s_1, s_2, \ldots\}$ et un ensemble de symboles de fonctions $\mathcal{F} = \{f, f_1, f_2, \ldots\}$. Ces symboles sont donnés avec des arités de la forme $\mathsf{arité}(f) = s_1 \times \cdots \times s_k \to s$ où $k \geq 0$. Lorsque $k = 0$, on dit que le symbole est une *constante* et on écrit s pour son arité.

Nous considérons un ensemble de *noms* et un ensemble de *variables*. Ces noms et ces variables sont typés et ordonnés. Ainsi nous considérons l'ensemble de noms $\mathcal{N} = \cup_{s\in\mathcal{S}}\mathcal{N}_s$ où $\mathcal{N}_s = \{n_{s1}, n_{s2}, \ldots\}$ et l'ensemble de variables $\mathcal{X} = \cup_{s\in\mathcal{S}}\mathcal{X}_s$ où $\mathcal{X}_s = \{x_{s1}, x_{s2}, \ldots\}$.

Signatures non typées

Une *signature non typée* ou simplement *signature* peut être définie comme une signature typée $(\mathcal{S}, \mathcal{F})$ dans laquelle $\mathcal{S}$ est un singleton. D'une part, il ne sera plus nécessaire de décrire l'ensemble $\mathcal{S}$ dans la donnée de la signature. D'autre part, nous considérerons que l'arité d'un symbole est simplement le nombre de ses arguments. Ainsi lorsque nous considérerons une signature non typée $\mathcal{F}$, nous noterons $\mathcal{F}_n$ l'ensemble des symboles d'arité n de $\mathcal{F}$. La donnée d'une signature non typée est donc la donnée d'un ensemble de symboles $\{f_1/i_1, \ldots, f_n/i_n\}$ où les i_j représentent l'arité des f_j.

2.1.2 Termes

Définition

Nous pouvons alors définir de manière inductive l'ensemble des termes de type s de la manière suivante :

t	::=		terme de type s
	|	x	variable x de type s
	|	n	nom n de type s
	|	$f(t_1, \ldots, t_k)$	application du symbole $f \in \mathcal{F}$

où chaque t_i est un terme de type s_i et $\mathsf{arité}(f) = s_1 \times \cdots \times s_k \to s$. L'ensemble des termes $T(\mathcal{F}, \mathcal{N}, \mathcal{X})$ est l'union des ensembles de termes de type s pour tout $s \in \mathcal{S}$.

Description

Nous notons $\mathsf{type}(t)$ le type de t. Nous écrirons aussi $\mathsf{var}(t)$ et $\mathsf{noms}(t)$ pour l'ensemble des variables et des noms apparaissant dans t. Un terme t est *clos* si et seulement si $\mathsf{var}(t) = \emptyset$. Nous notons $T(\mathcal{F}, \mathcal{N})$ l'ensemble des termes clos.

La *taille* de t, c'est-à-dire le nombre d'occurences de symboles qu'il comporte est noté $|t|$.

Les *positions* $Pos(t)$ sont définies de la manière habituelle comme un ensemble de mots sur $\mathbb{N}$, où Λ est le mot vide et $\cdot$ représente la concaténation des mots. Ainsi $Pos(u) = \{\Lambda\}$ quand $u \in \mathcal{N} \cup \mathcal{X}$ et $Pos(f(t_1, \ldots, t_n)) = \{\Lambda\} \cup \{i \cdot \pi \mid 1 \leq i \leq n, \pi \in Pos(t_i)\}$. Nous notons $t|_p$ le *sous-terme* de t à la position p et $t[u]_p$ le résultat du remplacement de t à la position p par u. La *hauteur* d'un terme correspond à la longueur maximale parmi les longueurs de ses positions, en considérant que la longueur de Λ est 0. Nous notons $St(t)$ l'ensemble des sous-termes du terme t, et nous étendons cette notation aux ensembles de termes. Nous dirons qu'un symbole de fonction f *apparaît* dans un terme t s'il y a des termes $t_1, \ldots, t_n$ tels que $f(t_1, \ldots, t_n) \in St(t)$. Dans ce même cas, nous dirons aussi que t *comporte* f. Dans le cas particulier où t est de la forme $f(u_1, \ldots, u_n)$, nous dirons que f *apparaît en tête* de t.

Exemple 3 Soit $(\mathcal{S}, \mathcal{F})$ la signature typée telle que $\mathcal{S} = \{E, G\}$, $\mathcal{F} = \{0, s, exp\}$ et $\mathsf{arité}(0) = E$, $\mathsf{arité}(s) = E \to E$, $\mathsf{arité}(exp) = E \to G$. Soient $n_E \in \mathcal{N}$ un nom de type E, $n_G \in \mathcal{N}$ un nom de type G, $x_E \in \mathcal{X}$ une variable de type E, $x_G \in \mathcal{X}$ une variable de type G.

Nous avons donc $0, s(x_E), s(s(0)), s(n_E) \in T(\mathcal{F},\mathcal{N},\mathcal{X})$. De même $exp(0)$, $exp(s(s(0)))$, $n_G \in T(\mathcal{F},\mathcal{N})$. Par contre $s(exp(s(0))), exp(exp(0)), exp(n_G) \notin T(\mathcal{F},\mathcal{N},\mathcal{X})$.

$\mathsf{type}(s(0)) = E$, $\mathsf{type}(exp(s(s(0)))) = G$, $\mathsf{var}(exp(s(s(0)))) = \emptyset$ et donc $exp(s(s(0)))$ est clos. $\mathsf{var}(s(x_E)) = \{x_E\}$, $\mathsf{noms}(s(n_E)) = \{n_E\}$. $|s(x_E)| = 2$.

$Pos(exp(s(s(0)))) = \{\Lambda, 1, 1{\cdot}1, 1{\cdot}1{\cdot}1\}$, $exp(s(s(0)))_{|1} = s(s(0))$, $exp(s(s(0)))[0]_{|1} = exp(0)$. La hauteur de $exp(s(s(0)))$ est 3.

0, s et exp apparaissent dans $exp(s(s(0)))$ et exp apparaît en tête de $exp(s(s(0)))$.

2.1.3 Substitutions

Une *substitution* σ, notée $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ de domaine $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_n\}$, est une application de $\{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ dans $T(\mathcal{F},\mathcal{N},\mathcal{X})$. Nous considérerons uniquement des substitutions *bien typées*, c'est-à-dire des substitutions dans lesquelles x_i et t_i ont le même type. Une substitution est dite *close* si tous les t_i sont clos. La restriction d'une substitution σ à un ensemble de variables V est la substitution σ' telle que $dom(\sigma') = dom(\sigma) \cap V$ et si $x_i \mapsto t_i \in \sigma$ et $x_i \in dom(\sigma')$ alors $x_i \mapsto t_i \in \sigma'$. L'*application* de la substitution σ au terme t remplace simultanément chaque occurrence de variable par son image par σ. Nous notons $t\sigma$ l'application d'une substitution σ à un terme t.

2.1.4 Contextes

Un *contexte* C est défini à la manière d'un lambda-terme de la forme $\lambda x_1.\ldots.\lambda x_n.t_C$ où les x_i peuvent apparaître dans le terme t_C. Par abus de notation, nous écrirons simplement $C[x_1, \ldots, x_n]$ au lieu de $\lambda x_1.\ldots.\lambda x_n.t_C$ et $C[t_1, \ldots, t_n]$ au lieu de $(\ldots(\lambda x_1.\ldots.\lambda x_n.t_C)t_1 \ldots t_n)$. Pour notre part, nous ne traiterons que des cas où les t_i sont clos. Dans ce cas précis, l'application de $(\ldots(\lambda x_1.\ldots.\lambda x_n.t_C)t_1 \ldots t_n)$ est égal à $t_C\sigma$ où $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. En d'autres termes $C[t_1, \ldots, t_n]$ est le résultat du remplacement de chaque x_i par t_i dans t_C. Un contexte est *public* s'il ne comporte pas de noms.

2.2 Théories équationnelles et systèmes de réécriture

2.2.1 Théories équationnelles

Une *équation* est une égalité $t = u$ dans laquelle t et u sont deux termes du même type. Une *théorie équationnelle* E est un ensemble fini d'équations.

Soit une signature $(\mathcal{S},\mathcal{F})$ et des ensembles de noms $\mathcal{N}$ et de variables $\mathcal{X}$. Une relation $\equiv$ sur $T(\mathcal{F},\mathcal{N},\mathcal{X})$ est une relation de congruence si $t_i \equiv u_i$ et $\mathsf{type}(t_i) = \mathsf{type}(u_i) = s_i$ alors pour tout symbole $f \in \mathcal{F}$ tel que $\mathsf{type}(f) = s_1 \times \cdots \times s_n \to s$ nous avons

$$f(t_1, \ldots, t_n) \equiv f(u_1, \ldots, u_n)$$

Nous notons $=_E$ la plus petite relation de congruence sur $T(\mathcal{F},\mathcal{N},\mathcal{X})$ telle que $t\sigma =_E u\sigma$ pour tout $t = u \in E$ et pour toute substitution σ. Nous dirons qu'un symbole f est *libre* dans E si f n'apparaît pas dans E.

Exemple 4 (Théorie équationnelle AC du symbole $\oplus$) À titre d'exemple, nous présentons la théorie équationnelle de l'associativité et de la commutativité, notée AC pour un symbole $\oplus$ en notation infixe.

$$x \oplus y = y \oplus x \qquad (x \oplus y) \oplus z = x \oplus (y \oplus z)$$

Considérons trois constantes a, b et c. Nous aurons entre autres les égalités suivantes

$$(a \oplus b) \oplus c =_{AC} (b \oplus a) \oplus c =_{AC} b \oplus (a \oplus c)$$

La classe d'équivalence d'une terme t modulo une théorie équationnelle E noté $[t]_E$ est l'ensemble des termes t' tels que $t' =_E t$. Un ensemble de termes S est *clos* sous une théorie équationnelle E, si pour tout terme $t \in S$, tout terme $t' \in [t]_E$ appartient à S.

2.2.2 Systèmes de réécriture

Un système de réécriture $\mathcal{R}$ est un ensemble de *règles de réécriture* $l \to r$ où $l \in T(\mathcal{F}, \mathcal{N}, \mathcal{X})$, $r \in T(\mathcal{F}, \mathcal{N}, \mathsf{var}(l))$ et l et r sont de même type. Un terme $u \in T(\mathcal{F}, \mathcal{N}, \mathcal{X})$ se réécrit en v par $\mathcal{R}$, s'il y a une règle de réécriture $l \to r \in \mathcal{R}$, une position p et une substitution σ telle que $u|_p = l\sigma$ et $v = u[r\sigma]_p$. Dans ce cas, nous écrivons $u \to_{\mathcal{R}} v$. La clôture réflexive et transitive de $\to_{\mathcal{R}}$ est notée $\to^*_{\mathcal{R}}$. Un terme est en *forme normale modulo* $\mathcal{R}$ s'il n'y a pas de terme u tel que $t \to_{\mathcal{R}} u$.

Exemple 5 Considérons la signature non typée $\{\oplus/2, 0/0, a/0\}$. Soit le système de réécriture $\mathcal{R}$ suivant qui modélise partiellement le ou exclusif.

$$\begin{array}{lcll} x \oplus 0 & \to & x & (r_u) \\ x \oplus x & \to & 0 & (r_i) \end{array}$$

$(a \oplus 0) \oplus a$ se réécrit en 0. En effet $(a \oplus 0) \oplus a \to_{\mathcal{R}} a \oplus a \to_{\mathcal{R}} 0$. En revanche $(0 \oplus a) \oplus a$ ne se réécrit pas en 0. Notons au passage que $(0 \oplus a) \oplus a$ est en forme normale.

2.2.3 Réécriture modulo E

Étant donné un ensemble d'équations E, u se réécrit modulo E par $\mathcal{R}$ en v, noté $u \to_{\mathcal{R}/E} v$, s'il y a un contexte t, une position p dans t, une règle $l \to r$ dans $\mathcal{R}$ et une substitution σ tels que $u =_E t[l\sigma]_p$ et $t[r\sigma]_p =_E v$.

$\mathcal{R}$ *termine modulo* E (ou est E*-terminant*) s'il n'y a pas de chaîne infinie $t_1 \to_{\mathcal{R}/E} t_2 \to_{\mathcal{R}/E} \ldots$. $\mathcal{R}$ est confluent modulo E (ou E*-confluent*) si et seulement si pour tous termes u et v tels que $t \to_{\mathcal{R}/E} u$ and $t \to_{\mathcal{R}/E} v$, il existe des termes u',v' tels que $u \to^*_{\mathcal{R}/E} u'$, $v \to^*_{\mathcal{R}/E} v'$, et $u' =_E v'$.

$\mathcal{R}$ est convergent modulo E (ou E*-convergent*) s'il est E-terminant et E-confluent.

Un terme est en *forme normale modulo* $\mathcal{R}/E$ s'il n'y a pas de terme u tel que $t \to_{\mathcal{R}/E} u$. Si $t \to^*_{\mathcal{R}/E} u$ et u est en forme normale, on dit que u est une forme normale de t. Quand cette forme normale est unique modulo E, nous écrivons $s = t\downarrow_{\mathcal{R}/E}$.

Exemple 6 (suite de l'exemple 5) Considérons à nouveau le système de réécriture $\mathcal{R}$ ainsi que la théorie équationnelle AC de l'exemple 4. Il est désormais vrai que $(0 \oplus a) \oplus a$ se réécrit en 0. En effet $(0 \oplus a) \oplus a =_{AC} (a \oplus 0) \oplus a \to_{\mathcal{R}} a \oplus a \to_{\mathcal{R}} 0$. 0 est la forme normale de $(0 \oplus a) \oplus a$ modulo $\mathcal{R}/AC$. On notera donc $(0 \oplus a) \oplus a \downarrow_{\mathcal{R}/AC} = 0$.

2.2.4 Décomposition de théories équationnelles

Nous disons que $(\mathcal{R}, E)$ est une *décomposition* de E' quand $\mathcal{R}$ est E-convergent, et $u =_{E'} v$ si et seulement si pour tous termes u, v, $u \downarrow_{\mathcal{R}/E} = v \downarrow_{\mathcal{R}/E}$.

2.3 Quelques primitives cryptographiques

Nous rappelons ici quelques propriétés de primitives cryptographiques que nous allons utiliser par la suite. Nous adoptons une notation infixe, plus habituelle, et présentons les propriétés en considérant les symboles $+$, $\cdot$, $\oplus$. Nous noterons l'application d'un symbole d'exponentiation à deux termes u et v, u^v.

2.3.1 Propriétés atomiques

Commençons par décrire certaines propriétés des opérateurs, exprimables par une unique équation.

Associativité (A)

Un symbole $+$ est associatif s'il satisfait la propriété d'*associativité* représentée par l'équation :

$$((x \ + \ y) \ + \ z) = (x \ + \ (y \ + \ z))$$

Commutativité (C)

Cette propriété est souvent associée avec la propriété de *commutativité*. Un symbole $+$ est commutatif s'il satisfait la propriété de commutativité représentée par l'équation :

$$(x \ + \ y) = (y \ + \ x)$$

Unité (U)

Considérons un ensemble S muni d'une loi de composition interne c. c a un *élément neutre* s'il existe un élément $x \in S$ tel que pour tout élément $y \in S$, $x \ c \ y = y \ c \ x = y$. La plupart du temps, nous nommerons cet élément 0 ou 1, cela sera précisé. En général, on utilise 0 pour représenter l'élément neutre d'une addition et 1 pour représenter l'élément neutre d'une multiplication.

Ainsi nous dirons qu'un symbole $+$ satisfait la propriété d'élément neutre ou d'*unité*, s'il satisfait la propriété représentée par l'équation suivante où 0 représente l'élément neutre :

$$(x \ + \ 0) = (0 \ + \ x) = 0$$

Nilpotence (N)

Considérons un ensemble S muni d'une loi de composition interne c. En supposant que c a un élément neutre 0, c est nilpotente si la composition d'un élément de S avec lui même est égale à 0.

Ainsi nous dirons qu'un symbole $\oplus$ est nilpotent s'il satisfait la propriété de nilpotence représentée par l'équation suivante où 0 représente un élément neutre pour $\oplus$:

$$x \ \oplus \ x = 0$$

Inverse (I)

Une loi de composition interne c sur un ensemble S a un *inverse* si pour tout élément x il y a un élément y tel que $x\ c\ y = y\ c\ x = 0$, où 0 est un élément neutre pour c. Cet élément sera parfois noté $-x$.

Ainsi nous disons qu'un symbole + satisfait la propriété d'inverse s'il satisfait la propriété représentée par l'équation suivante :

$$x\ +\ -x = 0 \qquad -x\ +\ x = 0$$

Distributivité (D)

Nous disons qu'un symbole $\cdot$ est *distributif à gauche* sur un symbole +, s'ils satisfont la propriété représentée par l'équation suivante :

$$x\ \cdot\ (y\ +\ z) = (x\ \cdot\ y)\ +\ (x\ \cdot\ z)$$

Nous disons qu'un symbole $\cdot$ est *distributif à droite* sur un symbole +, s'ils satisfont la propriété représentée par l'équation suivante :

$$(x\ +\ y)\ \cdot\ z = (x\ \cdot\ z)\ +\ (y\ \cdot\ z)$$

Nous disons qu'un symbole $\cdot$ est *distributif* sur un symbole +, s'il est distributif à gauche et à droite sur ce symbole.

2.3.2 Propriétés composées

Ou exclusif (ACUN)

Un symbole $\oplus$ représente de manière adéquate le ou exclusif, si comme nous l'avons noté il satisfait les propriétés d'associativité, de commutativité, d'unité et de nilpotence.

Exemple 7 Comme nous l'avons déjà fait dans les exemples 5 et 6, on utilise le symbole $\oplus$ pour représenter le ou exclusif et 0 pour représenter l'élément neutre de $\oplus$. Nous exprimerons donc les propriétés de $\oplus$ grâce à la théorie suivante.

$$\begin{array}{rcll} x \oplus (y \oplus z) & = & (x \oplus y) \oplus z & \text{(A)} \\ x \oplus y & = & y \oplus x & \text{(C)} \\ x \oplus 0 & = & x & \text{(U)} \\ x \oplus x & = & 0 & \text{(N)} \end{array}$$

Groupe commutatif (ACUI)

Un symbole + représente de manière adéquate les opérations qui ont lieu dans un groupe commutatif s'il satisfait les propriétés d'associativité et de commutativité, et si en outre il satisfait les propriétés d'élément neutre et d'inverse.

$$\begin{array}{rcll} x + (y + z) & = & (x + y) + z & \text{(A)} \\ x + y & = & y + x & \text{(C)} \\ x + 0 & = & x & \text{(U)} \\ x + (-x) & = & 0 & \text{(I)} \end{array}$$

Comme nous le ferons dans la partie II, il est possible d'adopter une notation multiplicative ($\cdot$ ou parfois $*$) pour les opérations qui ont lieu dans un groupe cyclique. Un groupe cyclique est un groupe de cardinal fini tel qu'il existe un élément dont la composition avec lui-même permet d'obtenir tous les éléments. Nous noterons 1 l'élément neutre. Comme annoncé, nous notons l'application d'un symbole d'exponentiation à deux termes u et v, u^v. Nous considérons alors que tout élément du groupe peut être représenté par un symbole g exponentié par une certaine valeur. En considérant les deux équations suivantes :

$$g^x \cdot g^y = g^{(x+y)} \qquad g^0 = 1$$

on obtient comme conséquence les quatre nouvelles équations à suivre qui nous montrent que l'on peut considérer que $\cdot$ représente les opérations qui ont lieu dans un groupe

$$\begin{aligned} g^x \cdot (g^y \cdot g^z) &= (g^x \cdot g^y) \cdot g^z && \text{(A)} \\ g^x \cdot g^y &= g^y \cdot g^x && \text{(C)} \\ g^x \cdot 1 &= g^x && \text{(U)} \\ g^x \cdot g^{-x} &= 1 && \text{(I)} \end{aligned}$$

Anneau commutatif (ACUID)

Deux symboles $+$ et $\cdot$ représentent les opérations qui ont lieu dans un anneau commutatif si l'un de ces symboles (nous chosissons $+$) représente les opérations qui ont lieu dans un groupe commutatif et si en plus l'autre symbole ($\cdot$) est associatif, commutatif et distributif sur $+$ et $\cdot$ a un élément neutre.

Exemple 8 Il est courant d'utiliser les symboles $+$ et $\cdot$ pour représenter les opérations usuelles dans un anneau. On utilise aussi souvent 0 pour représenter l'élement neutre de $+$ et 1 est l'élément neutre de $\cdot$. Nous exprimerons donc les propriétés de $+$ et $\cdot$ grâce à la théorie suivante.

$$\begin{aligned} (x+y)+z &= x+(y+z) && \text{(A)} & \qquad (x \cdot y) \cdot z &= x \cdot (y \cdot z) && \text{(A)} \\ x+y &= y+x && \text{(C)} & \qquad x \cdot y &= y \cdot x && \text{(C)} \\ 1 \cdot x &= x && \text{(N)} & \qquad 0+x &= x && \text{(N)} \\ x+(-x) &= 0 && \text{(I)} & \qquad x \cdot (y+z) &= (x \cdot y)+(x \cdot z) && \text{(D)} \end{aligned}$$

Nous pouvons remarquer que du fait de l'associativité et la commutativité de $\cdot$, l'équation (D) impose la distributivité à gauche mais aussi à droite de $\cdot$ sur $+$.

Exponentiation modulaire (E)

Considérons un symbole $\cdot$ représentant les opérations de groupe. Comme précédemment, nous notons l'application d'un symbole d'exponentiation à deux termes u et v, u^v. Ce symbole invisible satisfait les équations suivantes où 1 est l'élément neutre pour $\cdot$:

$$\begin{aligned} (x^y)^z &= x^{y \cdot z} \\ x^1 &= x \end{aligned}$$

Nous considérerons aussi une variante plus forte où $\cdot$ représente une opération distributive sur un troisième symbole $+$.

Première partie

Analyse des protocoles de groupe

Chapitre 3

Aperçu des protocoles de groupe et de leur analyse logique

Sommaire

Dans ce chapitre, nous expliquons brièvement les caractéristiques essentielles des protocoles de groupe et leurs utilisations (section 3.1). Nous illustrons ensuite notre propos en présentant deux exemples de tels protocoles (section 3.2). Enfin, nous passons en revue certains travaux visant à l'automatisation d'une telle analyse (section 3.3).

3.1 Qu'est-ce qu'un protocole de groupe ?

3.1.1 Spécificité essentielle des protocoles de groupe

La particularité des protocoles de groupe consiste simplement à être spécifié de manière à ce qu'un nombre non borné de participants puissent y prendre part. Dans ce cadre, le travail que nous nous proposons d'effectuer consiste à développer des méthodes permettant d'analyser ces protocoles tout en considérant cette particularité.

3.1.2 Autres spécificités des protocoles de groupe

De cette caractéristique découlent d'autres caractéristiques des protocoles de groupe. Un des aspects des protocoles de groupe rendant leur vérification plus ardue est l'évolution possible du groupe. Certains protocoles comme GKE présenté dans la sous-section 3.2.2, prennent en compte cette possibilité.

Différents types d'évolutions peuvent avoir lieu. L'ajout de certains participants, qui peut se faire participant par participant, mais qui peut aussi se décrire comme l'ajout massif d'un groupe de participants. Nous pouvons aussi évoquer le retrait de certains participants qui peut lui aussi se faire participant par participant de manière négociée, c'est-à-dire en respectant un sous-protocoles, ou de manière subite, typiquement dans le cas d'une panne de réseau. Toute une partie du groupe peut donc le quitter de manière brutale. La thèse récente de N. Chridi [Chr09] passe en revue de manière exhaustive toutes ces caractéristiques en particulier dans le chapitre 2.

3.1.3 Pourquoi analyser les protocoles de groupe en tant que tel ?

En effet, une méthode simple d'analyse des protocoles de groupe pourrait consister à considérer que le nombre de participants n'excédera jamais une certaine borne b. Cette hypothèse nous autoriserait alors à utiliser les méthodes habituelles d'analyse des protocoles sur les spécifications concernant tous les nombres possibles de participants inférieurs à b.

Ce n'est pas l'option que nous avons choisie. Plusieurs raisons peuvent justifier ce choix. La première est simple et pragmatique. Avant de renoncer à faire une analyse prenant en compte cette particularité, il faut au moins essayer de la faire. Deuxièmement, renoncer à cette analyse impose un travail consistant à déterminer soit une borne raisonnable pour tous les protocoles, soit établir pour chaque protocole ou chaque classe de protocoles (en supposant qu'une classification ait été faite) une borne raisonnable. Un tel travail ne semble pas si simple à première vue. Il concernerait davantage une étude pratique de l'utilisation des protocoles et des buts dans lesquels ils sont établis. Cela ne mènerait donc pas nécessairement à une analyse unifiée.

Enfin des raisons plus scientifiques s'imposent elles aussi. D'une part si l'on considère les protocoles de groupe comme un objet théorique, pourquoi ne mériteraient-ils pas d'être étudiés en tant que tel ? Une telle étude de la décidabilité de certaines propriétés de sécurité de ces protocoles peut élargir la connaissance que nous avons de la nature des protocoles. Ainsi de même que la connaissance de la décidabilité ou de la complexité d'un problème nous renseigne sur celui-ci, indépendamment du fait qu'il existe des heuristiques performantes pour le résoudre, savoir si une classe de protocoles est décidable ou non nous renseigne sur cette classe.

D'autre part il y a un argument théorique poussant à une étude des protocoles de groupe en tant que tel. Pour toute borne b au nombre de participants, il est possible de construire un protocole qui est sûr pour un nombre de participants inférieur ou égal à b et non sûr pour un nombre de participants supérieur à ce nombre.

Nous pouvons enfin ajouter une dernière remarque justifiant une analyse propre aux protocoles de groupe. Si sous certaines hypothèses nous parvenions à fixer une telle borne b, cette borne serait grande. Or dans ce cas l'analyse habituelle prend trop de temps et devient infaisable en pratique.

Nous avons donc tenté d'élaborer une méthode d'analyse qui permet de prouver automatiquement la sécurité des protocoles pour tout nombre de participants.

3.1.4 Applications actuelles et recueil

De nombreuses applications peuvent nécessiter l'établissement de communications sécurisées pour un nombre de participants indéterminé. Nous pouvons entre autres citer les systèmes

de réplication et de synchronisation [TvS02], la vidéo conférence, le travail coopératif supporté par ordinateur [Gre91a, Gre91b], certaines applications pair à pair [AS04].

De nombreux protocoles de groupe existent. Plusieurs recueils de ces protocoles peuvent être cités parmi lesquels [MS07] et [Man06]. Nous pouvons aussi noter que l'atelier « International Workshop on Group-Oriented Cryptographic Protocols » a été exclusivement dédié à ce sujet et a eu lieu en 2007.

3.2 Exemples de protocoles de groupe

Nous nous proposons de présenter deux protocoles distincts. Ces protocoles sont des protocoles d'établissement de clé. Le but de ces protocoles est de donner la possibilité à chacun de leurs participants de calculer une valeur commune censée être secrète. Cette valeur servira de clé afin de chiffrer des messages pendant un temps limité. Cette valeur sera donc appelée *clé de session*.

Le premier protocole, GDH-2 [STW96] (Diffie-Hellman de Groupe), est simple et bien connu. Il nous permettra d'illustrer le propos qui va suivre en demeurant relativement concis. L'autre protocole, GKE [BCEP04] (« Group Key Exchange ») est plus complexe, mais il nous permet de mettre en avant l'utilité des différentes primitives cryptographiques que nous avons choisi de modéliser, en particulier l'opération de ou exclusif (ACUN) et l'exponentiation modulaire (E). GKE est composé de plusieurs sous-protocoles dont nous ne présenterons que la partie GKE-setup.

Nous pouvons distinguer deux types de protocoles de groupe. D'une part, nous trouvons des protocoles comme GDH-2 [STW96] où tous les participants sont des pairs. Aucun serveur n'est nécessaire bien que parfois un des participants ait un statut spécial (initiateur par exemple). D'autre part, il existe des protocoles comme GKE.setup [BCEP04] qui nécessitent l'existence d'un serveur.

Dans les exemples suivants, nous représenterons une liste de messages $m_1, \ldots, m_n$ grâce à la notation $[m_1, \ldots, m_n]$. D'autre part, nous considérons que préalablement à chaque exécution, les participants au protocole s'accordent sur un groupe $\mathbb{G}$ dans lequel le problème de décision de Diffie-Hellman (DDH) est difficile, et sur un générateur α de ce groupe. Cette hypothèse garantit qu'étant donné trois éléments α^a, α^b et α^c, il est difficile de déterminer si le produit de a et de b est égal à c. Cela garantit en particulier qu'il est difficile d'obtenir a à partir de α^a.

3.2.1 Le protocole GDH-2

Le but de ce protocole présenté dans [STW96] est l'établissement d'une clé de session secrète entre n participants sans recourir à un serveur.

Dans une session à n participants, chaque participant, que nous pouvons désigner par un entier i génère un nonce $\mathcal{N}_i$. Comme nous l'avons annoncé dans la sous-section 1.3.5, nous pouvons représenter la fraîcheur de ces éléments grâce à des indices liant ce nonce au participant qui l'a généré, à la session dans laquelle il a été généré ou au nombre de participants dans la session dans laquelle il a été généré.

Le premier participant envoie $[\alpha, \alpha^{\mathcal{N}_1}]$ au second. Le i-ème participant (pour $1 < i < n$) attend de la part de son prédécesseur une liste de messages $[\alpha^{x_1}, \ldots, \alpha^{x_i}]$, et envoie $[\alpha^{x_i}, \alpha^{x_1 \cdot \mathcal{N}_i}, \ldots, \alpha^{x_i \cdot \mathcal{N}_i}]$. Le dernier participant, en recevant $[\alpha^{x_1}, \ldots, \alpha^{x_n}]$, envoie $[\alpha^{x_1 \cdot \mathcal{N}_n}, \ldots, \alpha^{x_{n-1} \cdot \mathcal{N}_n}]$ à tous les autres participants.

Exemple 1 (GDH-2 pour quatre participants) Dans le cas d'une exécution du protocole pour 4 participants la séquence de messages suivante est envoyée.

Envois	Message
$1 \rightarrow 2$	$[\alpha, \alpha^{\mathcal{N}_1}]$
$2 \rightarrow 3$	$[\alpha^{\mathcal{N}_1}, \alpha^{\mathcal{N}_2}, \alpha^{\mathcal{N}_1 \cdot \mathcal{N}_2}]$
$3 \rightarrow 4$	$[\alpha^{\mathcal{N}_1 \cdot \mathcal{N}_2}, \alpha^{\mathcal{N}_1 \cdot \mathcal{N}_3}, \alpha^{\mathcal{N}_2 \cdot \mathcal{N}_3}, \alpha^{\mathcal{N}_1 \cdot \mathcal{N}_2 \cdot \mathcal{N}_3}]$
$4 \rightarrow (1,2,3)$	$[\alpha^{\mathcal{N}_1 \cdot \mathcal{N}_2 \cdot \mathcal{N}_4}, \alpha^{\mathcal{N}_1 \cdot \mathcal{N}_3 \cdot \mathcal{N}_4}, \alpha^{\mathcal{N}_2 \cdot \mathcal{N}_3 \cdot \mathcal{N}_4}]$

La clé commune calculée est alors $\alpha^{\mathcal{N}_1 \cdot \mathcal{N}_2 \cdot \mathcal{N}_3 \cdot \mathcal{N}_4}$. Remarquons que chaque participant i pour $i < n$, peut calculer cette clé à partir du message envoyé par le dernier participant puisqu'il connaît le nonce manquant $\mathcal{N}_i$.

3.2.2 Le protocole GKE

Le protocole d'échange de clés GKE (Group Key Exchange) présenté dans [BCEP04] consiste en quatre algorithmes ou sous-protocoles décrits comme suit où nous notons $\mathcal{G}_c$ et $\mathcal{J}$ des groupes de participants. On suppose que $\mathcal{G}_c$ est un groupe de participants communiquant avec le serveur dont on s'attend à ce que les membres partagent une clé de session.

- GKE.KGen sert à établir des clés à long-termes.
- GKE.setup($\mathcal{J}$) initialise une session qui établit une clé de session entre les participants au groupe $\mathcal{J}$. On obtient ainsi $\mathcal{G}_c = \mathcal{J}$.
- GKE.join($\mathcal{J}$) est destiné à faire entrer les participants appartenant à $\mathcal{J}$ dans $\mathcal{G}_c$ sans lancer une procédure d'initialisation. Le but est de faire en sorte que les participants du nouveau groupe partagent une nouvelle clé de session.
- GKE.remove($\mathcal{J}$) est destiné à faire sortir les participants appartenant à $\mathcal{J}$ de $\mathcal{G}_c$, là encore sans lancer une procédure d'initialisation. Le but est donc de faire en sorte que les participants restants partagent une nouvelle clé de session.

Nous nous concentrons sur le GKE.setup($\mathcal{J}$). Dans une session à n participants, chaque participant i génère un nonce $\mathcal{N}_i$. Chaque participant envoie au serveur $[\alpha^{\mathcal{N}_i}, \sigma(\alpha^{\mathcal{N}_i})]$ où $\sigma(\alpha^{\mathcal{N}_i})$ représente une signature numérique. Trois fonctions de hachage $\mathcal{H}$, $\mathcal{H}_0$ et $\mathcal{H}_1$ sont utilisées. Après vérification des signatures des participants, le serveur augmente le compteur c, génère un nonce $\mathcal{N}$ et calcule la clé partagée suivante :

$$K = \mathcal{H}_0(c \parallel \{\alpha_i^{\mathcal{N}_i \mathcal{N}}\}_{i \in \mathcal{J}})$$

où $\mathcal{H}_0$ est une fonction de hachage, $\parallel$ représente une opération de concaténation et $\{\alpha_i^{\mathcal{N}_i \mathcal{N}}\}_{i \in \mathcal{J}}$ représente la concaténation des contributions des participants exponentiées par $\mathcal{N}$. Il envoie ensuite à chaque participant la valeur de c et $K_i = K \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_i \mathcal{N}})$.

Chaque participant calcule alors la clé de session comme cela est décrit ci-dessous :

$$K = K_i \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_i \mathcal{N}}) \qquad \text{et} \qquad sk = \mathcal{H}(K \parallel \mathcal{J} \parallel S)$$

où $\mathcal{J}$ est l'ensemble des identités des participants au protocole et S est l'identité du serveur.

Exemple 2 (GKE.setup pour quatre participants) Dans le cas d'une exécution du protocole pour 4 participants la séquence de messages suivante est envoyée.

Envoyeur	Message
$1 \to S$	$[\alpha^{\mathcal{N}_1}, \sigma(\alpha^{\mathcal{N}_1})]$
$2 \to S$	$[\alpha^{\mathcal{N}_2}, \sigma(\alpha^{\mathcal{N}_2})]$
$3 \to S$	$[\alpha^{\mathcal{N}_3}, \sigma(\alpha^{\mathcal{N}_3})]$
$4 \to S$	$[\alpha^{\mathcal{N}_4}, \sigma(\alpha^{\mathcal{N}_4})]$
$S \to 1$	$[c, \mathcal{H}_0(c\|\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_1\mathcal{N}})]$
$S \to 2$	$[c, \mathcal{H}_0(c\|\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_2\mathcal{N}})]$
$S \to 3$	$[c, \mathcal{H}_0(c\|\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_3\mathcal{N}})]$
$S \to 4$	$[c, \mathcal{H}_0(c\|\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(c \parallel \alpha^{\mathcal{N}_4\mathcal{N}})]$

La clé commune calculée est alors $\mathcal{H}(\mathcal{H}_0(c\|\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \parallel \mathcal{J} \parallel S)$.

Remarque 3. Ce type de protocoles a pour but de garantir certaines propriétés de sécurité en présence d'un intrus actif. En particulier, lors de la première phase, la présence de signatures $\sigma(\alpha^{\mathcal{N}_i})$ vise à garantir l'authenticité des participants distincts du serveur. Comme nous ne nous intéressons pas à un intrus actif et que nous nous concentrons sur la propriété de secret de certains messages, nous pouvons éviter de représenter ces signatures. De même la présence du compteur c vise à éviter les attaques par rejeu. Nous pouvons aussi omettre de les représenter.

Exemple 4 Ainsi d'après la remarque 3 une exécution de GKE.setup à 4 participants est représentée comme suit :

Envoyeur	Message
$1 \to S$	$\alpha^{\mathcal{N}_1}$
$2 \to S$	$\alpha^{\mathcal{N}_2}$
$3 \to S$	$\alpha^{\mathcal{N}_3}$
$4 \to S$	$\alpha^{\mathcal{N}_4}$
$S \to 1$	$\mathcal{H}_0(\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(\alpha^{\mathcal{N}_1\mathcal{N}})$
$S \to 2$	$\mathcal{H}_0(\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(\alpha^{\mathcal{N}_2\mathcal{N}})$
$S \to 3$	$\mathcal{H}_0(\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(\alpha^{\mathcal{N}_3\mathcal{N}})$
$S \to 4$	$\mathcal{H}_0(\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \oplus \mathcal{H}_1(\alpha^{\mathcal{N}_4\mathcal{N}})$

La clé commune calculée serait alors $\mathcal{H}(\mathcal{H}_0(\alpha^{\mathcal{N}_1\mathcal{N}} \parallel \ldots \parallel \alpha^{\mathcal{N}_4\mathcal{N}}) \parallel \mathcal{J} \parallel S)$.

3.3 Tentatives d'analyse automatique des protocoles de groupe

Le domaine de l'analyse logique des protocoles est bien exploré lorsque l'on considère un nombre borné de participants, et ce même pour un nombre non borné de sessions. Il y a moins de résultats concernant les protocoles dont l'exécution peut mettre en jeu un nombre non borné de participants.

Néanmoins différents travaux ont été menés dans le but d'analyser ces protocoles. G. Steel et A. Bundy ont développé l'outil CORAL [SB05, SB06], qui permet de chercher des attaques.

Plusieurs failles ont d'ailleurs été découvertes grâce à ce travail. Leur approche repose sur une technique visant à établir des preuves inductives appelée *preuve par consistance*. Cette méthode est complète pour la réfutation, ce qui permet de garantir la découverte des failles si elles existent. Néanmoins dans le cas où le protocole est sûr, il n'y a pas de garantie de terminaison de l'outil. Dans ce travail, des propriétés équationnelles visant par exemple à établir l'identité de certaines clés ont été prises en compte. Néanmoins des théories équationnelles plus classiques comme celles du ou exclusif (ACUN) ou de l'exponentiation modulaire (E) n'ont pas été prises en compte.

O. Pereira et J.J. Quisquater [PQ03, PQ06] ont quant à eux analysé un type spécifique de protocoles et ont prouvé l'impossibilité de construire des protocoles sûrs pour des propriétés d'authentification, en utilisant des primitives cryptographiques comme l'exponentiation modulaire et en se restreignant à un certain type de clé.

Les travaux les plus proches de notre approche sont ceux de R. Küsters et T. Truderung [Tru05, KT07] ainsi que le tout récent résultat de N. Chridi M. Rusinowitch et M. Turuani [CTR09].

[Tru05, KT07] proposent de représenter les protocoles de groupe par des clauses de Horn. Ils considèrent ce qu'ils nomment des protocoles récursifs où les calculs effectués par les participants peuvent être récursifs. Ceci autorise la représentation de protocoles où les messages échangés ont une taille non bornée. Leur approche permet en outre de représenter un intrus actif dont le pouvoir est lui aussi représenté par des clauses de Horn. Le résultat central de [Tru05] établit la décidabilité de ces protocoles récursifs sans considérer de théorie équationnelle sur les symboles représentant les opérations. Dans [KT07] ce résultat est étendu au cas où le protocole comporte des opérations ou exclusif. Malgré la puissance de ces résultats, deux inconvénients peuvent être relevés. Le premier est le fait que le chiffrement considéré doit être représenté par des clés atomiques. Le second est le fait qu'il ne considère pas davantage de théories équationnelles et de combinaisons de théories équationnelles.

[CTR09] ont quant à eux obtenu un résultat de décidabilité pour une classe de protocoles appelée « Protocoles bien-tagués avec des clés autonomes ». Certains protocoles (comme le protocole d'« asokan-ginzboorg »[AG99]) étant bien-tagués, ils ont défini une procédure qui leur permet d'obtenir des résultats concernant de tels protocoles. La seconde condition qui concerne les clés autonomes, qui leur est nécessaire pour prouver la terminaison de leur algorithme d'analyse demeure forte. Cette condition restreint la possibilité d'indicer des éléments des termes apparaissant en position de clé. Par exemple, la procédure d'analyse qu'ils proposent termine sur le protocole d'Asokan-Ginzboorg. Néanmoins, ce protocole ne satisfait pas la condition de clés autonomes.

Comme nous l'avons relevé, ces travaux ont pour but l'analyse de protocoles en présence d'un intrus actif, capable d'interagir avec le protocole et non pas simplement d'enregistrer tous les messages et d'essayer d'en obtenir quelque chose d'imprévu. Notre approche consiste en une analyse des protocoles en présence d'un intrus passif, ce qui est plus faible, mais qui a l'avantage de nous permettre de couvrir une classe plus large de protocoles. La propriété sur laquelle nous nous concentrons est une propriété de secret au sens de *l'atteignabilité*.

L'ensemble des résultats que nous présentons dans cette première partie a fait l'objet d'une publication [KMT08].

Chapitre 4

Représentation formelle des protocoles

Sommaire

Dans ce chapitre, nous définissons un cadre formel visant à représenter une classe de protocoles de groupe. Par « cadre formel », nous entendons la donnée d'une signature (section 4.1) munie d'une théorie équationnelle (section 4.2) et d'un système de construction de termes (section 4.4). La signature et sa théorie visent à représenter les messages échangés au cours d'un protocole, ceux qui sont construits par les participants (section 4.3), ainsi que ceux qui peuvent être construits par l'intrus. Le système de construction a pour but de représenter le pouvoir de l'intrus. Nous illustrons cette définition par la représentation des protocoles GDH et GKE. Nous définissons dans ce cadre la propriété de secret sur laquelle nous nous concentrons (section 4.5) .

4.1 Signature

Afin de représenter une classe de protocoles suffisamment large, qui comprend en particulier les protocoles GDH et GKE présentés dans les sous-sections 3.2.1 et 3.2.2, nous nous donnons la signature non typée suivante.

$$\mathcal{F} = \{\mathsf{pair}/2, \mathsf{enc}/2, \mathsf{exp}/2, \mathsf{mult}/2, \mathsf{xor}/2, H/1\} \uplus \mathcal{F}_0$$

Nous allons utiliser des notations plus habituelles pour certains des symboles de $\mathcal{F}$. Pour désigner une paire $\mathsf{pair}(u, v)$, nous écrirons ainsi $\langle u, v\rangle$, pour un chiffrement $\mathsf{enc}(u, v)$ nous écrirons $\{u\}_v$ et l'exponentiation $\mathsf{exp}(u, v)$ sera notée u^v. La multiplication mult et le ou

exclusif xor seront désignés par les notations infixes $\cdot$ et $\oplus$. Le symbole H désigne un symbole de fonction de hachage unaire. $\mathcal{F}_0$ est un ensemble infini de symboles de constantes pouvant représenter des nonces, des clés de chiffrement et éventuellement des éléments d'une structure algébrique comme le générateur d'un groupe.

Notons que la construction de paires peut nous servir à représenter la concaténation de messages. En effet, habituellement, pour représenter la concaténation, on utilise un opérateur associatif, qui peut être vu comme une paire associative. Or puisque nous nous restreignons à un intrus passif, il est aisé de constater que si un intrus peut déduire une paire de la forme $\langle\langle t_1, t_2\rangle, t_3\rangle$ par l'application des règles $proj_i$ et $pair$ l'intrus peut déduire $\langle t_1, \langle t_2, t_3\rangle\rangle$. Ainsi sans considérer explicitement un opérateur associatif pour représenter la concaténation on a que, si l'intrus possède une paire, il possède tous les éléments de sa classe d'équivalence modulo l'associativité de la paire.

4.2 Théorie équationnelle générique

Nous enrichissons $\mathcal{F}$ d'une théorie équationnelle représentée par le système de réécriture $\mathcal{R}_I$ modulo AC de $\oplus$ et $\cdot$.

$$\begin{array}{rcl} x \oplus 0 & \rightarrow & x \\ x \oplus x & \rightarrow & 0 \\ ((x)^y)^z & \rightarrow & x^{y\cdot z} \\ \langle x, y\rangle^z & \rightarrow & \langle x^z, y^z\rangle \end{array}$$

FIG. 4.1 – Système de réécriture $\mathcal{R}_I$

Les deux premières règles représentent l'élément neutre et la nilpotence du ou exclusif. La troisième règle de réécriture permet de normaliser les exponentiations imbriquées. Une première limitation provient du fait que nous ne représentons ni élément neutre ni inverse pour la multiplication.

La dernière règle permet de normaliser une liste de termes exponentiés avec le même terme. Notons que cela est utile pour représenter le protocole GKE.

Exemple 5 (utilisation de la règle $\langle x, y\rangle^z \rightarrow \langle x^z, y^z\rangle$ pour représenter GKE) Considérons une constante g représentant le générateur d'un groupe et un ensemble de constantes $\{N\} \cup \{N_i | i \leq k\}$ représentant des nonces. En admettant que nous puissions représenter la concaténation par la paire comme nous l'avons évoqué dans la sous-section 4.1, nous pouvons représenter la clé K calculée par le serveur par le terme $\mathcal{H}(\langle g^{N_1 N} \ldots \langle g^{N_{k-1} N}, g^{N_k N}\rangle \ldots\rangle)$. Grâce à cette règle nous pouvons aussi représenter ce terme comme suit : $\mathcal{H}(\langle g^{N_1} \ldots \langle g^{N_{k-1}}, g^{N_k}\rangle \ldots\rangle^N)$.

La terminaison et la confluence de ce système ont été prouvées en utilisant l'outil C*i*ME [CMMU00]. Dans le reste du papier nous considérerons uniquement des termes en forme normale modulo $\mathcal{R}_I/AC$. Nous considérerons que toutes les opérations sur les termes produisent des termes en forme normale. Nous pensons ici aux opérations qui peuvent être réalisées par un intrus représenté par le système décrit dans le paragraphe suivant.

4.3 Spécification des protocoles

Nous décrirons les protocoles par deux fonctions $e : \mathbb{N} \to 2^{\mathcal{T}(\Sigma)}$ et $k : \mathbb{N} \to 2^{\mathcal{T}(\Sigma)}$. e associe à un entier n l'ensemble des termes représentant les messages émis durant une exécution du protocole avec n participants. k associe à n un ensemble de secrets qui sont produits au cours d'une exécution, elle aussi avec n participants. Dans le cas d'un protocole d'établissement de clé, il s'agira typiquement de la clé de session établie.

Exemple 6 (Représentation de GDH-2 introduit dans la sous-section 3.2.1) .

$$e_{\mathrm{GDH}}(n) = \begin{cases} \emptyset & \text{si } n < 2 \\ \{t_1^n, \ldots, t_n^n\} & \text{sinon} \end{cases} \qquad \text{et} \qquad k_{\mathrm{GDH}}(n) = \begin{cases} \emptyset & \text{si } n < 2 \\ \{g^{N_1^n \cdot \ldots \cdot N_n^n}\} & \text{sinon} \end{cases}$$

où

$$\begin{aligned} t_1^n &= \langle g, g^{N_1^n} \rangle \\ t_i^n &= \langle g^{N_1^n \cdot \ldots \cdot N_{i-1}^n}, t_{i-1}^{N_i^n} \rangle \qquad (1 < i < n) \\ t_n^n &= \langle g^{N_2^n \cdot \ldots \cdot N_n^n}, \langle \ldots, \langle g^{N_1^n \cdot \ldots \cdot N_{j-1}^n \cdot N_{j+1}^n \cdot \ldots \cdot N_n^n}, \langle \ldots, g^{N_1^n \cdot \ldots \cdot N_{n-2}^n \cdot N_n^n} \rangle \rangle \rangle \ldots \rangle \end{aligned}$$

Exemple 7 (Représentation de GKE.setup introduit dans la sous-section 3.2.2) .

$$k_{\mathrm{GKE}}(n) = \begin{cases} \emptyset \text{ si } n < 2 \\ \{H(\langle H(\langle \langle g^{N_1^n}, \ldots \langle g^{N_{n-1}^n}, g^{N_n^n} \rangle \ldots \rangle^{N^n}), \langle 1, \ldots \langle 4, S \rangle \ldots \rangle \rangle)\} \end{cases}$$

et

$$e_{\mathrm{GKE}}(n) = \begin{cases} \emptyset & \text{si } n < 2 \\ \{t_{11}^n, t_{12}^n, \ldots, t_{n1}^n, t_{n2}^n\} & \text{sinon} \end{cases}$$

où

$$\begin{aligned} t_{i1}^n &= g^{N_i^n} \\ t_{i2}^n &= H(\langle g^{N_1^n}, \ldots \langle g^{N_{n-1}^n}, g^{N_n^n} \rangle \ldots \rangle^{N^n}) \oplus H(g^{N_1^n N^n}) \end{aligned}$$

4.4 Représentation formelle de l'intrus

Dans le modèle des protocoles cryptographiques jusqu'ici défini, nous pourrions représenter les capacités d'attaque de l'intrus, qui sont ici des capacités de déduction de nombreuses manières différentes. Nous choisissons de les représenter par un système de déduction. C'est une façon habituelle de procéder. Ce système est l'union du système DY décrit par la figure 4.2 et du système Ext décrit par la figure 4.3. I est le système complet, en d'autres termes $I = DY \cup Ext$. Dans les définitions des systèmes DY et Ext, un séquent $S \vdash t$, où $S \cup \{t\}$ est un ensemble de termes, exprime le fait que l'intrus peut déduire t de S.

Le système DY représente la capacité habituelle de déduction de l'intrus de Dolev-Yao, c'est-à-dire chiffrer (*enc*), déchiffrer un message avec une clé qu'il connaît (*dec*), extraire un des messages d'une paire (*proj*$_1$,*proj*$_2$), appliquer une fonction de hachage (*hash*), et enfin construire des paires (*pair*).

Comme il est d'usage de le faire, étant donnée une règle ou une instance de règle $\frac{S \vdash u_1, \ldots, S \vdash u_n}{S \vdash t}$, nous appelons t la *conclusion* et $u_1, \ldots, u_n$ les *prémisses* de cette règle. Notons que concernant

$$\frac{}{S \vdash t}\ axiom \text{ si } t \in S \qquad \frac{S \vdash t_1 \qquad S \vdash t_2}{S \vdash \langle t_1, t_2 \rangle}\ pair$$

$$\frac{S \vdash \langle t_1, t_2 \rangle}{S \vdash t_1}\ proj_1 \qquad \frac{S \vdash \langle t_1, t_2 \rangle}{S \vdash t_2}\ proj_2$$

$$\frac{S \vdash t_1 \qquad S \vdash t_2}{S \vdash \{t_1\}_{t_2}}\ enc \qquad \frac{S \vdash \{t_1\}_{t_2} \qquad S \vdash t_2}{S \vdash t_1}\ dec$$

$$\frac{S \vdash t}{S \vdash H(t)}\ hash$$

FIG. 4.2 – Système de Dolev-Yao DY

le système DY, la conclusion est en forme normale par rapport à $\mathcal{R}_I/AC$ quand ses prémisses le sont. Il n'est donc pas nécessaire de normaliser la conclusion.

Le système I étend les capacités de DY par des règles additionnelles qui permettent de représenter la capacité de l'intrus d'appliquer des fonctions, qui sont modélisées par des symboles apparaissant dans le système de réécriture $\mathcal{R}_I$.

$$\frac{S \vdash t_1 \qquad S \vdash t_2 \text{ et } t_2 \in \Sigma_0}{S \vdash t_1^{t_2} \downarrow_{\mathcal{R}_I/AC}}\ exp \qquad \frac{S \vdash t_1 \ \cdots \ S \vdash t_n}{S \vdash t_1 \oplus \cdots \oplus t_n \downarrow_{\mathcal{R}_I/AC}}\ Gxor$$

FIG. 4.3 – Extension de DY : Ext

Dans ce cas, on doit normaliser les termes qui apparaissent dans la conclusion de ces règles puisque l'application de l'exponentiation ou du $\oplus$ peut créer des redexes. La règle $Gxor$ est une généralisation de la règle xor binaire, permettant d'appliquer $\oplus$ à deux termes. Cette généralisation est possible grâce à l'associativité et à la commutativité de $\oplus$. Notons d'ailleurs que cette règle représente en réalité une infinité de règles, une règle pour chaque nombre possible de prémisses.

Comme cela est fait habituellement, les règles *pair*, *enc*, *hash*, *exp* et *Gxor* seront appelées règles de *construction* et $proj_1$, $proj_2$, et *dec*, règles de *déconstruction*. Remarquons que *Gxor* peut aussi être utilisée pour déduire un sous-terme d'un terme grâce à la nilpotence de $\oplus$. Si nous considérons par exemple deux constantes a et b de $\mathcal{F}_0$, appliquer *Gxor* aux séquents $S \vdash a \oplus b$ et $S \vdash b$ permet de déduire a, un sous-terme de $a \oplus b$.

Définition 1 (Déduction) *Une déduction de t à partir de S dans un système D est un arbre fini où tout nœud est étiqueté par un séquent. Un nœud étiqueté avec $S \vdash u$ a n fils $S \vdash v_1, \ldots, S \vdash v_n$ tels que $\frac{S \vdash v_1, \ldots, S \vdash v_n}{S \vdash u}$ est une instance d'une des règles de D. La racine est étiquetée par $S \vdash t$.*

Ainsi pour un ensemble de termes clos S et un terme clos t on écrit $S \vdash_D t$, s'il existe

une déduction du terme t à partir de l'ensemble S dans le système de déduction D. Nous parlerons ainsi de déduction de $S \vdash_D t$, pour désigner une déduction de $S \vdash t$ dans le système D. Nous omettrons d'écrire D lorsque le système D est clair à partir du contexte. Dans ce cas nous écrirons simplement $S \vdash t$.

Remarque 8. Si S est un ensemble de termes en forme normale, alors pour tout nœud $S \vdash u$ d'une déduction dans I, u est en forme normale.

La *taille* d'une déduction π notée $|\pi|$ est le nombre de nœuds dans l'arbre. Étant donné un séquent $S \vdash u$, une déduction π de $S \vdash_D u$ est *minimale* si pour toute déduction π' de $S \vdash_D u$, $|\pi| \leq |\pi'|$.

La clôture déductive d'un ensemble de termes S est l'ensemble de tous les termes qu'un intrus peut déduire à partir de S en utilisant D.

Définition 2 (Clôture déductive) *Soit D un système de déduction et T un ensemble de termes clos. La clôture déductive de T par D est*

$$D(T) = \{t \mid T \vdash_D t\}$$

4.5 Propriété de secret

4.5.1 Formalisation de la propriété de secret

Dans la suite, ce que nous appellerons une *spécification* de protocoles est simplement une paire d'ensembles éventuellement infinis de termes (E, K) où $E = \bigcup_{n\in\mathbb{N}} e(n)$ et $K = \bigcup_{n\in\mathbb{N}} k(n)$, où e et k sont les fonctions descriptives du protocole comme cela a été décrit dans la section 4.3. De plus, nous considérons que les constantes 0 et 1 qui sont respectivement les éléments neutres de $\oplus$ et $\cdot$ appartiennent toujours à une spécification E.

Étant donné une spécification de protocole (E, K) la sécurité d'un protocole, en rapport avec la propriété de secret (atteignabilité) reviendra donc à se demander si un élément supposé secret de K est déductible d'un ensemble de messages émis dans n'importe quel session, c'est-à-dire si $I(E) \cap K \stackrel{?}{=} \emptyset$. On supposera que $e(n)$ et $k(n)$, et donc E et K, sont clos sous associativité et commutativité du ou exclusif et de la multiplication.

Considérer la spécification d'un protocole comme la réunion des messages émis au cours des exécutions pour toutes les possibilités de nombres de participants est une approximation. En effet, il serait considéré qu'un protocole n'est pas sûr si un ensemble d'exécutions permettait de calculer un secret d'une exécution à j participants. Cette propriété de secret est plus forte que la propriété attendue qui impose simplement que le secret calculé dans une session à n participants ne soit pas accessible à partir des éléments émis pendant cette session. Remarquons en effet que $I(E) \cap K = \emptyset$ implique que, pour tout n, $I(e(n)) \cap k(n) = \emptyset$.

4.5.2 Analyse de la propriété de secret

Afin d'analyser le problème de sécurité des protocoles cryptographiques présentés comme deux fonctions e et k, nous opérons une série d'abstractions et de réductions que nous décrivons dans les paragraphes suivants. Le passage d'un problème 1 à un problème 2 grâce à l'une de ces abstractions ou réductions sera représenté par la flêche $\Downarrow$.

D'une analyse pour chaque nombre de participants à une analyse pour tous nombres de participants

Il s'agit là simplement de l'abstraction que nous avons présentée ci-dessus (sous-section 4.5.1).

$$\begin{array}{c} \forall n \geq 2\ I(e(n)) \cap k(n) \stackrel{?}{=} \emptyset \\ \Downarrow \\ I(\bigcup_{n\geq 2} e(n)) \cap \bigcup_{n\geq 2} k(n) \stackrel{?}{=} \emptyset \end{array}$$

Étant donnée la manière dont nous avons défini les spécifications (E, K), cette abstraction revient à l'abstraction suivante :

$$\begin{array}{c} \forall n \geq 2\ I(e(n)) \cap k(n) \stackrel{?}{=} \emptyset \\ \Downarrow \\ I(E) \cap K \stackrel{?}{=} \emptyset \end{array}$$

De I à DY (chapitre 5)

Dans le chapitre 5, nous établissons certaines conditions sur les spécifications (E, K), que nous appellerons conditions de bonne formation. Ces conditions nous permettent de considérer simplement un intrus DY habituel au lieu de considérer l'intrus I qui est capable d'appliquer des règles qui comportent des symboles pris dans des théories équationnelles.

$$\begin{array}{c} I(E) \cap K \stackrel{?}{=} \emptyset \\ \Downarrow \\ DY(E) \cap K \stackrel{?}{=} \emptyset \end{array}$$

Représentation par les $VTAM$ à contraintes structurelles (chapitre 6)

Dans le chapitre 6, afin de décider automatiquement le problème $DY(E) \cap K \stackrel{?}{=} \emptyset$, nous proposons de représenter les ensembles E et K par des automates d'arbres. Ces automates d'arbres sont des automates d'arbres particuliers, les $VTAM$ à contraintes structurelles. Une représentation exacte de ces ensembles semblant difficile, nous représentons des surapproximations des ensembles E et K, $App(E)$ et $App(K)$.

$$\begin{array}{c} DY(E) \cap K \stackrel{?}{=} \emptyset \\ \Downarrow \\ DY(App(E)) \cap App(K) \stackrel{?}{=} \emptyset \end{array}$$

Pour prendre en compte certaines restrictions liées aux $VTAM$ à contraintes structurelles, nous devrons coder par une fonction que nous nommerons ρ les termes des ensembles $(App(E))$ et $(App(K))$.

$$\begin{array}{c} DY(App(E)) \cap App(K) \stackrel{?}{=} \emptyset \\ \Downarrow \\ DY(\rho(App(E))) \cap \rho(App(K)) \stackrel{?}{=} \emptyset \end{array}$$

Afin de représenter l'associativité et la commutativité de certaines fonctions, nous choisissons de représenter un ensemble de témoins des classes d'équivalence modulo AC. Ces témoins seront obtenus par une fonction W.

$$\begin{array}{c} DY(\rho(App(E))) \cap \rho(App(K)) \stackrel{?}{=} \emptyset \\ \Downarrow \\ DY(W(\rho(App(E))) \cap W(\rho(App(K))) \stackrel{?}{=} \emptyset \end{array}$$

Nous pourrons alors définir des automates représentant ces ensembles.

$$\begin{array}{c} DY(W(\rho(App(E)))) \cap W(\rho(App(K))) \stackrel{?}{=} \emptyset \\ \Downarrow \\ DY(\mathcal{A}_{W(\rho(App(E)))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset \end{array}$$

Restera enfin à montrer que la clôture sous le système de DY est représentable par une méthode de complétion de ces automates.

$$\begin{array}{c} DY(\mathcal{A}_{W(\rho(App(E)))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset \\ \Downarrow \\ \mathcal{A}_{DY(W(\rho(App(E))))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset \end{array}$$

En bref

Le chemin est long, mais on passe du problème $\forall n \geq 2\ I(e(n)) \cap k(n) \stackrel{?}{=} \emptyset$
au problème $\mathcal{A}_{DY(W(\rho(App(E))))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset$.

Chapitre 5

Réduction de I à DY

Sommaire

Le but de ce chapitre est d'établir des critères permettant de simplifier le pouvoir de déduction de l'intrus. Nous définissons ici une notion de *bonne formation* des protocoles (section 5.2). Derrière cette définition de bonne formation, il n'y a rien de spécialement bon si ce n'est le fait que, sous les contraintes que cette définition impose aux protocoles qui les satisfont, il est possible réduire un intrus représentable par le système I à un intrus plus faible représentable par le système DY. Cette réduction est l'objet de la section 5.3. La classe dessinée par cette définition de bonne formation est assez large pour couvrir des protocoles existants. Elle prend en effet en compte à la fois des opérations modélisant les opérations d'exponentiation modulaire et le ou exclusif.

Avant de définir cette classe de protocoles et de prouver la possibilité de simplifier le pouvoir de l'intrus, nous établissons dans une première section (section 5.1) le caractère essentiellement syntaxique de l'une des règles de réécriture $\mathcal{R}_I$, et nous prouvons quelques lemmes techniques.

5.1 Hypothèses

5.1.1 Simplification du système de réécriture

Comme nous l'avons expliqué plus haut pour tenter d'être clair nous avons commencé par définir un système de déduction qui est plus pratique pour la représentation des protocoles.

Définition 3 (Système I') *Soit $\mathcal{R}'_I$ le système de réécriture $\mathcal{R}_I \setminus \{\langle x, y\rangle^z \to \langle x^z, y^z\rangle\}$. I' le système de déduction I où le système de réécriture utilisé dans les règles de Ext est $\mathcal{R}'_I$.*

Nous montrons d'abord que n'importe quel terme qui peut être déduit dans I peut aussi l'être grâce à I'. Ceci permet d'utiliser le système I' qui est plus pratique pour faire les démonstrations.

Lemme 1 *E un ensemble de termes en forme normale. Si $E \vdash_I t$ alors $E \vdash_{I'} t$, et si $E \vdash_{I \setminus Gxor} t$ alors $E \vdash_{I' \setminus Gxor} t$.*

Démonstration. La démonstration est la même pour les deux systèmes de déduction, donc on présente seulement la démonstration pour le système I. Soit π une déduction de $E \vdash_I t$.

Nous remplaçons de manière itérative les parties de π de la forme

$$\frac{E \vdash \langle u, v \rangle \qquad E \vdash c}{E \vdash \langle u', v' \rangle} exp$$

par

$$\dfrac{\dfrac{\dfrac{E \vdash \langle u, v \rangle}{E \vdash u}\, proj_1 \qquad E \vdash c}{E \vdash u^c}\, exp \qquad \dfrac{\dfrac{E \vdash \langle u, v \rangle}{E \vdash v}\, proj_2 \qquad E \vdash c}{E \vdash v^c}\, exp}{E \vdash \langle u^c, v^c \rangle \downarrow_{\mathcal{R}_I/AC}}\, pair$$

Il est facile de voir que cette transformation termine et conduit à une déduction de $\langle u', v' \rangle$. De plus, la déduction de $\langle u^c, v^c \rangle \downarrow_{\mathcal{R}_I/AC}$ que nous avons ainsi obtenue est telle que u et v n'ont pas un symbole paire à leur tête. Si tel n'était pas le cas nous aurions pu appliquer la transformation à nouveau. Donc le pas de normalisation dans $\mathcal{R}_I/AC$ coïncide avec le pas de normalisation dans $\mathcal{R}'_I/AC$ et nous avons ainsi une déduction dans I'. □

Il nous faut néanmoins noter qu'en toute généralité il est faux que $E \vdash_{I'} t$ implique que $E \vdash_I t$. Il est en effet possible de déduire un terme de la forme $\langle u, v \rangle^c$ dans I'. $\langle u, v \rangle^c$ ne serait pas en forme normale par rapport à $\mathcal{R}_I$.

5.1.2 Remarques sur les ensembles considérés

Remarque 9. De manière analogue à la remarque 8, si E est un ensemble de termes en forme normale, alors pour tout nœud $E \vdash u$ d'une déduction dans I', u est en forme normale.

D'après la définition de la spécification des protocoles section 4.5, pour toute spécification (E, K), $0 \in E$. Ceci nous permet de faire la remarque suivante qui nous permettra de simplifier certaines démonstrations.

Remarque 10. Les ensembles E que nous considérons dans les lemmes comportent tous 0.

Étant donné que la forme normale d'un terme est par définition équivalente à ce terme, on peut sans perte de généralité faire la remarque suivante.

Remarque 11. Les ensembles E considérés dans les lemmes et théorèmes de ce chapitre (chapitre 5) sont supposés être des ensembles de termes en forme normale.

5.1.3 Lemmes techniques

Nous donnons une définition descriptive de certains termes comportant des $\oplus$ en tête et nous prouvons les lemmes techniques suivants.

Définition 4 (Somme) *Un terme t est une* somme *si t est de la forme $t_1 \oplus t_2$.*

Lemmes techniques concernant le système de réécriture $\mathcal{R}'_I/AC$

Lemme 2 *Soit $t = u_1 \oplus \cdots \oplus u_n$ un terme en forme normale tel que les u_i ne sont pas des sommes et $t \neq 0$. Pour tout $v_1, \ldots, v_m$ qui ne sont pas des sommes, si $v_1 \oplus \cdots \oplus v_m \to^*_{\mathcal{R}'_I/AC} t$ alors pour tout $i \leq n$ il existe un $j \leq m$ tel que $v_j \to^*_{\mathcal{R}'_I/AC} u_i$.*

Démonstration. Par induction sur le nombre l d'étapes de réécriture.

Cas de base : $l = 0$. $u_1 \oplus \cdots \oplus u_n = v_1 \oplus \cdots \oplus v_m$. Trivialement, pour tout i il existe un j tel que $v_j \to^*_{\mathcal{R}'_I/AC} u_i$.

Cas inductif : $l = k + 1$. Soit $v'_1, \ldots, v'_{n'}$ des termes qui ne sont pas des sommes, tels que $v'_1 \oplus \cdots \oplus v'_{n'} \to^*_{\mathcal{R}'_I/AC} u_1 \oplus \cdots \oplus u_n$ en k étapes. Par hypothèse d'induction, pour tout $i \leq n$, il existe un $j \leq n'$ tel que $v'_j \to_{\mathcal{R}'_I/AC} u_i$. Remarquons que puisque $u_i \neq 0$ (t est en forme normale et $t \neq 0$) alors $v'_j \neq 0$. Montrons que pour tous termes $v_1, \ldots, v_m$ tels que $v_1 \oplus \ldots \oplus v_m \to_{\mathcal{R}'_I/AC} v'_1 \oplus \ldots \oplus v'_{n'}$ pour tout $i \leq n'$ tel que $v'_i \neq 0$ il existe un j tel que $v_j \to^*_{\mathcal{R}'_I/AC} v'_i$ (soit $v_j = v'_i$ soit $v_j \to_{\mathcal{R}'_I/AC} v'_i$, ce deuxième cas se produisant exactement pour un j) et v_j n'est pas une somme. Le résultat est immédiat si l'on considère les différentes possibilités d'application d'une règle de réécriture de $\mathcal{R}'_I/AC$. □

Lemme 3 *Soient $u_1, \ldots, u_n$ et t des termes en forme normale tels que $t \neq 0$ et les u_i et t ne sont pas des sommes. Si $u_1 \oplus \cdots \oplus u_n \to^*_{\mathcal{R}'_I/AC} t$, alors il existe un $m \leq n$ impair tel qu'il y a exactement m u_i tels que $u_i \neq 0$.*

Démonstration. Par induction sur le nombre l de pas de réécriture.

Cas de base : $l = 0$. $u_1 \oplus \cdots \oplus u_n = t$, et comme $t \neq u \oplus v$, $n = 1$ et donc n est impair et $u_1 = t \neq 0$.

Cas inductif : $l = k + 1$. On a donc $u_1 \oplus \cdots \oplus u_n \to^{k+1}_{\mathcal{R}'_I/AC} t$. Soit $v_1, \ldots, v_{n'}$ tels que $u_1 \oplus \cdots \oplus u_n \to_{\mathcal{R}'_I/AC} v_1 \oplus \cdots \oplus v_{n'} \to^k_{\mathcal{R}'_I/AC} t$ où les v_i ne sont pas des sommes. On doit donc considérer les deux possibilités suivantes d'applications de règle de réécriture pour $u_1 \oplus \cdots \oplus u_n \to_{\mathcal{R}'_I/AC} v_1 \oplus \cdots \oplus v_{n'}$.

- Soit la règle de réécriture est $x \oplus 0 \to x$. D'après l'hypothèse d'induction il existe un $m' \leq n'$ impair tel qu'il y a exactement m' v_i tels que $v_i \neq 0$. Soit $m = m'$. On a donc m est impair, $m \leq n$ et donc il y a exactement $m \leq n$ u_i tels que $u_i \neq 0$, ce qui correspond aux m' v_j.

– Soit la règle de réécriture utilisée est $x \oplus x \to 0$ et $x \neq 0$ (si $x = 0$ on suppose que l'on est dans le premier cas). Par hypothèse d'induction il existe un $m' \leq n'$ impair tel que il y a exactement m' v_i tels que $v_i \neq 0$. Soit $m = m' + 2$. Comme au moins un des v_i est 0, $m' < n'$. Comme d'autre part, $n = n' + 1$ on a $m \leq n$, m est impair et il y a exactement m u_i tels que $u_i \neq 0$, ce qui correspond aux m' v_j plus deux u_i qui sont égaux aux valeurs de x dans l'application de la règle de réécriture.

□

Lemmes techniques concernant le système I'

Lemme 4 *Pour toute instance* $\frac{E\vdash v_1 \ \ldots \ E\vdash v_n}{E\vdash u_1\oplus\cdots\oplus u_m}$ *de la règle Gxor tel que les* u_i *ne sont pas des sommes et les* v_i *sont en forme normale, pour tout* $i \leq m$ *il y a un* j *tel que soit* $v_j = u_i$ *soit* $v_j =_{AC} u_i \oplus u$ *pour un certain* u*.*

Démonstration. Le résultat découle directement des remarques 9 et 11 et du lemme 2. □

Le lemme suivant est similaire au Lemme 1 de [CS03]. Nous l'avons simplement adapté à notre contexte.

Lemme 5 *Soit* E *un ensemble de termes en forme normale. Si* π *est une déduction minimale dans* I' *d'une des formes suivantes :*

$$\frac{\dfrac{\vdots}{E \vdash \langle u, v\rangle}}{E \vdash u}\, proj_1 \qquad \frac{\dfrac{\vdots}{E \vdash \langle u, v\rangle}}{E \vdash v}\, proj_2 \qquad \frac{\dfrac{\vdots}{E \vdash \{u\}_v} \quad \dfrac{\vdots}{E \vdash v}}{E \vdash u}\, dec$$

Alors $\langle u, v\rangle \in St(E)$ *(resp.* $\langle u, v\rangle \in St(E)$*, resp.* $\{u\}_v \in St(E)$*).*

Démonstration. Supposons que la dernière règle soit $proj_1$. Les autres cas sont analogues. Nous affirmons que pour tout $n \geq 0$,

– soit il existe une branche dans π, c'est-à-dire une séquence de nœuds de longueur n, partant de $E \vdash \langle u, v\rangle$ telle que pour tout nœud $E \vdash t$ de cette branche, on a $\langle u, v\rangle \in St(t)$.
– soit il existe une telle branche de longueur $l \leq n$ qui est maximale, en ce sens que le dernier nœud est obtenu par la règle *axiom*.

Comme π est finie, il existe toujours un entier n tel que si notre affirmation est juste, nous sommes dans le second cas, et on conclut ainsi que $\langle u, v\rangle \in St(E)$.

Prouvons cette affirmation par induction sur n.

Cas de base. Si $n = 1$ nous concluons de manière immédiate car $\langle u, v\rangle \in St(\langle u, v\rangle)$.

Cas inductif. Supposons que l'affirmation est vraie pour n et montrons qu'elle l'est aussi pour $n+1$. Ou bien il y a une branche maximale de longueur $l \leq n$ telle que pour tout nœud $E \vdash t$ de cette branche on a $\langle u, v\rangle \in St(t)$. Alors c'est aussi le cas pour $n+1$. Ou bien il existe une séquence de règles de longueur n telle que pour tout nœud $E \vdash t$ on a $\langle u, v\rangle \in St(t)$. Soit $E \vdash t_f$ le dernier nœud de cette séquence. Considérons toutes les règles r' qui pourraient l'introduire. Si $r' = axiom$, la séquence de longueur n ($\leq n+1$) était une branche maximale.

Sinon il faut montrer qu'il y a une prémisse $E \vdash t'_f$ de r' telle que $\langle u, v\rangle \in St(t'_f)$. Si r' est une règle de déconstruction, comme par induction $\langle u, v\rangle \in St(t_f)$, $\langle u, v\rangle$ est trivialement un sous-terme de l'une des prémisses. Sinon nous sommes dans un des cas suivants.

- $t_f = \langle u, v\rangle$. Par minimalité de π $r' \neq pair$. r' ne peut être une règle exp, enc ou $hash$ à cause de la forme de t_f. r' est donc forcément une règle de déconstruction.
- $t_f \neq \langle u, v\rangle$. Par hypothèse d'induction on a $\langle u, v\rangle \in St(t_f)$.
  - Si $r' = pair$ alors $t_f = \langle u', v'\rangle$, et soit $\langle u, v\rangle \in St(u')$ soit $\langle u, v\rangle \in St(v')$. Comme $E \vdash u'$ et $E \vdash v'$ sont des prémisses de r' on peut conclure. On fait un raisonnement analogue si $r' = enc$ ou $r' = hash$.
  - Si $r' = Gxor$, si $t_f = u_1 \oplus \cdots \oplus u_n$ alors il existe un entier i tel que $\langle u, v\rangle \in St(u_i)$. Par le lemme 4, r' a une prémisse $E \vdash t'_f$ telle que $t'_f = u_i$ ou $t'_f =_{AC} u_i \oplus u'_i$ pour un certain u_i. On peut ainsi conclure que r' a une prémisse $E \vdash t'_f$ telle que $\langle u, v\rangle \in St(t'_f)$.
  - Si $r' = exp$ comme la déduction est dans I', $t_f = u'^{c_1 \cdots \cdot c_n}$ pour un certain u'. Si $\langle u, v\rangle \in St(t_f)$, alors $\langle u, v\rangle \in St(u')$.

□

5.2 Les protocoles bien formés

Afin de définir la bonne formation, nous définissons une fonction C sur les termes. Cette fonction représente le calcul d'une sur-approximation des constantes qui sont déductibles dans un terme donné. Cette fonction nous sera utile pour définir les conditions de bonne formation des protocoles.

Définition 5 *Soit $C : \mathcal{T}(\mathcal{F}) \to 2^{\mathcal{F}_0}$ la fonction définie de manière inductive comme suit :*

$$\begin{array}{rcll} C(c) & = & \{c\} & \text{si } c \in \mathcal{F}_0 \\ C(\langle u, v\rangle) & = & C(u) \cup C(v) & \\ C(\{u\}_v) & = & C(u) & \\ C(u_1 \oplus \cdots \oplus u_n) & = & \bigcup_{1 \leq i \leq n} C(u_i) & \text{où } u_1 \oplus \cdots \oplus u_n \text{ est en forme normale} \\ C(f(u_1, \ldots, u_n)) & = & \emptyset & \text{si } f \neq \langle ., .\rangle, \{.\}. \end{array}$$

On étend la définition d'ensembles de termes de manière naturelle, en ce sens que $c \in C(E)$ s'il existe $u \in E$ tel que $c \in C(u)$.

Le lemme suivant impose que si une constante c est dans $C(t)$, où t peut être déduit à partir d'un ensemble E, alors c est lui aussi dans $C(E)$. Une conséquence directe est que si une constante c peut être déduite à partir de E, alors c est dans $C(E)$.

Lemme 6 *Soit $c \in \mathcal{F}_0$ et t un terme tel que $c \in C(t)$. Si $E \vdash_{I'} t$ alors $c \in C(E)$.*

Démonstration. Soit π une déduction de $E \vdash_{I'} t$. Prouvons le résultat par induction sur $|\pi|$. Soit r la dernière règle de π.

Cas de base : $|\pi| = 1$. On a donc $r = axiom$. Puisque $t \in E$ et $c \in C(t)$ alors $c \in C(E)$.

Cas inductif. On considère toutes les possibilités pour r.

- $r = proj_1$. La racine de π a donc un unique successeur, étiqueté par $E \vdash \langle t, u\rangle$ pour un certain terme u. Comme $c \in C(t)$, par la définition 5, $c \in C(\langle t, u\rangle)$. On applique l'hypothèse d'induction sur la déduction π' de $E \vdash \langle t, u\rangle$ et on conclut que $c \in C(E)$.
- $r = proj_2$ ou $r = dec$. Ces cas sont similaires au cas où $r = proj_1$.
- $r = pair$. Il doit y avoir des termes u et v tels que $t = \langle u, v\rangle$. Par la définition 5 $C(\langle u, v\rangle) = C(u) \cup C(v)$. Comme $c \in C(\langle u, v\rangle)$ soit $c \in C(u)$ soit $c \in C(v)$. Soit π_1 et π_2 des déductions de $E \vdash u$ et $E \vdash v$. Par hypothèse d'induction sur π_1 ou π_2 on conclut que $c \in C(E)$.
- $r = enc$. Ce cas est similaire au cas où $r = pair$.
- $r = hash$. On a alors $t = H(u)$. Par la définition 5, $C(t) = \emptyset$. Ceci contredit le fait que $c \in C(t)$. Donc $r \neq hash$.
- $r = exp$. Il y a donc un u et un v tels que $t = u^v$. Par la définition 5, $C(t) = \emptyset$. Ceci contredit le fait que $c \in C(t)$. Donc $r \neq exp$.
- $r = Gxor$. Puisque E est un ensemble de termes en forme normale, par le lemme 4, une des prémisses de r est soit de la forme $E \vdash_{I'} t$ soit de la forme $E \vdash_{I'} t'$ où $t' =_{AC} t \oplus u$ pour un certain u.
  Dans le premier cas, on conclut immédiatement par les hypothèses d'induction. Considérons maintenant le cas où $E \vdash_{I'} t'$. Par la définition 5, $C(t') = C(t) \cup C(u)$. Donc $C(t) \subseteq C(t')$ et $c \in C(t')$. Par hypothèse d'induction $c \in C(E)$.

□

Corollaire 1 *Si $c \in \mathcal{F}_0$ et $E \vdash_{I'} c$ alors $c \in C(E)$.*

Démonstration. La démonstration est une application du lemme 6. □

Il nous faut imposer un certain nombre de restrictions sur les spécifications (E, K). Ces restrictions concernent l'utilisation de l'exponentiation modulaire et du ou exclusif pendant l'exécution du protocole. Ces restrictions concernent plus particulièrement les messages émis au cours d'un protocole E. Nous imposons aussi des contraintes sur l'ensemble des termes supposés secrets K.

Définition 6 (Bonne formation) *On dit que la spécification (E, K) d'un protocole est* bien formée *si elle satisfait les contraintes suivantes.*

1. *Si $t \in E$ et si $\oplus$ apparaît dans t, alors $t = u \oplus v$ pour certains termes u et v où*
   - *$\oplus$ n'apparaît ni dans u ni dans v*
   - *$u, v \notin DY(E)$.*
2. *Soit $t = u^{c_1 \cdot \cdots \cdot c_n}$ et $c_i \in \mathcal{F}_0$ pour tout $1 \leq i \leq n$. Si $t \in St(E \cup K)$ alors $c_1, \ldots, c_n \notin C(E)$*
3. *Pour n'importe quel $t \in K$, on a que $\oplus$ n'apparaît pas dans t.*

La contrainte 1 implique que $\oplus$ apparaît seulement en position de racine dans les termes de E et en plus $\oplus$ est d'arité 2. Remarquons que cette contrainte ne s'applique pas à l'intrus en ce sens qu'elle ne le contraint pas dans sa capacité de former des messages représentés par des termes où $\oplus$ a une arité supérieure à 2. En plus, la contrainte 1 requiert que ni u ni v ne puisse être déductible par un intrus Dolev-Yao.

La contrainte 2 requiert qu'on ne puisse pas accéder « facilement » à une constante apparaissant comme exposant dans un des termes de $E \cup K$. Cet accès facile à un terme est représenté par le fait qu'il se trouve dans $C(E)$. L'image par C de E étant en effet une surapproximation de l'ensemble des termes accessibles. L'ajout de cette contrainte semble assez naturel, puisque l'exponentiation est généralement utilisée pour cacher l'exposant. C'est par exemple le cas du protocole GDH-2.

Les contraintes que nous avons imposées sont restrictives, mais elles sont néanmoins satisfaites par un certain nombre de protocoles parmi lesquels figurent naturellement les protocoles donnés en exemple section 3.2.1 et 3.2.2. En particulier, la dernière contrainte impose que la spécification des ensembles de clefs ne doit pas comporter de $\oplus$.

Exemple 12 (Bonne formation de GDH-2) Pour constater que GDH-2 est bien formé, on passe en revue les différentes contraintes. La contrainte 1, est trivialement satisfaite puisque la spécification donnée dans l'exemple 6 du protocole ne comporte aucun $\oplus$. Il en est de même pour la contrainte 3. Pour s'assurer que la contrainte 2 est bien satisfaite, il est suffisant de constater que les constantes N_i^j qui sont les seuls éléments apparaissant comme exposant n'apparaissent jamais à une autre position dans les termes. Par définition de C on obtient que GDH-2 est bien formé.

Exemple 13 (Bonne formation de GKE) La contrainte 3 est trivialement satisfaite d'après la spécification donnée dans l'exemple 7. Pour s'assurer que la contrainte 2 est bien satisfaite, nous faisons la même remarque que dans l'exemple précédent (exemple 12). Enfin pour la contrainte 1, concernant le premier alinéa cela est simple, on constate que d'après la spécification, seuls les messages t_{i2}^n comportent un $\oplus$ et celui-ci est binaire et en tête. Pour le second alinéa, cela est moins direct. Puisque la contrainte 2 est satisfaite, l'intrus ne dispose pas des exposants des termes qui apparaissent comme sous-terme des termes, qui sont des arguments de $\oplus$. En particulier, il ne dispose pas de constantes de type N^n. Or il lui est nécessaire de former des termes comportant cette constante en exposant, puisque tous les termes qui sont des arguments de $\oplus$ en comportent. Il ne peut donc pas les obtenir par un simple système DY.

Nous n'avons pas trouvé de protocoles de groupe existant ne satisfaisant pas ces contraintes. Il serait possible d'en construire. Nous avons en revanche des exemples de protocoles à nombre fixé de participants qui ne satisfont pas ces contraintes.

Exemple 14 Nous présentons le protocole SmartRight view-only [Tho01] présenté dans [GTV03]. On peut le trouver dans le répertoire Spore [Spo]. Ce protocole est utilisé pour la protection contre la copie. Il est spécifié entre deux participants A et B de la manière suivante :

Envois	Message
$A \to B$	$\{N_K, CW \oplus N_R\}_{K_{AB}}$
$B \to A$	N_{Ri}
$A \to B$	$N_R, \{h(N_{Ri})\}_{N_K}$

Nous ne détaillerons pas la manière dont ce protocole est utilisé. Mais nous remarquons que le $\oplus$ est utilisé dans un message dans lequel il n'est pas en tête, ce qui n'est pas autorisé par la contrainte 1 de la Bonne formation.

5.3 DY est suffisant pour les protocoles bien formés

Le principal théorème de cette partie établit que si un protocole bien formé est attaquable par un intrus I alors il est attaquable par un intrus DY.

Théorème 1 *Pour tout (E, K) bien formé $I(E) \cap K = DY(E) \cap K$.*

La démonstration de ce théorème s'appuie sur de nombreux lemmes techniques. Nous commencerons dans un premier temps par montrer que si un séquent $E \vdash t$ peut se prouver sans utiliser la règle $Gxor$, alors il peut être prouvé sans utiliser la règle exp. Puis nous montrerons comment il est possible de montrer que l'on peut se passer de $Gxor$. Nous pourrons alors prouver le théorème 1.

5.3.1 Elimination de exp dans les déductions ne comportant pas de $Gxor$

Le lemme suivant (lemme 7) établit que s'il est possible de déduire un terme t à partir d'un ensemble E dans le système $I' \setminus Gxor$ alors cela est possible dans le système DY. L'hypothèse du lemme imposant que pour tout $u^{c_1 \cdots c_n} \in St(E, t)$ tel que $c_i \in \mathcal{F}_0$ on a $c_i \notin C(E)$, correspond à l'hypothèse selon laquelle t appartiendrait à un ensemble K tel que (E, K) satisfait la contrainte 2 de la bonne formation.

Lemme 7 *Soit E un ensemble de termes et t un terme. Si pour tout $u^{c_1 \cdots c_n} \in St(E, t)$ tel que $c_i \in \mathcal{F}_0$ on a $c_i \notin C(E)$ et si $E \vdash_{I' \setminus Gxor} t$ alors $E \vdash_{DY} t$.*

Démonstration. Soit π une déduction minimale de $E \vdash_{I' \setminus Gxor} t$. Par induction sur $|\pi|$ montrons que $E \vdash_{DY} t$. Soit r la dernière règle de π.

Cas de base : $|\pi| = 1$. Dans ce cas $r = axiom$ et donc π est une déduction dans DY.

Cas inductif : $|\pi| = n$. On considère les différentes possibilités pour la règle r.

- $r = pair$. On a donc $t = \langle u, v \rangle$ pour certains u et v et il existe une déduction π_1 de $E \vdash_{I' \setminus Gxor} u$ et une déduction π_2 de $E \vdash_{I' \setminus Gxor} v$ telles que $|\pi_1|, |\pi_2| \leq n$. Comme $u, v \in St(t)$, on a que pour tout $w^{c_1 \cdots c_n} \in St(E, u)$, $c_i \notin C(E)$, et pour tout $w^{c_1 \cdots c_n} \in St(E, v)$, $c_i \notin C(E)$. On peut donc appliquer l'hypothèse d'induction et obtenir que $E \vdash_{DY} u$ et $E \vdash_{DY} v$ et ainsi construire une déduction π' de $E \vdash_{DY} t$.
- $r = hash$ ou $r = enc$. Ces cas sont similaires au cas où $r = pair$.
- $r = proj_1$. Soit $E \vdash_{I' \setminus Gxor} \langle t, u \rangle$ la prémisse de r. Par hypothèse nous avons que pour tout $u^{c_1 \cdots c_n} \in St(E, t)$, $c_i \notin C(E)$. Par le lemme 5 $\langle t, u \rangle \in St(E)$. Il s'en suit que pour tout $u^{c_1 \cdots c_n} \in St(E, \langle t, u \rangle)$, $c_i \notin C(E)$. Puisque nous avons une déduction de $E \vdash_{I' \setminus Gxor} \langle t, u \rangle$ alors on peut appliquer notre hypothèse d'induction et conclure que l'on a une induction de $E \vdash_{DY} \langle t, u \rangle$ et par une application de $proj_1$, on obtient une déduction de $E \vdash_{DY} t$.
- $r = proj_2$ ou $r = dec$. Ces cas sont analogues à $r = proj_1$.
- $r = exp$. On a donc $t = v^{c_1 \cdots c_n}$. De plus, une des prémisses de exp est $c \in \{c_1, \cdots, c_n\}$. D'une part par hypothèse $c \notin C(E)$. D'autre part, on a une déduction π_1 de $E \vdash_{I'} c$. Par le corollaire 1, on a que $c \in C(E)$. On obtient ainsi une contradiction et on conclut que $r \neq exp$.

□

5.3.2 Elimination des règles *Gxor*

Le but de cette sous-section est de prouver le corollaire 2 établissant le fait que si un terme $t \in K$ est déductible par le système I' à partir de E tel que (E, K) est bien formée, il existe une déduction sans *Gxor*. Combiné avec le lemme 7, cela nous permettra d'établir le théorème 1.

Dans les lemmes de cette sous-section, nous allons considérer que les ensembles E que nous évoquons sont des ensembles qui pourraient apparaître dans des spécifications (E, K) bien formées, qui satisfont donc en particulier les contraintes 1 et 2 de la définition 6.

Le lemme suivant (lemme 8) établit que les déductions minimales de termes qui sont des sommes dans le système I' en partant d'un ensemble E sont soit l'application d'un simple axiome, soit une déduction dont la dernière règle est un *Gxor*.

Lemme 8 *Soit E un ensemble de termes satisfaisant la contrainte 1 de la définition 6 et soit π une déduction minimale de $E \vdash_{I'} u \oplus v$. π est alors de l'une des formes suivantes :*

$$\frac{}{E \vdash u \oplus v}\ axiom \qquad\qquad \frac{\begin{array}{ccc}\pi_1 & & \pi_n \\ \vdots & & \vdots \\ E \vdash t_1 & \dots & E \vdash t_n\end{array}}{E \vdash u \oplus v}\ Gxor$$

Démonstration. Soit r la dernière règle de π. Puisqu'il est exclu que r soit une règle de construction à cause de la forme de $u \oplus v$, r est une règle de déconstruction ($proj_1$, $proj_2$, dec), un axiome ou *Gxor*.

Supposons alors que la dernière règle soit $proj_1$, les autres cas étant similaires. Comme π est minimale et se termine par $proj_1$, d'après le lemme 5, pour un certain t, $\langle u \oplus v, t\rangle \in St(E)$. Par ailleurs, E satisfaisant la contrainte 1 de la définition 6, pour tout terme $t' \in E$ contenant $\oplus$, $t' = u' \oplus v'$ pour certains u' et v' et $\oplus$ n'est contenu ni dans u' ni dans v'. On obtient ainsi une contradiction avec le fait que $\langle u \oplus v, t\rangle \in St(E)$. Donc r ne peut être une règle de déconstruction.

Les seules possibilités qui restent pour r sont donc *axiom* et *Gxor*. □

En s'appuyant sur le lemme précédent, le lemme 9 établit qu'étant donné E, la conclusion $E \vdash t$ d'une déduction minimale dont la dernière règle serait une application de *Gxor*, aurait pour prémisse soit des sommes binaires, soit des termes qui ne sont pas des sommes. Par somme binaire on entend une somme de la forme $u \oplus v$ telle que ni u ni v ne sont des sommes.

Lemme 9 *Soit E un ensemble de termes satisfaisant la contrainte 1 de la définition 6, et soit π une déduction minimale de $E \vdash t$ dans I' de la forme suivante :*

$$\frac{\begin{array}{ccc}\pi_1 & & \pi_n \\ \vdots & & \vdots \\ \dfrac{}{E \vdash t_1}\ r_1 & \dots & \dfrac{}{E \vdash t_n}\ r_n\end{array}}{E \vdash t}\ Gxor$$

telle que les π_i sont des déductions de $E \vdash_{I' \setminus Gxor} t_i$. Alors pour tout t_i soit $t_i = u \oplus v$ ou $t_i = u$ pour certains u et v qui ne sont pas des sommes.

Démonstration. Par minimalité de π pour tout i, $r_i \neq Gxor$. D'après le lemme 8, si $t_i = u \oplus v$, $E \vdash t_i$ est introduit par un *axiom*. Donc, $t_i \in E$. Comme E satisfait la contrainte 1 de la définition 6, si $t_i = u \oplus v$, ni u ni v ne sont des sommes. □

Grâce aux lemmes 8 et 9 ainsi qu'un des lemmes techniques de la sous-section 3, nous établissons si la seule règle $Gxor$ de la déduction apparaît en dernier, alors t est une somme. Notons que nous utilisons aussi le lemme 7 grâce au fait que par hypothèse les prémisses de la dernière règle ne comportent elles-même pas de $Gxor$.

Lemme 10 *Soit E un ensemble de termes satisfaisant les contraintes 1 et 2 de la définition 6, et soit π la déduction minimale dans I' de la forme suivante*

$$\dfrac{\begin{array}{ccc}\pi_1 & & \pi_n \\ \vdots & & \vdots \\ E \vdash t_1 & \ldots & E \vdash t_n\end{array}}{E \vdash t}\, Gxor$$

telle que tout π_i est une déduction de $E \vdash t_i$ dans $I' \setminus Gxor$, alors t est une somme.

Démonstration. Par l'absurde. Supposons que t n'est pas une somme. Comme π est minimal et par la remarque 10, $0 \in E$ alors $t \neq 0$. D'après le lemme 4 il y a un i tel que soit $t_i = t$, soit $t_i =_{AC} t \oplus u$ pour un certain terme u. Par minimalité de π, il n'y a pas de t_i tels que $t_i = t$. Il y a donc un t_i tel que $t_i =_{AC} t \oplus u$.

Soit $t_1 \oplus \cdots \oplus t_n$ un terme égal à $v_1 \oplus \cdots \oplus v_m$ tel que les v_i ne sont pas des sommes. Par minimalité de π, tous $t_j \neq 0$. Comme les t_j sont en forme normale par définition de I', $t_j \neq_{AC} u \oplus 0$. Il s'en suit que tout $v_i \neq 0$. Comme $t \neq 0$ et puisque t n'est pas une somme et comme $t_1 \oplus \cdots \oplus t_n \rightarrow^*_{\mathcal{R}'_I/AC} t$, par le lemme 3, m est impair.

Par le lemme 9, cela implique qu'il y a un k tel que t_k n'est pas une somme. Par ailleurs, la preuve π est de la forme suivante :

$$\dfrac{\begin{array}{ccccc}\pi_i & & \pi_k & & \\ \vdots & & \vdots & & \\ E \vdash t \oplus u & & E \vdash t_k & \ldots\end{array}}{E \vdash t}\, Gxor$$

où ni u ni t ni t_k ne sont des sommes et tels que $u \oplus t_k \oplus \cdots =_{\oplus} 0$. Alors soit $u = t_k$ soit t_k apparaît dans un t_l où $l \neq i$. t_k n'apparaît pas seul dans t_l par minimalité (sinon les deux occurences de t_k dans la preuve s'annuleraient), donc $t_l =_{AC} t_k \oplus v$. Il existe donc un l tel que $t_l =_{AC} t_k \oplus v$ pour un certain v.

Par le lemme 8, $t_k \oplus v \in E$, et donc $t_k \in St(E)$. Comme E satisfait la contrainte 2, alors pour tout $u^{c_1 \cdots c_n} \in St(E, t_k)$ tel que $c_i \in \mathcal{F}_0$, $c_i \notin C(E)$. Comme π_i est une déduction de $E \vdash_{I' \setminus Gxor} t_k$, par le lemme 7, $E \vdash_{DY} t_k$. Comme $t_k \oplus v \in E$ on obtient une contradiction avec le fait que E satisfait la contrainte 1 de la définition 6. □

Le lemme suivant est obtenu grâce aux contraintes de bonne formation et au lemme 5 prouvé dans la sous-section 5.1.3. Il établit que si l'on considère une sous-déduction maximale (d'une certaine déduction minimale), ne comportant pas de $Gxor$ ayant des feuilles comportant des sommes, alors ces sommes sont des sous-termes de la conclusion.

Lemme 11 *Soit E un ensemble satisfaisant les contraintes 1 et 2 de la définition 6. Soit π une déduction minimale de $E \vdash_{I'} t$. Soit π' le sous-arbre de π construit comme suit : en commençant par la racine $E \vdash_{I'} t$ on suit chaque branche jusqu'à ce que l'on atteigne soit une feuille soit un nœud obtenu par l'application d'une règle Gxor. Dans le second cas, on coupe la branche telle que la feuille de cette branche soit un nœud correspondant à la conclusion de la règle Gxor. Alors s'il existe une feuille de π', qui est étiquetée par $E \vdash_{I'} u \oplus v$, on a que $u \oplus v \in St(t)$.*

Démonstration. Par induction sur $|\pi'|$.

Cas de base : $|\pi'| = 1$. La seule feuille est $E \vdash_{I'} t$. Donc si $t = u \oplus v$, trivialement $u \oplus v \in St(t)$.

Cas inductif : $|\pi'| = n$. Soit r la dernière règle de π'. On considère les différentes possibilités pour la dernière règle r.

- $r = pair$. $t = \langle t_1, t_2 \rangle$ pour certains t_1 et t_2 et $E \vdash_{I'} t_1$ et $E \vdash_{I'} t_2$ sont les prémisses de r. Soit π'_1 et π'_2 les sous-arbres définis de manière identique à π' mais dont les racines sont $E \vdash_{I'} t_1$ et $E \vdash_{I'} t_2$. S'il y a une feuille de π', qui est étiquetée $E \vdash_{I'} u \oplus v$, alors c'est aussi le cas pour π'_1 ou π'_2. Comme $|\pi'_1| < n$ et $|\pi'_2| < n$, on peut appliquer l'hypothèse d'induction. On obtient alors que $u \oplus v \in St(t_1)$ ou $u \oplus v \in St(t_2)$. Comme $t = \langle t_1, t_2 \rangle$, $u \oplus v \in St(t)$.
- $r = enc$, $r = exp$, $r = hash$. Ces cas sont analogues au précédent.
- $r = proj_1$. On montre que cela impliquerait une contradiction avec le fait que E satisfait les contraintes 1 et 2 de la définition 6. Soit $E \vdash_{I'} \langle t, w \rangle$ la prémisse de r. Soit π'^{-} le sous-arbre défini de manière identique à π' mais dont la racine est $E \vdash_{I'} \langle t, w \rangle$. S'il y a une feuille de π', qui est étiquetée $E \vdash_{I'} u \oplus v$, alors c'est aussi le cas pour π'^{-}. Comme $|\pi'^{-}| < n$, on peut appliquer l'hypothèse d'induction. Nous avons donc que $u \oplus v \in St(\langle t, w \rangle)$. De plus comme π est une déduction minimale de $E \vdash_{I'} t$, par le lemme 5 $\langle t, w \rangle \in St(E)$. Comme $u \oplus v \in St(\langle t, w \rangle)$ et $u \oplus v \neq \langle t, w \rangle$, c'est une contradiction avec la contrainte 1 de la définition 6. Donc $r \neq proj_1$.
- $r = proj_2$, $r = dec$. Ces cas sont similaires à $r = proj_1$.
- $r \neq Gxor$ par construction de π'.

□

Nous prouvons le lemme 12, qui est véritablement central en ce sens qu'il combine les différents lemmes établis jusqu'ici, et qu'il nous permet d'obtenir le résultat central (corollaire 2) de cette sous section.

Lemme 12 *Soit E un ensemble de termes satisfaisant les contraintes 1 et 2 de la définition 6. Soit π une déduction minimale de $E \vdash_{I'} t$. Si π comporte une application d'une règle Gxor, alors $\oplus$ apparaît dans t.*

Démonstration. Par induction sur le nombre $|\pi|_{Gxor}$ d'applications de la règle $Gxor$ dans π.

Cas de base : $|\pi|_{Gxor} = 0$. π ne comporte pas d'instance de la règle $Gxor$. On a donc trivialement le résultat.

Cas inductif : $|\pi|_{Gxor} = n$**.** Distinguons deux cas.

- Toute instance de règle *Gxor* apparaît dans une branche différente de π. Soit π' le sous-arbre maximal de π, qui ne contient pas d'instance de règle *Gxor* dont la racine est $E \vdash_{I'} t$, défini à la manière du lemme 11. Soit $E \vdash_{I'} t_f$ une feuille de π' introduite dans π par une application de la règle *Gxor* r. Comme la déduction des prémisses de r ne comporte aucune application de *Gxor* et comme π est minimal, d'après le lemme 10, $t_f = u \oplus v$ pour certains u et v. D'après le lemme 11, $u \oplus v \in St(t)$.
- Il existe une branche qui comporte au moins 2 règles *Gxor*. Considérons une telle branche. Soit r l'instance de la règle *Gxor* la plus basse, c'est-à-dire celle qui est la plus proche de la racine. Soit $E \vdash_{I'} t_f$ le nœud introduit par r.
  - Si t_f est une somme, nous faisons un raisonnement similaire à celui du cas où toutes les instances de *Gxor* sont sur des branches distinctes de π.
  - Si t_f n'est pas une somme alors on obtient une contradiction avec la contrainte 1 de la définition 6. En effet r est alors de la forme suivante :

$$\dfrac{\begin{array}{ccc}\pi_1 & & \pi_n \\ \vdots & & \vdots \\ E \vdash t_1 & \dots & E \vdash t_n\end{array}}{E \vdash t_f}\, Gxor$$

  Comme π est minimale et par la remarque 10, $0 \in E$ alors $t_f \neq 0$. Par le lemme 4, il existe un entier i tel que soit $t_i = t_f$ ou $t_i =_{AC} t_f \oplus u$ où u est un terme. Par la minimalité de π, il n'y a pas de t_i tel que $t_i = t_f$. Donc, il y a un t_i tel que $t_i =_{AC} t_f \oplus u$.
  Soit $t_1 \oplus \cdots \oplus t_n$ un terme de la forme $v_1 \oplus \cdots \oplus v_m$ où les v_i ne sont pas de sommes. Par minimalité de π, tous $t_i \neq 0$. Comme les t_i sont en forme normale, par définition de I', $t_i \neq_{AC} u \oplus 0$. Il s'en suit que tous les $v_i \neq 0$. Comme $t_f \neq 0$ et t_f n'est pas une somme et comme $t_1 \oplus \cdots \oplus t_n \to^*_{\mathcal{R}'_I} t_f$, d'après le lemme 3, m est impair.
  D'après le lemme 9, ceci implique qu'il existe un k tel que t_k n'est pas une somme. Par ailleurs, la preuve π est de la forme suivante :

$$\dfrac{\dfrac{\begin{array}{ccc}\pi_i & & \pi_k \\ \vdots & & \vdots \\ E \vdash t_f \oplus u & E \vdash t_k & \dots\end{array}}{E \vdash t_f}\, Gxor}{\vdots}$$

  où ni u ni t ni t_k ne sont des sommes et tels que $u \oplus t_k \oplus \cdots =_{\oplus} 0$. Alors soit $u = t_k$ soit t_k apparaît dans un t_l où $l \neq i$. t_k n'apparaît pas seul dans t_l par minimalité (sinon les deux occurences de t_k dans la preuve s'annuleraient), donc $t_l =_{AC} t_k \oplus v$. Il existe donc un l tel que $t_l =_{AC} t_k \oplus v$ pour un certain v.
  Il n'y a pas d'application de *Gxor* dans les $\pi_{i'}$ les que $t_{i'}$ est une somme puisque par le lemme 8 et la minimalité de π, $\pi_{i'} = \frac{}{E \vdash t_{i'}}$. Alors il y a une application de la règle *Gxor* dans une déduction $\pi_{i''}$ de $E \vdash_{I'} t_{i''}$ tel que $t_{i''}$ n'est pas une somme. Comme $|\pi_{i''}|_{Gxor} < n$, par hypothèse d'induction, $\oplus$ apparaît dans $t_{i''}$. Comme les t_i de la forme $t_{i''} \oplus v$ a été introduit par un axiome, on a une contradiction avec la contrainte 1 de la définition 6.

□

Le corollaire suivant permet d'établir que dans le cas d'un protocole bien formé (E, K), s'il est possible de déduire un terme t de K à partir de E, alors une déduction minimale de $E \vdash t$ ne comporte pas de règle $Gxor$. Ce corollaire est obtenu grâce à la contrainte de bonne formation portant sur K et grâce au lemme 12.

Corollaire 2 *Soit (E, K) un protocole bien formé. Toute déduction minimale de $E \vdash_{I'} t$, telle que $t \in K$, ne comporte pas d'application de la règle $Gxor$.*

Démonstration. Par l'absurde. Soit π un déduction minimale de $E \vdash_{I'} t$ qui comporte une règle $Gxor$. Comme (E, K) est un protocole bien formé, d'après le lemme 12, $\oplus$ apparaît dans t. Cependant, comme $t \in K$ et (E, K) est bien formé, par la contrainte 3, $\oplus$ n'apparaît pas dans t. □

5.3.3 Preuve du théorème 1

Nous sommes désormais en mesure de prouver le théorème 1.

Démonstration (du théorème 1). De manière immédiate $DY(E) \cap K \subseteq I(E) \cap K$ puisque DY est un sous-système de I. Afin de prouver l'autre direction, considérons un terme t de K et supposons que $E \vdash_I t$. D'après le lemme 1, il y a une déduction de $E \vdash_{I'} t$. Comme (E, K) est bien formé et $t \in K$, d'après le corollaire 2, il existe une déduction de $E \vdash_{I' \setminus Gxor} t$. Donc par le lemme 7, il existe une déduction de $E \vdash_{DY} t$. □

Chapitre 6

Représentation des protocoles par les automates d'arbres

Sommaire

Après avoir prouvé que sous certaines conditions nous pouvons ne considérer qu'un intrus représenté par le système DY, nous proposons une représentation des messages échangés et des secrets établis. Une des difficultés essentielles que l'on rencontre en essayant de trouver une représentation des protocoles de groupe est la représentation de l'infinité des ensembles possibles de messages échangés. Pour cela nous avons choisi une certaine classe d'automates d'arbres. Ces *automates d'arbres avec visibilité, mémoire et contraintes* (VTAM à contraintes) ont été introduits dans [CJP08]. Nous gardons le sigle anglais VTAM qui signifie *visibly tree automata with memory*.

Nous commençons donc par décrire ces automates (section 6.1). Nous expliquons certaines adaptations de notre signature nécessaires pour obtenir une représentation dans les VTAM (section 6.2). Nous expliquons ensuite comment nous représentons certaines caractéristiques de notre théorie équationnelle (section 6.3). Puis nous décrivons comment il est possible de représenter les capacité d'un intrus modélisé par un système DY dans ce modèle par automates (section 6.4). Nous proposons enfin d'écrire la représentation du protocole GDH-2 (section 6.5).

6.1 Les automates d'arbres à mémoire, avec visibilité et contraintes structurelles

Nous rappelons ici quelques notions de base concernant les automates d'arbres et définissons les VTAM à contraintes présentés dans [CJP08]. Précisons avant tout que de même que dans le chapitre 5, les signatures que nous évoquons dans ce chapitre sont des signatures non typées.

6.1.1 Automates d'arbres

Nous utilisons des automates d'arbres tels qu'ils ont été définis dans [CDG+07]. Plus particulièrement, nous utilisons des automates ascendants sur des termes clos finis dans $T(\mathcal{F})$.

Formellement un automate d'arbres ascendant (TA) $\mathcal{A}$ sur une signature $\mathcal{F}$ est un triplet (Q, Q_f, Δ), où

- $\mathcal{F}$ est la signature de calcul,
- Q est un ensemble fini de symboles d'états disjoint de $\mathcal{F}$,
- $Q_f \subseteq Q$ est l'ensemble des états finaux
- et Δ est un ensemble de règles de réécriture de la forme : $f(q_1, \ldots, q_n) \to q$ telles que $f \in \mathcal{F}$ et $q_1, \ldots, q_n \in Q$.

Cet ensemble de règles Δ peut ainsi être vu comme un système de réécriture permettant de réécrire un terme dans un état. Un terme est *accepté* (ou *reconnu*) par $\mathcal{A}$ dans l'état q si et seulement si $t \to_\Delta^* q$. Le langage $L(\mathcal{A}, q)$ de $\mathcal{A}$ dans l'état q est l'ensemble des termes clos acceptés par q. Le langage $L(\mathcal{A})$ de $\mathcal{A}$ est $\bigcup_{q \in Q_f} L(\mathcal{A}, q)$.

Exemple 15 Nous présentons l'exemple utilisé dans [CDG+07]. Soit $\mathcal{F}$ la signature $\{f/2, g/1, a/0\}$, et $\mathcal{A}$ l'automate (Q, Q_f, Δ) tel que

- $Q = \{q_a, q_g, q_f\}$
- $Q_f = \{q_f\}$
- Δ est l'ensemble de transitions suivant :

$$\begin{array}{llllllll} \{ & a & \to & q_a, & & g(q_a) & \to & q_g, \\ & g(q_g) & \to & q_g, & & f(q_g, q_g) & \to & q_f \quad \} \end{array}$$

Donnons deux exemples de réduction :

$$f(a, a) \to_\Delta f(q_a, a) \to_\Delta f(q_a, q_a)$$

$$f(g(a), g(a)) \to_\Delta^* f(g(q_a), g(q_a)) \to_\Delta^* f(q_g, q_g) \to_\Delta q_f)$$

Étant donné que $q_f \in Q_f$, $f(g(a), g(a))$ est accepté par $\mathcal{A}$.

6.1.2 Automates d'arbres avec mémoire

Les automates d'arbres avec mémoire (TAM) sont une extension des automates d'arbres dans laquelle une mémoire est maintenue tout au long de la réécriture par l'automate. On considère donc une signature non typée $\mathcal{G}$ permettant de représenter la mémoire. On distinguera désormais la *signature d'entrée* $\mathcal{F}$ et la *signature de mémoire* $\mathcal{G}$.

Comme pour les automates d'arbres présentés ci-dessus, on peut utiliser un système de réécriture pour représenter les calculs effectués par un automate d'arbres à mémoire. Dans ce but, on ajoute un argument aux symboles d'états, qui contiendra la mémoire. Une configuration d'un TAM qui est dans l'état q et dont le contenu de la mémoire est un terme clos $m \in T(\mathcal{G})$ est noté $q(m)$.

Les TAM sont donc définis comme suit.

Définition 7 (TAM) *Un* automate d'arbres ascendant avec mémoire (TAM) *sur une signature $\mathcal{F}$ est un tuple $(\mathcal{G}, Q, Q_f, \Delta)$ où*

- *$\mathcal{G}$ est la signature de la mémoire,*
- *Q est un ensemble fini de symboles d'états unaires, disjoint de $\mathcal{F} \cup \mathcal{G}$,*
- *$Q_f \subseteq Q$ est l'ensemble des états finaux*
- *et Δ est un ensemble de règles de réécriture de la forme $f(q_1(m_1), \ldots, q_n(m_n)) \to q(m)$ où $f \in \mathcal{F}_n$, $q_1, \ldots, q_n, q \in Q$ et $m_1, \ldots, m_n, m \in T(\mathcal{G}, \mathcal{X})$.*

Un terme t est *accepté* par $\mathcal{A}$ dans l'état $q \in Q$ avec la mémoire $m \in T(\mathcal{G})$ si et seulement si $t \to_\Delta^* q(m)$, et le langage de cet automate $L(\mathcal{A}, q)$ est défini comme suit :

$$L(\mathcal{A}, q) = \{t \mid \exists m \in T(\mathcal{G}), t \to_\Delta^* q(m)\}$$

Le langage de l'automate $\mathcal{A}$ est l'union des langages de $\mathcal{A}$ dans ses états finaux, noté $L(\mathcal{A}) = \bigcup_{q \in Q_f} L(\mathcal{A}, q)$.

Exemple 16 Nous modifions l'exemple 15 afin d'obtenir un automate d'arbres à mémoire. Soit $\mathcal{F}$ la signature $\{f/2, g/1, a/0\}$ et $\mathcal{A}_m = (\mathcal{G}, Q, Q_f, \Delta)$ tel que

- $\mathcal{G} = \{S/1, \bot/0\}$
- $Q = \{q_a, q_g, q_f\}$
- $Q_f = \{q_f\}$
- Δ est l'ensemble de transitions suivant :

$$\begin{array}{llll} \{ & a & \to & q_a(\bot), \\ & g(q_g(x)) & \to & q_g(x), \\ & g(q_a(x)) & \to & q_g(S(x)), \\ & f(q_g(x), q_g(\bot)) & \to & q_f(S(\bot)), \\ & f(q_g(x), q_a(y)) & \to & q_f(y) \qquad \} \end{array}$$

Donnons deux exemples de réduction :

$$f(g(a), g(a)) \to_\Delta^* f(g(q_a(\bot)), g(q_a(\bot))) \to_\Delta^* f(q_g(S(\bot)), q_g(S(\bot)))$$

Contrairement à l'exemple 15, $f(g(a), g(a))$ n'est pas accepté par $\mathcal{A}_m$. En effet, du fait des contraintes concernant la mémoire imposées par les transitions, il n'est pas possible de réécrire $f(q_g(S(\bot)), q_g(S(\bot)))$ en $q_f(t)$ quelle que soit la mémoire t. En revanche la réduction suivante est possible :

$$f(g(a), a) \to_\Delta^* f(g(q_a(\bot)), q_a(\bot)) \to_\Delta^* f(q_g(S(\bot)), q_a(\bot)) \to_\Delta q_f(\bot)$$

$f(g(a), a)$ est donc accepté par $\mathcal{A}_m$.

6.1.3 Conditions de visibilité

Le formalisme présenté ci-dessus est trop expressif pour obtenir la décidabilité de l'appartenance d'un terme au langage d'un automate ou la décidabilité des opérations booléennes entre langages. On ajoute des restrictions sur l'ensemble des transitions de ces systèmes. Ces contraintes inspirées de [AM04, JLT99] sont donc appelées contraintes de visibilité et permettent d'établir la décidabilité des problèmes évoqués ci-dessus.

Intuitivement la condition de visibilité impose que l'on puisse déterminer le type d'opération qui est opéré sur la mémoire uniquement à partir du symbole de fonction qui est lu. Par exemple on doit pouvoir déterminer si l'on a ajouté ou retiré quelque chose de la mémoire selon le symbole lu, mais pas nécessairement savoir ce que l'on a ajouté ou retiré.

Pour les prochaines définitions nous devrons considérer une partition de la signature $\mathcal{F}$.

$$\begin{aligned}\mathcal{F} \;=\; & \mathcal{F}_{\text{PUSH}} \uplus \mathcal{F}_{\text{POP}_{11}} \uplus \mathcal{F}_{\text{POP}_{12}} \uplus \mathcal{F}_{\text{POP}_{21}} \\ & \uplus \mathcal{F}_{\text{POP}_{22}} \uplus \mathcal{F}_{\text{INT}_0} \uplus \mathcal{F}_{\text{INT}_1} \uplus \mathcal{F}_{\text{INT}_2}\end{aligned}$$

De plus les symboles de fonction devront être d'arité 0 ou 2. Ces restrictions semblent avoir uniquement pour but de simplifier les preuves de décidabilité. Nous plongerons néanmoins notre signature dans une signature satisfaisant ces restrictions afin que notre résultat repose sur les preuves de décidabilité telles qu'elles sont faites. On supposera aussi que $\mathcal{G}$ contient le symbole de constante $\perp$.

Définition 8 (VTAM) *Un* automate d'arbres ascendant avec mémoire et visibilité *sur une signature finie $\mathcal{F}$ est un tuple $(\mathcal{G}, Q, Q_f, \Delta_{VTAM})$ où $\mathcal{G}$,Q,Q_f sont définis comme dans la définition 7 et Δ_{VTAM} est un ensemble de règles de réécriture de l'une des formes suivantes :*

PUSH	a	$\rightarrow$	$q(c)$	$a \in \mathcal{F}_{\text{PUSH}}$
PUSH	$f(q_1(y_1), q_2(y_2))$	$\rightarrow$	$q(h(y_1, y_2))$	$f \in \mathcal{F}_{\text{PUSH}}$
POP_{1i}	$f(q_1(h(y_{11}, y_{12}), q_2(y_2)))$	$\rightarrow$	$q(y_{1i})$	$f \in \mathcal{F}_{\text{POP}_{1i}}, 1 \leq i \leq 2$
POP_{1i}	$f(q_1(\perp), q_2(y_2))$	$\rightarrow$	$q(\perp)$	$f \in \mathcal{F}_{\text{POP}_{1i}}, 1 \leq i \leq 2$
POP_{2i}	$f(q_1(y_1), q_2(h(y_{21}, y_{22})))$	$\rightarrow$	$q(y_{2i})$	$f \in \mathcal{F}_{\text{POP}_{2i}}, 1 \leq i \leq 2$
POP_{2i}	$f(q_1(y_1), q_2(\perp))$	$\rightarrow$	$q(\perp)$	$f \in \mathcal{F}_{\text{POP}_{2i}}, 1 \leq i \leq 2$
INT_0	a	$\rightarrow$	$q(\perp)$	$a \in \mathcal{F}_{\text{INT}_0}$
INT_i	$f(q_1(y_1), q_2(y_2))$	$\rightarrow$	$q(y_i)$	$f \in \mathcal{F}_{\text{INT}_i}, 1 \leq i \leq 2$

où $q_1, q_2, q \in Q$, y_1, y_2 sont des variables distinctes de $\mathcal{X}$, $c, h \in \mathcal{G}$.

Exemple 17 Remarquons tout d'abord que l'automate $\mathcal{A}_m$ décrit dans l'exemple 16 n'est pas un VTAM pour plusieurs raisons. D'une part, la signature de $\mathcal{A}_m$ comporte des symboles, comme les symboles unaires, qui ne sont pas pris en compte dans la définition des VTAM. D'autre part, le symbole f se trouverait à la fois dans $\mathcal{F}_{\text{PUSH}}$ à cause de la transition $f(q_g(x), q_g(\perp)) \rightarrow q_f(S(\perp))$ et dans $\mathcal{F}_{\text{INT}_2}$ à cause de la transition $f(q_g(x), q_a(y)) \rightarrow q_f(y)$.

Définissons le VTAM $\mathcal{A}_v = (\mathcal{G}, Q, Q_f, \Delta)$ sur la signature $\mathcal{F} = \{f/2, g/2, a/0\}$ comme suit :

- $\mathcal{G} = \{S/2, \perp/0\}$
- $Q = \{q_a, q_g, q_f, q_{acc}\}$
- $Q_f = \{q_{acc}\}$

– Δ est l'ensemble de transitions suivant :

$$\{ \quad a \to q_a(\bot), \qquad g(q_a(x), q_a(y)) \to q_g(S(x,y)),$$
$$g(q_g(x), q_a(y)) \to q_g(S(x,y)), \qquad f(q_g(S(x,y)), q_a(z)) \to q_f(x),$$
$$f(q_f(S(x,y)), q_a(z)) \to q_f(x), \qquad f(q_f(\bot), q_a(\bot)) \to q_{acc}(\bot) \quad \}$$

Remarquons que a et g sont dans $\mathcal{F}_{\text{PUSH}}$ et f est dans $\mathcal{F}_{\text{POP}_{11}}$.

Considérons le terme $f(f(g(a,a),a),a)$ qui est accepté par $\mathcal{A}_v$ par la séquence de réécriture suivante :

$$\begin{aligned} & f(f(g(a,a),a),a) \\ \to_\Delta^* \; & f(f(g(q_a(\bot),q_a(\bot)),q_a(\bot)),q_a(\bot)) \\ \to_\Delta \; & f(f(q_g(S(\bot,\bot)),q_a(\bot)),q_a(\bot)) \\ \to_\Delta \; & f(q_f(\bot),q_a(\bot)) \\ \to_\Delta \; & q_{acc}(\bot) \end{aligned}$$

On peut en revanche remarquer que $f(f(g(g(a,a),a),a),a)$ n'est pas accepté par $\mathcal{A}_v$.

6.1.4 Contraintes structurelles

De manière à accroître le pouvoir d'expression de ces VTAM, c'est à dire définir des langages de manière plus fine, on peut proposer une version de ces automates où l'application des transitions de type INT peut être conditionnée par un test des mémoire des états sur lesquels on l'applique.

De manière à jouir de la décidabilité des opérations booléennes et du problème du vide de ces VTAM on considère des contraintes structurelles définies comme suit. L'utilisation de contraintes syntaxiques mène à l'indécidabilité de certains calculs d'après le corollaire 3.11 de [CJP08].

Deux termes t_1 et t_2 sont structurellement équivalents, noté $t_1 \equiv t_2$, s'ils sont égaux lorsque l'on identifie tous les symboles de même arité. Formellement $\equiv$ est la plus petite relation d'équivalence sur les termes clos satisfaisant

– $a \equiv b$ pour tout a, b d'arité 0,

– $f(s_1,s_2) \equiv g(t_1,t_2)$ si $s_1 \equiv t_1$ et $s_2 \equiv t_2$, pour tout f et g d'arité 2.

Afin de définir les VTAM avec contraintes structurelles, on considère une signature partitionnée comme suit :

$$\begin{aligned} \mathcal{F} \;=\; & \mathcal{F}_{\text{PUSH}} \uplus \mathcal{F}_{\text{POP}_{11}} \uplus \mathcal{F}_{\text{POP}_{12}} \uplus \mathcal{F}_{\text{POP}_{21}} \uplus \mathcal{F}_{\text{POP}_{22}} \\ & \uplus \mathcal{F}_{\text{INT}_0} \uplus \mathcal{F}_{\text{INT}_1} \uplus \mathcal{F}_{\text{INT}_2} \uplus \mathcal{F}^{\equiv}_{\text{INT}_1} \uplus \mathcal{F}^{\equiv}_{\text{INT}_2} \end{aligned}$$

Définition 9 (VTAM avec contraintes structurelles [CJP08]) *Un* VTAM avec contraintes structurelles *(VTAM$^{\equiv}_{\not\equiv}$) sur une signature d'entrée finie $\mathcal{F}$ est un tuple $(\mathcal{G}, \equiv, Q, Q_f, \Delta)$ où $\mathcal{G}$,Q,Q_f sont définis de manière identique à la définition 8, $\equiv$ est la relation sur $\mathcal{T}(\mathcal{G})$ définie ci-dessus et Δ est l'ensemble des règles de réécriture $\Delta_{VTAM} \cup \Delta_C$ où Δ_{VTAM} est défini de manière identique à la définition 8 et Δ_C est l'une des règles suivantes :*

$$\text{INT}_i^{\equiv} \qquad f(q_1(y_1), q_2(y_2)) \overset{y_1 \equiv y_2}{\to} q(y_i) \qquad f \in \mathcal{F}_{\text{INT}_i^{\equiv}}, 1 \le i \le 2$$
$$\text{INT}_i^{\not\equiv} \qquad f(q_1(y_1), q_2(y_2)) \overset{y_1 \not\equiv y_2}{\to} q(y_i) \qquad f \in \mathcal{F}_{\text{INT}_i^{\equiv}}, 1 \le i \le 2$$

où $q_1, q_2, q \in Q$, y_1, y_2 sont des variables distinctes de $\mathcal{X}$, $c, h \in \mathcal{G}$.

Un VTAM$^{\equiv}_{\not\equiv}$ peut appliquer une transition de type $\text{INT}_i^{\equiv}$ (resp. $\text{INT}_i^{\not\equiv}$) à un terme $f(q_1(m_1), q_2(m_2))$ seulement quand $m_1 \equiv m_2$ (resp. $m_1 \not\equiv m_2$). Un terme t est accepté par un VTAM$^{\equiv}_{\not\equiv}$ $\mathcal{A}$ dans un état $q \in Q$ et avec une mémoire $m \in T(\mathcal{G})$ ssi $t \to^* q(m)$.

Exemple 18 Définissons le VTAM$^{\equiv}_{\not\equiv}$ $\mathcal{A}_{vc}$. Soit $\mathcal{F}$ la signature non typée $\{f/2, g/2, a/0\}$ et $\mathcal{A}_v = (\mathcal{G}, Q, Q_f, \Delta)$ tel que
- $\mathcal{G} = \{S/2, \bot/0\}$
- $Q = \{q_a, q_g, q_f\}$
- $Q_f = \{q_f\}$
- Δ est l'ensemble de transitions suivant :

$$\{ \quad a \to q_a(\bot), \qquad g(q_a(x), q_a(y)) \to q_g(S(x,y)), \qquad g(q_g(x), q_a(y)) \to q_g(S(x,y)), \qquad f(q_g(x), q_g(y)) \overset{x\equiv y}{\to} q_f(x) \quad \}$$

Considérons le terme $f(g(a,a), g(a,a))$ qui est accepté par $\mathcal{A}_v$ par la séquence de réécriture suivante :

$$\begin{aligned} & f(g(a,a), g(a,a)) \\ \to_\Delta^* \; & f(g(q_a(\bot), q_a(\bot)), g(q_a(\bot), q_a(\bot))) \\ \to_\Delta \; & f(q_g(S(\bot,\bot)), q_g(S(\bot,\bot))) \\ \to_\Delta \; & q_f(S(\bot,\bot)) \end{aligned}$$

On peut en revanche remarquer que ni $f(f(g(g(a,a),a),a),a)$ ni $f(f(f(g(a,a),a),a),a)$ ne sont acceptés par $\mathcal{A}_v$.

Théorème 2 ([CJP08]) *La classe des langages reconnaissables par VTAM$^{\equiv}_{\not\equiv}$ est close sous les opérations booléennes et le problème du vide d'un VTAM$^{\equiv}_{\not\equiv}$ est décidable.*

Remarquons que la clôture sous les opérations de $\cup$, $\cap$ suppose la même partition de la signature d'entrée $\mathcal{F}$ en $\mathcal{F}_{\text{PUSH}}, \mathcal{F}_{\text{POP}_{11}}$ etc...

6.2 Encodage de signatures infinies

La signature $\mathcal{F}$ utilisée dans la spécification d'un protocole peut être infinie. Ceci est dû en particulier à la présence de constantes qui sont indicées par le nombre de participants à une session. Pour être capable de définir un VTAM$^{\equiv}_{\not\equiv}$ qui reconnaît E et K on doit trouver la signature finie $\mathcal{F}'$ qui contient seulement des constantes et des symboles binaires ainsi qu'une fonction $\rho : T(\mathcal{F}) \to T(\mathcal{F}')$ appropriée. Nous étendrons la fonction ρ de manière naturelle aux ensembles de termes. Nous montrerons alors que $\rho(DY(E)) \cap \rho(K) = \emptyset$. Remarquons que cela implique $DY(E) \cap K = \emptyset$ indépendamment du choix de ρ, bien qu'en pratique ρ sera un homomorphisme injectif. Si ρ est injective alors le fait que $DY(E)$ et K sont disjoints est équivalent au fait que $\rho(DY(E))$ et $\rho(K)$ le soient.

Ceci correspond à l'étape suivante présentée sous-section 4.5.2 :

$$\begin{array}{c} DY(App(E)) \cap App(K) \overset{?}{=} \emptyset \\ \Downarrow \\ DY(\rho(App(E))) \cap \rho(App(K)) \overset{?}{=} \emptyset \end{array}$$

Dans la suite de réductions et d'abstraction présentée section 4.5.2 l'étape précédant l'étape ci-dessus, permettant de passer du problème $DY(E) \cap K \stackrel{?}{=} \emptyset$ au problème $DY(App(E)) \cap App(K) \stackrel{?}{=} \emptyset$ sera présentée section 6.5.

Nous allons nous concentrer sur l'exemple GDH-2 (présenté à la sous-section 3.2.1) pour des raisons de simplicité.

Exemple 19 (suite de l'exemple 6) La signature décrivant ce protocole contient des constantes N_i^j représentant le nonce du participant i à la session j (où $i \leq j$). Remarquons que nous pourrions aussi considérer des constantes K_i^j pour les clés symétriques entre les participants i et j (où $i \leq j$).

Nous choisissons une signature finie $\mathcal{F}'$ consistant en un ensemble de constantes $\mathcal{F}'_0 = \{0, g\}$, et l'ensemble de symboles de fonctions binaires

$$\mathcal{F}'_2 = \{\mathsf{pair}, \mathsf{enc}, \mathsf{exp}, \mathsf{mult}, \mathsf{xor}, t, H, N, K, s, s'\}$$

La fonction $\rho\colon T(\mathcal{F}) \to T(\mathcal{F}')$ pour l'exemple 6 est définie comme suit en utilisant une fonction auxiliaire $\rho'\colon \{(i,j) \mid i \leq j\} \to T(\mathcal{F}')$:

$$\begin{array}{rclrcl}
g & \to & g & \mathsf{pair}(u,v) & \to & \mathsf{pair}(\rho(u),\rho(v)) \\
0 & \to & 0 & \mathsf{enc}(u,v) & \to & \mathsf{enc}(\rho(u),\rho(v)) \\
K_i^j & \to & K(0,\rho'(i,j)) & \mathsf{exp}(u,v) & \to & \mathsf{exp}(\rho(u),\rho(v)) \\
N_i^j & \to & N(0,\rho'(i,j)) & \mathsf{mult}(u,v) & \to & \mathsf{mult}(\rho(u),t(0,\rho(v))) \\
H(u) & \to & H(0,\rho(u)) & \mathsf{xor}(u,v) & \to & \mathsf{xor}(\rho(u),\rho(v))
\end{array}$$

tels que

$$\begin{array}{rcll}
\rho'(i,j) & = & s'(0,\rho'(i-1,j-1)) & \text{if } i > 0 \\
\rho'(0,j) & = & s(0,\rho'(0,j-1)) & \text{if } j > 0 \\
\rho'(0,0) & = & 0 &
\end{array}$$

Par exemple $\rho'(1,3) = s'(0,s(0,s(0,0)))$. Cet encodage des paires ainsi que l'encodage du symbole mult et en particulier la présence du symbole t ont été choisis pour faciliter la construction de l'automate dans la sous-section 6.5.2. Nous revenons plus précisément sur l'explication de ce symbole dans la section 6.5.2.

Enfin nous devons adapter le système de déduction DY à la traduction de la signature, menant à la modification du système de déduction DY' tel que $\rho(DY(S)) = DY'(\rho(S))$ pour n'importe quel $S \subseteq T(\mathcal{F})$.

Exemple 20 (suite de l'exemple 19) Dans l'exemple 19, nous avons juste à adapter la règle *hash* et à la remplacer par la variante suivante :

$$\frac{S \vdash t}{S \vdash H(0,t)}\, hash'$$

Le lemme suivant établit que DY' a bien la propriété attendue, à savoir que $\rho(DY(S)) = DY'(\rho(S))$ pour n'importe quel $S \subseteq T(\mathcal{F})$.

Lemme 13 *$\rho(DY(S)) = DY'(\rho(S))$ pour ρ définie comme dans l'exemple 19.*

Démonstration. Prouvons d'abord que $\rho(DY(S)) \subseteq DY'(\rho(S))$. Soit t' un terme de $\rho(DY(S))$. Par définition il existe $t \in DY(S)$ tel que $t' = \rho(t)$. D'après la définition 2, $S \vdash_{DY} t$. Nous affirmons que si $S \vdash_{DY} t$, alors $\rho(S) \vdash_{DY'} \rho(t)$, donc $\rho(S) \vdash_{DY'} t'$, et enfin $t' \in DY'(\rho(S))$.

Montrons que s'il existe une déduction π de $S \vdash_{DY} t$, alors il existe aussi une déduction π' de $\rho(S) \vdash_{DY'} \rho(t)$. Par induction sur $|\pi|$.

Cas de base. $|\pi| = 0$. Alors la déduction consiste simplement en une application de *axiom*, et donc $t \in S$ et $\rho(t) \in \rho(S)$ et on a directement $\rho(S) \vdash_{DY'} \rho(t)$.

Cas inductif. $|\pi| = n$. On considère les différents cas pour la dernière règle r.

- $r = pair$. On doit donc avoir $t = \mathsf{pair}(u, v)$ pour certains termes u et v. Comme π est une déduction de $S \vdash_{DY} \mathsf{pair}(u, v)$, il existe une déduction π_1 de $S \vdash_{DY} u$ et une déduction π_2 de $S \vdash_{DY} v$ telles que $|\pi_1|, |\pi_2| \leq n$. Par induction il existe une déduction π_1' de $\rho(S) \vdash_{DY'} \rho(u)$ et une déduction π_2' de $\rho(S) \vdash_{DY'} \rho(v)$. Par une application de la règle *pair* on obtient une déduction π' de $\rho(S) \vdash_{DY'} \mathsf{pair}(\rho(u), \rho(v))$. Par définition, $\rho(\mathsf{pair}(u, v)) = \mathsf{pair}(\rho(u), \rho(v))$, et donc $\rho(S) \vdash_{DY'} \rho(\mathsf{pair}(u, v))$.
- $r = proj_1$, $r = proj_2$, $r = dec$, $r = enc$. Ces cas sont similaires à $r = pair$.
- $r = hash$. On doit donc avoir $t = H(u)$ pour un certain u. Comme π est une déduction de $S \vdash_{DY} H(u)$, il existe une déduction π^- de $S \vdash_{DY} u$ telle que $|\pi^-| \leq n$. Par induction il existe $\pi^{-'}$ a déduction de $\rho(S) \vdash_{DY'} \rho(u)$. Par une application de la règle $hash'$ on obtient une déduction π' de $\rho(S) \vdash_{DY'} H(0, \rho(u))$. Par définition, $\rho(H(u)) = H(0, \rho(u))$, et donc $\rho(S) \vdash_{DY'} \rho(H(u))$.

Prouvons que $DY'(\rho(S)) \subseteq \rho(DY(S))$. Soit t' un terme dans $DY'(\rho(S))$. Par définition, il existe une déduction π' de $\rho(S) \vdash_{DY'} t'$. Il nous faut montrer que si $\rho(S) \vdash_{DY'} t'$ alors il existe un terme $t \in DY(S)$ tel que $t' = \rho(t)$. Donc $t' \in \rho(DY(S))$.

Montrons que s'il existe une déduction π' de $\rho(S) \vdash_{DY'} t'$ alors il existe un terme $t \in DY(S)$ tel que $t' = \rho(t)$. Par induction sur la taille de π'. Soit r la dernière règle de π'.

Cas de base. $|\pi'| = 0$. Dans ce cas $r = axiom$ et on a $t' \in \rho(S)$. Il existe donc un terme $t \in S$ tel que $t' = \rho(t)$ et de manière triviale $t \in DY(S)$.

Cas inductif. $|\pi'| = n$. On considère les différentes possibilités pour la dernière règle r.

- $r = pair$. On doit donc avoir $t' = \mathsf{pair}(u', v')$ pour certains u' et v'. Comme π' est une déduction de $\rho(S) \vdash_{DY'} \mathsf{pair}(u', v')$, il existe deux déductions π_1' de $\rho(S) \vdash_{DY'} u'$ et π_2' de $\rho(S) \vdash_{DY'} v'$ telles que $|\pi_1'|, |\pi_2'| \leq n$. Par hypothèse d'induction il existe $u, v \in DY(S)$ tel que $u' = \rho(u)$ et $v' = \rho(v)$. En une application de *pair* on a une déduction π de $S \vdash_{DY} \mathsf{pair}(u, v)$, donc $\mathsf{pair}(u, v) \in DY(S)$. Or par définition, $\rho(\mathsf{pair}(u, v)) = \mathsf{pair}(\rho(u), \rho(v))$ et donc $\rho(\mathsf{pair}(u, v)) = \mathsf{pair}(u', v')$.
- Tous les autres cas sont similaires à $r = pair$.

□

Le choix du système DY ou DY' que nous utilisons dépendant de la signature $\mathcal{F}$ ou $\mathcal{F}'$, il sera souvent clair selon le contexte. Nous écrirons donc DY au lieu de DY' lorsque cela ne suscite pas d'ambiguité.

6.3 Représentation de l'associativité et de la commutativité de xor et mult

Comme pour les automates d'arbres classiques, les langages reconnus par VTAM$^{\equiv}_{\neq}$ ne sont en général pas clos sous associativité et commutativité. Pour régler cette difficulté, on définit une fonction témoin W sur $T(\mathcal{F}')$. Cette représentation des classes d'équivalence d'un terme t par rapport à AC par un terme témoin obtenu par la fonction W correspond à l'étape suivante présentée dans la sous-section 4.5.2 :

$$DY(\rho(App(E))) \cap \rho(App(K) \stackrel{?}{=} \emptyset$$
$$\Downarrow$$
$$DY(W(\rho(App(E))) \cap W(\rho(App(K))) \stackrel{?}{=} \emptyset$$

Cette fonction W, définie sur $T(\mathcal{F}')$, associe à chaque terme t l'élément minimal de la classe d'équivalence $[t]_{AC}$ par rapport à l'ordre lexicographique sur les chemins (lexicographic path ordering) [Der87] que nous noterons lpo, défini en considérant la précédence $<_{\mathcal{F}'}$ sur $\mathcal{F}'$:

$$0 <_{\mathcal{F}'} g <_{\mathcal{F}'} s <_{\mathcal{F}'} s' <_{\mathcal{F}'} N <_{\mathcal{F}'} K <_{\mathcal{F}'} K^+ <_{\mathcal{F}'}$$
$$K^- <_{\mathcal{F}'} H <_{\mathcal{F}'} t <_{\mathcal{F}'} \mathsf{xor} <_{\mathcal{F}'} \mathsf{mult} <_{\mathcal{F}'} \mathsf{exp} <_{\mathcal{F}'} \mathsf{enc} <_{\mathcal{F}'} \mathsf{pair}$$

Rappelons la définition de lpo en considérant la précédence $<_{\mathcal{F}'}$:

Définition 10 (lpo) *Soient u et v deux termes de $T(\mathcal{F}')$:*
$u <_{lpo} v$ *si* $v = f(v_1, \ldots, v_n)$ *et soit*
– $u = f(u_1, \ldots, u_n)$ *et* $\exists i$

$$\forall j <_{\mathbb{N}} i \; u_j = v_j \qquad u_i <_{lpo} v_i \qquad \forall j >_{\mathbb{N}} i \; u_j < v$$

– $u = g(u_1, \ldots, u_m)$ *et* $g <_{\mathcal{F}'} f$ *et* $\forall j \; u_j <_{lpo} v$
– $\exists i \; u <_{lpo} v_i$ *ou* $u = v_i$

Remarque 21. $\leq_{lpo}$ est une congruence et $<_{lpo}$ un ordre total et bien fondé.

Remarque 22. On vérifie facilement que $\rho(N_i^j) <_{lpo} \rho(N_{i'}^{j'})$ si et seulement si soit $i <_{\mathbb{N}} i'$ soit $i = i'$ et $j <_{\mathbb{N}} j'$.

Nous pouvons désormais aisément définir la fonction témoin W comme suit.

Définition 11 *La fonction $W : T(\mathcal{F}') \mapsto T(\mathcal{F}')$ assigne à tout $t' \in T(\mathcal{F}')$ tel que $t' = \rho(t)$ l'élément minimal de $\rho([t]_{AC})$.*

Cette fonction s'étend de manière naturelle aux ensembles de termes. Maintenant, le fait que les ensembles de termes S_1 et S_2 qui sont clos sous AC sont disjoints est équivalent au fait que $W(S_1)$ et $W(S_2)$ sont disjoints.

Théorème 3 *Si S est clos sous AC alors $W(DY'(S)) = DY'(W(S))$.*

Afin de prouver ce théorème nous avons d'abord besoin de prouver les deux lemmes suivants.

Lemme 14 *Soient S_1 et S_2 des ensembles de termes clos sous associativité et commutativité de* xor *et* mult*. Pour tout t, $t \in S_1 \cap S_2$ si et seulement si $W(t) \in W(S_1) \cap W(S_2)$.*

Démonstration. Montrons que si $t \in S_1 \cap S_2$ alors $W(t) \in W(S_1) \cap W(S_2)$. Soit t un terme de $S_1 \cap S_2$, alors $t \in S_1$ et $t \in S_2$. Par l'extension aux ensembles de termes de la définition 11, $W(t) \in W(S_1)$ et $W(t) \in W(S_2)$, et donc $W(t) \in W(S_1) \cap W(S_2)$.

Montrons que si $W(t) \in W(S_1) \cap W(S_2)$ alors $t \in S_1 \cap S_2$. Si $W(t) \in W(S_1) \cap W(S_2)$, alors $W(t) \in W(S_1)$, c'est-à-dire qu'il y a un $t_1 \in S_1$ avec $W(t) = W(t_1)$. Par définition de W cela implique que $t \equiv_{AC} t_1$, et donc que $t \in S_1$ puisque S_1 est clos sous AC. L'argument pour $t \in S_2$ est analogue. □

Lemme 15 *Pour tout symbole $f \in \{\mathsf{pair}, \mathsf{enc}, H\}$ et pour tous termes u et v, on a $W(f(u,v)) = f(W(u), W(v))$.*

Démonstration. D'abord, comme f n'est ni xor ni mult, $W(f(u,v)) = f(u',v')$ pour certains u' et v', et $u \equiv_{AC} u'$ et $v \equiv_{AC} v'$. Par minimalité de $W(f(u,v))$ dans la classe d'équivalence de $f(u,v)$ et par la remarque 21 ($\leq_{lpo}$ est une congruence), u' est minimal dans la classe d'équivalence de u et v' est minimal dans la classe d'équivalence de v. Il s'en suit $u' = W(u)$ and $v' = W(v)$. □

Démonstration (du théorème 3). Nous montrons successivement que $W(DY'(S)) \subseteq DY'(W(S))$ et que $DY'(W(S)) \subseteq W(DY'(S))$.

$W(DY'(S)) \subseteq DY'(W(S))$. Soit $t' \in W(DY'(S))$. Il existe $t \in DY'(S)$ tel que $t' = W(t)$. On affirme que si $S \vdash_{DY'} t$, alors $W(S) \vdash_{DY'} W(t)$. Ceci implique que $t' \in DY'(W(S))$.

Montrons par induction sur $|\pi|$ que si $S \vdash_{DY'} t$, alors il existe une déduction π' de $W(S) \vdash_{DY'} W(t)$.

Cas de base : $|\pi| = 0$. Alors la déduction π consiste uniquement en une application de la règle *axiom*. Donc $t \in S$, et par conséquent $W(t) \in W(S)$ et $W(S) \vdash_{DY'} W(t)$.

Cas inductif : $|\pi| = n$. On fait la démonstration dans le cas où la dernière règle r de π est *pair* car les autres cas sont analogues. Lorsque $r = pair$, $t = \mathsf{pair}(u,v)$ pour certains u et v. Comme π est une déduction de $S \vdash_{DY'} \mathsf{pair}(u,v)$, il y a des déductions π_1 de $S \vdash_{DY'} u$ et π_2 de $S \vdash_{DY'} v$ telles que $|\pi_1|, |\pi_2| \leq n$. Par hypothèse d'induction il existe une déduction π_1' de $W(S) \vdash_{DY'} W(u)$ et une déduction π_2' de $W(S) \vdash_{DY'} W(v)$. Par une application de la règle *pair* on obtient une déduction π' de $W(S) \vdash_{DY'} \mathsf{pair}(W(u), W(v))$. Par le lemme 15 $W(\mathsf{pair}(u,v)) = \mathsf{pair}(W(u), W(v))$, et donc $W(S) \vdash_{DY'} W(\mathsf{pair}(u,v))$.

$DY'(W(S)) \subseteq W(DY'(S))$. Soit $t' \in DY'(W(S))$. On affirme que si $W(S) \vdash_{DY'} t'$ alors il existe un terme t tel que $t' = W(t)$ et une déduction π de $S \vdash_{DY'} t$. Donc $t \in DY'(S)$ et $t' \in W(DY'(S))$. Montrons par induction sur $|\pi'|$, que s'il existe une déduction π' de $W(S) \vdash_{DY'} t'$ alors il existe un terme t tel que $t' = W(t)$ et une déduction π de $S \vdash_{DY'} t$.

Cas de base. $|\pi'| = 0$. Alors la déduction π' consiste simplement en une application de la règle *axiom* et $t' \in W(S)$, c'est à dire $t' = W(t)$ pour un certain $t \in S$. Donc $S \vdash_{DY'} t$.

Cas inductif. $|\pi'| = n$. On fait la démonstration dans le cas où la dernière règle r de π' est *pair*. Les autres cas sont analogues. Lorsque $r = pair\ t' = \mathsf{pair}(u', v')$ pour certains u' et v'. Comme π' est une déduction de $W(S) \vdash_{DY'} \mathsf{pair}(u', v')$, il existe deux déductions π'_1 de $W(S) \vdash_{DY'} u'$ et π'_2 de $W(S) \vdash_{DY'} v'$ telles que $|\pi'_1|, |\pi'_2| \leq n$. Par hypothèse d'induction il existe un terme u tel que $u' = W(u)$ et une déduction π_1 de $S \vdash_{DY'} u$, et un terme v tel que $v' = W(v)$ et une déduction π_2 de $S \vdash_{DY'} v$. En appliquant la règle *pair* nous obtenons une déduction π de $S \vdash_{DY'} \mathsf{pair}(u, v)$. D'après le lemme 15 $W(\mathsf{pair}(u, v)) = \mathsf{pair}(W(u), W(v))$, donc $W(\mathsf{pair}(u, v)) = \mathsf{pair}(u', v')$, et $S \vdash_{DY'} \mathsf{pair}(u, v)$. □

6.4 Clôture sous DY et compatibilité avec la clôture sous AC

Nous montrons désormais comment il est possible de représenter l'intrus *DY* grâce à un automate d'arbre. Cela correspond à l'étape suivante présentée dans la sous-section 4.5.2.

$$DY(\mathcal{A}_{W(\rho(App(E)))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset$$
$$\Downarrow$$
$$\mathcal{A}_{DY(W(\rho(App(E))))} \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset$$

Théorème 4 *Pour tout VTAM$^{\equiv}_{\not\equiv}$ $\mathcal{A}$, tel que $\mathsf{pair}, \mathsf{enc} \notin \{\mathcal{F}'_{\mathrm{INT}^{\equiv}_1} \cup \mathcal{F}'_{\mathrm{INT}^{\equiv}_2}\}$ et le seul symbole de constante $\mathcal{G}$ est $\perp$, il existe un VTAM$^{\equiv}_{\not\equiv}$ $\mathcal{A}_{DY}$ tel que $L(\mathcal{A}_{DY}) = DY'(L(\mathcal{A}))$.*

La preuve repose sur une technique classique de complétion d'automates (voir [Gou00]). Dans notre cas, il faut porter une attention particulière aux contraintes sur la mémoire. Nous montrons ainsi comment nous pouvons étendre la méthode de complétion de [Gou00] des automates d'arbres classiques sous le système de DY aux VTAM$^{\equiv}_{\not\equiv}$.

À chaque partition possible p de $\mathcal{F}'$ mise à part la partition où $\mathsf{pair}, \mathsf{enc} \in \{\mathcal{F}'_{\mathrm{INT}^{\equiv}_1} \cup \mathcal{F}'_{\mathrm{INT}^{\equiv}_2}\}$, nous associons un système de complétion DY_p. $DY_p(\mathcal{A})$ est le point fixe de l'application des règles de DY_p.

Définition 12 *Soit p la partition de $\mathcal{F}'$ et $\mathcal{A} = (\mathcal{G}, Q, Q_f, \Delta)$ un VTAM$^{\equiv}_{\not\equiv}$ sur p. Le système de complétion DY_p est l'ensemble de règles*

$$\{Proj_1, Proj_2, Dec, Pair, Enc, Hash\}$$

défini comme suit après avoir ajouté quatre nouveaux états à Q_f : q_{pair}, q_{enc}, q_0 et q_H.

- *$Proj_1$.*
  - *Si $\mathsf{pair} \in \mathcal{F}'_{\mathrm{PUSH}}$, pour tout état $q \in Q_f$, si $\mathsf{pair}(q_1(x), q_2(y)) \to q(h(x, y)) \in \Delta$ pour un certain symbole h, on ajoute q_1 à Q_f si $L(\mathcal{A}, q_2) \neq \emptyset$. Cette dernière vérification est possible par le corollaire 3.12 de [CJP08].*
  - *Si $\mathsf{pair} \in \mathcal{F}'_{\mathrm{INT}_1}$ ou $\mathsf{pair} \in \mathcal{F}'_{\mathrm{INT}_2}$. Ces cas sont analogues au précédent.*
  - *Si $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{22}}$, pour tout état $q \in Q_f$, si $\mathsf{pair}(q_1(x), q_2(h(x, y))) \to q(y) \in \Delta$ pour un certain symbole h, soit $\mathcal{A}_h$ reconnaissant l'ensemble des termes sur $T(\mathcal{G})$ ayant h à leur tête, et soit $M(\mathcal{A}_{q_2})$ l'automate d'arbres reconnaissant le langage de mémoire de q_2 (cet automate peut être construit par le lemme 3.7 de [CJP08]) . Si $M(\mathcal{A}, q_2) \cap \mathcal{A}_h \neq \emptyset$ on ajoute q_1 à Q_f.*
    De manière similaire, si $\mathsf{pair}(q_1(x), q_2(\perp)) \to q(\perp) \in \Delta$, on ajoute q_1 à Q_f au cas où $M(\mathcal{A}, q_2) \cap \mathcal{A}_\perp \neq \emptyset$.

  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{11}}$, $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{12}}$ *ou* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{21}}$, *les opérations sont similaires aux cas précédents.*
  - $\mathsf{pair} \notin \mathcal{F}'_{\mathrm{INT}_1^{\equiv}}$ *et* $\mathsf{pair} \notin \mathcal{F}'_{\mathrm{INT}_2^{\equiv}}$.
- $Proj_2$ *est défini de manière similaire à* $Proj_1$. *On ajoute* q_2 *à* Q_f *au lieu de* q_1 *en adaptant les vérifications effectuées avant cet ajout.*
- *Dec.*
  - *Si* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{PUSH}}$, *pour tout état* $q \in Q_f$, *si* $\mathsf{enc}(q_1(x), q_2(y)) \to q(h(x,y)) \in \Delta$ *pour un certain symbole* h, *on ajoute* q_1 *à* Q_f *si* $L(\mathcal{A}, q_2) \cap L(\mathcal{A}) \neq \emptyset$. *Cette dernière vérification est possible par le corollaire 3.12 et le théorème 3.15 de [CJP08].*
  - *Si* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{INT}_1}$ *ou* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{INT}_2}$ *les opérations sont similaires au cas précédent.*
  - *Si* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{22}}$, *pour tout état* $q \in Q_f$, *si* $\mathsf{enc}(q_1(x), q_2(h(x,y))) \to q(y) \in \Delta$ *pour un certain symbole* h, *on construit le VTAM*$^{\equiv}_{\neq}$ $\mathcal{A}_\cap = L(\mathcal{A}, q_2) \cap \mathcal{A}$. *Si* $\mathcal{A}_\cap = \emptyset$ *on n'ajoute pas d'état à* Q_f. *Sinon on vérifie que* $M(A_\cap) \cap \mathcal{A}_h \neq \emptyset$, $\mathcal{A}_h$ *étant défini comme pour* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{22}}$. *Si ce n'est pas vide on ajoute* q_1 *à* Q_f.
    Le cas où $\mathsf{enc}(q_1(x), q_2(\bot)) \to q(\bot) \in \Delta$ *est analogue.*
  - *Si* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{11}}$, $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{12}}$, $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{21}}$ *alors les opérations sont similaires au cas où* $\mathsf{enc} \in \mathcal{F}'_{\mathrm{POP}_{22}}$.
  - $\mathsf{enc} \notin \mathcal{F}'_{\mathrm{INT}_1^{\equiv}}$ *et* $\mathsf{enc} \notin \mathcal{F}'_{\mathrm{INT}_2^{\equiv}}$.
- *Pair.*
  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{PUSH}}$, *pour tout* $q_1, q_2 \in Q_f$, *on ajoute* $\mathsf{pair}(q_1(x), q_2(y)) \to q_{pair}(h(x,y))$ *à* Δ *pour un symbole* $h \in \mathcal{G}$ *quelconque.*
  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{INT}_1}$, *pour tous* $q_1, q_2 \in Q_f$, *on ajoute* $\mathsf{pair}(q_1(x), q_2(y)) \to q_{pair}(x)$ *à* Δ.
  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{INT}_2}$, *pour tous* $q_1, q_2 \in Q_f$, *on ajoute* $\mathsf{pair}(q_1(x), q_2(y)) \to q_{pair}(y)$ *à* Δ.
  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{22}}$, *pour tous* $q_1, q_2 \in Q_f$ *et pour tous symboles* h *dans* $\mathcal{G}$, *on ajoute* $\mathsf{pair}(q_1(x), q_2(h(y,z))) \to q_{pair}(z)$ *à* Δ. *On ajoute aussi* $\mathsf{pair}(q_1(x), q_2(\bot)) \to q_{pair}(\bot)$ *à* Δ.
  - *Si* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{11}}$, $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{12}}$, $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{21}}$, *ces cas sont similaires à* $\mathsf{pair} \in \mathcal{F}'_{\mathrm{POP}_{22}}$.
  - $\mathsf{pair} \notin \mathcal{F}'_{\mathrm{INT}_1^{\equiv}}$ *et* $\mathsf{pair} \notin \mathcal{F}'_{\mathrm{INT}_2^{\equiv}}$.
- *Enc. Ce cas est similaire à Pair, mis à part le fait que l'état que l'on considère dans les transitions n'est pas* q_{pair} *mais* q_{enc}.
- *Hash. Soit* $0 \to q_0(\bot)$ *de nouvelles transitions dans* Δ.
  - $H \in \mathcal{F}'_{\mathrm{PUSH}}$. *Pour tout* $q_1 \in Q_f$, *on ajoute* $H(q_0(x), q_1(y)) \to q_H(h(x,y))$ *à* Δ *pour certains symbole* $h \in \mathcal{G}$.
  - $H \in \mathcal{F}'_{\mathrm{INT}_1}$. *Pour tout* $q_1 \in Q_f$, *on ajoute* $H(q_0(x), q_1(y)) \to q_H(x)$.
  - $H \in \mathcal{F}'_{\mathrm{INT}_2}$. *Pour tout* $q_1 \in Q_f$, *on ajoute* $H(q_0(x), q_1(y)) \to q_H(y)$.
  - $H \in \mathcal{F}'_{\mathrm{POP}_{22}}$. *Pour tout* $q_1 \in Q_f$ *et pour tout symbole* h *dans* $\mathcal{G}$, $H(q_0(x), q_1(h(y,z))) \to q_H(z)$ *à* Δ. *On ajoute aussi* $H(q_0(x), q_1(\bot)) \to q_H(\bot)$ *à* Δ.
  - $H \in \mathcal{F}'_{\mathrm{POP}_{11}}$, $H \in \mathcal{F}'_{\mathrm{POP}_{12}}$, $H \in \mathcal{F}'_{\mathrm{POP}_{21}}$. *Ces cas sont similaires à* $H \in \mathcal{F}'_{\mathrm{POP}_{22}}$.
  - $H \notin \mathcal{F}'_{\mathrm{INT}_1^{\equiv}}$ *et* $H \notin \mathcal{F}'_{\mathrm{INT}_2^{\equiv}}$.

Nous illustrons la définition 12 par le système de complétion obtenu dans le cas où $\mathsf{pair}, \mathsf{enc}, H \in \mathcal{F}_{\mathrm{PUSH}}$. La clôture de $\mathcal{A}$ étend $\mathcal{A}$ par de nouveaux états finaux q_{pair}, q_{enc}, et q_H. On ajoute aussi de nouvelles transitions et transformons certains états en états finaux :

$$\frac{q_1, q_2 \in Q_f}{\mathsf{pair}(q_1(x), q_2(y)) \to q_{pair}(h(x,y))}\ Pair$$

$$\frac{\mathsf{pair}(q_1(x), q_2(y)) \to q(h(x,y)) \qquad q \in Q_f \qquad L(\mathcal{A}, q_{3-i}) \neq \emptyset}{q_i \in Q_f}\ Proj_i,\ 1 \leq i \leq 2$$

$$\frac{q_1, q_2 \in Q_f}{\mathsf{enc}(q_1(x), q_2(y)) \to q_{enc}(h(x,y))}\ Enc$$

$$\frac{\mathsf{enc}(q_1(x), q_2(y)) \to q(h(x,y)) \qquad q \in Q_f \qquad L(\mathcal{A}, q_2) \cap L(\mathcal{A}) \neq \emptyset}{q_1 \in Q_f}\ Dec$$

$$\frac{q_1 \in Q_f}{H(q_0(x), q_1(y)) \to q_H(h(x,y))}\ Hash$$

6.5 Représentation d'une sur-approximation de GDH-2

Nous proposons ici une approximation des messages échangés et des messages supposés secrets obtenus lors d'un nombre infini de sessions d'un protocole (une session pour chaque nombre de participants). Cela correspond à l'étape suivante présentée sous-section 4.5.2 :

$$DY(E) \cap K \stackrel{?}{=} \emptyset$$
$$\Downarrow$$
$$DY(App(E)) \cap App(K) \stackrel{?}{=} \emptyset$$

6.5.1 Définition des sur-approximations

On propose ici une sur-approximation de l'ensemble des clés calculées durant un nombre infini de sessions d'un protocole (une session pour chaque nombre de participants).

$$K = \{g^{N_{j_1}^{j_1} \cdot N_{(j_1-1)}^{j_2} \cdots N_1^{j_{j_1}}}\}$$

ainsi qu'une sur-approximation de l'ensemble des messages émis obtenue comme $E_1 \cup E_2$

$$E_1 = \{g^{N_{i_1}^{j_1} \cdot \ldots \cdot N_{i_k}^{j_k}} | k < i_1\} \qquad E_2 = \{g^{N_{i_1}^{j_1} \cdot N_{(i_1-1)}^{j_2} \cdots N_1^{j_{i_1}}} | i_1 < j_1\}$$

Dans la sur-approximation de K se trouvent tous les termes $g^{N_{j_1}^{j_1} \cdot N_{(j_1-1)}^{j_2} \cdots N_1^{j_{j_1}}}$ qui contiennent tous les termes ayant en exposant N_j^j et pour tous $i < j$ il y a exactement un $N_i^{j'}$ pour un j' quelconque supérieur ou égal à i. Dans la sur-approximation de K il y a donc les termes $\{g^{N_j^j \cdot N_{(j-1)}^j \cdots N_1^j}\}$ qui sont les clés appartenant réellement à la spécification de GDH-2, mais en plus il contient des termes du type $g^{N_{j_1}^{j_1} \cdot N_{(j_1-1)}^{j_2} \cdots N_1^{j_{j_1}}}$ où $j_{j_i} \neq j$.

6.5.2 Définition des automates représentant les sur-approximations

Ceci correspond à l'étape suivante présentée dans la sous-section 4.5.2 :

$$DY(W(\rho(App(E)))) \cap W(\rho(App(K))) \stackrel{?}{=} \emptyset$$
$$\Downarrow$$
$$DY(\mathcal{A}_{W(\rho(App(E)))}) \cap \mathcal{A}_{W(\rho(App(K)))} \stackrel{?}{=} \emptyset$$

Automate $\mathcal{A}_K$ détaillé

On détaille la construction d'un automate $\mathcal{A}_K$ tel que $W(\rho(K)) \subseteq L(\mathcal{A}_K) \subseteq \rho(K)$. La preuve de cette assertion est la preuve du lemme 16. On utilise la partition suivante de la signature $\mathcal{F}'$ dans l'automate :

$$\begin{array}{ll} \mathcal{F}_{PUSH} = \{s', \mathsf{exp}, 0, g\} & \mathcal{F}_{POP_{22}} = \{t\} \\ \mathcal{F}_{INT_2^{\equiv}} = \{\mathsf{mult}\} & \mathcal{F}_{INT_2} = \{s, N\} \end{array}$$

Les autres symboles peuvent appartenir à n'importe quelle autre partie de la signature. Nous définissons $\mathcal{G} = \{S, S', h, \bot\}$. L'automate $\mathcal{A}_K$ est défini comme suit (q_{acc} est le seul état final) :

$$0 \to q_t(\bot) \qquad 0 \to q_d(\bot) \qquad g \to q_g(\bot)$$

Les transitions suivantes vérifient le fait que si un terme $t \to^* q_{nent}(m)$ alors t est de la forme $N(0, s'(0, \ldots s'(0,0)\ldots))$ et $m = S'(\bot, \ldots S'(\bot, \bot)\ldots)$ et le nombre de S' dans m est égal au nombre de s' dans t. Alors t représente un nonce de K tel que $i = j$.

$$\begin{array}{rcl} s'(q_t(m), q_t(m')) & \to & q_{s'ent}(S'(m, m')) \\ s'(q_t(m), q_{s'ent}(m')) & \to & q_{s'ent}(S'(m, m')) \\ N(q_t(m), q_{s'ent}(m')) & \to & q_{nent}(m') \end{array}$$

Les transitions suivantes sont similaires mais elles autorisent aussi la présence de S entre les S' et la constante 0. En mémoire, on ne compte que le nombre de S'. On vérifie aussi que les termes menant à $q_{nonly1s'}$ ne comportent qu'au plus une occurrence du symbole s'.

$$\begin{array}{rcl} s(q_t(m), q_d(m')) & \to & q_d(m') \\ s'(q_t(m), q_d(m')) & \to & q_{only1s'}(S'(m, m')) \\ s'(q_t(m), q_{only1s'}(m')) & \to & q_{s'}(S'(m, m')) \\ s'(q_t(m), q_{s'}(m')) & \to & q_{s'}(S'(m, m')) \\ N(q_t(m), q_{s'}(m')) & \to & q_n(m') \\ N(q_t(m), q_{only1s'}(m')) & \to & q_{nonly1s'}(m') \end{array}$$

Les transitions suivantes retirent un symbole S' de la mémoire. L'utilisation de t trouve ici sa justification. En effet, le symbole mult ne peut pas permettre à la fois de tester l'égalité des mémoires représentant indirectement l'indice du participant (l'indice i dans un nonce N_i^j) et de retirer un symbole S' de la mémoire. Puisque le test de la mémoire, conjoint au fait d'en retirer un élément n'est pas possible du fait de la partition de la signature, nous avons introduit le codage ρ utilisant ce symbole t.

$$\begin{array}{rcl} t(q_t(m), q_{nent}(S'(m', m''))) & \to & q_{nt}(m'') \\ t(q_t(m), q_{narg}(S'(m', m''))) & \to & q_{nt}(m'') \end{array}$$

La transition suivante peut être appliquée entre un terme qui représente un nonce et un terme qui représente un produit ou un nonce N_j^j sur lequel on a appliqué une des transitions ci-dessus.

$$\mathsf{mult}(q_n(m), q_{nt}(m')) \overset{m \equiv m'}{\to} q_{narg}(m)$$

La transition suivante s'applique seulement si m' est $S'(\bot,\bot)$. Dans ce cas le terme est considéré comme un terme possible de $\rho(K)$.

$$\mathsf{mult}(q_{nonly1s'}(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{exp}(m)$$

Il est alors possible d'appliquer la dernière transition :

$$\mathsf{exp}(q_g(m), q_{exp}(m')) \to q_{acc}(h(m,m'))$$

Exécution de $\mathcal{A}_K$

Afin de donner un peu plus d'intuition concernant la définition de l'automate $\mathcal{A}_K$, nous nous proposons ici de décrire une de ses exécutions sur $\rho(t)$ pour un $t = g^{N_3^3\cdot(N_2^3\cdot N_1^3)}$.

Nous avons donc par définition de ρ, $\rho(t)$ correspond au terme suivant :

$$\mathsf{exp}(g, \mathsf{mult}(N(0,s'(0,s'(0,s'(0,0)))), t(\mathsf{mult}(N(0,s'(0,s'(0,s(0,0)))), t(N(0,s'(0,s(0,s(0,0))))))))$$

On peut vérifier que $W(\rho(t))$ est le terme :

$$\mathsf{exp}(g, \mathsf{mult}(N(0,s'(0,s(0,s(0,0)))), t(\mathsf{mult}(N(0,s'(0,s'(0,s(0,0)))), t(N(0,s'(0,s'(0,s'(0,0))))))))$$

Montrons d'abord que par la transition $0 \to q_t(\bot)$ on a trivialement que $0 \to q_t(\bot)$.

Par la transition $s'(q_t(m), q_d(m')) \to q_{s'ent}(S(m,m'))$:

$$s'(0,0) \to^* q_{s'ent}(S'(\bot,\bot))$$

Par la transition $s'(q_t(m), q_{s'ent}(m')) \to q_{s'ent}(S(m,m'))$:

$$s'(0,s'(0,s'(0,0))) \to^* q_{s'ent}(S'(\bot,S'(\bot,S'(\bot,\bot))))$$

Par la transition $N(q_t(m), q_{s'ent}(m')) \to q_{nent}(m')$:

$$N(0,s'(0,s'(0,s'(0,0)))) \to^* q_{nent}(S'(\bot,S'(\bot,S'(\bot,\bot))))$$

Par le même type d'analyse on vérifie que :

$$N(0,s'(0,s'(0,s(0,0)))) \to^* q_n(S'(\bot,S'(\bot,\bot)))$$

et que

$$N(0,s'(0,s(0,s(0,0)))) \to^* q_{nonly1s'}(S'(\bot,\bot))$$

Par la transition $t(q_d(m), q_{nent}(S'(m',m''))) \to q_{nt}(m'')$:

$$t(N(0,s'(0,s'(0,s'(0,0))))) \to^* q_{nt}(S'(\bot,S'(\bot,\bot)))$$

Par la transition $\mathsf{mult}(q_n(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{narg}(m)$:

$$\mathsf{mult}(N(0,s'(0,s'(0,s(0,0)))), t(N(0,s'(0,s'(0,s'(0,0)))))) \to^* q_{narg}(S'(\bot,S'(\bot,\bot)))$$

Par la transition $t(q_d(m), q_{narg}(S'(m',m''))) \to q_{nt}(m'')$:

$$t(\mathsf{mult}(N(0,s'(0,s'(0,s(0,0)))), t(N(0,s'(0,s'(0,s'(0,0))))))) \to^* q_{nt}(S'(\bot,\bot))$$

Par la transition $\mathsf{mult}(q_{nonly1s'}(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{exp}(m)$:

$$smult(N(0,s'(0,s(0,s(0,0)))), t(\mathsf{mult}(N(0,s'(0,s'(0,s(0,0)))), t(N(0,s'(0,s'(0,s'(0,0))))))))$$
$$\to^* \quad q_{exp}(S'(\bot,\bot))$$

Enfin comme $g \to q_g(\bot)$ par la transition $g \to q_g(\bot)$,

$$\rho(t) \to^* q_{acc}(\bot, S'(\bot,\bot))$$

Correction de la représentation de $W(\rho(K))$ par $\mathcal{A}_K$

Le lemme suivant établit que notre automate reconnaît en fait une légère sur-approximation de $W(\rho(K))$ puisqu'il reconnaît aussi des termes qui ne sont pas des témoins (mais qui sont néanmoins dans $\rho(K)$.

Lemme 16 *$W(\rho(K)) \subseteq L(\mathcal{A}_K) \subseteq \rho(K)$.*

Nous prouvons ce lemme en plusieurs étapes dont témoignent les lemmes suivants. Nous commencerons par montrer que $W(\rho(K)) \subseteq L(\mathcal{A}_K)$ (lemme 20) ce qui nécessitera de prouver le lemme 19. Ensuite nous montrerons que $L(\mathcal{A}_K) \subseteq \rho(K)$ (lemme 21).

Avant tout prouvons le lemme technique 17 utilisé dans la démonstration du lemme 18. La démonstration du lemme 19 consistera en une application directe du lemme 18.

Lemme 17 *Soit $s = N_{i_1}^{j_{i_1}}, \ldots, N_{i_n}^{j_{i_n}}$ une séquence de termes de $T(\mathcal{F})$ où $n \geq 1$. $\rho(N_{i_1}^{j_{i_1}} \cdot (\cdots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$ où p est un produit de la forme $(u_1 \cdot u_2) \cdot v$.*

Démonstration. Par induction sur la longueur l de s.

Cas de base : $l = 1$. Montrons que $\rho(N_{i_1}^{j_{i_1}}) <_{lpo} \rho((u_1 \cdot u_2) \cdot v)$. Puisque $N <_{\mathcal{F}'} \mathsf{mult}$, $s' <_{\mathcal{F}'} \mathsf{mult}$, $s <_{\mathcal{F}'} \mathsf{mult}$ et $0 <_{\mathcal{F}'} \mathsf{mult}$, nous obtenons par la définition de lpo et de ρ que $\rho(N_{i_1}^{j_{i_1}}) <_{lpo} \rho((u_1 \cdot u_2) \cdot v)$. En effet $\rho(N_{i_1}^{j_{i_1}})$ est de la forme $N(0, \underbrace{s'(0, \ldots s'}_{i_1 \text{fois}}(\underbrace{s(0, \ldots s}_{j_1 - i_1 \text{fois}}(0,0) \ldots)) \ldots))$

Cas inductif : $l = k+1$. Nous devons montrer que $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$ où p est un produit de la forme $(u_1 \cdot u_2) \cdot v$.

Puisque $N <_{\mathcal{F}'} \mathsf{mult}$, $s' <_{\mathcal{F}'} \mathsf{mult}$, $s <_{\mathcal{F}'} \mathsf{mult}$ et $0 <_{\mathcal{F}'} \mathsf{mult}$, nous obtenons par la définition de lpo que $\rho(N_{i_1}^{j_{i_1}}) <_{lpo} \rho(u_1 \cdot u_2)$.

Il nous reste donc à montrer que $t(0, \rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots))) <_{lpo} \rho(p)$. Cela revient à montrer que $\rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$. Puisque la longueur de $N_{i_2}^{j_{i_2}}, \ldots, N_{i_n}^{j_{i_n}}$ est k, par induction nous obtenons que $\rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$. Puisque d'une part $\rho(N_{i_1}^{j_{i_1}}) <_{lpo} \rho(u_1 \cdot u_2)$ et d'autre part $t(0, \rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots))) <_{lpo} \rho(p)$, nous avons $\rho(N_{i_1}^{j_{i_1}} \cdot (\cdots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$ où p est un produit de la forme $(u_1 \cdot u_2) \cdot v$. □

Lemme 18 *Soit $s = N_{i_1}^{j_{i_1}}, \ldots, N_{i_n}^{j_{i_n}}$ une séquence de termes de $T(\mathcal{F})$ telle que $i_1 <_{\mathbb{N}} \cdots <_{\mathbb{N}} i_n$. Soit $s_p = N_{k_1}^{j_{k_1}}, \ldots, N_{k_m}^{j_{k_m}}$ une séquence de termes et pour tous i, j si $N_i^j \in s$ alors $N_i^j \in s_p$. Si p est un produit des éléments de s_p syntaxiquement distinct de $N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$ alors $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$.*

Démonstration. Soient l_s la longueur de s et l_p la longueur de p. Nous prouvons le résultat par induction sur les paires des longueurs (l_s, l_p) ordonnées selon l'ordre lexicographiques.

Cas de base : $l_s = 1$. Nous avons que $l_p > 1$. En effet, si tel n'était pas le cas, nous aurions que $l_p = 1$ et puisque pour tous i, j si $N_i^j \in s$ alors $N_i^j \in s_p$ alors $p = N_{i_1}^{j_{i_1}}$. Le produit p ne serait pas syntaxiquement distinct de $N_{i_1}^{j_{i_1}}$. Nous avons donc que $\rho(N_{i_1}^{j_{i_1}})$ est un sous-terme strict de $\rho(p)$, ce qui implique par définition de lpo que $\rho(N_{i_1}^{j_{i_1}}) <_{lpo} \rho(p)$.

Cas inductif : $l_s > 1$. Nous supposons que pour tout couple $(l_s^-, l_p^-) <_{lex} (l_s, l_p)$, les séquences $s^- = N_{i_1^-}^{j_{i_1^-}}, \ldots, N_{i_n^-}^{j_{i_n^-}}$ et $s_p^- = N_{k_1^-}^{j_{k_1^-}}, \ldots, N_{k_m^-}^{j_{k_m^-}}$ satisfont la propriété exprimée par le lemme.

(1) Soit p est de la forme $(u_1 \cdot u_2) \cdot v$. Par le lemme 17 nous obtenons directement le résultat.

(2) Soit $p = N_{k_1}^{j_{k_1}} \cdot p_2$ avec $N_{k_1}^{j_{k_1}} = N_{i_1}^{j_{i_1}}$. Dans ce cas soient $s_2 = N_{i_2}^{j_{i_2}}, \ldots, N_{i_n}^{j_{i_n}}$ la séquence de longueur l_{s2} et $s_{p2} = N_{k_2}^{j_{k_2}}, \ldots, N_{k_m}^{j_{k_m}}$ de longueur l_{p2}. Nous pouvons constater que pour tous i, j si $N_i^j \in s_2$ alors $N_i^j \in s_{p2}$. p_2 est syntaxiquement distinct de $N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$. Si ce n'était pas le cas, p ne serait pas syntaxiquement distinct de $N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$. Par hypothèse d'induction, $\rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p_2)$. Par la définition de lpo et de ρ, ceci nous permet de conclure que dans ce cas $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$.

(3) Soit $p = N_{k_1}^{j_{i_1}} \cdot p_2$ avec $i_1 < k_1$. Dans ce cas, par la remarque 22, $N_{i_1}^{j_{i_1}} <_{lpo} N_{k_1}^{j_{k_1}}$. D'autre part, p est syntaxiquement distinct de $N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$ car p contient $N_{i_1}^{j_{i_1}}$. Par hypothèse d'induction appliquée à $s_2 = N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$ et s_p, $\rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$. Puisque $0, t <_{\mathcal{F}'} \mathsf{mult}$, nous avons $t(0, \rho(N_{i_2}^{j_{i_2}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots))) <_{lpo} \rho(p)$. Par la définition de lpo, ceci nous permet de conclure que dans ce cas $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$ où p est un produit des éléments de s_p.

(4) $p = N_{k_1}^{j_{i_1}} \cdot p_2$ avec $i_1 > k_1$. Si $p_2 = N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$, $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$ car $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots))$ est un sous-terme strict de $\rho(p)$. Sinon p_2 est syntaxiquement distinct de $N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)$. Par hypothèse d'induction appliquée à s et $s_{p_2} = N_{k_2}^{j_{k_2}}, \ldots, N_{k_m}^{j_{k_m}}$, $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p2)$. Remarquons que l'application de l'hypothèse d'induction est possible puisque $i_1 > k_1$, et donc $N_{k_1}^{j_{i_1}}$ n'apparaît pas dans s. Nous avons donc que $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} t(0, \rho(p2))$ et donc $\rho(N_{i_1}^{j_{i_1}} \cdot (\ldots (N_{i_{n-1}}^{j_{i_{n-1}}} \cdot N_{i_n}^{j_{i_n}}) \ldots)) <_{lpo} \rho(p)$.

□

Lemme 19 *Soit $N_1^{j_1}, \ldots, N_n^{j_n}$, où $n > 2$, une séquence de termes de $T(\mathcal{F})$ et $N_{i'_1}^{j_{i'_1}}, \ldots, N_{i'_n}^{j_{i'_n}}$ une de ses permutations. Pour tout produit $p = N_{i'_1}^{j_{i'_1}} \cdot \cdots \cdot N_{i'_n}^{j_{i'_n}}$ syntaxiquement distinct de $N_1^{j_1} \cdot (\ldots (N_{n-1}^{j_{n-1}} \cdot N_n^{j_n}) \ldots)$, $\rho(N_1^{j_1} \cdot (\ldots (N_{n-1}^{j_{n-1}} \cdot N_n^{j_n}) \ldots)) <_{lpo} \rho(p)$.*

Démonstration. Il suffit de remarquer que $N_1^{j_1}, \ldots, N_n^{j_n}$, où $n > 2$ est une séquence de termes $s = N_{i_1}^{j_{i_1}}, \ldots, N_{i_n}^{j_{i_n}}$ de $T(\mathcal{F})$ telle que $i_1 <_{\mathbb{N}} \cdots <_{\mathbb{N}} i_n$, et d'autre part qu'une permutation $N_{i'_1}^{j_{i'_1}}, \ldots, N_{i'_n}^{j_{i'_n}}$ est une séquence s_p de termes telle que pour tous i, j si $N_i^j \in s$ alors $N_i^j \in s_p$. Il nous suffit alors d'appliquer le lemme 18 pour obtenir que pour tout produit $p = N_{i'_1}^{j_{i'_1}} \cdot \cdots \cdot N_{i'_n}^{j_{i'_n}}$ syntaxiquement distinct de $N_1^{j_1} \cdot (\ldots (N_{n-1}^{j_{n-1}} \cdot N_n^{j_n}) \ldots)$, $\rho(N_1^{j_1} \cdot (\ldots (N_{n-1}^{j_{n-1}} \cdot N_n^{j_n}) \ldots)) <_{lpo} \rho(p)$. □

Lemme 20 $W(\rho(K)) \subseteq L(\mathcal{A}_K)$.

Démonstration. D'après le lemme **??**, étant donné un terme $g^{N_n^n \cdot N_{n-1}^{j_{n-1}} \cdot \ldots \cdot N_1^{j_1}}$ de K, l'élément minimal de sa classe modulo AC est obtenu en appliquant ρ à l'élément $g^{(N_1^{j_1} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)}$.

Montrons donc que $\rho(g^{(N_1^{j_1} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)}) \in L(\mathcal{A}_K)$

Pour cela nous allons commencer par montrer que

$$\rho(N_n^n) \quad \to^*_{\mathcal{A}_K} \quad q_{nent}(\underbrace{S'(\bot, \ldots S'}_{n \text{ fois}}(\bot, \bot) \ldots)) \tag{6.1}$$

Cela montré, nous montrerons que pour tout i tel que $0 \leq i \leq n-1$ on peut vérifier que

$$\rho((N_{n-i}^{j_{n-i}} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \quad \to^*_{\mathcal{A}_K} q_{narg} \quad (\underbrace{S'(\bot, \ldots S'}_{n-i \text{ fois}}(\bot, \bot) \ldots)) \tag{6.2}$$

Enfin on montrera que

$$\rho((N_1^{j_1} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \quad \to^*_{\mathcal{A}_K} \quad q_{exp}(S'(\bot, \bot)) \tag{6.3}$$

Il sera immédiat de constater que

$$\rho(g^{(N_1^{j_1} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)}) \quad \to^*_{\mathcal{A}_K} \quad q_{acc}(h(\bot, S'(\bot, \bot)))$$

En effet $g \to q_g(\bot)$ et par les assertions précédentes on peut utiliser la transition

$$\mathsf{exp}(q_g(m), q_{exp}(m')) \ \to \ q_{acc}(h(m, m'))$$

Preuve de l'équation 6.1. Par définition de la fonction ρ,

$$\rho(N_n^n) = N(0, \underbrace{s'(0, \ldots s'}_{n \text{ fois}}(0, 0) \ldots)$$

Montrons par induction sur n que

$$\underbrace{s'(0, \ldots s'}_{n \text{ fois}}(0, 0) \ldots) \to^*_{\mathcal{A}_K} q_{s'ent}(\underbrace{S'(\bot, \ldots S'}_{n \text{ fois}}(\bot, \bot) \ldots))$$

Si $n = 1$, remarquons d'abord que $0 \to q_t(\bot)$. En utilisant la transition $s'(q_t(m), q_t(m')) \to q_{s'ent}(S'(m, m'))$, on a le résultat.

Supposons que la propriété est vraie pour $n = i$. On a donc que

$$\underbrace{s'(0,\dots s'}_{i \text{ fois}}(0,0)\dots) \to^*_{\mathcal{A}_K} q_{s'ent}(\underbrace{S'(\bot,\dots S'}_{i \text{ fois}}(\bot,\bot)\dots))$$

En remarquant à nouveau que $0 \to q_t(\bot)$, et en utilisant la transition $s'(q_t(m), q_{s'ent}(m')) \to q_{s'ent}(S'(m,m'))$, on obtient

$$\underbrace{s'(0,\dots s'}_{i+1 \text{ fois}}(0,0)\dots) \to^*_{\mathcal{A}_K} q_{s'ent}(\underbrace{S'(\bot,\dots S'}_{i+1 \text{ fois}}(\bot,\bot)\dots))$$

En utilisant la transition $N(q_t(m), q_{s'ent}(m')) \to q_{nent}(m')$ on obtient l'équation 6.1

Preuve de l'équation 6.2. Par induction sur i. Si $i = 1$ cela revient à montrer que

$$\rho((N_{n-1}^{j_{n-1}} \cdot N_n^n)) \quad \to^*_{\mathcal{A}_K} q_{narg} \quad (\underbrace{S'(\bot,\dots S'}_{n-1 \text{ fois}}(\bot,\bot)\dots)) \tag{6.4}$$

Par définition de ρ,

$$\rho((N_{n-1}^{j_{n-1}} \cdot N_n^n)) = \mathsf{mult}(\rho(N_{n-1}^{j_{n-1}}), \rho(t(0, \rho(N_n^n))))$$

En utilisant l'équation 6.1, $0 \to q_t(\bot)$ et la transition $t(q_t(m), q_{nent}(S'(m',m'')) \to q_{nt}(m'')$, nous avons

$$\rho(t(0,\rho(N_n^n))) \to^*_{\mathcal{A}_K} q_{nt}(\underbrace{S'(\bot,\dots S'}_{n-1 \text{ fois}}(\bot,\bot)\dots))$$

D'autre part en effectuant une preuve similaire à la preuve de l'équation 6.1,

$$\begin{aligned}
s(q_t(m), q_d(m')) &\to q_d(m') \\
s'(q_t(m), q_d(m')) &\to q_{only1s'}(S'(m,m')) \\
s'(q_t(m), q_{only1s'}(m')) &\to q_{s'}(S'(m,m')) \\
s'(q_t(m), q_{s'}(m')) &\to q_{s'}(S'(m,m')) \\
N(q_t(m), q_{s'}(m')) &\to q_n(m')
\end{aligned}$$

on obtient

$$\rho(N_{n-1}^{j_{n-1}}) \to^*_{\mathcal{A}_K} q_n(\underbrace{S'(\bot,\dots S'}_{n-1 \text{ fois}}(\bot,\bot)\dots))$$

On peut alors utiliser la transition $\mathsf{mult}(q_n(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{narg}(m)$ et obtenir

$$\rho((N_{n-1}^{j_{n-1}} \cdot N_n^n)) \quad \to^*_{\mathcal{A}_K} \quad q_{narg}(\underbrace{S'(\bot,\dots S'}_{n-1 \text{ fois}}(\bot,\bot)\dots))$$

Par induction supposons que pour $i = k$ nous avons

$$\rho((N_{n-k}^{j_{n-k}} \cdot \dots (N_{n-1}^{j_{n-1}} \cdot N_n^n)\dots)) \quad \to^*_{\mathcal{A}_K} \quad q_{narg}(\underbrace{S'(\bot,\dots S'}_{n-k \text{ fois}}(\bot,\bot)\dots))$$

Montrons

$$\rho((N_{n-(k+1)}^{j_{n-(k+1)}} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_{narg}(\underbrace{S'(\bot, \ldots S'}_{n-(k+1) \text{ fois}}(\bot, \bot) \ldots))$$

Cela revient à montrer que

$$\rho((N_{n-(k+1)}^{j_{n-(k+1)}} \cdot (N_{n-k}^{j_{n-k}} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots))) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_{narg}(\underbrace{S'(\bot, \ldots S'}_{n-(k+1) \text{ fois}}(\bot, \bot) \ldots))$$

ou encore par définition de ρ :

$$\mathsf{mult}(\rho(N_{n-(k+1)}^{j_{n-(k+1)}}), t(0, \rho((N_{n-k}^{j_{n-k}} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)))) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_{narg}(\underbrace{S'(\bot, \ldots S'}_{n-(k+1) \text{ fois}}(\bot, \bot) \ldots))$$

En appliquant la transition $t(q_t(m), q_{nent}(S'(m', m'')) \rightarrow q_{nt}(m'')$, nous obtenons par hypothèse d'induction que

$$t(0, \rho((N_{n-k}^{j_{n-k}} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots))) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_{nt}(\underbrace{S'(\bot, \ldots S'}_{n-(k+1) \text{ fois}}(\bot, \bot) \ldots))$$

Par ailleurs en utilisant les transitions

$$\begin{array}{rcl} s(q_t(m), q_d(m')) & \rightarrow & q_d(m') \\ s'(q_t(m), q_d(m')) & \rightarrow & q_{only1s'}(S'(m, m')) \\ s'(q_t(m), q_{only1s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ s'(q_t(m), q_{s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ N(q_t(m), q_{s'}(m')) & \rightarrow & q_n(m') \end{array}$$

nous obtenons

$$\rho(N_{n-(k+1)}^{j_{n-(k+1)}}) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_n(\underbrace{S'(\bot, \ldots S'}_{n-(k+1) \text{ fois}}(\bot, \bot) \ldots))$$

On peut donc appliquer $\mathsf{mult}(q_n(m), q_{nt}(m')) \overset{m \equiv m'}{\rightarrow} q_{narg}(m)$ ce qui nous permet de conclure.

Preuve de l'équation 6.3 .

Par l'équation 6.2 nous avons

$$\rho((N_2^{j_2} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \quad \rightarrow_{\mathcal{A}_K}^* q_{narg} \quad (\underbrace{S'(\bot, \ldots S'}_{2 \text{ fois}}(\bot, \bot) \ldots))$$

Nous pouvons constater que

$$\rho(N_1^{j_1}) \quad \rightarrow_{\mathcal{A}_K}^* \quad q_{nonly1s'}(S'(\bot, \bot))$$

Comme

$$\rho((N_1^{j_1} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \quad = \quad \mathsf{mult}(\rho((N_1^{j_1}), t(0, \rho((N_2^{j_2} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)))))$$

on peut aisément appliquer la transition

$$\mathsf{mult}(q_{nonly1s'}(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{exp}(m)$$

et on obtient le résultat. □

Lemme 21 *$L(\mathcal{A}_K) \subseteq \rho(K)$.*

Démonstration. Soit $t \in L(\mathcal{A}_K)$. Comme il n'y a qu'un seul état acceptant dans $\mathcal{A}_K$ et que la seule transition menant à cet état est $\mathsf{exp}(q_g(m), q_{exp}(m')) \to q_{acc}(h(m, m'))$, il existe deux termes $m, m' \in T(\mathcal{G})$ et deux termes de $u, v \in T(\mathcal{F}')$ tels que $t = exp(u, v)$ et $t \to^*_{\mathcal{A}_K} q_{acc}(h(m, m')$.

Puisque la seule transition dont l'état d'arrivée est q_g est $g \to q_g(\bot)$ on peut conclure que $u = g$ et $m = \bot$. t est donc de la forme $exp(g, v)$.

Par ailleurs, la seule transition menant à q_{exp} est

$$\mathsf{mult}(q_{nonly1s'}(m), q_{nt}(m')) \overset{m\equiv m'}{\to} q_{exp}(m)$$

il existe deux termes $m_v, m'_v \in T(\mathcal{G})$ et deux termes $v_1, v_2 \in T(\mathcal{F}')$ tels que $v = \mathsf{mult}(v_1, v_2)$ et $v \to^*_{\mathcal{A}_K} q_{exp}(m'_v)$ et $m_v \equiv m'_v$. Puisqu'en plus $v_1 \to^*_{\mathcal{A}_K} q_{nonly1s'}(m'_v)$ on peut aisément déduire que $v_1 = \rho(N_1^j)$ pour $j \geq 1$ et $m'_v = S'(\bot, \bot)$.

D'autre part, si $v_2 \to^*_{\mathcal{A}_K} q_{nt}(S'(\bot, \bot))$, montrons par induction sur la taille de v_2 que $v_2 = t(0, \rho((N_2^{j_2} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots))$.

Soit v_2 est le plus petit possible pour que pour obtenir le fait que $v_2 \to^*_{\mathcal{A}_K} q_{nt}(S'(\bot, \bot))$, on a utilisé la transition $t(q_t(m), q_{nent}(S'(m', m'')) \to q_{nt}(m'')$. Ceci implique que v_2 est de la forme $t(v'_2, v''_2)$ tels que $v'_2 \to q_t(m)$ et $v''_2 \to^*_{\mathcal{A}_K} q_{nent}(S'(m', S'(\bot, \bot)))$. Sans ambiguïté $v'_2 = 0$ et $m = \bot$. D'autre part v''_2 est de la forme $N(v_2^-, v_2^{-'})$ où $v_2^- \to^*_{\mathcal{A}_K} q_t(m)$ ce qui implique que $v_2^- = 0$ et $v_2^{-'} \to^*_{\mathcal{A}_K} q_{s'ent}(S'(m', S'(\bot, \bot)))$. Il est alors facile de constater que $v_2^{-'} = s'(0, s'(0, 0))$. Ceci nous permet de déduire par définition de ρ que $v_2 = t(0, \rho(N_2^2))$.

Soit v_2 n'est pas le plus petit possible pour obtenir le fait que $v_2 \to^*_{\mathcal{A}_K} q_{nt}(S'(\bot, \bot))$. Dans ce cas, la transition employée a été $t(q_t(m), q_{narg}(S'(m', m'')) \to q_{nt}(m'')$. Ceci implique que v_2 est de la forme $t(v'_2, v''_2)$ tels que $v'_2 \to q_t(m)$ et $v''_2 \to^*_{\mathcal{A}_K} q_{narg}(S'(m', S'(\bot, \bot)))$. Sans ambiguïté $v'_2 = 0$ et $m = \bot$. D'autre part v''_2 est de la forme $\mathsf{mult}(v_2^-, v_2^{-'})$ où $v_2^- \to^*_{\mathcal{A}_K} q_n(m)$ et $v_2^{-'} \to^*_{\mathcal{A}_K} q_{nt}(m')$ et $m = m' = S'(m', S'(\bot, \bot))$. D'une part on peut vérifier que si $v_2^- \to^*_{\mathcal{A}_K} q_n(m)$ cela implique que $v_2 = \rho(N_2^j)$ où $j \geq 2$, et par induction on a que $v_2 = t(0, \rho((N_3^{j_3} \cdot \ldots (N_{n-1}^{j_{n-1}} \cdot N_n^n) \ldots)) \to^*_{\mathcal{A}_K})$. On peut alors conclure. □

Démonstration (du lemme 16). Application directe des lemmes 20 et 21. □

Définitions de E_1 et E_2

On donne ici simplement les définitions des automates représentant E_1 et E_2.

Définition de $\mathcal{A}_{E_1}$: $Q_{\mathcal{A}_{E_1}} = \{q_d, q_{1s'}, q_s, q_{nt}, q_{narg}, q_g, q_{acc}\}$ $Q_{f\mathcal{A}_{E_1}} = \{q_{acc}\}$

$$\begin{array}{rcl} 0 & \rightarrow & q_t(\bot) \\ 0 & \rightarrow & q_d(\bot) \\ g & \rightarrow & q_g(\bot) \end{array}$$

$$\begin{array}{rcl} s(q_t(m), q_d(m')) & \rightarrow & q_d(m') \\ s'(q_t(m), q_d(m')) & \rightarrow & q_{1s'}(S(m, m')) \\ s'(q_t(m), q_{1s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ s'(q_t(m), q_{s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ s(q_t(m), q_{s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ N(q_t(m), q_{s'}(m')) & \rightarrow & q_n(m') \end{array}$$

$$\begin{array}{rcl} t(q_t(m), q_n(S'(m', m'')) & \rightarrow & q_{nt}(m'') \\ t(q_t(m), q_{narg}(S'(m', m'')) & \rightarrow & q_{nt}(m'') \end{array}$$

$$\begin{array}{rcl} \mathsf{mult}(q_n(m), q_{nt}(m')) & \stackrel{m \equiv m'}{\rightarrow} & q_{narg}(m) \\ \mathsf{mult}(q_n(m), q_{nt}(m')) & \stackrel{m \not\equiv m'}{\rightarrow} & q_{narg}(m) \end{array}$$

$$\begin{array}{rcl} \mathsf{exp}(q_g(m), q_{nt}(m')) & \rightarrow & q_{acc}(h(m, m')) \\ \mathsf{exp}(q_g(m), q_n(m')) & \rightarrow & q_{acc}(h(m, m')) \end{array}$$

Définition de $\mathcal{A}_{E_2}$: $Q_{\mathcal{A}_{E_2}} = \{q_d, q_s, q_{nt}, q_{narg}, q_g, q_{acc}\}$ $Q_{f\mathcal{A}_{E_2}} = \{q_{acc}\}$

$$\begin{array}{rcl} 0 & \rightarrow & q_t(\bot) \\ 0 & \rightarrow & q_d(\bot) \\ g & \rightarrow & q_g(\bot) \end{array}$$

$$\begin{array}{rcl} s(q_t(m), q_d(m')) & \rightarrow & q_s(m') \\ s'(q_t(m), q_s(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ s'(q_t(m), q_{s'}(m')) & \rightarrow & q_{s'}(S'(m, m')) \\ N(q_t(m), q_{s'}(m')) & \rightarrow & q_n(m') \end{array}$$

$$\begin{array}{rcl} t(q_t(m), q_n(S'(m', m'')) & \rightarrow & q_{nt}(m'') \\ t(q_t(m), q_{narg}(S'(m', m'')) & \rightarrow & q_{nt}(m'') \\ \mathsf{mult}(q_n(m), q_{nt}(m')) & \stackrel{m \equiv m'}{\rightarrow} & q_{narg}(m) \\ \mathsf{exp}(q_g(m), q_{nt}(m')) & \rightarrow & q_{acc}(h(m, m')) \\ \mathsf{exp}(q_g(m), q_n(m')) & \rightarrow & q_{acc}(h(m, m')) \end{array}$$

6.5.3 Bonne formation de notre représentation

Lemme 22 *$(L(\mathcal{A}_{E_1}) \cup L(\mathcal{A}_{E_2}), L(\mathcal{A}_K))$ est bien formé.*

Démonstration. Comme aucune transition n'a de côté gauche qui a un xor en tête, les contraintes (1) et (3) de la définition 6 sont satisfaites. On peut vérifier sur la construction des automates $\mathcal{A}_{E_1}$ et $\mathcal{A}_{E_2}$ que chaque terme accepté par ces automates est de la forme $\mathsf{exp}(u, v)$ pour certains u et v. Par définition 5, cela implique que $C(L(\mathcal{A}_{E_1}) \cup L(\mathcal{A}_{E_2})) = \emptyset$. □

Remarques conclusives

Nous avons montré par une série de réductions et de sur-approximations une maniére d'obtenir une analyse automatique d'une approximation de la propriété de secret pour une classe de protocoles de groupe en présence d'un intrus passif. Cela peut être obtenu en utilisant une classe particulière d'automates d'arbres les VTAM$^{\equiv}_{\neq}$. Cette méthode n'est pas exacte du fait de certaines approximations.

Nous avons montré comment représenter les indices des variables, comment représenter l'associativité et la commutativité de certains opérateurs, et la clôture sous Dolev-Yao de cette classe d'automates.

Bien que notre approche permette de représenter des classes de protocoles comportant à la fois des opérations comme le ou exclusif et une exponentiation modulaire simplifiée, nous pouvons critiquer certains défaut de notre travail. Bien que certaines conditions de la bonne formation puissent paraître naturelles, d'autres ne le sont pas. On pourrait espérer trouver d'autres conditions permettant d'aboutir à un résultat similaire mais qui restent néanmoins plus naturelles.

Un autre défaut de notre modèle est le fait que la représentation de l'exponentiation modulaire que nous avons est quelque peu restreinte. En effet, l'opération $\cdot$ qui prend en argument les éléments de l'exposant n'est pas réellement une opération représentant celle qui a lieu dans un groupe puisqu'elle n'autorise pas la représentation et donc la manipulation d'un inverse.

Enfin, le problème le plus important est le manque d'automaticité de l'extraction du modèle d'un protocole dans les VTAM$^{\equiv}_{\neq}$ à partir d'une spécification. En effet, le lecteur scrupuleux aura pu constater la longueur et surtout la lourdeur des démonstrations d'une adéquation entre le modèle pour les VTAM$^{\equiv}_{\neq}$ et le simple ensemble de termes représentant une infinité d'exécutions du protocoles. Cette automatisation prendrait véritablement son sens si l'on pouvait éprouver notre modélisation. Et ceci nécessite d'implémenter cette classe particulière d'automates, ce qui d'ailleurs pourrait être un travail fastidieux mais intéressant en soi.

Deuxième partie

Réduction de théories équationnelles

Chapitre 7

Équivalence statique et combinaison de théories équationnelles

Sommaire

Dans ce chapitre, nous présentons le contexte dans lequel se situe le résultat de la partie II publié dans [KMT09a]. Nous commençons ainsi par présenter la notion d'équivalence statique sur laquelle il porte. Nous présentons ensuite des résultats de combinaison de théories équationnelles qui ont déjà été obtenus, ce qui nous permettra de mieux situer notre propre résultat de combinaison. Enfin nous présentons la théorie du couplage bilinéaire sur lequel il s'applique.

7.1 Équivalence statique

Comme nous l'avons annoncé, dans cette partie nous nous concentrons sur la vérification de la propriété de secret au sens de l'indistinguabilité et ce dans le cas d'un intrus passif.

7.1.1 Équivalence dans le cas passif

Nous analyserons ce que l'on appelle l'*équivalence statique*. Cette notion trouve son origine dans le pi-calcul appliqué [AF01]. Intuitivement la notion d'équivalence statique modélise le

fait qu'un intrus ne peut pas distinguer deux suites de messages, appelées plus tard *cadres* (*frames* en anglais), en exhibant une relation entre les messages qui existent dans une suite mais pas dans l'autre. Remarquons cependant que cette notion d'équivalence statique a aussi été utilisée pour l'analyse de protocoles en présence d'un intrus actif, afin de caractériser des équivalences de processus [AF01] et les attaques par dictionnaire hors ligne [CDE04, Bau05]. Il existe désormais des algorithmes exacts [AC06], et approximatifs [ABF08] pour décider l'équivalence statique, et ce parfois en présence de théories équationnelles comme dans [AC06].

7.1.2 Les cadres

Une différence importante avec le problème de l'atteignabilité est que l'on ne considère plus simplement l'ensemble des messages que l'intrus a intercepté. On considère aussi l'ordre dans lequel ces messages ont été émis.

Une autre nouveauté est que l'on va représenter la *fraîcheur* de certains éléments des messages. Par fraîcheur nous entendons le fait qu'ils soient propres à une session du protocole. Un exemple de partie de message dont on peut s'attendre à ce qu'il soit frais est le nonce. Rappelons que dans la partie I, nous devions représenter le fait pour un élément d'être propre à un protocole par des indices (sections 4.1, 4.2 et 4.3), c'est-à-dire au niveau du choix de la signature. Grâce aux cadres, c'est avant même le choix d'une quelconque signature que la modélisation de la fraîcheur est permise. Ces deux aspects, l'ordre des messages et la fraîcheur de certaines valeurs, sont donc au centre de la définition des cadres.

Définition 13 (cadre) *Un* cadre *est une expression $\phi = \nu\tilde{n}_\phi.\sigma_\phi$ où $\tilde{n}_\phi$ est un ensemble de* noms liés *et σ_ϕ est une substitution.*

Notons que les noms liés représentent les valeurs fraîches que nous venons d'évoquer.

Exemple 1 Considérons la signature non typée $\mathcal{F}_{enc} = \{enc/2, dec/2, 0/0, 1/0\}$ et la théorie équationnelle $E_{enc} = dec(enc(x,y)y) = x$ définie sur $\mathcal{F}_{enc}$. Le cadre $\phi_1 = \nu k.\{x \mapsto enc(0,k), y \mapsto k\}$ représente une séquence de messages émis. Le premier est le chiffrement de la valeur 0 par une clé fraîche k, le second message est la clé elle-même.

$|\phi|$ est la taille de ϕ, c'est-à-dire le nombre d'éléments dans $\mathrm{dom}(\sigma_\phi)$. Nous appellerons σ_ϕ la *substitution sous-jacente* de ϕ. Nous étendons la notation *dom* aux cadres de la manière suivante : $\mathrm{dom}(\nu\tilde{n}.\sigma) = \mathrm{dom}(\sigma)$. Nous écrirons $\phi =_\alpha \psi$ lorsque deux cadres ϕ et ψ sont égaux à alpha-renommage près des noms liés.

Considérons $\phi = \nu\tilde{n}_\phi.\sigma_\phi$ et $\psi = \nu\tilde{n}_\psi.\sigma_\psi$ deux cadres tels que $\mathrm{dom}(\phi) \cap \mathrm{dom}(\psi) = \emptyset$ et $\tilde{n}_\phi \cap \tilde{n}_\psi = \emptyset$. La *composition disjointe* de ϕ et ψ, notée $\phi\psi$ est définie comme suit : $\phi\psi = \nu(\tilde{n}_\phi \cup \tilde{n}_\psi).\sigma_\phi\sigma_\psi$. Remarquons que $\tilde{n}_\phi \cap \tilde{n}_\psi = \emptyset$ est toujours possible par alpha-renommage des noms liés de ϕ et ψ. Le type d'un cadre ϕ est l'ensemble $S = \{\mathsf{type}(x) | x \in \mathrm{dom}(\phi)\}$, et nous disons alors que ϕ est *S-typé*.

L'application d'un cadre $\phi = \nu\tilde{n}.\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ à un contexte M notée $M\phi$ est défini comme $M[t_1, \ldots, t_n]$.

Nous avons pu définir un contexte public comme étant un contexte qui ne comporte pas de noms (sous-section 2.1.4), car, sans perte de généralité, nous pouvons considérer uniquement des cadres $\phi = \nu\tilde{n}.\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ tels que tous les noms utilisés dans ce cadre sont liés, c'est-à-dire pour lesquels $\tilde{n} = \mathsf{noms}(t_1, \ldots, t_n)$. Ainsi dans le cas où nous aurions voulu considérer un contexte qui comporte un nom a libre, il est suffisant de considérer une variable x_a à la place de ce nom dans ce contexte et d'ajouter une application $x_a \mapsto a$ au cadre.

7.1.3 Équivalence statique formellement

Les cadres étant définis, il est possible d'établir une notion de déduction similaire à celle que nous avons définie dans la partie I, tenant compte de la spécificité des cadres.

Définition 14 (Déduction) *Soit E une théorie équationnelle, M un terme clos et $\nu\tilde{n}.\sigma$ un cadre où tous les noms sont liés. Alors $\nu\tilde{n}.\sigma \vdash M$ si et seulement si il existe un contexte public C tel que $C\sigma =_E M$.*

Rappelons un exemple donné dans [AC06] permettant d'illustrer plus précisément le besoin d'établir une notion plus forte que la simple déduction.

Exemple 2 (suite de l'exemple 1) Nous considérons les mêmes signatures et théories équationnelles que dans l'exemple 1. Nous considérons à nouveau le cadre $\phi_1 = \nu k.\{x \mapsto enc(0,k), y \mapsto k\}$, mais aussi le cadre $\phi_2 = \nu k.\{x \mapsto enc(1,k), y \mapsto k\}$. Puisque 0 et 1 sont des constantes, l'intrus peut déduire le même ensemble de termes des deux cadres.

Définissons maintenant formellement cette notion d'indistinguabilité de cadres.

Définition 15 (Égalité dans un cadre [AF01]) *On dit que deux termes M et N sont* égaux dans un cadre ϕ *pour la théorie équationnelle E, et on écrit $(M =_E N)\phi$, si et seulement si $\phi =_\alpha \nu\tilde{n}.\sigma$, $M\sigma =_E N\sigma$, et $\{\tilde{n}\} \cap (\mathsf{noms}(M) \cup \mathsf{noms}(N)) = \emptyset$.*

Définition 16 (Équivalence statique [AF01]) *Deux cadres ϕ et ψ sont* statiquement équivalents *pour la théorie équationnelle E, noté $\phi \approx_E \psi$, si et seulement si $dom(\phi) = dom(\psi)$, et pour tous termes M et N, on a $(M =_E N)\phi$ si et seulement si $(M =_E N)\psi$.*

Pour deux cadres ϕ et ψ, nous dirons que deux termes M, N sont des *distingueurs* de ϕ et ψ pour une théorie equationnelle E si $(M =_E N)\phi$ et $(M \neq_E N)\psi$ ou si $(M \neq_E N)\phi$ et $(M =_E N)\psi$. Lorsque la théorie équationnelle est claire à partir du contexte dans lequel « distingueur » est employé, nous ne préciserons pas de quelle théorie il s'agit.

Exemple 3 (suite de l'exemple 2) Considérons les cadres ϕ_1 et ϕ_2 définis dans l'exemple 2. Ces cadres sont distingables et donc non équivalents. Les distingeurs sont en effet $M = dec(x,y)$ et $N = 0$. Nous pouvons aisément vérifier que $(M =_E N)\phi_1$ et $(M \neq_E N)\phi_2$.

De même, nous pouvons remarquer que les cadres $\nu k.\{x \mapsto enc(0,k)\}$ et $\nu k.\{x \mapsto enc(1,k)\}$ ne sont pas distinguables.

7.1.4 Problème de décision de l'équivalence statique

Nous sommes désormais en mesure de définir le problème vers lequel ce travail est orienté :

Données : deux cadres ϕ et ψ,
une théorie équationnelle E.
Question : $\phi \approx_E \psi$?

Des résultats de décidabilité relatifs à de nombreuses théories équationnelles ont été établis. Les théories sous-termes convergentes sont décidables [AC06, CDK09]. Les théories monoïdales parmi lesquelles se trouvent (ACU, ACUI, ACUN) sont décidables [CD07]. Des résultats concernant d'autres théories que nous ne détaillerons pas ont été obtenus dans [AC06, CDK09, BBC09]. Ces théories représentent la signature en aveugle, le chiffrement homomorphique, la « preuve à vérificateur désigné » ou encore le « Trapdoor bit-commitment ».

7.2 Combinaison de théories équationnelles

Le résultat que nous présentons dans cette deuxième partie peut être vu de deux manières distinctes : une réduction ou une combinaison.

Le but de notre travail est d'établir des critères permettant de *réduire* de façon algorithmique le problème de l'équivalence statique modulo une théorie équationnelle, à d'autres problèmes d'équivalence statique modulo des théories équationnelles plus simples. Il s'agit alors de faire en sorte que, pour ces théories plus simples, le problème auquel nous nous intéressons soit décidable. Ces critères définissent une classe $\mathcal{E}$ de théories équationnelles.

Plus formellement, il s'agit d'établir une application associant à chaque triplet (ϕ, ψ, E) où ϕ et ψ sont des cadres et $E \in \mathcal{E}$ un n-uplet $((\phi_1, \psi_1, E_1), \ldots, (\phi_n, \psi_n, E_n))$ tel que :

$$\phi \approx_E \psi \text{ ssi } (\phi_1 \approx_{E_1} \psi_1 \text{ et } \ldots \text{ et } \phi_n \approx_{E_n} \psi_n)$$

A l'inverse, nous pouvons considérer que ce travail débouche sur un résultat de combinaison. Il s'agit de voir à quelles conditions des théories équationnelles peuvent être combinées en ce sens que leur réunion préserve la décidabilité du problème de décision de l'équivalence statique. Il s'agit aussi de définir une classe $\mathcal{E}$ de théories telles que si pour des théories $E_1 \ldots E_n$ le problème de décision de l'équivalence statique est décidable, alors si $E_1 \cup \cdots \cup E_n \in \mathcal{E}$, $E_1 \cup \cdots \cup E_n$ est décidable.

Nous avons présenté ici la combinaison de théories pour l'équivalence statique. Ce principe de combinaison de théorie peut bien évidemment être décliné pour tout problème comportant parmi ses données une théorie équationnelle. On peut par exemple parler de combinaison pour le problème de déduction, de satisfaisabilité d'un système de contraintes etc...

Nous pouvons distinguer deux types de combinaisons que nous présentons successivement :
- combinaison de théories disjointes,
- combinaison de théories non-disjointes.

Nous illustrerons les deux premières en indiquant certains résultats déjà obtenus.

7.2.1 Combinaison de théories disjointes

Un premier type de combinaison est la combinaison de théories disjointes. Il s'agit de considérer deux signatures disjointes $\mathcal{F}_1$ et $\mathcal{F}_2$ telles que $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ et deux théories E_1 et E_2 spécifiées respectivement dans $\mathcal{F}_1$ et dans $\mathcal{F}_2$. Remarquons que cela implique que $E_1 \cap E_2 = \emptyset$.

Exemple 4 ([ACD07]) Ces résultats constituent des résultats de combinaison pour des théories disjointes. Ils s'articulent en deux étapes. La première établit la combinaison des théories disjointes pour le problème de la déduction. La seconde étape établit la combinaison de deux théories disjointes E_1 et E_2 pour l'équivalence statique à condition que non seulement l'équivalence statique mais aussi la déduction soient décidables pour les theories E_1 et E_2. Ce dernier point constitue une condition supplémentaire pour le résultat de combinaison pour le problème de l'équivalence statique. Remarquons que ces résultats ne dépendent pas des types des signatures, ils sont donc valables, que les signatures sur lesquelles portent E_1 et E_2 soient typées ou non.

7.2.2 Combinaison de théories non-disjointes

Lorsque l'on considère une combinaison de théories non-disjointes, les choses sont moins claires. Soient deux signatures $\mathcal{F}_1$ et $\mathcal{F}_2$ et deux théories E_1 et E_2 telles que

- E_1 est spécifiée sur $\mathcal{F}_1$,
- E_2 est spécifiée sur $\mathcal{F}_2$,
- et $\mathcal{F}_1 \cap \mathcal{F}_2 \neq \emptyset$.

Il existe un résultat pour ce type de combinaison.

Exemple 5 (Combinaison pour les théories hiérarchiques [CR08]) . Il s'agit d'un résultat de combinaison pour les problèmes de déduction et de satisfaisabilité de systèmes de contraintes déterministes. Les contraintes imposées aux signatures et aux théories équationnelles reposent sur la notion de *mode*, qui est un assouplissement de la notion de type, en ce sens qu'il permet d'envisager des termes *mal modés*. Pour notre part, nous ne considérons que des termes bien typés. Notons de prime abord que la notion de mode est une notion descriptive qui permet aux auteurs de définir une notion de sous-terme et par là de localité. Contrairement à la notion de type, il ne s'agit pas d'une notion définissant l'ensemble des termes. Dans [CR08], deux signatures disjointes $\mathcal{F}_0$ et $\mathcal{F}_1$ sont considérées et deux théories équationnelles E_0 et E_1, la première étant spécifiée sur $\mathcal{F}_0$ et la seconde sur $\mathcal{F}_0 \cup \mathcal{F}_1$. C'est précisément de la possibilité de spécifier la théorie E_1 sur l'union de $\mathcal{F}_0$ et $\mathcal{F}_1$ que vient la non-disjonction, les équations de E_1 pouvant comporter des symboles apparaissant dans $\mathcal{F}_0$.

Nous évoquerons à nouveau cet exemple lorsque nous présenterons la notion centrale de réductibilité. Cette notion de réductibilité a certains liens avec les contraintes imposées dans [CR08] pour obtenir les résultats de combinaison.

7.2.3 Ce que nous proposons

Nous établissons un résultat de combinaison pour des théories non disjointes sur des signatures typées. À cela s'ajoutent deux caractéristiques importantes. Le résultat est obtenu dans un cas où :

- trois théories E_1, E_2 et E_3 sont combinées,
- les théories combinées E_1 et E_2 spécifiées sur deux signatures $\mathcal{F}_1$ et $\mathcal{F}_2$ ne sont pas disjointes,
- E_3 est spécifiée sur la signature $(\mathcal{F}_1 \cup \mathcal{F}_2) \setminus (\mathcal{F}_1 \cap \mathcal{F}_2)$.

7.3 Théorie du couplage bilinéaire

Nous allons illustrer nos définitions spécifiques ainsi que nos lemmes par l'exemple d'une théorie équationelle représentant les opérations en jeu dans le couplage bilinéaire.

7.3.1 Description cryptographique

Cette primitive cryptographique utilise deux groupes distincts $\mathbb{G}_1$ et $\mathbb{G}_2$ et une opération de couplage e associant à deux éléments de $\mathbb{G}_1$ un élément de $\mathbb{G}_2$.

Une définition cryptographique concrète est par exemple donnée dans [BF01]. De manière générale, une opération de couplage associe les éléments d'un groupe aux éléments d'un autre groupe de la manière suivante, où nous adoptons une notation multiplicative :

$$\begin{array}{rccc} e: & \mathbb{G}_1 \times \mathbb{G}_1 & \rightarrow & \mathbb{G}_2 \\ & e(g_1^a, g_2^b) & = & e(g_1, g_2)^{ab} \end{array}$$

Dans certains protocoles, par exemple celui de Joux [Jou00] décrit plus bas, on a $g_1 = g_2$. Nous utiliserons cette supposition pour simplifier nos notations. En outre, nous utiliserons une notation multiplicative pour représenter les éléments de $\mathbb{G}_1$. Nous écrirons ainsi $exp_1(x)$ aussi bien pour g_1^x que pour g_2^x.

Exemple 6 (Protocole de Joux [Jou00]) Le protocole est spécifié pour établir une clé secrète entre trois participants. Il s'agit d'une variante à trois participants du protocole de Diffie-Hellman.

Envoyeur	Message
$1 \to (2,3)$	$g_1^{\mathcal{N}_1}$
$2 \to (1,3)$	$g_1^{\mathcal{N}_2}$
$3 \to (1,2)$	$g_1^{\mathcal{N}_3}$

La clé commune calculée est alors $e(g_1, g_1)^{\mathcal{N}_1\mathcal{N}_2\mathcal{N}_3}$. Décrivons simplement comment 1 calcule cette clé. Le calcul est similaire pour les autres participants.

$$e(g_1, g_1)^{\mathcal{N}_1\mathcal{N}_2\mathcal{N}_3} \quad = \quad e(g_1^{\mathcal{N}_2}, g_1^{\mathcal{N}_3})^{\mathcal{N}_1}$$

7.3.2 Théorie équationnelle du couplage bilinéaire

Soit $\mathcal{S}_{\mathsf{CB}}$ l'ensemble de types $\{R, G_1, G_2\}$. R est le type des termes représentant les exposants pour un générateur du groupe donné de $\mathbb{G}_i$, et G_1, respectivement G_2, sont les types des éléments des groupes $\mathbb{G}_1$, respectivement $\mathbb{G}_2$. Soit $\mathcal{F}_{\mathsf{CB}}$ l'ensemble suivant de symboles :

$+, \cdot :$	$R \times R \to R$		addition, multiplication
$-:$	$R \to R$		inverse
$0_R, 1_R :$	R		constantes
$exp_i :$	$R \to G_i$	$i \in \{1,2\}$	exponentiation
$*_i :$	$G_i \times G_i \to G_i$	$i \in \{1,2\}$	multiplications dans $\mathbb{G}_i$
$e :$	$G_1 \times G_1 \to G_2$		couplage

Nous nous permettrons d'écrire $*$ au lieu de $*_i$, le type de $*$ étant toujours clair selon le contexte. Nous utiliserons aussi l'abbréviation t^i qui représente $\underbrace{t * \ldots * t}_{i\times}$. Les propriétés de ces symboles de fonction sont définies par E_{CB} la théorie équationnelle définie comme suit :

$$\begin{array}{rclcrcl} x+y & = & y+x & & 0_R + x & = & x \\ (x+y)+z & = & x+(y+z) & & x+(-x) & = & 0_R \\ x \cdot y & = & y \cdot x & & x \cdot (y+z) & = & (x \cdot y)+(x \cdot z) \\ (x \cdot y) \cdot z & = & x \cdot (y \cdot z) & & 1_R \cdot x & = & x \end{array}$$

$$\begin{array}{rcll} exp_i(x) *_i exp_i(y) & = & exp_i(x+y) & i \in \{1,2\} \\ e(exp_1(x), exp_1(y)) & = & exp_2(x \cdot y) & \end{array}$$

Cette signature et cette théorie équationnelle représentent les opérations réalisées dans des protocoles où les messages échangés sont des éléments des groupes $\mathbb{G}_i$. Le symbole e représente l'opération de couplage. Notons aussi que les huit premières équations définissent la théorie de des anneaux commutatifs.

Exemple 7 (suite de l'exemple 6) La sécurité du protocole de Joux repose sur l'hypothèse de Diffie-Hellman Bilinéaire (BDH) qui peut être modélisée formellement en utilisant le concept d'équivalence statique comme suit :

$$\nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c), y_1 \mapsto exp_2(a \cdot b \cdot c)\}$$
$$\approx_{E_{\mathsf{CB}}}$$
$$\nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c), y_1 \mapsto exp_2(r)\}$$

Chapitre 8

Réductibilité

Sommaire

Dans ce chapitre, nous définissons une classe de théories équationnelles, pour lesquelles nous montrons qu'il est possible de ramener le problème de décision de l'équivalence statique à ce même problème, mais pour des théories plus simples.

Nous commençons par définir un concept de *valve* qui caractérise certains symboles d'une signature typée. Cela nous permet de définir la possibilité de réduire un problème de décision à un autre, comme celle de simuler certains calculs par d'autres. Nous comparons enfin la réductibilité ainsi définie avec la notion plus classique de complétude suffisante et avec les hypothèses sur les signatures nécessaires aux résultats de [CR08].

8.1 Réductibilité

8.1.1 Les valves

Intuitivement, comme cela est suggéré par le nom « valve », une valve f est un symbole dont l'application sur des termes de type r produit un terme t de type s, tel que t ne peut être un sous-terme d'un terme de type r. Une fois qu'on passe dans le type s, on ne revient pas dans le type r.

Afin de définir cette notion formellement, nous utilisons certaines notions de la théorie des graphes.

Définition 17 (Graphe de la signature) *Soit $(\mathcal{S}, \mathcal{F})$ une signature typée. Le graphe $\mathcal{G}(\mathcal{S}, \mathcal{F})$ est le graphe orienté et étiqueté (V, E) où $V = \mathcal{S}$, $E \subseteq V \times V \times \mathcal{F}$ et $(r, s, f) \in E$ ssi $\mathsf{type}(f) = s_1 \times \cdots \times s_n \to s$ et il y a un i tel que $s_i = r$.*

Rappelons qu'un *chemin* dans un graphe est une séquence d'arêtes telle que pour deux arêtes consécutives (r, s, f) et (r', s', f') on a $s = r'$.

Définition 18 (valve) *Un symbole f d'arité $r \times \cdots \times r \to s$ est une* valve *de r à s si et seulement si tout chemin de r à s dans $\mathcal{G}(\mathcal{S}, \mathcal{F})$ contient (r, s, f) et il n'y a pas de chemin de s à r.*

Exemple 8 Considérons la signature typée $(\mathcal{S}_{\mathsf{CB}}, \mathcal{F}_{\mathsf{CB}})$ introduite à la sous-section 7.3.2. $\mathcal{G}(\mathcal{S}_{\mathsf{CB}}, \mathcal{F}_{\mathsf{CB}})$ est donné dans la figure 8.1.

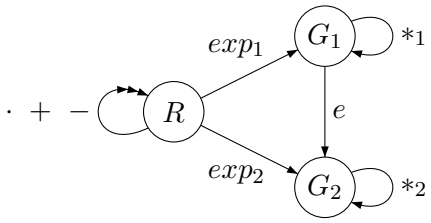

FIG. 8.1 – $\mathcal{G}(\mathcal{S}_{\mathsf{CB}}, \mathcal{F}_{\mathsf{CB}})$

Dans la signature de la figure 8.1, e est une valve de G_1 à G_2 puisque (G_1, G_2, e) est dans tous les chemins qui vont de G_1 à G_2, et puisqu'il n'y a pas de chemin de G_2 à G_1. Nous pouvons aussi remarquer que exp_1 est une valve de R à G_1. Cependant exp_2 n'est pas une valve de R à G_2 parce que la séquence $(R, G_1, exp_1), (G_1, G_2, e)$ constitue elle aussi un chemin de R à G_2.

8.1.2 Les signatures réductibles

Définissons à présent la notion de réductibilité.

Définition 19 (Réductible) *Soient r et s deux types et f une valve de r à s. Une théorie équationnelle E est* réductible *pour f si et seulement si pour tout $n \geq 0$ il existe m contextes publics $T_1[x_1, \ldots, x_n], \ldots, T_m[x_1, \ldots, x_n]$ d'arité $r \times \cdots \times r \to s$ tels que pour tous contextes publics $C_1[x_1, \ldots, x_n], \ldots, C_k[x_1, \ldots, x_n]$ d'arité $r \times \cdots \times r \to r$ il existe un contexte public $D[y_1, \ldots, y_m]$ d'arité $s \times \cdots \times s \to s$ tel que pour toute séquence de termes clos $t_1, \ldots, t_n$ de type r*

$$f(C_1, \ldots, C_k)[t_1, \ldots, t_n] =_E D[T_1, \ldots, T_m][t_1, \ldots, t_n]$$

Intuitivement, la réductibilité pour une valve f signifie qu'étant donnée une cardinalité n d'ensembles de termes de type r, on peut construire de manière uniforme un ensemble de termes tels que, quelles que soient les opérations qui ont pu être réalisées avant d'appliquer f, il y a une manière de simuler ces opérations sur le terme obtenu avec les contextes T_i. Simuler signifie que l'on peut obtenir un résultat identique au calcul initial en ne manipulant les termes de type r qu'à travers les contextes T_i. L'uniformité que nous venons d'évoquer repose sur le fait que les contextes T_i ne dépendent que de l'entier n et non pas des contextes C_i. Nous pouvons remarquer que le nombre de contextes T_i doit être fini contrairement au nombre de contextes C_i. Notons aussi que D ne peut contenir de f du fait des contraintes imposées par les types, alors que les T_i le peuvent. Nous verrons l'exemple concret de la réductibilité du couplage bilinéaire dans la section suivante.

Lemme 23 *Soient r et s deux types, f une valve de r à s et E une théorie équationnelle. Si pour tout $n \geq 0$, l'ensemble des contextes publics $C_i[x_1, \ldots, x_n]$ d'arité $r \times \cdots \times r \to r$ est fini, alors E est réductible pour f.*

Démonstration. Soit $n \geq 0$, et n_C le nombre de contextes publics $C_i[x_1, \ldots, x_n]$ d'arité $r \times \cdots \times r \to r$. Nous définissons $m = n_C^{n_C}$ contextes T d'arité $r \times \cdots \times r \to s$ indicés par un vecteur d'entier $i \leq n_C$ de longueur k comme suit :

$$T_{i_1,\ldots i_k}[x_1, \ldots, x_n] = f(C_{i_1}, \ldots, C_{i_k})[x_1, \ldots, x_n]$$

Dans ce cas, considérons k contextes publics $C_{j_1}[x_1, \ldots, x_n]$, $\ldots, C_{j_k}[x_1, \ldots, x_n]$. Nous définissons D de la manière suivante :

$$\lambda y_{1,\ldots,1}. \ldots . \lambda y_{n_C,\ldots,n_C}. y_{j_1,\ldots,j_n}$$

et de manière immédiate, nous avons pour toute séquence de termes clos $t_1, \ldots, t_n$ de type r

$$f(C_{j_1}, \ldots, C_{j_k})[t_1, \ldots, t_n] =_E D[T_{1,\ldots,1}, \ldots, T_{n_C,\ldots n_C}][t_1, \ldots, t_n]$$

□

8.2 Réductibilité du couplage bilinéaire

Nous illustrons cette notion en montrant que la théorie E_{CB} (présentée dans la sous-section 7.3.2) est réductible pour e.

Proposition 1 *E_{CB} est réductible pour e si $\mathcal{N}_{G_1} = \emptyset$.*

Démonstration. Soit n un entier. Nous définissons $m = n + \frac{n*(n+1)}{2}$ contextes publics.

$$\begin{array}{lcll} T_i & = & \lambda x_1. \ldots . \lambda x_n. e(x_i, exp_1(1_R)) & \text{pour } 1 \leq i \leq n \\ T_{ij} & = & \lambda x_1. \ldots . \lambda x_n. e(x_i, x_j) & \text{pour } 1 \leq i \leq j \leq n \end{array}$$

Tout contexte public $C_i[x_1, \ldots, x_n]$ d'arité $G_1 \times \cdots \times G_1 \to G_1$ est de la forme

$$\lambda x_1. \ldots . \lambda x_n. x_1^{e_{i1}} * \cdots * x_n^{e_{in}} * exp_1(p_i)$$

où $p_i =_{E_{\mathsf{CB}}} 1_R + \cdots + 1_R$ (l_i fois). Donc $exp_1(p_i) =_{E_{\mathsf{CB}}} exp_1(1_R)^{l_i}$.

Montrons par induction sur la taille des contextes C_i qu'il existe un contexte D tel que pour toute séquence de termes clos $t_1, \ldots, t_n$

$$e(C_1, C_2)[t_1, \ldots t_n] =_{E_{\mathsf{CB}}} D[T_1, \ldots, T_n, T_{11}, \ldots, T_{nn}][t_1, \ldots, t_n]$$

Tout contexte peut être écrit comme un produit de variables et de constantes.

Cas de base. Nous distinguons quatre cas :

1. $C_1 = \lambda x_1.\ldots.\lambda x_n.x_i$ and $C_2 = \lambda x_1.\ldots.\lambda x_n.x_j$.
   Pour toute séquence de termes $t_1, \ldots, t_n$,

$$e(C_1, C_2)[t_1, \ldots, t_n] = e(t_i, t_j) = T_{ij}[t_1, \ldots, t_n]$$

   Soit $D = \lambda y_1.\ldots.\lambda y_n.\lambda y_{11}.\ldots.\lambda y_{nn}.y_{ij}$. Nous avons alors que $e(C_i, C_j)[t_1, \ldots, t_n] =_{E_{\mathsf{CB}}} D[T_1, \ldots, T_n, T_{11}, \ldots, T_{nn}][t_1, \ldots, t_n]$.
2. $C_1 = \lambda x_1.\ldots.\lambda x_n.x_i$ et $C_2 = exp_1(1_R)^l$.
   Pour toute séquence de termes $t_1, \ldots, t_n$,

$$e(C_1, C_2)[t_1, \ldots, t_n] = e(t_i, exp_1(1_R)^l) =_{E_{\mathsf{CB}}} (T_i[t_1, \ldots, t_n])^l$$

   Soit $D = \lambda y_1.\ldots.\lambda y_n.\lambda y_{11}.\ldots.\lambda y_{nn}.y_i^l$. Nous obtenons donc $e(C_i, C_j)[t_1, \ldots, t_n] =_{E_{\mathsf{CB}}} D[T_1, \ldots, T_n, T_{11}, \ldots, T_{nn}][t_1, \ldots, t_n]$.
3. $C_1 = exp_1(1_R)^l$ et $C_2 = \lambda x_1.\ldots.\lambda x_n.x_i$.
   Puisque $C_1 * C_2 =_{E_{\mathsf{CB}}} C_2 * C_1$ ce cas est similaire au cas 2.
4. $C_1 = exp_1(1_R)^{l_1}$ et $C_2 = exp_1(1_R)^{l_2}$.
   Nous concluons de manière immédiate en posant

$$D = \lambda y_1.\ldots.\lambda y_n.\lambda y_{11}.\ldots.\lambda y_{nn}.exp_2(1_R)^{l_1 \cdot l_2}$$

Cas inductif : $C_i = C_{i1} * C_{i2}$. Soit $i = 1$. Le cas où $i = 2$ est similaire. Remarquons que tout terme de type R peut être écrit comme une somme de produits de noms de type R. Plus formellement, pour tous contextes $C_{11}[x_1, \ldots, x_n]$, $C_{12}[x_1, \ldots, x_n]$, $C_2[x_1, \ldots, x_n]$, pour toute séquence de termes $t_1, \ldots, t_n$ nous avons $C_{11}[t_1, \ldots t_n] = exp_1(p_{11})$, $C_{12}[t_1, \ldots t_n] = exp_1(p_{12})$ et $C_2[t_1, \ldots t_n] = exp_1(p_2)$, pour certains eléments de type R. Remarquons comme $\mathcal{N}_{G_1} = \emptyset$, la théorie équationnelle implique que $e(C_{11} * C_{12}, C_2) =_{E_{\mathsf{CB}}} e(C_{11}, C_2) * e(C_{12}, C_2)$.

Par induction, il y a des contextes D_1 et D_2 tels que

$$e(C_{11}, C_2)[t_1, \ldots, t_n] =_{E_{\mathsf{CB}}} D_1[T_1, \ldots, T_m][t_1, \ldots, t_n]$$

et $e(C_{12}, C_2)[t_1, \ldots, t_n] =_{E_{\mathsf{CB}}} D_2[T_1, \ldots, T_m][t_1, \ldots, t_n]$. On peut donc conlure en définissant D comme $D_1 * D_2$. □

Exemple 9 Pour $n = 2$ nous avons

$$T_1 = e(x_1, exp_1(1)) \qquad T_2 = e(x_2, exp_1(1))$$
$$T_{1,1} = e(x_1, x_1) \quad T_{1,2} = e(x_1, x_2) \quad T_{2,2} = e(x_2, x_2)$$

Soit $C_1 = \lambda x_1 \lambda x_2.x_1$ et $C_2 = \lambda x_1 \lambda x_2.x_2 * x_2 * exp_1(1 + 1)$. Nous définissons

$$D = \lambda y_1 \lambda y_2 \lambda y_{1,1} \lambda y_{1,2} \lambda y_{2,2}.y_{1,2} * y_{1,2} * y_1 * y_1$$

car $e(t_1, t_2 * t_2 * exp_1(1 + 1)) = e(t_1, t_2) * e(t_1, t_2) * e(t_1, exp_1(1)) * e(t_1, exp_1(1))$ pour tous termes *clos* t_1, t_2.

Remarque 10. Le lemme 1 nécessite que nous n'ayons pas de noms de type G_1. Nous affirmons que ce n'est pas restrictif dans le contexte des protocoles. Comme les termes de type G_1 représentent les éléments d'un groupe dont le générateur est fixé, chaque élément du groupe $\mathbb{G}_1$ peut être représenté par un terme $exp_1(t)$ où t est de type R. Si l'on veut représenter un élément de ce groupe dont nous ne saurions déterminer la puissance par rapport au générateur, on considérera un terme $exp_1(r)$ où r est un nom lié de type R.

8.3 Réductibilité et concepts existants

8.3.1 Réductibilité et complétude suffisante

On aurait pu s'attendre à ce que la réductibilité pour un symbole f soit liée au fait d'être *suffisamment complet par rapport à* f comme cela a été défini dans [Com01].

Définition 20 (suffisamment complet) *E est une théorie équationnelle* suffisamment complète *par rapport à $f \in \mathcal{F}$ si pour tout terme clos $t \in T(\mathcal{F},\mathcal{N})$, il existe un terme clos $u \in T(\mathcal{F} \setminus \{f\},\mathcal{N})$ tel que $t =_E u$.*

Cependant les deux prochains lemmes montrent que ces deux notions sont indépendantes.

Lemme 24 *La réductibilité d'une théorie équationnelle E pour un symbole f n'implique pas qu'elle soit suffisamment complète par rapport à f.*

Démonstration. Soit $\mathcal{S} = \{r, s\}$ et $\mathcal{F} = \{f\}$, avec $\mathsf{type}(f) = r \to s$, et $E = \emptyset$. Nous montrons que E est réductible pour f mais pas suffisamment complète par rapport à f.

Considérons un entier n. Comme les seuls contextes possibles de type r sont de la forme $C_i[x_1, \ldots, x_n] = \lambda x_1 \ldots \lambda x_n . x_i$, il sont en nombre fini. Par le lemme 1, E est donc réductible pour f.

Afin de montrer que E n'est pas suffisamment complète par rapport à f, nous remarquons que f est libre. Ainsi pour tout i, le terme $f(n_i)$ n'est équivalent à aucun terme ne comportant pas de f. □

Lemme 25 *La complétude suffisante de E par rapport à f n'implique pas sa réductibilité pour f.*

Démonstration. Nous exhibons une théorie équationnelle E et une valve f telles que E est suffisamment complète par rapport à f mais non réductible pour f.

Nous définissons une signature avec deux types r et s, sans noms, et les symboles de fonction et constantes définis ci-dessous :

$$\begin{aligned} 0_r &: r \\ s_r &: r \to r \\ f &: r \to s \\ 0_s &: s \\ s_s &: s \to s \end{aligned}$$

Le symbole de fonction f est une valve. Nous considérons la théorie équationnelle suivante :

$$\begin{aligned} f(s_r(x), y)) &= s_s(f(x, y)) \\ f(0_r, s_r(y)) &= f(s_r(y), y) \\ f(0_r, 0_r) &= 0_s \end{aligned}$$

Nous identifions les termes clos construits seulement avec les symboles 0_r et s_r, ainsi que ceux construits seulement avec les symboles 0_s et s_s, avec les entiers naturels. Dans cette notation nous voyons aisément que pour tout $n, m \in \mathbf{N}$:

$$f(n, m) = n + \frac{m * (m + 1)}{2}$$

En particulier, remarquons que E est suffisamment complète pour f puisqu'il n'y a pas de noms de type r. Supposons maintenant que E est réductible pour f. Choisissons $n = 1$. Remarquons qu'à cause des restrictions liées aux types, tout contexte $C[x]$ est de la forme $s_r^c(x)$ pour certaines constantes c, que l'on écrit selon notre notation comme $c+x$. Un contexte $D[y_1, \ldots, y_m]$ est de la forme $\lambda y_1, \ldots, y_m.s_s^d(y_i)$, ce que l'on écrit $d+y_i$, et tout contexte $T_i[x]$ est de la forme $s_s^{n_i}(f(s_r^{m_i}(x), s_r^{k_i}(x)))$.

Selon la notion de réductibilité, il y a un nombre fini de n_i, m_i, et k_i (pour $1 \leq i \leq m$) tels que pour tout c (nous choisissons l'identité pour le premier contexte, le second contexte est $\lambda x.s_r^c(x)$), il existe un indice i_0 et un contexte $D = \lambda y_1, \ldots, y_m.s_s^d(y_i)$ tels que pour tout t :

$$f(t, c + t) = d + n_{i_0} + f(m_{i_0} + t, k_{i_0} + t)$$

Avec l'équation ci-dessus pour f nous obtenons

$$t + \frac{(c + t) * (c + t + 1)}{2} = d + n_{i_0} + m_{i_0} + t + \frac{(k_{i_0} + t) * (k_{i_o} + t + 1)}{2}$$

ce qui se réduit en

$$\frac{c^2}{2} + ct + \frac{c}{2} = d + n_{i_0} + m_{i_0} + \frac{k_{i_0}^2}{2} + k_{i_0}t + \frac{k_{i_0}}{2}$$

D'après la définition de la réductibilité, pour tout c il existe d et i_0 tels que l'équation ci-dessus est vraie pour tout t. Choisissons une valeur c_0 distincte de tous les k_i. Nous obtenons ainsi pour un certain d et i_0

$$\forall t. \left(\frac{c_0^2}{2} + c_0 t + \frac{c_0}{2} = d + n_{i_0} + m_{i_0} + \frac{k_{i_0}^2}{2} + k_{i_0} t + \frac{k_{i_0}}{2} \right)$$

En instanciant cela avec les valeurs $t = 0$ et $t = 1$, nous obtenons respectivement

$$\begin{aligned} \frac{c_0^2}{2} + \frac{c_0}{2} &= d + n_{i_0} + m_{i_0} + \frac{k_{i_0}^2}{2} + \frac{k_{i_0}}{2} \\ \frac{c_0^2}{2} + c_0 + \frac{c_0}{2} &= d + n_{i_0} + m_{i_0} + \frac{k_{i_0}^2}{2} + k_{i_0} + \frac{k_{i_0}}{2} \end{aligned}$$

En soustrayant ces deux équations, nous obtenons

$$c_0 = k_{i_0}$$

Ceci entre en contradiction avec le fait que nous ayons choisi c_0 comme étant différent de tous les k_i. □

8.3.2 Réductibilité et combinaison hiérarchique

Cette sous-section est issue d'une remarque faite dans une relecture pour le workshop Secret09 où une version préliminaire de ce résultat [KMT09b] a été présentée.

Il y était suggéré une définition de notre notion de valve à l'aide du concept de mode défini dans [CR08] cité dans la sous-section 7.2.2. Cette redéfinition nous invite alors à comparer, autant que cela est possible, les contraintes qu'elle impose sur la signature à laquelle elle appartient, ainsi que celles de la notion de réductibilité, avec les contraintes nécessaires aux résultats de combinaison de [CR08].

Étant donné que le problème en vue duquel le résultat de combinaison est établi n'est pas le même dans nos résultats respectifs, nous ne chercherons pas à comparer les méthodes qui ont permis d'obtenir nos résultats. En revanche, nous allons comparer les notions de types et de modes. Nous montrerons ensuite comment définir la notion de valve en terme de mode. Enfin nous comparerons les contraintes que nous imposons respectivement sur nos signatures et nos théories équationelles.

Types et modes

Comme cela est indiqué dans [CR08], ces deux notions diffèrent. En effet, la notion de type sert directement à définir l'ensemble des termes. Ceci suppose, comme cela est bien noté dans [CR08] que tous les termes sont « bien typés ». La notion de mode quant à elle est une notion descriptive, en ce sens qu'elle permet de décrire les termes, et de saisir certaines de leurs parties.

La notion de valve portant sur les types, impose en particulier que les arguments de cette valve soient d'un certain type. A l'opposé, la notion de « bon modage », ne contraint pas le mode des arguments des symboles. Elle sera utilisée pour repérer certains sous-termes d'un terme. En l'occurence, il s'agira de repérer les termes dit mal-modés. Ceci permet aux auteurs de [CR08] d'exprimer et d'imposer aux systèmes de l'intrus considéré une certaine notion de localité, en particulier par le biais de l'hypothèse 1 nécessaire aux théorèmes 1 et 2.

Valves et théories hiérarchiques

Dans la remarque que nous avons évoquée plus haut, il nous était suggéré qu'une théorie comportant une valve entre deux types était un cas particulier d'une signature hiérarchisée.

Précisons le sens de cette hiérarchisation de ces signatures. Comme nous l'avons dit dans la sous-section 7.2.2, le résultat de [CR08] concerne des combinaisons de deux théories portant sur deux signatures $\mathcal{F}_0$ et $\mathcal{F}_1$. Les auteurs définissent une fonction SIG, qui associe à chaque symbole $f \in \mathcal{F}$ et à chaque variable un entier $0, 1$ ou 2 correspondant à la signature à laquelle ils appartiennent.

$$\begin{aligned} SIG &: \mathcal{F} \cup \mathcal{X} \to \{0,1,2\} \\ SIG(f) &= \begin{cases} i \text{ si } f \in \mathcal{F}_i \cup \mathcal{X}_i \text{ pour } i \in \{0,1\} \\ 2 \text{ sinon} \end{cases} \end{aligned}$$

Ils considèrent une notion de fonction de mode $M(.,.)$ telle que $M(f,i)$ est définie pour tous les symboles $f \in \mathcal{F}$ et pour tout entier i tels que si $\mathsf{arité}(f) = s_1 \times \cdots \times s_k \to s$, $i \leq k$. Ils imposent en plus que pour tout f, i, on ait $M(f,i) \in \{0,1\}$ et $M(f,i) \leq SIG(f)$. Une

position $p \cdot i$ (où p est une position et i un entier strictement positif) dans un terme t est *bien modée* par rapport à M si et seulement si $SIG(t_{|p.i}) = M(f, i)$ où f est le symbole qui apparaît en tête de $t_{|p}$.

Il nous était suggéré que le concept de valve pouvait être généralisé grâce au concept de mode. Ils nous semble que cela signifiait qu'il était possible de définir une valve f comme un symbole tel que pour tout i, $M(f, i) = 0$ et $SIG(f) = 1$ et tel qu'il existe une possibilité de moder la signature de sorte que tout terme « bien typé » soit bien modé.

Considérons en effet une signature $(\mathcal{S}, \mathcal{F})$ et une valve f d'un type r à un type s. Alors on peut partitionner $\mathcal{S}$ en $\mathcal{S}_1$, $\mathcal{S}_2$, et $\mathcal{S}_3$ tels que pour tout type $s_1 \in \mathcal{S}_1$ il existe un chemin dans $\mathcal{G}(\mathcal{S}, \mathcal{F})$ menant de s_1 à r et pour tout type $s_2 \in \mathcal{S}_2$ il existe un chemin menant de s à s_2 et $\mathcal{S}_3$ correspond à une partie non connexe de $\mathcal{G}(\mathcal{S}, \mathcal{F})$. On peut alors vérifier que si l'on attribue le mode 0 aux termes de type $s \in \mathcal{S}_1$ et 1 aux termes de type $s \in \mathcal{S}_2$ on obtient que tout terme bien typé est bien modé.

Ainsi on peut toujours considérer qu'une signature comportant une valve est une théorie hiérarchisable en ce sens que tout terme bien typé est bien modé.

Réductibilité et théories hiérarchiques

Il semble néanmoins que les utilisations respectives qui sont faites des valves et des modes diffèrent fortement. Pour cela nous allons nous centrer sur l'hypothèse 1 de [CR08] qui doit être satisfaite dans les théorèmes 1 et 2. Il semble ici que nos approches vont dans des directions opposées.

L'hypothèse 1 de [CR08] implique en effet que si un calcul utilisant des contextes de la signature $\mathcal{F}_1$ produisant des termes mal modés qui ne seraient pas préalablement des sous-termes de l'ensemble de départ, alors il est possible d'effectuer des calculs en utilisant uniquement des termes de la signature $\mathcal{F}_0$. Ainsi le principe sous-jacent est d'imposer que tout calcul qui utilise des symboles de $\mathcal{F}_1$ produisant un terme qui n'est pas « local », peut être simulé en travaillant avec des symboles de $\mathcal{F}_0$. Quant à nous, notre notion de réductibilité impose à l'inverse que modulo un nombre fini de calculs fixés (les contextes T_i dépendant de la taille de l'ensemble de départ) nous pouvons simuler dans $\mathcal{F}_1$ ce qui pouvait être calculé dans $\mathcal{F}_0$.

Ainsi alors que [CR08] impose que tout calcul non local peut être fait avant de passer d'une signature $\mathcal{F}_0$ à une signature $\mathcal{F}_1$, nous imposons la possibilité de projeter la plus grande partie des calculs de $\mathcal{F}_0$ dans $\mathcal{F}_1$.

Chapitre 9

Simplification des théories réductibles

Sommaire

Dans ce chapitre, nous présentons le résultat central de la partie II. Nous montrons que si une théorie équationnelle E est réductible pour une valve f de r à s alors il est possible de ne plus considérer le symbole f. En ajoutant d'autres conditions, ceci permettra de considérer des sous-théories de E pour décider l'équivalence statique.

Ainsi après avoir défini des notions utiles pour l'obtention de nos resultats, nous commençons par montrer que la décision de l'équivalence statique sur des cadres $\{r, s\}$-typés en présence d'une valve de r à s peut être réduite à la décision de deux équivalences statiques, l'une sur des cadres r-typés et l'autre sur des cadres s-typés (lemme 27).

Ensuite nous montrons que sous certaines conditions sur la théorie équationnelle, la décision de l'équivalence statique pour une théorie équationnelle peut être réduite à la décision de l'équivalence statique pour une théorie équationnelle qui ne comporte pas de symboles réductibles (théorème 5). On obtiendra ainsi comme corollaire la possibilité de séparer la théorie équationnelle en deux théories équationnelles plus simples.

Nous établirons enfin un critère permettant de décider certaines conditions imposées pour obtenir notre résultat de réduction, ce qui nous permettra d'établir aisément la décidabilité de la théorie du couplage bilinéaire E_{CB} (section 7.3).

9.1 Quelques définitions et remarques supplémentaires

9.1.1 Réductions et restrictions des cadres

Définition 21 (réduction) *Soit E une théorie équationnelle réductible pour f, où f est une valve de r à s, et soit $\phi = \nu\tilde{n}.\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ un cadre de type $\{r\}$. La* réduction *de ϕ est définie comme $\overline{\phi} = \nu\tilde{n}.\{y_1 \mapsto T_1[t_1, \ldots, t_n], \ldots, y_m \mapsto T_m[t_1, \ldots, t_n]\}$ où les T_i sont les contextes dont l'existence est imposée par la définition 19.*

Remarquons que $\overline{\phi}$ est $\{s\}$-typée. Avant de donner un exemple illustrant la construction de $\overline{\phi}$, nous définissons la notation suivante.

Définition 22 (s-restriction) *Soit $\phi = \nu\tilde{n}.\sigma$ un cadre $\{s_1, \ldots, s_n\}$-typé. La* s_i-restriction *de ϕ, notée $\phi_{|s_i}$ est le cadre $\nu\tilde{n}.\sigma_{|s_i}$ où $\sigma_{|s_i}$ est la substitution σ restreinte aux variables de type s_i.*

Exemple 11 Soit ϕ_{BDH} la G_1-restriction des cadres présentés dans l'exemple 6 : $\phi_{BDH} = \nu a, b, c, r.\{x_1 \mapsto exp_1(a), x_2 \mapsto exp_1(b), x_3 \mapsto exp_1(c)\}$. En utilisant l'ensemble de termes T_i et T_{ij} défini dans la preuve de la proposition 1, on obtient

$$\begin{aligned}\overline{\phi}_{BDH} = \nu a, b, c, r.\{ \quad & y_1 \mapsto e(exp_1(a), exp_1(1)), \quad y_{12} \mapsto e(exp_1(a), exp_1(b)), \\ & y_2 \mapsto e(exp_1(b), exp_1(1)), \quad y_{13} \mapsto e(exp_1(a), exp_1(c)), \\ & y_3 \mapsto e(exp_1(c), exp_1(1)), \quad y_{23} \mapsto e(exp_1(b), exp_1(c)) \quad \}\end{aligned}$$

9.1.2 Théories équationnelles suffisantes

Grâce à la définition suivante, nous identifions une condition suffisante pour se débarasser du symbole f pour la décision de l'équivalence statique entre des cadres qui ne comportent pas ce symbole. Dans la section 9.4, nous définirons un critère syntaxique suffisant pour obtenir une telle théorie.

Définition 23 (Théorie équationnelle suffisante) *Soit $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ une signature typée et E une théorie équationnelle. Une théorie équationnelle E' est* suffisante pour E sans f *si et seulement si pour tous termes $u, v \in T(\mathcal{F}, \mathcal{N})$, $u =_E v$ si et seulement si $u =_{E'} v$ et E' ne comporte pas de symbole f.*

Dans la section 9.4 nous décrivons des critères suffisants simples permettant de savoir qu'une théorie est suffisante dans un symbole f.

9.1.3 Remarques concernant la signature considérée dans les démonstrations

Remarque 12. Dans la suite, lorsque nous considérerons des signatures $(\mathcal{S}, \mathcal{F})$ où $f \in \mathcal{F}$ est une valve de r à s, nous effectuons les preuves en considérant que les seuls types accessibles à partir de r dans $\mathcal{G}(\mathcal{S}, \mathcal{F})$ sont r et s, et d'autre part que les seuls types accessibles à partir de s est s lui-même.

Ceci peut apparaître comme une restriction forte bien qu'elle soit suffisante pour notre exemple. Le but est de simplifier les preuves. Si nous avions voulu être réellement généraux, deux solutions auraient été possibles. Soit il aurait fallu considérer une notion de valve non pas entre deux type r et s mais entre deux ensembles de types. Soit il aurait fallu prendre en compte la possibilité d'avoir des distingueurs de cadres $\{r,s\}$-typés qui ne soient ni de type r ni de type s. Cela aurait permis d'obtenir des preuves plus générales, mais aussi beaucoup plus complexes, tout cela sans que nous ayons d'exemples qui nécessitent un cadre aussi général.

9.2 Se ramener à des cadres à un seul type

Nous prouvons un lemme technique qui sera utilisé pour transférer les tests d'un cadre testé sur sa réduction.

Lemme 26 *Soit $(\mathcal{S},\mathcal{F})$ une signature telle que $f \in \mathcal{F}$ est une valve de r à s, et E une théorie équationnelle qui est réductible pour f. Pour tout entier n et pour tout contexte public M de type s, il existe un contexte public M' tel que pour tout cadre $\{r,s\}$-typé ϕ de taille n, $M\phi =_E M'\overline{\phi_{|r}}\phi_{|s}$.*

Démonstration. Montrons cela par induction sur la hauteur de M.

Si la hauteur de M est 0, alors M est soit une constante soit une variable. En posant $M' = M$, si M est une constante, alors $M\phi =_E M'\overline{\phi_{|r}}\phi_{|s}$ est trivialement vrai et si $M = y \in \mathcal{X}$ alors $y\overline{\phi_{|r}}\phi_{|s} = y\phi$ puisque le type de y est s.

Si la hauteur de M est non-nulle alors le symbole à la tête de M peut être une valve f de r à s, ou un symbole $f' \neq f$.

Si $M = f(C_1,\ldots,C_k)[x_1,\ldots,x_{n'}]$ alors $\mathsf{type}(x_i) = r$, et donc $M\phi = M\phi_{|r}$ où $\phi_{|r} = \{x_1 \mapsto t_1,\ldots,x_{n'} \mapsto t_{n'}\}$. Comme E est réductible pour f, nous pouvons définir $\overline{\phi_{|r}}$ comme $\{y_1 \mapsto T_1[t_1,\ldots,t_{n'}],\ldots,y_m \mapsto T_m[t_1,\ldots,t_{n'}]\}$. D'après la définition 19, il existe un contexte public $D[y_1,\ldots,y_m]$ tel que

$$f(C_1,\ldots,C_k)[t_1,\ldots,t_{n'}] =_E D[T_1,\ldots,T_m][t_1,\ldots,t_{n'}]$$

En considérant que $M' = D$ nous avons $M\phi_{|r} =_E M'\overline{\phi_{|r}}$, et donc $M\phi =_E M'\overline{\phi_{|r}}\phi_{|s}$.

Si $M = f'(C_1,\ldots,C_{k'})[x_1,\ldots,x_n,y_1,\ldots,y_m]$ avec $f' \neq f$ alors $\mathsf{type}(C_i) = s$ car f est une valve. Par induction il existe des contextes publics $M_1,\ldots,M_{k'}$ tels que pour tous les cadres $\{r,s\}$-typés de taille n, $C_{i'}\phi =_E M_{i'}\overline{\phi_{|r}}\phi_{|s}$. Nous définissons $M' = f'(M_1,\ldots,M_{k'})$, et nous obtenons $M\phi =_E M'\overline{\phi_{|r}}\phi_{|s}$. □

Le lemme suivant permet de ramener la décision de l'équivalence statique de cadres $\{r,s\}$-typés à la décision de l'équivalence sur des cadres r-typés et s-typés.

Lemme 27 *Pour tous cadres $\{r,s\}$-typés ϕ_1 et ϕ_2 construits sur $(\mathcal{S},\mathcal{F})$, et pour une valve f de r à s, si E est une théorie équationnelle réductible pour f alors $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$.*

Démonstration. Prouvons tour à tour les deux implications de l'équivalence. Étant donné un cadre ϕ nous notons $\overrightarrow{x_\phi}$ les variables de $\phi_{|r}$, $\overrightarrow{y_\phi}$ les variables de $\overline{\phi_{|r}}$ et $\overrightarrow{z_\phi}$ les variables de $\phi_{|s}$. Comme dans la plupart des cas, le cadre ϕ dont il est question est clair selon le contexte, nous écrivons simplement $\overrightarrow{x}$, $\overrightarrow{y}$ et $\overrightarrow{z}$. Nous étendons cette notation aux vecteurs de termes.

(⇒) Si $\phi_1 \approx_E \phi_2$, alors $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$.

Supposons d'abord que $\phi_{1|r} \not\approx_E \phi_{2|r}$. Alors sans perte de généralité il y a deux termes $M[\overrightarrow{x}]$ et $N[\overrightarrow{x}]$ tels que $(M =_E N)\phi_{1|r}$ et $(M \neq_E N)\phi_{2|r}$. Puisque $dom(\phi_{i|r}) \subseteq dom(\phi_i)$ et $dom(\phi_{1|r}) = dom(\phi_{2|r})$, nous avons que $(M =_E N)\phi_1$ et $(M \neq_E N)\phi_2$, et donc $\phi_1 \not\approx_E \phi_2$.

Supposons maintenant que $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_E \overline{\phi_{2|r}}\phi_{2|s}$. Il existe alors deux termes $M[\overrightarrow{y}, \overrightarrow{z}]$ et $N[\overrightarrow{y}, \overrightarrow{z}]$ tels que sans perte de généralité $(M =_E N)\overline{\phi_{1|r}}\phi_{1|s}$ et $(M \neq_E N)\overline{\phi_{2|r}}\phi_{2|s}$. Puisque $\overline{\phi_{i|r}}$ est une réduction de $\phi_{i|r}$ pour $i \in \{1, 2\}$ et $dom(\phi_1) = dom(\phi_2)$, il existe un ensemble de termes $T_j[\overrightarrow{x}]$ tels que $M[\overrightarrow{y}, \overrightarrow{z}]\overline{\phi_{i|r}}\phi_{i|s} =_E M[\overrightarrow{T[\overrightarrow{x}]}, \overrightarrow{z}]\phi_i$ et $N[\overrightarrow{y}, \overrightarrow{z}]\overline{\phi_{i|r}}\phi_{i|s} =_E N[\overrightarrow{T[\overrightarrow{x}]}, \overrightarrow{z}]\phi_i$. Nous concluons alors que $M[\overrightarrow{T[\overrightarrow{x}]}, \overrightarrow{z}]$ et $N[\overrightarrow{T[\overrightarrow{x}]}, \overrightarrow{z}]$ distinguent ϕ_1 et ϕ_2.

Alors en supposant que la conjonction de ces deux assertions $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$ est fausse, l'une d'entre elles est fausse. Comme la négation de chacune d'elle implique $\phi_1 \not\approx_E \phi_2$, nous obtenons que $\phi_1 \approx_E \phi_2$ implique que $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$.

(⇐) Si $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$, alors $\phi_1 \approx_E \phi_2$. Supposons que $\phi_1 \not\approx_E \phi_2$. Il existe donc deux termes $M[\overrightarrow{x}, \overrightarrow{y}]$ et $N[\overrightarrow{x}, \overrightarrow{y}]$ tels que sans perte de généralité $(M =_E N)\phi_1$ et $(M \neq_E N)\phi_2$. Comme $(M =_E N)\phi_1$, M et N ont le même type. Nous considérons deux cas $\mathsf{type}(M) = r$ ou $\mathsf{type}(M) = s$.

Si $\mathsf{type}(M) = r$, puisque f est une valve, la seule possibilité de changement de type va de r à s. Ainsi toutes les variables de M sont de type r. Donc M et N distinguent $\phi_{1|r}$ et $\phi_{2|r}$.

Si $\mathsf{type}(M) = s$, comme $\mathrm{dom}(\phi_{1|r}) = \mathrm{dom}(\phi_{2|r})$ d'après le lemme 26, il existe des termes M' et N' tels que $M\phi_i =_E M'\overline{\phi_{i|r}}\phi_{i|s}$ et $N\phi_i =_E M'\overline{\phi_{i|r}}\phi_{i|s}$, pour $i \leq 2$. Alors si M et N distinguent ϕ_1 et ϕ_2 alors sans perte de généralité $(M =_E N)\phi_1$ et $(M \neq_E N)\phi_2$. Donc $(M =_E N)\overline{\phi_{1|r}}\phi_{1|s}$ et $(M \neq_E N)\overline{\phi_{2|r}}\phi_{2|s}$. □

9.3 Réduction des théories réductibles pour f et suffisantes sans f

L'objet de cette section est de prouver le théorème 5, établissant la possibilité de considérer des théories plus simples (sous-section 9.3.1) en vue de décider le problème de l'équivalence statique. Ce théorème établi, le corollaire 3 permettra d'effectuer un pas de plus dans la simplification des théories (sous-section 9.3.2).

9.3.1 Retrait de la valve

Commençons donc par énoncer le théorème central.

Théorème 5 *Soit E une théorie équationnelle sur la signature typée $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ telle que*
- *f est une valve de r à s,*
- *E est une théorie équationnelle réductible pour f,*
- *E est suffisamment complète par rapport à f.*

S'il existe une théorie équationnelle E' suffisante pour E sans f alors pour tous cadres $\{r, s\}$-typés ϕ_1 et ϕ_2, $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|r} \approx_{E'} \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$.

La preuve du théorème 5 repose sur le lemme 28.

Lemme 28 *Soient ϕ_1 et ϕ_2 deux cadres $\{r\}$-typés, E une théorie équationnelle, et f une valve de r à un type distinct s, qui est libre dans E. Si pour tous termes M et N de type r $(M =_E N)\phi_1$ si et seulement si $(M =_E N)\phi_2$, alors pour tous termes M et N de type s, $(M =_E N)\phi_1$ si et seulement si $(M =_E N)\phi_2$.*

Afin de prouver ce lemme, nous utilisons la fonction cut et le lemme 29 introduit dans [BCK09]. Rappelons-les préalablement. Étant donné $u = f(u_1, \ldots, u_n)$ où f est libre et un nom a du même type que u, la *fonction de découpage* $\mathsf{cut}_{u,a}$ est définie récursivement comme suit : $\mathsf{cut}_{u,a}(u) = u$ si u est une variable ou un nom, et

$$\mathsf{cut}_{u,a}(g(t_1, \ldots, t_k)) = \begin{cases} a \text{ si } g = f,\ k = n \text{ et } \forall 1 \leq i \leq n, u_i =_E t_i \\ g(\mathsf{cut}_{u,a}(t_1), \ldots, \mathsf{cut}_{u,a}(t_k)) \text{ sinon.} \end{cases}$$

L'effet de la fonction $\mathsf{cut}_{u,a}(t)$ est donc de substituer certains sous-termes de t (mais pas tous) égaux à u modulo E par a. Dans le cas où f est une valve, l'effet de $\mathsf{cut}_{u,a}(t)$ est de substituer tout sous-terme de t égal à u modulo E par a.

Lemme 29 ([BCK09]) *Soit E une théorie équationnelle où f est libre. Soit $u = f(u_1, \ldots, u_n)$ un terme tel que f est un symbole libre. Soit a un nom du même type que u. Pour tous termes M et N,*

$$\text{si } M =_E N \text{ alors } \mathsf{cut}_{u,a}(M) =_E \mathsf{cut}_{u,a}(N).$$

Lemme 30 *Soit ϕ_1 et ϕ_2 deux cadres $\{r\}$-typés, f une valve de r à s et E une théorie équationnelle où f est libre. Si pour toute paire de termes M', N' de type r $M'\phi_1 =_E N'\phi_1$ ssi $M'\phi_2 =_E N'\phi_2$, alors pour toute paire de termes M, N de type s comportant un f en tête $M\phi_1 =_E N\phi_1$ ssi $M\phi_2 =_E N\phi_2$.*

Démonstration. Soit $M = f(C_1^M, \ldots, C_k^M)$ and $N = f(C_1^N, \ldots, C_k^N)$. Puisque f est libre $M\phi_1 =_E N\phi_1$ ssi pour tout $i \leq k$ $C_i^M\phi_1 =_E C_i^N\phi_1$. Comme pour toute paire de termes M', N' de type r $(M' =_E N')\phi_1$ ssi $(M' =_E N')\phi_2$, en particulier pour tout $i \leq k$ $C_i^M\phi_1 =_E C_i^N\phi_1$ ssi pour tout $i \leq k$ $C_i^M\phi_2 =_E C_i^N\phi_2$. À nouveau, puisque f est libre pour tout $i \leq k$ $C_i^M\phi_2 =_E C_i^N\phi_2$ ssi $M\phi_2 =_E N\phi_2$. □

Nous pouvons enfin passer à la preuve du lemme 28.

Démonstration (du lemme 28). Nous allons définir deux fonctions de remplacement σ_1 (resp. σ_2) dont le domaine est constitué des paires (α, p) où α identifie M (ou N) et p est une position dans $M\phi_1$ (ou $N\phi_1$) (resp. $M\phi_2, N\phi_2$) telle que f apparaît en tête de $M\phi_1|_p$ (ou $N\phi_1|_p$). Le co-domaine de σ_1 (resp. σ_2) est un ensemble de noms n'apparaissant ni dans ϕ_1 ni dans ϕ_2. Nous allons commencer par montrer que $M\phi_1\sigma_1 =_E M\phi_2\sigma_2$ et $N\phi_1\sigma_1 =_E N\phi_2\sigma_2$ sous la condition que pour toute paire de termes M, N de type r, $(M =_E N)\phi_1$ ssi $(M =_E N)\phi_2$. Ceci implique trivialement que $M\phi_1\sigma_1 =_E N\phi_1\sigma_1$ est équivalent à $M\phi_2\sigma_2 =_E N\phi_2\sigma_2$ (équivalence (1) de la figure 9.1). Par la suite, nous montrons que $M\phi_i =_E N\phi_i$ ssi $M\phi_i\sigma_i =_E N\phi_i\sigma_i$ pour $i \in \{1, 2\}$ en se servant de l'hypothèse selon laquelle pour tous termes M et N de type r, $(M =_E N)\phi_1$ ssi $(M =_E N)\phi_2$ (équivalences (2) et (3) de la figure 9.1). Pour cela nous montrons d'abord que σ_1 (resp. σ_2) correspond à une séquence d'applications de la fonction de découpage, et nous utilisons le lemme 29 et le fait que σ_1 et σ_2 sont bijectives. La conjonction de ces assertions implique que pour toute paire de termes M, N de type s, $(M =_E N)\phi_1$ ssi $(M =_E N)\phi_2$.

$$\begin{array}{ccc} M\phi_1 =_E N\phi_1 & \underset{(2)}{\Longleftrightarrow} & M\phi_1\sigma_1 =_E N\phi_1\sigma_1 \\ & & \Big\Updownarrow (1) \\ M\phi_2 =_E N\phi_2 & \underset{(3)}{\Longleftrightarrow} & M\phi_2\sigma_2 =_E N\phi_2\sigma_2 \end{array}$$

FIG. 9.1 – Schéma de raisonnement

Définissons σ_1 et σ_2. Nous considérons un ensemble de noms de la forme $a_{\alpha,\beta}$ où $\alpha \in \{\mathcal{M}, \mathcal{N}\}$ et $\beta \in Pos(M) \cup Pos(N)$. Intuitivement les fonctions σ_i remplacent un même terme $f(C_1, \ldots, C_k)$ (modulo $=_E$) par un même nom $a_{\alpha,\beta}$. Ici α et β identifient la première occurence de $f(C_1, \ldots, C_k)$ durant un parcours en largeur de $M\phi_i$ suivi d'un parcours en largeur de $N\phi_i$. Plus formellement nous définissons un ordre total $<$ sur le produit cartésien $\{\mathcal{M}, \mathcal{N}\} \times (Pos(M) \cup Pos(N))$ comme suit. Soient p_1 et p_2 deux positions dans $Pos(M) \cup Pos(N)$. $(\alpha_1, p_1) < (\alpha_2, p_2)$ ssi :

- $\alpha_1 = \mathcal{M}$ et $\alpha_2 = \mathcal{N}$,
- ou $|p_1| < |p_2|$,
- ou $|p_1| = |p_2|$ et $p_1 <_{lex} p_2$ où $<_{lex}$ est l'ordre lexicographique.

Remarquons que M (resp. N) est de la forme $C_{\mathcal{M}}[f(C^1_{\mathcal{M}1}, \ldots, C^1_{\mathcal{M}k}), \ldots, f(C^m_{\mathcal{M}1}, \ldots, C^m_{\mathcal{M}k})]$ (resp. $C_{\mathcal{N}}[f(C^1_{\mathcal{N}1}, \ldots, C^1_{\mathcal{N}k}), \ldots, f(C^l_{\mathcal{N}1}, \ldots, C^l_{\mathcal{N}k})]$) où toutes les variables de M (resp. N) sont dans $f(C^i_{\mathcal{M}1}, \ldots, C^i_{\mathcal{M}k})$ (resp. $f(C^i_{\mathcal{N}1}, \ldots, C^i_{\mathcal{N}k})$), et les contextes $C_{\mathcal{M}}$, $C_{\mathcal{N}}$, $C^i_{\mathcal{M}j}$, $C^i_{\mathcal{N}j}$ ne contiennent pas f (autrement dit, toutes les occurrences de f sont explicites dans cette écriture de M et N). Nous avons donc $M\phi_i = C_{\mathcal{M}}[f(C^1_{\mathcal{M}1}, \ldots, C^1_{\mathcal{M}k})\phi_i, \ldots, f(C^m_{\mathcal{M}1}, \ldots, C^m_{\mathcal{M}k})\phi_i]$ et $N\phi_i = C_{\mathcal{N}}[f(C^1_{\mathcal{N}1}, \ldots, C^1_{\mathcal{N}k})\phi_i, \ldots, f(C^l_{\mathcal{N}1}, \ldots, C^l_{\mathcal{N}k})\phi_i]$. Nous écrivons $Pos(M\phi_i) \cup Pos(N\phi_i)_{|f}$ pour l'ensemble des positions de $M\phi_i$ et $N\phi_i$ qui comportent un f en tête. Nous pouvons donc faire les remarques suivantes :

Remarque 13. $Pos(M\phi_1)_{|f} = Pos(M\phi_2)_{|f} = Pos(M)_{|f}$ et $Pos(N\phi_1)_{|f} = Pos(N\phi_2)_{|f} = Pos(N)_{|f}$.

Remarque 14. Pour toute position $p \in Pos(M)_{|f}$ il y a un i tel que $M\phi_1|_p = f(C^i_{\mathcal{M}1}, \ldots, C^i_{\mathcal{M}k})\phi_1$ et $M\phi_2|_p = f(C^i_{\mathcal{M}1}, \ldots, C^i_{\mathcal{M}k})\phi_2$, et pour toute position $p \in Pos(N)_{|f}$ il y a i tel que $N\phi_1|_p = f(C^i_{\mathcal{N}1}, \ldots, C^i_{\mathcal{N}k})\phi_1$ et $N\phi_2|_p = f(C^i_{\mathcal{N}1}, \ldots, C^i_{\mathcal{M}k})\phi_2$.

Définissons σ_i sur le domaine $\mathcal{M} \times Pos(M)_{|f} \cup \mathcal{N} \times Pos(N)_{|f}$ (remarque 13) et notons τ la fonction associant M à $\mathcal{M}$ et N à $\mathcal{N}$. Soit $(\alpha, p) \in \mathrm{dom}(\sigma_i)$, soit (α', p') l'élément minimal tel que $\tau(\alpha')\phi_i|_{p'} =_E \tau(\alpha)\phi_i|_p$, alors $\sigma_i((\alpha, p)) = a_{\alpha',p'}$.

Pour n'importe quel terme M, on écrit $M\phi_i\sigma_i$ le terme $M\phi_i[\sigma_i((\mathcal{M}, p))]_p$ pour $p \in Pos(M)_{|f}$.

Montrons que pour tout préfixe p d'éléments de $Pos(\tau(\alpha))_{|f}$, $\tau(\alpha)_{|p}\phi_1\sigma_1 = \tau(\alpha)_{|p}\phi_2\sigma_2$. Ainsi dans le cas particulier où $p = \Lambda$ nous obtiendrons de manière immédiate que $\tau(\alpha)\phi_1\sigma_1 = \tau(\alpha)\phi_2\sigma_2$. Montrons cela par induction descendante sur p.

Cas de base : p est maximal. De manière immédiate, $p \in Pos(\tau(\alpha))_{|f}$, donc $C_{\alpha|p}$ est une variable. Soit (α', p') la paire minimale telle que $\tau(\alpha')\phi_{1|p'} =_E \tau(\alpha)\phi_{1|p}$. Alors $\tau(\alpha)_{|p}\phi_1\sigma_1 = a_{\alpha',p'}$.

Par la remarque 14, il y a des i et i' tels que $\tau(\alpha)\phi_{1|p} = f(C^i_{\alpha 1}, ..., C^i_{\alpha k})\phi_1$, $\tau(\alpha)\phi_{2|p} = f(C^i_{\alpha 1}, ..., C^i_{\alpha k})\phi_2$, $\tau(\alpha')\phi_{1|p'} = f(C^{i'}_{\alpha' 1}, ..., C^{i'}_{\alpha' k})\phi_1$ et $\tau(\alpha')\phi_{2|p'} = f(C^{i'}_{\alpha' 1}, ..., C^{i'}_{\alpha' k})\phi_2$.

D'après le lemme 30, comme $\tau(\alpha')\phi_{1|p'} =_E \tau(\alpha)\phi_{1|p}$, on a $\tau(\alpha')\phi_{2|p'} =_E \tau(\alpha)\phi_{2|p}$.

S'il y avait une paire $(\alpha'', p'') < (\alpha', p')$ telle que $\tau(\alpha'')\phi_{2|p''} =_E \tau(\alpha)\phi_{2|p}$, par un raisonnement analogue, on aurait $\tau(\alpha'')\phi_{1|p''} =_E \tau(\alpha)\phi_{1|p}$ donc cela entrerait en contradiction avec le fait que (α', p') est la paire minimale telle que $\tau(\alpha')\phi_{1|p'} =_E \tau(\alpha)\phi_{1|p}$. Donc $\tau(\alpha)_{|p}\phi_2\sigma_2 = a_{\alpha',p'}$. Donc $\tau(\alpha)_{|p}\phi_1\sigma_1 = \tau(\alpha)_{|p}\phi_2\sigma_2$.

Cas inductif. On suppose la propriété vraie pour les positions $p_1, \ldots, p_n \in Pos(\tau(\alpha))_{|f}$ dont p est un préfixe, et on la prouve pour p. Ceci est immediat.

Nous passons désormais à la seconde partie de la démonstration correspondant aux équivalences (2) et (3) de la figure 9.1. Pour cela nous montrons d'abord que σ_1 (resp. σ_2) correspond à une séquence d'applications de la fonction de découpage.

Montrons ainsi par induction sur le nombre de noms dans le co-domaine de σ_i, qu'il y a une séquence de fonction $\mathsf{cut}_{u_1,a_1}, \ldots, \mathsf{cut}_{u_n.a_n}$ telle que $M\phi_i\sigma_i = \mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(M\phi_i))\ldots)$ et $N\phi_i\sigma_i = \mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(N\phi_i))\ldots)$.

Base case. Il n'y a pas de noms dans le co-domaine de σ_i. Donc σ_i est vide et $M\phi_i\sigma_i = M\phi_i$ et $N\phi_i\sigma_i = N\phi_i$. La séquence dont nous cherchons à montrer l'existence est la séquence vide.

Inductive case. Considérons le nom $a_{\alpha,p}$ tel que pour tout nom $a_{\alpha',p'}$ dans le co-domaine de σ_i, $(\alpha, p) > (\alpha', p')$. On note $\sigma_{i\setminus a_{\alpha,p}}$ la fonction σ_i dont nous avons retiré les éléments de la forme $(c, a_{\alpha,p})$ et $\sigma_i|_{a_{\alpha,p}}$ l'ensemble des éléments de σ_i de la forme $(c, a_{\alpha,p})$.

Par induction, il existe une séquence $\mathsf{cut}_{u_1,a_1}, \ldots, \mathsf{cut}_{u_n.a_n}$ telle que $M\phi_i\sigma_{i\setminus a_{\alpha,p}} = \mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(M\phi_i))\ldots)$ et $N\phi_i\sigma_{i\setminus a_{\alpha,p}} = \mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(N\phi_i))\ldots)$. Par définition de σ_i si l'image de plusieurs paires $(\alpha_1, p_1), \ldots, (\alpha_n, p_n)$ est $a_{\alpha,p}$, cela signifie que $\tau(\alpha_1)\phi_i|_{p_1} = \cdots = \tau(\alpha_n)\phi_i|_{p_n} = \tau(\alpha)\phi_i|_p$ et nous avons aussi que $\tau(\alpha)\phi_i|_p$ comporte un f en tête. Donc appliquer $\sigma_i|_{\alpha,p}$ revient à appliquer $\mathsf{cut}_{\tau(\alpha)\phi_i|_p,a_{\alpha,p}}$. Comme $M\phi_i\sigma_i = M\phi_i\sigma_{i\setminus a_{\alpha,p}}\sigma_i|_{\alpha,p}$ et $M\phi\sigma_{i\setminus a_{\alpha,p}} = \mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(M\phi_i))\ldots)$, nous avons
$M\phi_i\sigma_i = \mathsf{cut}_{\tau(\alpha)\phi_i|_p,a_{\alpha,p}}(\mathsf{cut}_{u_n,a_n}(\ldots(\mathsf{cut}_{u_1.a_1}(M\phi_i))\ldots))$. On fait un raisonnement analogue pour N.

En itérant l'application du lemme 29 nous obtenons que si $M\phi_i =_E N\phi_i$ alors $M\phi_i\sigma_i =_E N\phi_i\sigma_i$. D'autre part, comme σ_i est bijective nous avons que si $M\phi_i\sigma_i =_E N\phi_i\sigma_i$ alors $M\phi_i =_E N\phi_i$. Il suffit de remplacer les noms frais que nous avons introduits par l'application de σ_i par leurs pré-images par σ_i. Donc $M\phi_i =_E N\phi_i$ ssi $M\phi_i\sigma_i =_E N\phi_i\sigma_i$ ce qui correspond aux équivalences (2) et (3) de la figure 9.1. □

On peut enfin passer à la preuve du théorème 5.

Démonstration (du théorème 5). Nous supposons que $\phi_1 \approx_E \phi_2$. Par le lemme 27, nous obtenons que $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|r} \approx_E \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}$.

Nous montrerons que

$$\begin{array}{rcl} \phi_{1|r} \approx_E \phi_{2|r}(p) & \wedge & \overline{\phi_{1|r}}\phi_{1|s} \approx_E \overline{\phi_{2|r}}\phi_{2|s}(q) \\ & \Leftrightarrow & \\ \phi_{1|r} \approx_{E'} \phi_{2|r}(p_1) & \wedge & \overline{\phi_{1|r}}\phi_{1|s} \approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}(q_1) \end{array}$$

Nous prouverons les trois assertions suivantes séparément :

$$(1)\neg q \Leftrightarrow \neg q_1 \qquad (2)\neg p \Rightarrow \neg p_1 \vee \neg q_1 \qquad (3)\neg p_1 \Rightarrow \neg p$$

La conjonction de ces assertions implique que $(p \wedge q) \Leftrightarrow (p_1 \wedge q_1)$.

(1) $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_E \overline{\phi_{2|r}}\phi_{2|s}$ si et seulement si $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$

Puisque E est suffisamment complète par rapport à $\{f\}$, nous pouvons supposer que $\overline{\phi_{1|r}}\phi_{1|s}$ et $\overline{\phi_{2|r}}\phi_{2|s}$ ne comportent pas f.

Si $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_E \overline{\phi_{2|r}}\phi_{2|s}$ il existe deux termes M et N distinguant $\overline{\phi_{1|r}}\phi_{1|s}$ et $\overline{\phi_{2|r}}\phi_{2|s}$. Puisque f est une valve, que les $\overline{\phi_{i|r}}\phi_{i|s}$ sont de type s et que E est suffisamment complète par rapport à $\{f\}$, nous pouvons considérer que M et N ne comportent pas f. Donc $M\overline{\phi_{i|r}}\phi_{i|s}$ et $N\overline{\phi_{i|r}}\phi_{i|s}$ ne comportent pas non plus f. Comme E' est suffisante pour E sans f nous obtenons que $M\overline{\phi_{i|r}}\phi_{i|s} =_E N\overline{\phi_{i|r}}\phi_{i|s}$ si et seulement si $M\overline{\phi_{i|r}}\phi_{i|s} =_{E'} N\overline{\phi_{i|r}}\phi_{i|s}$. Donc $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$.

Si $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$ il existe deux termes M et N distinguant $\overline{\phi_{1|r}}\phi_{1|s}$ et $\overline{\phi_{2|r}}\phi_{2|s}$ pour E'. Puisque f est une valve, que les $\overline{\phi_{i|r}}\phi_{i|s}$ sont de type s et que E est suffisamment complète par rapport à $\{f\}$, nous pouvons considérer que M et N ne comportent pas f. Donc $M\overline{\phi_{i|r}}\phi_{i|s}$ et $N\overline{\phi_{i|r}}\phi_{i|s}$ ne comportent pas non plus f. Comme E' est suffisante pour E sans f nous obtenons que $M\overline{\phi_{i|r}}\phi_{i|s} =_{E'} N\overline{\phi_{i|r}}\phi_{i|s}$ si et seulement si $M\overline{\phi_{i|r}}\phi_{i|s} =_E N\overline{\phi_{i|r}}\phi_{i|s}$. Donc $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_E \overline{\phi_{2|r}}\phi_{2|s}$.

(2) Si $\phi_{1|r} \not\approx_E \phi_{2|r}$ alors $\phi_{1|r} \not\approx_{E'} \phi_{2|r}$ ou $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$

Soient M et N deux termes distinguant $\phi_{1|r}$ et $\phi_{2|r}$.

Si M est de type r, comme f est une valve, M, N, $\phi_{1|r}$ et $\phi_{2|r}$ ne comportent pas de f. Donc $M\phi_{i|r}$ et $N\phi_{i|r}$ ne comportent pas de f. Comme E' est suffisante pour E sans f, nous obtenons que $M\phi_{i|r} =_E N\phi_{i|r}$ si et seulement si $M\phi_{i|r} =_{E'} N\phi_{i|r}$. Donc $\phi_{1|r} \not\approx_{E'} \phi_{2|r}$.

Si M est de type s, par le lemme 26 il existe des termes M' et N' tels que $M\phi_{i|r} =_E M'\overline{\phi_{i|r}}$ et $N\phi_{i|r} =_E N'\overline{\phi_{i|r}}$. Comme f est une valve et E est suffisamment complète par rapport à $\{f\}$, M' et N' ne comporte pas de symbole f. Par la complétude suffisante de E par rapport à $\{f\}$, nous pouvons considérer des cadres $\overline{\phi_{1|r}}$ et $\overline{\phi_{2|r}}$ qui ne comporte pas f. $M'\overline{\phi_{i|r}}$ et $N'\overline{\phi_{i|r}}$ ne comportent donc pas f non plus. Puisque E' est suffisante sans f nous avons que $M'\overline{\phi_{i|r}} =_E N'\overline{\phi_{i|r}}$ si et seulement si $M'\overline{\phi_{i|r}} =_{E'} N'\overline{\phi_{i|r}}$. Donc $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$.

Comme nous sommes soit dans le cas où M est de type r soit dans le cas où M est de type s, nous avons bien $\overline{\phi_{1|r}} \not\approx_{E'} \overline{\phi_{2|r}}$ ou $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$.

(3) si $\phi_{1|r} \not\approx_{E'} \phi_{2|r}$ alors $\phi_{1|r} \not\approx_E \phi_{2|r}$

Puisque $\phi_{1|r} \not\approx_{E'} \phi_{2|r}$ il existe deux termes M et N distinguant $\phi_{1|r}$ et $\phi_{2|r}$.

S'il n'y a pas de termes M et N de type r distinguant $\phi_{1|r}$ et $\phi_{2|r}$, d'après le lemme 28 il n'y a pas de termes de type s distinguant $\phi_{1|r}$ et $\phi_{2|r}$. Donc si $\phi_{1|r} \not\approx_{E'} \phi_{2|r}$ alors il y a des termes M et N distinguant $\phi_{1|r}$ et $\phi_{2|r}$ de type r.

Si M est de type r, comme f est une valve, M, N, $\phi_{1|r}$ et $\phi_{2|r}$ ne comportent aucun f. Donc $M\phi_{i|r}$ et $N\phi_{i|r}$ ne comportent pas de f. Comme E' est suffisante sans f, nous avons donc $M\phi_{i|r} =_{E'} N\phi_{i|r}$ si et seulement si $M\phi_{i|r} =_E N\phi_{i|r}$. Donc $\phi_{1|r} \not\approx_E \phi_{2|r}$. □

Remarque 15. Remarquons que d'après le point (1) de la démonstration du théorème 5, dans le cas où $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_E \overline{\phi_{2|r}}\phi_{2|s}$, et où on fait les même hypothèses sur les theories équationnelles et sur les signatures que dans ce théorème, $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$, grâce à des distingueurs M et N ne comportant pas de f.

9.3.2 Séparation par type

Nous notons E^{-r} la théorie équationnelle de E dont nous avons retiré les équations de type r. Il nous est donc possible d'énoncer le corollaire suivant.

Corollaire 3 *Soit E une théorie équationnelle sur la signature typée $(\mathcal{S}, \mathcal{F} \cup \{f\})$ telle que*
- *f est une valve,*
- *E est réductible pour f,*
- *E est suffisamment complète par rapport à $\{f\}$.*

S'il existe une théorie équationnelle E' suffisante pour E sans f alors pour tout cadre $\{r,s\}$-typé ϕ_1 et ϕ_2, nous avons que $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|r} \approx_{E'^{-s}} \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_{E'^{-r}} \overline{\phi_{2|r}}\phi_{2|s}$.

Démonstration. D'après le théorème 5, nous avons $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|r} \approx_{E'} \phi_{2|r}$ et $\overline{\phi_{1|r}}\phi_{1|s} \approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$.

D'après le lemme 28, si pour toute paire de termes M, N de type r $(M =_E N)\phi_1$ si et seulement si $(M =_E N)\phi_2$, alors pour toute paire de termes M, N de type s, $(M =_E N)\phi_1$ si et seulement si $(M =_E N)\phi_2$. Il est donc suffisant de considérer des termes de type r pour décider l'équivalence statique entre $\phi_{1|r}$ et $\phi_{2|r}$. Puisque f est une valve, pour tout terme M de type r, M ne comporte aucun sous-terme de type s. Nous pouvons donc ne considérer que E'^{-s} pour décider l'équivalence statique entre $\phi_{1|r}$ et $\phi_{2|r}$.

Montrons que $\overline{\phi_{1|r}}\phi_{1|s} \approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$ si et seulement si $\overline{\phi_{1|r}}\phi_{1|s} \approx_{E'^{-r}} \overline{\phi_{2|r}}\phi_{2|s}$.

$\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'} \overline{\phi_{2|r}}\phi_{2|s}$ si et seulement si il y a deux termes M et N qui distinguent $\overline{\phi_{1|r}}\phi_{1|s}$ et $\overline{\phi_{2|r}}\phi_{2|s}$. Par la remarque 15, M et N ne comportent pas de symbole f. Comme E est suffisamment complète par rapport à f, nous pouvons supposer que les cadres $\overline{\phi_{1|r}}\phi_{1|s}$ et $\overline{\phi_{2|r}}\phi_{2|s}$ ne comportent pas f. Donc $M\overline{\phi_{i|r}}\phi_{i|s}$ et $N\overline{\phi_{i|r}}\phi_{i|s}$ ne comportent pas de f non plus. Comme f est une valve, $M\overline{\phi_{i|r}}\phi_{i|s}$ et $N\overline{\phi_{i|r}}\phi_{i|s}$ ne comportent pas de sous-termes de type r. Nous avons donc que $M\overline{\phi_{i|r}}\phi_{i|s} =_{E'} N\overline{\phi_{i|r}}\phi_{i|s}$ si et seulement si $M\overline{\phi_{i|r}}\phi_{i|s} =_{E'^{-r}} N\overline{\phi_{i|r}}\phi_{i|s}$. Donc $\overline{\phi_{1|r}}\phi_{1|s} \not\approx_{E'^{-r}} \overline{\phi_{2|r}}\phi_{2|s}$. □

9.4 Critère pour les théories équationnelles suffisantes

Dans cette section, nous présentons une première tentative pour trouver des critères suffisants pour appliquer le théorème 5. Le critère que nous exposons ici trouve une application dans la section suivante.

Définition 24 (décomposition) *Une paire* $(\mathcal{R}, E')$ *est une* décomposition *d'une théorie équationnelle* E *si et seulement si*

- E' *est une théorie équationnelle,*
- $\mathcal{R}$ *est un système de réécriture convergent modulo* E'*,*
- *pour toute paire de termes* u *et* v $u =_E v$ *si et seulement si* $u \downarrow_{\mathcal{R}/E'} = v \downarrow_{\mathcal{R}/E'}$*.*

Définition 25 (définir exclusivement) *Soit* $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ *une signature typée. Un système de réécriture* $\mathcal{R}$ définit exclusivement f *si tout terme en forme normale modulo* $\mathcal{R}/E'$ *est dans* $T(\mathcal{F}, \mathcal{N})$ *et si pour toute règle de réécriture* $l \to r \in \mathcal{R}$*,* f *apparaît dans* l*.*

Lemme 31 *Soit* $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ *une signature. Si une théorie* E *sur* $(\mathcal{S}, \mathcal{F} \uplus \{f\})$ *a une décomposition* $(\mathcal{R}, E')$ *et si* $\mathcal{R}$ *définit exclusivement* f *alors* E' *est suffisante pour* E *sans* f*.*

Démonstration. Soient u et v deux termes qui ne comportent pas f. Comme $\mathcal{R}$ définit exclusivement f et comme u et v ne comportent pas de symbole f, aucune règle de $\mathcal{R}$ ne peut être appliquée. Donc $u =_E v$ si et seulement si $u =_{E'} v$. □

9.5 La théorie du couplage bilinéaire est décidable

Nous expliquons ici comment l'exemple de la sous-section 7.3.2 satisfait ce critère et nous permet d'obtenir la décidabilité de la théorie E_{CB}.

Exemple 16 (suite de l'exemple de la sous-section 7.3.2) Nous définissons $\mathcal{R}_{\mathsf{CB}}$ comme étant le système de réécriture obtenu en orientant la loi $e(exp_1(x), exp_1(y)) = exp_2(x \cdot y)$ de gauche à droite, et E'_{CB} la théorie équationnelle E_{CB} sans cette loi. Nous remarquons que $(\mathcal{R}, E'_{\mathsf{CB}})$ est une décomposition de E_{CB} et il est aisé de remarquer que $\mathcal{R}$ définit exclusivement e s'il n'y a pas de noms dans G_1. De même, il est aisé de remarquer que E_{CB} est suffisamment complète pour e.

Corollaire 4 *Si les ensembles de noms de type* G_1 *et* G_2 *sont vides, l'équivalence statique pour* E_{CB} *est décidable pour des cadres* $\{G_1, G_2\}$*-typés.*

Démonstration. Puisque $\mathcal{R}_{\mathsf{CB}}$ définit exclusivement e, d'après le lemme 31, nous obtenons que E'_{CB} est suffisant pour E_{CB} sans e. D'après la proposition 1, nous avons que E_{CB} est réductible pour f. Finalement, comme les ensembles des noms de type G_1 et G_2 sont vides, E_{CB} est suffisamment complète pour e. Donc d'après le corollaire 3, pour deux cadres ϕ_1 et ϕ_2, $\phi_1 \approx_E \phi_2$ si et seulement si $\phi_{1|G_1} \approx_{E'^{-G_2}_{\mathsf{CB}}} \phi_{2|G_1}$ et $\overline{\phi_{1|G_1}}\phi_{1|G_2} \approx_{E'^{-G_1}_{\mathsf{CB}}} \overline{\phi_{2|G_1}}\phi_{2|G_2}$.

Comme $E'^{-G_2}_{\mathsf{CB}}$ et $E'^{-G_1}_{\mathsf{CB}}$ correspondent toutes deux à la théorie modélisant habituellement la théorie de Diffie-Hellman, dont on sait qu'elle est décidable [KM07] pour les cadres dont les noms sont uniquement de type R, nous obtenons que l'équivalence statique est décidable pour E_{CB} sur les cadres $\{G_1, G_2\}$-typés. □

Remarque 17. Le corollaire 4 nécessite que nous n'ayons ni noms de type G_1 ni noms de type G_2. Nous affirmons que ce n'est pas restrictif, en vertu de la remarque 10 que l'on peut étendre au type G_2.

Remarques conclusives

Dans cette partie, nous avons défini les notions de valves et de réductibilité qui nous ont permis de simplifier les théories équationnelles pour la décision de l'équivalence statique. Cela constitue une première approche en vue d'un critère générique.

Le résultat obtenu ici s'applique au cas du couplage bilinéaire. Nous pensons que ce résultat pourrait s'appliquer à d'autres situations où de nombreuses structures algébriques sont utilisées dans le modèle pour le même opérateur cryptographique. En bref, nous continuons de rechercher dans les directions suivantes :

(1) Nous essayons d'identifier des critères de réductibilité dont il serait plus facile de décider s'ils sont satisfaits ou non. En effet, même pour notre exemple qui est plutôt simple, prouver la décidabilité est assez technique. Donc nous essayons de trouver soit un critère syntaxique sur la théorie équationnelle soit des propriétés plus classiques comme une forme restreinte de la complétude suffisante.

(2) Nous avons analysé le cas où il y a seulement une valve réductible. Il semble possible d'étendre la réductibilité au cas où de nombreuses valves appartiennent à la théorie équationnelle. Cela nécessiterait néanmoins de définir un ordre de priorité sur les réductions des différentes valves.

(3) Nous essayons aussi d'élargir la notion de valve. Dans la définition que nous avons proposée ici, une valve est définie d'un type donné à un autre. Cependant, des cas où des valves prennent en argument des termes de types différents peuvent être considérés. Nous pensons qu'une telle notion pourrait donner lieu à une notion plus large de réductibilité que celle que nous avons analysée ici. Il semble que nous ayons besoin de conditions sur les liens entre les arguments de telles valves.

Conclusion

Résumé

Dans cette thèse, nous avons contribué à l'élargissement des possibilités de vérification automatique des protocoles cryptographiques dans deux directions distinctes.

Dans la première partie, nous nous sommes concentrés sur un type particulier de protocoles appelés protocoles de groupe. Leur principale particularité est d'être spécifié pour un nombre non borné de participants. Nous y sommes parvenus dans l'analyse des protocoles pour le cas d'un intrus passif tout en représentant certaines propriétés des primitives cryptographiques et en analysant la propriété de secret définie comme non-atteignabilité d'un terme. Pour réaliser cela, nous avons identifié des conditions sur les spécifications de certains protocoles nous permettant de ne considérer qu'un intrus plus faible qu'attendu contre ces protocoles. Nous avons utilisé les automates d'arbres à mémoire, avec visibilité et contraintes structurelles pour représenter une approximation des exécutions de certains protocoles de groupe auxquels nous nous intéressons. Grâce à certaines propriétés de ces automates, nous avons prouvé la décidabilité d'une approximation du problème du secret. Ce premier résultat a été publié dans [KMT08].

Dans la partie II, nous nous sommes intéressés à l'établissement de conditions permettant de combiner des théories équationnelles tout en préservant la décidabilité de l'équivalence statique. Nous avons établi de cette manière un résultat de décidabilité pour une théorie particulière représentant le couplage bilinéaire. Ce second résultat a été publié dans [KMT09a].

Perspectives

Concernant la partie I, de nombreuses améliorations restent à apporter. D'abord, une chose qui semble intéressante et que nous aurions pu effectuer est l'implémentation de la classe d'automates d'arbres que nous avons utilisée. Cela nous aurait permis de faire des essais « réels » de notre méthode une fois une approximation définie. Cela nous aurait aussi évité de faire des preuves extrêmement longues et sans grand intérêt uniquement pour montrer qu'une approximation peut bien être représentée par un automate d'arbres. Une telle implémentation pourrait aussi potentiellement être utilisée dans n'importe quel domaine qui a recours à de tels automates qui sont plus expressifs que les automates d'arbres habituels. Concernant les approximations que nous avons évoquées, il aurait aussi fallu automatiser le passage d'une spécification de protocoles à une approximation par automate d'arbres, et vérifier en chemin que cette spécification satisfait bien les critères permettant d'obtenir la réductibilité d'un intrus utilisant les primitives cryptographiques exponentiation et ou exclusif en plus des possibilités de chiffrer, déchiffer, apparier, projeter et hacher à un intrus plus faible. Ces travaux, implémentation des automates et génération automatique d'approximations représentables par automate, iraient de pair et mèneraient à un outil complet qui prendrait en entrée une spécification et l'analyserait.

Concernant la partie II, comme nous l'avons annoncé dans nos remarques conclusives, nous pensons que l'on peut rendre la notion de réductibilité plus facile à utiliser en trouvant des critères suffisants qui l'impliquent. Il semble aussi possible d'envisager des notions de valve plus larges et surtout d'envisager le cas de signatures qui comportent plusieurs valves. Nous pouvons aussi espérer que la recherche dans le domaine de cryptographie conduise à l'élaboration de primitives similaires au couplage bilinéaire, en ce sens qu'elle permettrait d'associer des éléments d'une structure à une autre, sans qu'il soit possible d'établir de correspondance réciproque. Cela donnerait un nouveau champ d'applications à nos résultats.

Questions ouvertes

Une restriction, que nous considérons dans notre problème du couplage bilinéaire dans la partie II, est le fait que les éléments de l'anneau, qui sont des exposants des générateurs des groupes dont nous considérons les éléments, n'apparaissent pas eux-mêmes dans les cadres. Une question naturelle qui survient alors, est celle de savoir ce qu'il en est de la décidabilité du problème de l'équivalence statique lorsque nous considérons ces éléments dans les cadres. Ce qui débouche sur la question de la décidabilité de l'équivalence statique en présence de la théorie de l'anneau.

Ainsi pendant plusieurs mois, nous nous sommes intéressés à la question de la décidabilité du problème de l'équivalence statique en présence de la théorie équationnelle de l'anneau et à la question de la décidabilité du problème de la déduction en présence de cette même théorie.

Nous pensions au départ qu'en présence de cette théorie équationnelle, ces problèmes étaient indécidables. Il semblait en effet y avoir une proximité entre ces problèmes et le dixième problème de Hilbert. Ainsi nous avons espéré réduire ce dixième problème de Hilbert au problème de décision de l'équivalence statique ou au problème de la dérivabilité. Cette tentative a échoué.

Nous nous sommes finalement concentrés sur la question de la décidabilité de la déduction en présence de la théorie équationnelle de l'anneau, ce qui nous a conduit à nous interroger sur la décidabilité du problème suivant :

Données : n polynômes $P_0, \ldots, P_n$ de $\mathbb{Z}[X_1, \ldots, X_n]$
Question : Existe-t-il un polynôme Q tel que $P_0 = Q[P_1, \ldots, P_n]$?

C'est sur cette question que nous avons le plus particulièrement poussé nos recherches. Le but était de borner la taille de Q. À partir des monômes $c * x_1^{e_1} * \cdots * x_n^{e_n}$ apparaissant dans les polynômes $P_0, \ldots, P_n$, nous avons établi une borne sur le degré à partir de l'espace vectoriel engendré par les vecteurs $(e_1, \ldots, e_n)$.

Il y a aussi une certaine proximité entre le problème évoqué ci-dessus et celui de savoir si P_0 existe dans l'idéal engendré par $P_1, ..., P_n$. En d'autres termes, il s'agit de se demander s'il existe des polynômes $Q_1, \ldots, Q_n$ dans $\mathbb{Z}[X_1, \ldots, X_n]$ tels que $P_0 = P_1Q_1 + \cdots + P_nQ_n$. Mais il semble que les problèmes soient distincts.

À suivre ...

Bibliographie

[ABB+02] A. Armando, D. Basin, M. Bouallagui, Y. Chevalier, L. Compagna, S. Mödersheim, M. Rusinowitch, M. Turuani, L. Viganò et L. Vigneron : The AVISS security protocol analysis tool. *In Proceedings of the 14th International Conference on Computer Aided Verification (CAV '02)*, volume 2404 de *LNCS*, pages 349–353. Springer, 2002.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò et L. 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)*, volume 3576 de *LNCS*, pages 281–285. Springer, 2005.

[ABF08] M. Abadi, B. Blanchet et C. Fournet : Verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[AC06] M. Abadi et V. Cortier : Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1):2–32, 2006.

[ACD07] M. Arnaud, V. Cortier et S. Delaune : Combining algorithms for deciding knowledge in security protocols. *In Proceedings of the 6th International Symposium on Frontiers of Combining Systems (FroCoS'07)*, volume 4720 de *LNAI*, pages 103–117. Springer, 2007.

[AF01] M. Abadi et C. Fournet : Mobile values, new names, and secure communication. *In Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM, 2001.

[AG99] N. Asokan et P. Ginzboorg : Key agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 1999.

[AM04] R. Alur et P. Madhusudan : Visibly pushdown languages. *In Proceedings of the thirty-sixth annual ACM Symposium on Theory of Computing (STOC'04)*, pages 202–211. ACM, 2004.

[AR00] M. Abadi et P. Rogaway : Reconciling two views of cryptography (the computational soundness of formal encryption). *In Proceedings of the International Conference IFIP on Theoretical Computer Science, Exploring New Frontiers of Theoretical Informatics (IFIP TCS'00)*, volume 1872 de *LNCS*, pages 3–22. Springer, 2000.

[AS04] S. Androutsellis-Theotokis et D. Spinellis : A survey of peer-to-peer content distribution technologies. *ACM Computer Surveys*, 36(4):335–371, 2004.

[Ava] Automated Validation of Trust and Security of Service-oriented Architectures (AVANTSSAR). `http://www.avantssar.eu/`.

[Avi] Automated Validation of Internet Security Protocols and Applications (AVISPA). `http://avispa-project.org/`.

[Bau05] M. Baudet : Deciding security of protocols against off-line guessing attacks. *In Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05)*, pages 16–25. ACM, 2005.

[BBC09] M. Berrima, N. Ben Rajeb et V. Cortier : Deciding knowledge in security protocols under some e-voting theories. Research Report RR-6903, INRIA, 2009.

[BCEP04] E. Bresson, O. Chevassut, A. Essiari et D. Pointcheval : Mutual authentication and group key agreement for low-power mobile devices. *Computer Communications*, 27(17):1730–1737, 2004.

[BCK09] M. Baudet, V. Cortier et S. Kremer : Computationally sound implementations of equational theories against passive adversaries. *Information and Computation*, 207(4):496–520, 2009.

[BF01] D. Boneh et M. Franklin : Identity-based encryption from the Weil pairing. *In Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'01)*, volume 2139 de *LNCS*, pages 213–229. Springer, 2001.

[Blaa] B. Blanchet : Cryptoverif. `http://www.cryptoverif.ens.fr/`.

[Blab] B. Blanchet : Proverif. `http://www.proverif.ens.fr/`.

[Bla01] B. Blanchet : An efficient cryptographic protocol verifier based on Prolog rules. *In 14th IEEE Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society, 2001.

[Bla04] B. Blanchet : Automatic proof of strong secrecy for security protocols. *In Proceedings of the 25th IEEE Symposium on Security and Privacy (SSP'04)*, pages 86–100. IEEE Computer Society, 2004.

[Bla06] B. Blanchet : A computationally sound mechanized prover for secu- rity protocols. *In Proceedings of the 27th IEEE Symposium on Security and Privacy (SSP'06)*, pages 140–154. IEEE Computer Society, 2006.

[BN98] F. Baader et T. Nipkow : *Term rewriting and all that*. Cambridge University Press, 1998.

[CC05] H. Comon-Lundh et V. Cortier : Tree automata with one memory set constraints and cryptographic protocols. *Theoretical Computer Science*, 331(1): 143–214, 2005.

[CD07] V. Cortier et S. Delaune : Deciding knowledge in security protocols for monoidal equational theories. *In Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'07)*, volume 4790 de *LNAI*, pages 196–210. Springer, 2007.

[CDE04] R. Corin, J. Doumen et S. Etalle : Analysing password protocol security against off-line dictionary attacks. *In Proceedings of the 2nd International*

Workshop on Security Issues with Petri Nets and other Computational Models (WISP'04), volume 121 de *ENTCS*, pages 47–63. Elsevier, 2004.

[CDG+07] H. Comon-Lundh, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison et M. Tommasi : Tree automata techniques and applications. Diaponible : `http://www.grappa.univ-lille3.fr/tata`, 2007.

[CDK09] Ş. Ciobâcă, S. Delaune et S. Kremer : Computing knowledge in security protocols under convergent equational theories. *In Proceedings of the 22nd International Conference on Automated Deduction (CADE'09)*, LNAI, pages 355–370. Springer, 2009.

[CDL06] V. Cortier, S. Delaune et P. Lafourcade : A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[Chr09] N. Chridi : *Contributions à la vérification automatique de protocoles de groupes*. Thèse de doctorat, Université Henri Poincaré - Nancy, 2009. `http://tel.archives-ouvertes.fr/tel-00417290/en/`.

[CJ97] J. Clark et J. Jacob : A survey of authentication protocol literature, 1997. `http://www-users.cs.york.ac.uk/~jac/papers/drareviewps.ps`.

[CJP08] H. Comon-Lundh, F. Jacquemard et N. Perrin : Visibly tree automata with memory and constraints. *Logical Methods in Computer Science*, 4(2:8):1–36, 2008.

[CMMU00] E. Contejean, C. Marché, B. Monate et X. Urbain : *The CiME rewrite tool*, 2000. `http://cime.lri.fr`.

[CMSS03] R. Chadha, J. Mitchell, A. Scedrov et V. Shmatikov : Contract signing, optimism, and advantage. *In Proceedings of the 14th International Conference on Concurrency Theory (CONCUR'03)*, volume 2761 de *LNCS*, pages 366–382. Springer, 2003.

[Com01] H. Common : Inductionless induction. *In Handbook of Automated Reasoning*, volume 1, chapitre 14. Elsevier, 2001.

[CR08] Y. Chevalier et M. Rusinowitch : Hierarchical combination of intruder theories. *Information and Computation*, 206(2-4):352–377, 2008.

[Cre06] C. Cremers : *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[CRZ07] V. Cortier, M. Rusinowitch et E. Zălinescu : Relating two standard notions of secrecy. *Logical Methods in Computer Science*, 3(3):1–29, 2007.

[CS03] H. Comon-Lundh et V. Shmatikov : Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. *In Proceedings of the 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, volume 171, pages 271–280. IEEE Computer Society, 2003.

[CTR09] N. Chridi, M. Turuani et M. Rusinowitch : Decidable analysis for a class of cryptographic group protocols with unbounded lists. *In Proceedings of the 22st IEEE Computer Security Foundations Symposium (CSF'09)*, pages 277–289. IEEE Computer Society, 2009.

[Der87] N. Dershowitz : Termination of rewriting. *Journal of Symbolic Computation*, 3(1-2):69–116, 1987.

[DJ90] N. Dershowitz et J.-P. Jouannaud : Rewrite systems. *In Handbook of Theoretical Computer Science*, volume B, chapitre 6. Elsevier, 1990.

[DLMS99] N. Durgin, P. Lincoln, J. Mitchell et A. Scedrov : Undecidability of bounded security protocols. *In Proceeding of the Workshop on Formal Methods and Security Protocols (FMSP'99)*, 1999.

[DY83] D. Dolev et A. Yao : On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[EMM06] S. Escobar, C. Meadows et J. Meseguer : A rewriting-based inference system for the NRL Protocol analyzer and its meta-logical properties. *Theoretical Computer Science*, 367(1):162–202, 2006.

[Fev] Fédération du e-commerce et de la vente à distance. `http://www.fevad.com/`.

[Gou00] J. Goubault-Larrecq : A method for automatic cryptographic protocol verification (extended abstract). *In Parallel and Distributed Processing Symposium (IPDPS'00)*, volume 1800 de *LNCS*, pages 977–984. Springer, 2000.

[Gre91a] S. Greenberg : An annotated bibliography of computer-supported cooperative work. *ACM SIGCHI Bulletin*, 23(3):29–62, 1991.

[Gre91b] S. Greenberg : Computer-supported cooperative work and groupware : an introduction to the special issues. *International Journal of Man-Machine Studies*, 34(2):133–141, 1991.

[GTV03] T. Genet, Y.-M. Tang-Talpin et V. Viet Triem Tong : Verification of copy-protection cryptographic protocol using approximations of term rewriting systems. *In Proceedings of the Workshop on Issues in the Theory of Security (WITS'03)*, 2003.

[Ina98] `http://www.ina.fr/economie-et-societe/vie-sociale/video/CAC98051240/cadeaux-sur-le-web-achats-de-noel-sur-internet.fr.html`, 1998.

[Ina99] `http://www.ina.fr/economie-et-societe/vie-sociale/video/CAB99012806/internet-commerce-electronique.fr.html`, 1999.

[Ina07] `http://www.ina.fr/economie-et-societe/vie-sociale/video/CAC98051240/cadeaux-sur-le-web-achats-de-noel-sur-internet.fr.html`, 2007.

[JLT99] T. Jensen, D. Le Métayer et T. Thorn : Verification of control flow based security policies. *In Proceedings of the IEEE Symposium on Security and Privacy*, pages 89–103, 1999.

[Jou00] A. Joux : A one round protocol for tripartite Diffie-Hellman. *In Proceedings of the 4th International Symposium on Algorithmic Number Theory (ANTS-IV)*, volume 1838 de *LNCS*, pages 385–394. Springer, 2000.

[KKW06] D. Kähler, R. Küsters et Th. Wilke : A Dolev-Yao-based definition of abuse-free protocols. *In Proceedings of the 33rd International Colloqium on Automata, Languages, and Programming (ICALP'06)*, volume 4052 de *LNCS*, pages 95–106. Springer, 2006.

[KM07] S. Kremer et L. Mazaré : Adaptive soundness of static equivalence. *In Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 de *LNCS*, pages 610–625. Springer, 2007.

[KMT08] S. Kremer, A. Mercier et R. Treinen : Proving group protocols secure against eavesdroppers. *In Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, volume 5195 de *LNAI*, pages 116–131. Springer, 2008.

[KMT09a] S. Kremer, A. Mercier et R. Treinen : Reducing equational theories for the decision of static equivalence. *In Proceedings of the 13th Asian Computing Science Conference (ASIAN'09)*, LNCS. Springer, 2009. To appear.

[KMT09b] S. Kremer, A. Mercier et R. Treinen : Reducing equational theories for the decision of static equivalence (preliminary version). *In Preliminary Proceedings of the 4th International Workshop on Security and Rewriting Techniques (SecReT'09)*, 2009.

[KR02] S. Kremer et J.-F. Raskin : Game analysis of abuse-free contract signing. *In Proceedings of the 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 206–220. IEEE Computer Society, 2002.

[KT07] R. Küsters et T. Truderung : On the automatic analysis of recursive security protocols with XOR. *In Proceedings of the 24th Symposium on Theoretical Aspects of Computer Science (STACS'07)*, volume 4393 de *LNCS*. Springer, 2007.

[Low97] G. Lowe : A hierarchy of authentication specifications. *In Proceedings of the 10th IEEE Workshop on Computer Security Foundations (CSFW'97)*, page 31. IEEE Computer Society, 1997.

[Man06] M. Manulis : Security-focused survey on group key exchange protocols. Research report, Horst Görtz Institute for IT-Security Ruhr University Bochum, Germany, 2006. `http://www.manulis.eu/papers/TR0603-GKEPS.pdf`.

[MGK03] O. Markowitch, D. Gollmann et S. Kremer : On fairness in exchange protocols. *In Revised Papers of the 5th International Conference on Information Security and Cryptology (ICISC'02)*, volume 2587 de *LNCS*, pages 451–464. Springer, 2003.

[Mil99] R. Milner : *Communicating and mobile author systems : the pi-calculus*. Cambridge University Press, 1999.

[Min09] `http://www.interieur.gouv.fr/sections/a_votre_service/votre_securite/internet/faq-securite#a22`, 2009.

[MS07] R. Monroy et G. Steel : Faulty group protocols. `http://homepages.inf.ed.ac.uk/gsteel/group-protocol-corpus/`, 2007.

[MvOV96] A. Menezes, P. van Oorschot et S. Vanstone : *Handbook of applied cryptography*. CRC Press, 1996.

[PQ03] O. Pereira et J.-J. Quisquater : Some attacks upon authenticated group key agreement protocols. *Journal of Computer Security*, 11(4):555–580, 2003.

[PQ06] O. Pereira et J.-J. Quisquater : On the impossibility of building secure cliques-type authenticated group key agreement protocols. *Journal of Computer Security*, 14(2):197–246, 2006.

[RT01] M. Rusinowitch et M. Turuani : Protocol insecurity with finite number of sessions is np-complete. *In Proceedings of the 14th IEEE Workshop on Computer Security Foundations (CSFW'01)*, pages 174–190. IEEE Computer Society, 2001.

[SB05] G. Steel et A. Bundy : Attacking group multicast key management protocols using Coral. *In Proceedings of the Workshop on Automated Reasoning for Security Protocol Analysis (ARSPA'04)*, volume 125 de *ENTCS*, pages 125–144. Elsevier Science Publishers, 2005.

[SB06] G. Steel et A. Bundy : Attacking group protocols by refuting incorrect inductive conjectures. *Journal of Automated Reasoning*, 36(1-2):149–176, 2006.

[Sch94] B. Schneier : *Applied cryptography, protocols, algorithms, and source code in C*. John Wiley & Sons, 1994.

[Sch98] S. Schneider : Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9):741–758, 1998.

[Spo] Security protocol open repository. `http://www.lsv.ens-cachan.fr/Software/spore/`.

[Ste98] J. Stern : *La science du secret*. Odile Jacob, 1998.

[STW96] M. Steiner, G. Tsudik et M. Waidner : Diffie-Hellman key distribution extended to group communication. *In ACM Conference on Computer and Communications Security*, pages 31–37. ACM, 1996.

[Tho01] Thomson : Smartright technical white paper v1.0. Rapport technique, Thomson, 2001. `http://www.smartright.org`.

[Tru05] T. Truderung : Selecting theories and recursive protocols. *In Proceedings of the 16th International Conference on Concurrency Theory (CONCUR'05)*, volume 3653 de *LNCS*, pages 217–232. Springer, 2005.

[TvS02] A. Tanenbaum et M. van Steen : *Distributed systems : principles and paradigms*. Prentice Hall, 2002.