A. Boisseau

Signatures électroniques de contrats

Research Report LSV–02–4, Apr. 2002

Laboratoire Spécification et Vérification

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

http://www.lsv.ens-cachan.fr/Publis/
Research Report LSV-02-4, Lab. Spécification et Vérification, CNRS & ENS de Cachan, France, Apr. 2002

Signatures électroniques de contrats

Alexandre Boisseau
Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan, France
boisseau@lsv.ens-cachan.fr

Résumé

Nous considérons ici des protocoles des signatures électroniques de contrats. Nous proposons un formalisme combinant une approche dans le style Dolev & Yao et des systèmes de transitions alternés. Nous prenons également en compte dans ce modèle différents niveaux d'honnêteté possibles pour les participants. Le modèle est décrit de manière logique au moyen de clauses de Horn ce qui nous permet également de proposer une représentation symbolique des ensembles d'états (ainsi qu'une méthode de décision du vide pour de telles représentations). Les propriétés attendues du protocole se formalisent naturellement en utilisant la logique du temps alterné.

Abstract

We are considering here electronic contract signing protocols. Our formalism combines a Dolev & Yao approach together with alternating transition systems and, moreover, we take into account several honesty level for participants. Our model is described using a logical formalism with Horn clauses and that allows us tu propose a symbolic representation for state sets (together with an emptiness decision procedure for these representations). The properties of the protocol are specified using alternating time temporal logic.

On s'intéresse ici à une classe particulière de protocoles cryptographiques qui sont les protocoles de signatures électroniques de contrats [2, 5]. Les principales différences avec les protocoles plus classiques (authentification par exemple) sont :

- des propriétés attendues différentes, qui mettent l'accent sur les relations entre les différents participants du protocole autant que sur leurs relations avec le monde extérieur,
- des primitives cryptographiques différentes (dont l'intérêt est de permettre de rétablir une certaine symétrie dans l'échange des signatures). Cette symétrie existe dans les échanges de signatures classiques (dans le monde réel) mais est perdue dès que l'on essaie de décrire un protocole, par essence asynchrone, pour réaliser électroniquement ces signatures. Or la présence d'une asymétrie dans le protocole a généralement pour conséquence un déséquilibre du protocole en faveur de l'un des participants).

Des méthodes formelles ont déjà été appliquées à de tels protocoles, on peut citer notamment [8, 9, 3, 6]. Dans chaque cas, le protocole est modélisé par un système de transitions (de manière directe, ou bien en utilisant le formalisme *multiset rewriting*). L'étape de vérification est effectuée soit en utilisant un *model-checker* classique (dans le cas d'un système fini), soit en faisant les preuves à la main. Les hypothèses sous lesquelles la vérification est menée sont de deux sortes :

- les niveaux de corruption des participants (qu'ils soient honnêtes, fortement malhonnêtes (*ie* coopérant avec l'intrus) ou encore dans un état intermédiaire, dit faiblement malhonnêtes (plusieurs définitions sont possibles)),
- les niveaux de fiabilité des différents canaux de communication (là encore, on peut distinguer deux extrêmes et des cas intermédiaires).

Notons que ces hypothèses sont traitées de la manière suivante : tout d'abord on décide quel est le niveau de corruption de chaque participant et le niveau de fiabilité de chaque canal de communication puis on construit le modèle correspondant à cette distribution et on fait ensuite la vérification. De plus les propriétés sont des propriétés de traces (en vue d'une analyse avec Murϕ ou FDR) ou des propriétés de l'arbre des exécutions.

Les améliorations que l'on propose sont les suivantes :

- proposer un modèle dans lequel chaque participant et chaque canal de comunications peut agir suivant n'importe quel niveau de corruption,
- coder les hypothèses sur les niveaux de corruption à l'intérieur même des propriétés à vérifier (et par la même occasion, proposer un traitement plus modulaire des niveaux de corruption et du modèle considéré en général),
- utiliser la logique du temps alterné (ATL,[1]) pour modéliser les propriétés, cette logique permet en effet d'exprimer de manière très naturelle les propriétés typiques attendues des protocoles de signatures de contrat (en particulier, dire qu'un participant possède une stratégie pour obtenir quelquechose). On se base pour ceci sur [7],
- utiliser un formalisme logique pour décrire le modèle, ce qui donne également des pistes pour représenter symboliquement les ensembles d'états ainsi que pour manipuler de telles représentations.

Les deux premières sections ont pour but de décrire (informellement) un exemple de protocole de signatures électroniques de contrats ainsi que les propriétés que l'on attend d'un tel protocole. Les deux sections suivantes introduisent le formalisme que l'on a choisi pour modéliser un tel protocole ainsi que la description du modèle dans le cas particulier du protocole qui nous intéresse. Afin de justifier la pertinence de notre modèle, nous le comparons avec les modélisations plus classiques que l'on peut trouver dans la littérature. Enfin, nous proposons des spécifications formelles des propriétés attendues du protocole.

1 Description du protocole GJM

1.1 Le problème principal : l'asymétrie

Dans la vie réelle, les contrats sont signés simultanément. Dans le cas des signatures électroniques, c'est une séquence d'envois et réceptions de messages qui est censée permettre à chaque participant d'obtenir à la fin un contrat signé. Cette dissymétrie peut avoir pour effet de désavantager l'un des participants (typiquement, le premier à signer le contrat). Le but d'un protocole de signature de contrat est alors, entre autres, d'assurer une propriété d'équité (*fairness*) entre les participants, qui s'énonce informellement de la manière suivante : si un des participants a obtenu un contrat signé, il ne peut pas empêcher les autres d'en faire autant. Une autre propriété intéressante est l'(absence d')abus de confiance (*abuse-freeness*) : un participant ne peut pas tirer avantage du fait qu'il est en train de participer à la signature d'un contrat (par exemple, B ne doit pas pouvoir profiter du fait que A lui a envoyé le contrat signé pour obtenir de meilleures propositions d'un troisième personne C). Le protocole sur lequel on va s'appuyer est celui présenté dans [5].

1.2 Le cadre d'exécution

On suppose que deux participants A et B se sont mis d'accord sur leurs identités respectives, un contrat à signer C et l'identité d'une troisième partie de confiance (TTP) T. On suppose de plus que chaque participant X possède une clé privée $k(X)$ (il est d'usage de confondre l'identité d'un participant avec cette clé privée, ou plus exactement de le confondre avec l'ensemble des participants possèdant sa clé privée, mais on ne le fera pas ici).

1.3 Sous-protocole *exchange*

La manière la plus simple de formuler un protocole d'échange de signatures est probablement la suivante :

$$\begin{array}{llcll} 1) & A & \rightarrow & B & : \ \mathrm{Sig}_A(C) \\ 2) & B & \rightarrow & A & : \ \mathrm{Sig}_B(C) \end{array}$$

où $\mathrm{Sig}_X(M)$ dénote le message M signé par le participant X. Le problème est qu'après la première étape, B possède un avantage sur A : il peut par exemple montrer à une troisième personne que A est prêt à conclure un contrat avec lui, sans pour autant signer lui-même le contrat. Remarquons qu'il peut aussi choisir d'abandonner le protocole. Pour remédier à cela, le protocole commence par un échange de "pre"-signatures (*private contract signatures*) de la forme $\mathrm{PCS}_A(C, B, T)$ possédant les propriétés suivantes :

- $\mathrm{PCS}_A(C, B, T)$ peut être calculée aussi bien par A que par B, ce qui fait que, par exemple, B ne peut pas s'en servir comme preuve que A est impliqué dans le contrat,

– $\mathrm{PCS}_A(C, B, T)$ peut être convertie par T en une signature classique (que l'on notrera $\mathrm{Sig}_A^T(C)$ pour la distinguer de $\mathrm{Sig}_A(C)$ dans un but de vérification, cette distinction étant invisible pour les participants du protocole) ce qui permet à T de rétablir une certaine équité entre A et B.

Le protocole utilisé est alors le suivant (*Exchange subprotocol*) :

$$\begin{array}{llllll} 1) & A & \rightarrow & B & : & \mathrm{PCS}_A(C, B, T) \\ 2) & B & \rightarrow & A & : & \mathrm{PCS}_B(C, A, T) \\ 3) & A & \rightarrow & B & : & \mathrm{Sig}_A(C) \\ 4) & B & \rightarrow & A & : & \mathrm{Sig}_B(C) \end{array}$$

Il reste maintenant à mettre en place des protocoles pour :

– permettre à A et B de demander à T d'intervenir pour convertir une pre-signature en signature dans le cas où ils s'estiment lésés,
– permettre à A d'abandonner le protocole (B peut le faire simplement en n'envoyant pas le dernier message).

C'est le rôle des sous-protocoles *abort* et *resolve* décrits dans la section suivante.

1.4 Sous-protocoles *abort* et *resolve*

Commençons par le protocole permettant à A d'abandonner l'échange de signatures. Il envoie à T une demande qui se compose d'une marque d'abandon et des différentes valeurs caractéristiques de la session, ceci étant signé par A dans un but d'authentification. La réponse de T peut alors être :

– $\mathrm{Sig}_B(C)$ si B a déjà fait appel à T pour conclure l'échange (au moyen du sous-protocole *resolve*),
– $\mathrm{Sig}_B^T(C)$ si A à déjà demandé à T de conclure le contrat,
– une marque d'abandon dans les autres cas.

On obtient alors le protocole suivant (*Abort subprotocol*) :

$$\begin{array}{llllll} 1) & A & \rightarrow & T & : & \mathrm{Sig}_A(\langle C, \langle A, \langle B, \mathrm{abort}\rangle\rangle\rangle) \\ 2) & T & \rightarrow & A & : & \textbf{if } (\mathrm{resolved}(B)) \\ & & & & & \quad \textbf{then } \mathrm{Sig}_B(C) \\ & & & & & \quad \textbf{else if } (\mathrm{resolved}(A)) \\ & & & & & \quad\quad \textbf{then } \mathrm{Sig}_B^T(C) \\ & & & & & \quad\quad \textbf{else } \mathrm{Sig}_T(\mathrm{Sig}_A(\langle C, \langle A, \langle B, \mathrm{abort}\rangle\rangle\rangle)) \\ & & & & & \quad \mathrm{aborted} = \mathrm{true} \end{array}$$

Le protocole *resolve* permet à un participant de faire intervenir T pour conclure le contrat (en général, en convertissant une pre-signature en une signature classique). Il y a donc en fait deux sous-protocoles *resolve*, un pour A et un pour B. On va décrire ici le cas B (le protocole pour A étant symétrique). B lance ce protocole en envoyant à T la pre-signature de A (pour que T puisse la convertir) ainsi que sa propre signature du contrat (afin de prouver qu'il est prêt à le conclure). La réponse de T peut alors être :

– une marque d'abandon si A avait déjà demandé à abandonner,
– $\mathrm{Sig}_A(C)$ si A avait déjà demandé à T de conclure l'échange,

– $\mathrm{Sig}_A^T(C)$ obtenue par conversion de $\mathrm{PCS}_A(C,B,T)$ dans les autres cas.

Le protocole est alors le suivant (*Resolve subprotocol for B*) :

$$
\begin{array}{llllll}
1) & B & \rightarrow & T & : & \langle \mathrm{PCS}_A(C,B,T), \mathrm{Sig}_B(C)\rangle \\
2) & T & \rightarrow & B & : & \textbf{if } (\text{aborted}) \\
 & & & & & \quad \textbf{then } \mathrm{Sig}_T(\mathrm{Sig}_A(\langle C, \langle A, \langle B, \text{abort}\rangle\rangle\rangle)) \\
 & & & & & \quad \textbf{else} \quad \textbf{if } (resolved A) \\
 & & & & & \qquad\qquad \textbf{then } \mathrm{Sig}_A(C) \\
 & & & & & \qquad\qquad \textbf{else } \mathrm{Sig}_A^T(C) \\
 & & & & & \qquad \mathrm{resolved}(B) = \mathrm{true}
\end{array}
$$

Dans la suite, on appellera :
– *abort request* le message $\mathrm{Sig}_A(\langle C, \langle A, \langle B, \text{abort}\rangle\rangle\rangle)$,
– *abort token* le message $\mathrm{Sig}_T(\mathrm{Sig}_A(\langle C, \langle A, \langle B, \text{abort}\rangle\rangle\rangle))$,
– *resolve request for B* le message $\langle \mathrm{PCS}_A(C,B,T), \mathrm{Sig}_B(C)\rangle$.

2 Quelques propriétés du protocole

Décrivons informellement quelques propriétés qu'il serait intéressant de vérifier à propos de ce protocole :

– *Fairness* :
  – il est impossible pour un participant corrompu d'obtenir un contrat signé valide sans permettre aux autres de faire de même,
  – une fois qu'un participant honnête a obtenu un *abort token* de T, il est impossible pour un autre participant d'obtenir un contrat signé valide,
  – chaque participant honnête est sûr de pouvoir conclure le protocole (en obtenant soit un contrat signé valide, soit un *abort token*),
– *Completeness* : si les participants sont honnêtes, ils sont sûrs de pouvoir conclure le protocole (sous réserve que les canaux de communication soient suffisamment fiables),
– *Abuse-freeness* : il est impossible pour un participant isolé de prouver à un observateur extérieur qu'il a le pouvoir de déterminer l'issue du contrat (*ie* une signature ou un abandon),
– *Trusted third party accountability* : si l'un des participants est lésés à cause d'un mauvais comportement de T, alors il peut prouver que T s'est mal comporté.

3 (Lock-step synchronous) ATS et ATL

On va utiliser la logique temporelle alternée (*ATL*) pour modéliser les propriétés du protocole. Un modèle pour cette logique est un système de transition alterné (*ATS*). Dans le cas qui nous intéresse, on peut se restreindre à une sous-classe des *ATS* qui sont les *lock-step synchonous ATS*. Toutes les définitions présentées ici sont issues de [1], qui présente de plus de nombreuses autres variantes ainsi que des algorithmes de *model-checking* et des calculs de complexité.

Définition 1 *Un* lock-step synchonous ATS *(*ATS *dans la suite) est un triplet :*

$$(\Sigma, Q, \delta)$$

où :
- Σ *est l'ensemble fini des agents du système,*
- Q *est l'ensemble fini des états du système, de la forme* $Q = \prod_{a \in \Sigma} Q_a$*,*
- δ *est une famille de fonctions* $(\delta_a)_{a \in \Sigma}$ *où pour chaque* $a \in \Sigma$ *:*

$$\delta_a : Q \to \mathcal{P}(Q_a)$$

Pour $q, q' \in Q$*, il y a une transition de* q *vers* q' *(noté* $q \to q'$*) si et seulement si pour chaque* $a \in \Sigma$ *on a* $q'_a \in \delta_a(q)$*.*

Dans la suite de cette section, on suppose défini un *ATS* (Σ, Q, δ).

Définition 2 *Les formules de la logique* ATL *sur un ensemble de proposition atomiques* Π *et un ensemble d'agents* Σ *sont définies par la grammaire suivante :*

$$\begin{array}{rcll} \phi \in \mathrm{ATL}[\Pi, \Sigma] & := & p \in \Pi & \\ & | & \neg\phi & \\ & | & \phi \vee \psi & \\ & | & \langle\langle A \rangle\rangle \circ \phi & (A \subseteq \Sigma) \\ & | & \langle\langle A \rangle\rangle \Box \phi \ (A \subseteq \Sigma) & \\ & | & \langle\langle A \rangle\rangle \phi \mathcal{U} \psi \ (A \subseteq \Sigma) & \end{array}$$

Le A apparaissant dans les modalités signifiant (intuitivement) que les agents dans A peuvent coopérer de manière à rendre la formule concernée vraie. La définition précise de la sémantique de *ATL* fait intervenir la notion de stratégie pour un ensemble d'agents.

Définition 3 *Une stratégie pour* $a \in \Sigma$ *est une application* $f : Q^+ \to Q_a$ *telle que pour tout* $w \in Q^*$ *et pour tout* $q \in Q$ *:*

$$f(w.q) \in \delta_a(q)$$

Un ensemble de stratégies pour $A \subseteq \Sigma$ *est une application* f *qui envoie chaque* $a \in A$ *sur une stratégie* f_a *pour* a*, on note alors :*

$$\mathrm{out}(q, f) = \{w = q_0 q_1 \ldots \mid q_0 = q \text{ and } \forall i \geq 0. \forall a \in A. (q_{i+1})_a = f_a(q_0 \ldots q_i)\}$$

Définition 4 *Si* (Σ, Q, δ) *est un* ATS*,* $q \in Q$ *et si* π *est une valuation sur* Π *(*ie *une fonction qui a tout* $p \in \Pi$ *associe une valeur de vérité), on définit une relation* $q, \pi \models _ \subseteq \mathrm{ATL}[\Pi, \Sigma]$ *par induction sur* $\phi \in \mathrm{ATL}[\Pi, \Sigma]$ *de la manière suivante :*
- $q, \pi \models p \in \Pi$ *ssi* $p \in \pi(q)$
- $q, \pi \models \neg\phi$ *ssi* $q \not\models \phi$
- $q, \pi \models \phi \vee \psi$ *ssi* $q \models \phi$ *or* $q \models \psi$

- $q, \pi \models \langle\langle A \rangle\rangle \circ \phi$ *ssi il existe* f *ensemble de stratégies pour* A *tel que pour* $w = q_0 q_1 \ldots \in \mathrm{out}(q, f)$ *on ait* $q_1, \pi \models \phi$
- $q, \pi \models \langle\langle A \rangle\rangle \Box \phi$ *ssi il existe* f *ensemble de stratégies pour* A *tel que pour* $w = q_0 q_1 \ldots \in \mathrm{out}(q, f)$ *et* $i \geq 0$ *on ait* $q_i\, \pi \models \phi$
- $q, \pi \models \langle\langle A \rangle\rangle \phi \mathcal{U} \psi$ *ssi il existe* f *ensemble de stratégies pour* A *tel que pour* $w = q_0 q_1 \ldots \in \mathrm{out}(q, f)$ *il existe* $j \geq 0$ *tel que* $q_0, \pi \models \phi, \ldots, q_{j-1}, \pi \models \phi$ *et* $q_j, \pi \models \psi$

4 Modélisation

4.1 Principe de la modélisation

Le but que l'on se propose est de modéliser le protocole sous la forme d'un *ATS* et les propriétés à vérifier sous la forme de formules de la logique *ATL*. Pour rendre la définition du système de transitions plus simple, on va le décrire comme modèle d'un ensemble de clauses de Horn. Les ensembles d'états pourront alors être représentés de manière symbolique par des ensembles de clauses de Horn. On définira sur ces représentations symboliques les opérations ensemblistes usuelles et on constatera que le test du vide se ramène à vérifier si une certaine formule de la logique du premier ordre est conséquence d'un ensemble de clauses de Horn (ce qui sera décidable dans le cas particulier qui nous préoccuppe).

4.2 Principe de la modélisation logique d'un *ATS*

Supposons, pour simplifier, qu'il n'y a que deux agents a et b, décrits par les clauses :

$$\overline{agent(a)} \qquad \overline{agent(b)}$$

On dispose aussi d'un prédicat *event* qui décrit les événements possibles que peuvent réaliser les agents. Les états du système sont alors des couples (Qa, Qb) où Qa est l'état local de a et Qb l'état local de b, ce sont tous deux des listes d'événements :

$$\frac{\ldots}{event(E)}$$

$$\frac{event_list(Qa) \quad event_list(Qb)}{state((Qa, Qb))}$$

La relation de transition tr est définie à partir de deux relations de transitions locales tr_a et tr_b qui sont deux relations binaires entre états est listes d'états locaux. Intuitivement, $tr_a(Q, Qal)$ signifie que Qal est la liste des Qa vers

lesquels a peut effectuer une transition à partir de Q :

$$\frac{\ldots}{tr_a(Q, Qal)} \qquad \frac{\ldots}{tr_b(Q, Qbl)}$$

$$\frac{tr_a(Q, Qal) \quad member(Qa, Qal) \quad tr_b(Q, Qbl) \quad member(Qb, Qbl)}{tr(Q, (Qa, Qb))}$$

Supposons définit un prédicat $init$ qui représente l'ensemble des états initiaux :

$$\frac{\ldots}{init(Q)}$$

l'ensemble des états atteignables est alors décrit par :

$$\frac{init(Q)}{reach(Q)}$$

$$\frac{reach(Q) \quad tr(Q, Q')}{reach(Q')}$$

Notons $\mathcal{H}_0$ l'ensemble des clauses de Horn ainsi définit. La représentation symbolique d'un ensemble d'états est un couple $(P+, P-, \mathcal{H})$ dans lequel $P+$ et $P-$ sont des symboles de prédicats unaires et $\mathcal{H}$ est un ensemble de clauses de Horn tel que les deux ensembles :

$$S+ = \{q \mid \mathcal{H}_0 \cup H \vdash reach(q) \wedge P+(q)\}$$
$$S- = \{q \mid \mathcal{H}_0 \cup H \vdash reach(q) \wedge P-(q)\}$$

soient complémentaires dans l'ensemble :

$$\{q \mid \mathcal{H}_0 \vdash reach(q)\}$$

Les différentes opérations ensemblistes sont alors décrites sur les représentations symboliques de la manière suivante :

- l'ensemble vide est représenté par :

$$\left(P+, P-, \left\{\frac{}{P-(Q)}\right\}\right)$$

- si $(P_1+, P_1-, \mathcal{H}_1)$ et $P_2+, P_2-, \mathcal{H}_2)$ sont deux représentations symboliques, leur union et leur intersection est définie au moyen de deux nouveaux symboles $P+$ et $P-$ en posant pour l'union :

$$\left(P+, P-, \mathcal{H}_1 \cup \mathcal{H}_2 \cup \left\{\frac{P_1+(Q)}{P+(Q)}, \frac{P_2+(Q)}{P+(Q)}, \frac{P_1-(Q) \quad P_2-(Q)}{P-(Q)}\right\}\right)$$

et pour l'intersection :

$$\left(P+, P-, \mathcal{H}_1 \cup \mathcal{H}_2 \cup \left\{\frac{P_1+(Q) \quad P_2+(Q)}{P+(Q)}, \frac{P_1-(Q)}{P-(Q)}, \frac{P_2-(Q)}{P-(Q)}\right\}\right)$$

– si $(P+, P-, \mathcal{H})$ est une représentation symbolique on définit la représentation de son complémentaire en considérant deux nouveaux symboles de prédicats $P'+$ et $P'-$ et en posant :

$$\left(P'+, P'-, \left\{ \frac{P-(Q)}{P'+(Q)}, \frac{P+(Q)}{P'-(Q)} \right\}\right)$$

Il faut également expliquer la construction "Pre" utilisé dans l'algorithme de model-checking de ATL : pour un ensemble A d'agents et un ensemble $\mathcal{Q}$ d'états, on note :

$$\begin{aligned} Pre(A, \mathcal{Q}) = \{q \in Q \mid\ & \exists (qa)_{a\in A} \in \Pi_{a\in A}\delta(q,a). \\ & \forall (qb)_{b\notin A} \in \Pi_{b\notin A}\delta(q,b). \\ & ((qa)_{a\in A}, (qb)_{b\notin A}) \in \mathcal{Q}\} \end{aligned}$$

Dans le cas présent où il n'y a que deux agents, on a quatre possibilités pour A qui peut être soit vide, soit l'ensemble de tous les agents, soit réduit à a ou b, ce qui permet d'écrire pour Pre les quatre définitions suivantes :

$$\begin{aligned} &Pre(\varnothing, \mathcal{Q}) = \{q \in Q \mid \forall qa \in \delta_a(q). \forall qb \in \delta_b(q). (qa, qb) \in \mathcal{Q}\} \\ &Pre(\{a,b\}, \mathcal{Q}) = \{q \in Q \mid \exists qa \in \delta_a(q). \exists qb \in \delta_b(q). (qa, qb) \in \mathcal{Q}\} \\ &Pre(\{a\}, \mathcal{Q}) = \{q \in Q \mid \exists qa \in \delta_a(q). \forall qb \in \delta_b(q). (qa, qb) \in \mathcal{Q}\} \\ &Pre(\{b\}, \mathcal{Q}) = \{q \in Q \mid \exists qb \in \delta_b(q). \forall qa \in \delta_a(q). (qa, qb) \in \mathcal{Q}\} \end{aligned}$$

Si on considère une représentation symbolique $(P+, P-, \mathcal{H})$, la représentation symbolique des différents "Pre" est obtenue en considérant deux nouveaux symboles de prédicats $P'+$ et $P'-$ et en posant :

– pour $\varnothing$:

$$\left(P'+, P'-, \left\{ \begin{array}{c} \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad ff_P+(Qal,Qbl)}{P'+(Q)}, \\ \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad ee_P-(Qal,Qbl)}{P'-(Q)} \end{array} \right\}\right)$$

– pour $\{a,b\}$:

$$\left(P'+, P'-, \left\{ \begin{array}{c} \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad ee_P+(Qal,Qbl)}{P'+(Q)}, \\ \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad ff_P-(Qal,Qbl)}{P'-(Q)} \end{array} \right\}\right)$$

– pour $\{a\}$:

$$\left(P'+, P'-, \left\{ \begin{array}{c} \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad ef_P+(Qal,Qbl)}{P'+(Q)}, \\ \dfrac{tr_a(Q,Qal) \quad tr_b(Q,Qbl) \quad fe_P-(Qal,Qbl)}{P'-(Q)} \end{array} \right\}\right)$$

– pour $\{b\}$: *idem.*

Remarque : Ce que l'on désigne par $ef_P+(Qal, Qbl)$ c'est le fait qu'il existe Qa dans Qal tel que pour tout Qb dans Qbl on ait $P+((Qa, Qb))$ (et de même pour les constructions fe, ff, ee).

Finalement, pour tester si une représentation symbolique $(P+, P-, \mathcal{H})$ dénote un ensemble vide, il suffit de montrer l'une des deux propriétés suivantes :

$$\mathcal{H}_0, \mathcal{H} \vdash reach(Q) \rightarrow \neg P+(Q))$$
$$\mathcal{H}_0, \mathcal{H} \vdash reach(Q) \rightarrow P-(Q)$$

4.3 La modélisation en détails

Commençons par introduire un peu de terminologie :

- un principal est l'un des rôles du protocole :

$$R \in Principal := A \mid B \mid T$$

- les sous-protocoles sont :

$$p \in SubProtocol := Exch \mid Abrt \mid Rslv_A \mid Rslv_B$$

- On considère par exemple deux sessions :

$$s \in Session := s1 \mid s2$$

- les niveaux d'honnêteté (valables pour les participants aussi bien que pour les canaux de communication) sont :

$$l \in Level := honest \mid weakly \mid strongly$$

- les agents (dans le sens des ATS) sont :

$$X \in Agents := (s, R) \mid I$$

4.3.1 L'algèbre des messages

On considère un nombre fini de participants (par exemple un participant pour instancier chaque rôle de chaque session, mais on pourrait tout aussi bien supposer qu'un même participant instancie des rôles dans des session différentes) et chacun d'eux possède une clé privée :

$$x \in Participant := a1 \mid a2 \mid b1 \mid b2 \mid t1 \mid t2 \mid i \mid \ldots$$
$$k \in Key := k(a1) \mid k(a2) \mid \ldots$$

On considère également un certain nombre de données atomiques (les contrats à signer entre autres) :

$$d \in Data := \text{abort} \mid c1 \mid c2 \mid \ldots$$

A chaque session sont associées des valeurs caractéristiques (*attributs*) qui sont les identités des participants impliqués dans ces sessions et le texte du contrat à signer :

$$\overline{attr(s1, A, a1)} \quad \overline{attr(s1, B, b1)} \quad \overline{attr(s1, T, t1)} \quad \overline{attr(s1, C, c1)}$$

$$\overline{attr(s2, A, a2)} \quad \overline{attr(s2, B, b2)} \quad \overline{attr(s2, T, t2)} \quad \overline{attr(s2, C, c2)}$$

Les messages arbitraires sont construits à partir de la grammaire suivante :

$$\begin{array}{rcll} m \in Msg & := & x \mid k \mid d & \textit{(Atomic messages)} \\ & \mid & \langle m, m \rangle & \textit{(Pair construction)} \\ & \mid & \mathrm{Sig}_x(m) & \textit{(Classic signatures)} \\ & \mid & \mathrm{Sig}_x^y(m) & \textit{(Converted signatures)} \\ & \mid & \mathrm{PCS}_x(m, y, z) & \textit{(“Pre”-signatures)} \end{array}$$

où :

- $\mathrm{Sig}_x(m)$ dénote le message m signé par x,
- $\mathrm{Sig}_x^y(m)$ dénote le message m converti par y en une signature de x (rappelons que $\mathrm{Sig}_x(m)$ et $\mathrm{Sig}_x^y(m)$ sont égaux du point de vue des participants, et on note alors $\mathrm{Sig}_x^{\text{-}}(m)$),
- $\mathrm{PCS}_x(m, y, z)$ dénote le message m signé par x par rapport à y et à l'intention de z pour les conversions.

4.3.2 Connaissances et déductions des agents

La définition de ce qu'un agent (quelconque) peut déduire à partir d'un ensemble s de messages se fait en étendant le modèle classique Dolev & Yao [4] au nouveau type de signatures considéré. On note $s \Vdash m$ le fait que le message m puisse être produit à partir de la connaissance (*ie* la liste de messages) s :

- (*Induction basis*)
$$\frac{member(m, s)}{s \Vdash m}$$
- (*Pair operator*)
$$\frac{s \Vdash m_1 \quad s \Vdash m_2}{s \Vdash \langle m_1, m_2 \rangle}$$
$$\frac{s \Vdash \langle m_1, m_2 \rangle}{s \Vdash m_1} \qquad \frac{s \Vdash \langle m_1, m_2 \rangle}{s \Vdash m_2}$$
- (*Pre-signatures*)
$$\frac{s \Vdash k(y) \quad s \Vdash m \quad s \Vdash z \quad s \Vdash t}{s \Vdash \mathrm{PCS}_y(m, z, t)} \qquad \frac{s \Vdash y \quad s \Vdash m \quad s \Vdash k(z) \quad s \Vdash t}{X \Vdash s\mathrm{PCS}_y(m, z, t)}$$
$$\frac{s \Vdash \mathrm{PCS}_y(m, z, t)}{s \Vdash y} \qquad \frac{s \Vdash \mathrm{PCS}_y(m, z, t)}{s \Vdash m} \qquad \frac{s \Vdash \mathrm{PCS}_y(m, z, t)}{s \Vdash z} \qquad \frac{s \Vdash \mathrm{PCS}_y(m, z, t)}{s \Vdash t}$$

– (*Conventional signatures*)

$$\frac{s \Vdash k(y) \quad s \Vdash m}{s \Vdash \mathrm{Sig}_y^u(m)}$$

$$\frac{s \Vdash \mathrm{Sig}_y^u(m)}{s \Vdash y} \qquad \frac{s \Vdash \mathrm{Sig}_y^u(m)}{s \Vdash m}$$

– (*Converted signatures*)

$$\frac{s \Vdash \mathrm{PCS}_y(m, z, t) \quad s \Vdash k(t)}{X \Vdash s\mathrm{Sig}_y^t(m)}$$

$$\frac{s \Vdash \mathrm{Sig}_y^t(m)}{s \Vdash y} \qquad \frac{s \Vdash \mathrm{Sig}_y^t(m)}{s \Vdash m}$$

4.3.3 Evénements et états

Les événements sont des envois de messages, notés :

$$e \in Event := send(x, y, z, m, s, p, l)$$

où :
- x désigne l'émetteur du message,
- y est le destinataire (escompté)
- z est le receveur ($z = i$ pour modéliser un canal non fiable par exemple, alors que $z = y$ pour représenter une communication sûre),
- m est le message envoyé,
- s est la session du protocole dans laquelle intervient cet envoi de message,
- p est le sous-protocole considéré,
- l est le niveau d'honnêteté du participant lorsqu'il envoie ce message.

L'état local de chaque principal du protocole est alors une suite d'événements :

$$ql \in LState := Event^*$$

L'état local d'une session est le produit de l'état local de chaque principal :

$$qs \in SState := (qa, qb, qt) \in LState^3$$

et l'état global du système est constitué de l'état local de chaque session ainsi que de l'état local de l'intrus :

$$q \in State := (qs1, qs2, qi) \in SState \times SState \times LState$$

4.3.4 Quelques primitives de manipulation des états

Tout d'abord à chaque état q et chaque agent X on peut associer la composante qX de q relative à X. On représente ceci au moyen d'un prédicat

$proj(_,_,_) \subseteq State \times Agent \times LState$ définit de la manière suivante :

$$\overline{proj(((qa,qb,qt),qs2,qi),(s1,A),qa)} \quad \overline{proj((qs1,(qa,qb,qt),qi),(s2,A),qa)}$$

$$\overline{proj(((qa,qb,qt),qs2,qi),(s1,B),qb)} \quad \overline{proj((qs1,(qa,qb,qt),qi),(s2,B),qb)}$$

$$\overline{proj(((qa,qb,qt),qs2,qi),(s1,T),qt)} \quad \overline{proj((qs1,(qa,qb,qt),qi),(s2,T),qt)}$$

$$\overline{proj((qs1,qs2,qi),i,qi)}$$

Ensuite, définissons ce qu'un agent peut observer dans un état : il s'agit de tous les messages dont il est le receveur (réel). On définit donc deux prédicats :

- le premier, noté $rec(_,_,_) \subseteq State \times Agent \times Event_List$ décrit la liste des événements que peut observer un agent dans un certain état (*ie* les événements dont il est le receveur réel),
- le second, noté $obs(_,_,_) \subseteq State \times Agent \times Msg_List$ décrit la liste des messages que connait un agent dans un état (il suffit simplement de prendre la liste des messages apparaissant dans un événement dont l'agent en question est le receveur réel, augmenté de la clé privée de cet agent et des données atomiques que l'on considère connues universellement).

Finalement, on aura besoin d'un prédicat indiquant pour chaque état q, chaque session s, chaque principal R et chaque sous-protocole p à quel point du protocole p se trouve l'agent (s,R) dans l'état q. Ce prédicat est noté $loc(_,_,_,_,_) \subseteq State \times Session \times Principal \times SubProtocol \times Int$ et sa définition est simple mais fastidieuse. Ce prédicat joue un rôle important dans la définition du comportement honnête d'un agent, puisque par définition un agent honnête doit suivre le protocole, et donc agir en fonction du point du protocole où il se trouve. On aura également besion d'un prédicat $end_point(_,_) \subseteq Principal \times Int$ qui indique quel est le point qui correspont à la fin du protocole pour l'agent considéré.

4.3.5 Relation(s) de transition

A chaque agent X est associée une relation de transition locale $tr(_,X,_) \subseteq State \times LState^*$ qui possède une définition de la forme suivante :

$$\frac{tr(q,X,honest,qlh) \quad tr(q,X,weakly,qlw) \quad tr(q,X,strongly,qls)}{tr(q,X,qlh.qlw.qls)}$$

autrement dit chaque agent peut choisir d'agir honnêtement, faiblement malhonnêtement ou fortement malhonnêtement à chaque fois qu'il effectue une transition. Quelques précisions sur la manière dont sont définis ces différents comportements, en commençant tout d'abord par le cas où l'agent considéré n'est pas l'intrus :

- pour le comportement honnête, il faut que les actions effectuées soient en accord avec la position de l'agent dans le protocole considéré,

- pour le comportement fortement malhonnête, la seule action possible est de fournir sa clé à l'intrus,
- pour le comportement faiblement malhonnête, l'agent considéré peut envoyer à n'importe qui n'importe lequel des messages qu'il peut produire (parmi une liste finie de messages, par exemple les messages qui sont des instances des messages intervenant dans le protocole). On pourrait adopter d'autres définitions de faiblement malhonnête, par exemple convenir qu'un agent faiblement malhonnête est suceptible d'accepter des messages qui ne rentrent pas dans le cadre du protocole. De toutes façons notre construction du modèle est modulaire par rapport à de tels choix.

En ce qui concerne les comportements de l'intrus maintenant :

- pour le comportement honnête, il doit se contenter d'envoyer vers leur destinataire escompté *tous* les messages dont il est le destinataire réel,
- pour le comportement faiblement malhonnête, il peut ajouter des messages de son cru à la liste des messages qu'il doit renvoyer, mais pas en supprimer,
- pour le comportement fortement malhonnête, il peut tout aussi bien ajouter que supprimer des messages (on pourrait là encore envisager d'autres définitions possibles).

4.4 *ATS* associé

Cet ensemble de clauses de Horn décrit un ATS. Pour le décrire formellement, notons T l'ensemble des termes clos sur la signature induite par cet ensemble de clauses de Horn. On construit l'ATS associé en posant :

- Pour l'ensemble des agents :

$$\Sigma = \{a \in T \mid agent(a)\}$$

- Pour l'ensemble des états :

$$Q = \{q \in T \mid state(q)\} \simeq \prod_{a \in \Sigma} \{el \in T \mid event_list(el)\}$$

- pour la relation de transition :

$$\delta_a(q) = \{qa \in T \mid member(qa, qal) \text{ et } tr(q, a, qal)\}$$

5 Comparaison avec des modèles standards

5.1 Point de vue informel

Tout d'abord ce modèle permet de faire la distinction entre différents degrés de corruption, aussi bien pour les participants que pour les canaux de communications. On peut par exemple prendre ainsi en compte des canaux de communication semi-sûrs en ce sens qu'il est possible d'y écrire mais pas de supprimer les messages, ou bien qu'il est seulement possible de lire les messages qui y circulent,

etc. On peut aussi remarquer que ce modèle permet d'exprimer les capacités que possède un participant malhonnête mais agissant seul (*ie* sans l'identifier avec l'intrus). Imaginons par exemple trois participants d'un protocole a, b et c et intéressons-nous à ce que peut faire b contre a et c. Dans un modèle classique, on identifierait b à l'intrus ce qui permettrai à b d'espionner les communications entre a et c. On peut ici préciser que b est (faiblement) malhonnête, sans pour autant l'identifier à l'intrus. Il peut alors agir malhonnêtement (*ie* sans suivre le protocole) mais il ne peut plus espionner les communications entre a et c.

En ce qui concerne les propriétés formelles, on peut prendre en compte des propriétés de la forme "il existe une exécution telle que ...dans laquelle n'apparait aucune action de la forme..." (par exemple "il existe une action où les deux participants ont leur contrat signé sans intervention active de l'intrus" (*completeness*)). Signalons que les hypothèses concernant les participants et les canaux de comunications pourront être intégrées aux formules (au lieu de construire différents systêmes de transitions relatifs à chaque jeu d'hypothèses). On verra également que plusieurs propriétés formelles peuvent être associées à la même propriété intuitive, et il pourra être intéressant de les comparer (du point de vue de la validité dans le modèle).

Ces deux points nous conduisent à penser que l'on obtient ainsi un modèle plus précis que les modèles type Dolev & Yao [4] (ce qui permet, entre-autres de prouver plus de propriétés à propos du protocole).

5.2 Comparaison formelle

Formellement, on peut considérer que notre modèle (Se) est décrit par un certain nombre de règles, précisément :

- pour chaque principal R, on a trois règles, notées (Rh), (Rw) et (Rs) décrivant respectivement comment peut agir ce participant dans un cadre honnête (en respectant le protocole), faiblement malhonnête (sans respecter le protocole, mais sans coopérer avec l'intrus) ou fortement malhonnête (en coopérant avec l'intrus, *ie* en lui fournissant sa clé),
- trois règles pour l'intrus (Ih), (Iw) et (Is) décrivant les trois comportement possibles des canaux de communication (c'est à dire, se contenter de faire parvenir à leur destinataire les messages reçus dans le cadre d'un comportement honnête, éventuellement en ajouter de nouveaux dans le cadre d'un comportement faiblement malhonnête voire même en supprimer dans un comportement fortement malhonnête).

On pourrait bien sûr faire un autre choix pour la définition des comportements faiblement malhnnêtes, comme cela a été expliqué plus haut. On pourrait même, en ce qui concerne l'intrus, faire dépendre de comportement de la ligne de communications considérée (par exemple distinguer les lignes vers le TTP des autres). Pour un système classique (Sc) par ailleurs, on se contente en général des règles (Rh), (Rs) et (Is). Il est clair qu'une exécution de (Sc) est *a fortiori* une exécution de (Se). Réciproquement, on peut traduire une exécution de (Se) dans (Sc), il suffit pour cela de trouver la contrepartie dans (Sc) des règles (Rw), (Ih) et (Iw) :

- pour (Ih) et (Iw) c'est simple, car ces deux règles peuvent être réalisées par (Is) (mais on ne peut plus, bien sûr, considérér des propriétés de la forme "si l'intrus est au plus faiblement malhonnête, alors..."),
- pour (Rw) c'est un peu plus délicat. Comme cette règle concerne la construction d'un message à partir de la connaissance du rôle R, il faut tout d'abord que R fournisse sa clé à l'intrus, puis que l'intrus forme le message concerné. Ainsi, une application de (Rw) dans (Se) se traduit par une application de (Rs) suivie de (Is) dans (Sc).

Ainsi, chaque action faiblement malhonnête dans (Se) a pour contrepartie dans (Sc) l'identification du participant en question avec l'intrus. Il est par conséquent clair que certaines propriétés ne pourront pas être préservées par cette traduction. Imaginons un protocole dans lequel deux participants a et b veulent s'échanger des secrets (par exemple deux nonces). On aurait alors deux propriétés intéressantes :

- *fairness* : si b a obtenu le secret de a, alors a peut obtenir le secret de b,
- *secretness* : l'intrus ne connait pas les secrets échangés.

Imaginons en outre que a ne puisse pas obtenir le secret de b en restant dans le cadre du protocole, et qu'il soit par conséquent obligé de devenir faiblement malhonnête. Dans (Se), ceci n'induirait pas la violation de la propriété de *secretness*, alors que ce serait le cas dans (Sc).

6 Modelisation de propriétés avec *ATL*

6.1 Propositions atomiques

Il faut avant tout décider une fois pour toutes quelles sont les propriétés atomiques dont on va avoir besoin pour spécifier les propriétés du protocole. On distingue essentiellement trois catégories :

- *Corruption level* : on appelle *corruption level* d'un agent $X \in Agent$ dans un état $Q \in State$ la borne supérieure des niveaux de corruption des différentes actions qu'il a effectué (pour l'ordre $honest < weakly < strongly$). Pour $L \in Level$, $X \in Agent$ et $Q \in State$, on désigne par $corrupt(Q, X, L)$ le fait que X dans l'état Q a le niveau de corruption L,
- *Participant's knowledge* : pour un agent $X \in Agent$, un état $Q \in State$ et un message $M \in Msg$, on note $knows(Q, X, M)$ le fait que dans l'état Q, X puisse produire le message M,
- *End of protocol* : pour un agent $X \in Agent$, un état $Q \in State$, on désigne par $finish(Q, X)$ le fait que dans l'état Q, X a terminé le protocole.

Comme ces différentes propositions sont censées définir des ensembles d'états, il faut en fait leur associer une représentation symbolique. Pour le niveau de corruption, on considère tout d'abord un prédicat $level_in$ permettant de calculer

le niveau de corruption d'une suite d'actions :

$$\frac{}{level_in([], honest)}$$

$$\frac{level_in(El, L1) \quad level_of(E, L2) \quad glb(L1, L2, L)}{level_in([E|El], L)}$$

(les prédicats glb et $level_of$ représentant respectivement la borne supérieure dans l'ensemble des niveaux de corruption et le niveau de corruption d'un événement. On ne donnera pas leur définition ici). On considère alors deux nouveaux symboles de prédicats $corrupt+$ et $corrupt-$ qui permettent de construire la représentation symbolique de cette famille de propositions atomiques : il suffit de poser pour $X \in Agent$ et $L \in Level$:

$$\left(corrupt+(_, X, L), corrupt-(_, X, L), \left\{ \begin{array}{l} \dfrac{proj(Q, X, Qx) \quad level_in(Qx, L)}{corrupt+(Q, X, L)} \\ \\ \dfrac{proj(Q, X, Qx) \quad level_in(Qx, L') \quad L' \neq L}{corrupt-(Q, X, L)} \end{array} \right\} \right)$$

En ce qui concerne les connaissances des participants maintenant, le plus simple est de définir un prédicat $_ \not\Vdash _$ qui permet de dire qu'un message ne peut pas être produit à partir d'une liste de messages fournie en paramètre. On a alors la représentation symbolique :

$$\left(knows+(_, X, M), knows-(_, X, M), \left\{ \begin{array}{l} \dfrac{obs(Q, X, Ml) \quad Ml \Vdash M}{knows+(Q, X, M)} \\ \\ \dfrac{obs(Q, X, Ml) \quad Ml \not\Vdash M}{knows-(Q, X, M)} \end{array} \right\} \right)$$

Enfin, en ce qui concerne le fait de finir le protocole, on a la représentation symbolique suivante :

$$\left(finish+(_, X), finish-(_, X), \left\{ \begin{array}{l} \dfrac{loc(Q, X, Exch, C) \quad end_point(X, C)}{finish+(Q, X)} \\ \\ \dfrac{loc(Q, X, Exch, C) \quad end_point(X, C') \quad C \neq C'}{finish-(Q, X)} \end{array} \right\} \right)$$

Passons maintenant à la spécification des différentes propriétés.

6.2 *Completeness*

Cette propriété exprime que les principaux A et B d'une session s peuvent coopérer de manière à obtenir un contrat signé valide. Il faut toutefois raffiner ceci un peu :

– ce n'est intéressant que si les participants en question peuvent obtenir ce contrat signé tout en restant honnêtes (ce qui se traduit simplement dans notre modèle),
– pour pouvoir prouver cette propriété, il est nécessaire de supposer une certaine forme d'honnêteté des canaux de communication, l'intrus doit par conséquent coopérer.

On obtient donc l'énoncé suivant :

$$\langle\langle (s,A),(s,B),I\rangle\rangle\Diamond \left(\begin{array}{l} corrupt(q,(s,A),honest) \wedge corrupt(q,(s,B),honest) \\ \wedge corrupt(q,I,honest) \\ \wedge knows(q,(s,A),\mathrm{Sig}_b(c)) \wedge knows(q,(s,B),\mathrm{Sig}_a(c)) \end{array} \right)$$

Une forme plus faible consiste à se placer du point de vue des identités des participants (qui peuvent désigner plusieurs rôles dans plusieurs sessions) :

$$\langle\langle a,b,i\rangle\rangle\Diamond \left(\begin{array}{l} corrupt(q,a,honest) \wedge corrupt(q,b,honest) \\ \wedge corrupt(q,i,honest) \\ \wedge knows(q,a,\mathrm{Sig}_b(c)) \wedge knows(q,b,\mathrm{Sig}_a(c)) \end{array} \right)$$

6.3 *Fairness*

La première forme de cette propriété s'énonce en disant que si un participant a obtenu un contrat signé valide, alors il ne peut pas empêcher les autres d'en faire autant. Traitons par exemple le cas où le principal A de la session s obtient un contrat signé valide. On peut tout d'abord penser à spécifier que l'agent (s,A) n'a pas de stratégie pour empêcher (s,B) d'obtenir aussi le contrat signé :

$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \neg\langle\langle (s,A)\rangle\rangle\Box\neg knows(q,(s,B),\mathrm{Sig}_a^-(c))$$

On peut aussi vouloir que (s,B) possède une stratégie pour obtenir le contrat signé (c'est en effet plus intéressant, la distinction se traduit ici de manière naturelle) :

$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle (s,B)\rangle\rangle\Diamond knows(q,(s,B),\mathrm{Sig}_a^-(c))$$

En fait ce que l'on veut exactemenent dire (mais ici c'est la même chose, du fait que la connaissance des participants ne peut qu'augmenter) c'est que, a partir du moment où (s,A) possède le contrat signé, et *à tout moment à partir de là* (s,B) possède une stratégie pour obtenir le contrat signé :

$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle\rangle\rangle\Box\langle\langle (s,B)\rangle\rangle\Diamond knows(q,(s,B),\mathrm{Sig}_a^-(c))$$

Bien sûr, comme dans le cas de la *completeness*, on peut passer du point de vue des agents à celui des participants :

$$\langle\langle\rangle\rangle\Box knows(q,a,\mathrm{Sig}_b^-(c)) \Rightarrow \neg\langle\langle a\rangle\rangle\Box\neg knows(q,b,\mathrm{Sig}_a^-(c))$$
$$\langle\langle\rangle\rangle\Box knows(q,a,\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle b\rangle\rangle\Diamond knows(q,b,\mathrm{Sig}_a^-(c))$$

et on peut même effectuer diverses combinaisons :

$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \neg\langle\langle(s,A)\rangle\rangle\Box\neg knows(q,b,\mathrm{Sig}_a^-(c))$$
$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle b\rangle\rangle\Diamond knows(q,b,\mathrm{Sig}_a^-(c))$$
$$\langle\langle\rangle\rangle\Box knows(q,a,\mathrm{Sig}_b^-(c)) \Rightarrow \neg\langle\langle a\rangle\rangle\Box\neg knows(q,(s,B),\mathrm{Sig}_a^-(c))$$
$$\langle\langle\rangle\rangle\Box knows(q,a,\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle(s,B)\rangle\rangle\Diamond knows(q,(s,B),\mathrm{Sig}_a^-(c))$$

On peut aussi vouloir préciser, par exemple, que (s,B) a une stratégie pour obtenir le contrat signé, tout en restant honnête :

$$\langle\langle\rangle\rangle\Box knows(q,(s,A),\mathrm{Sig}_b^-(c)) \Rightarrow \langle\langle(s,B)\rangle\rangle\Diamond\left(\begin{array}{l} corrupt(q,(s,B),honest) \\ \wedge knows(q,(s,B),\mathrm{Sig}_a^-(c)) \end{array}\right)$$

La seconde forme de *fairness* concerne le cas ou l'un des participants a reçu un *abort token* du TTP. Il faut alors montrer que si ce participant est honnête, il est désormais impossible pour un autre participant d'obtenir un contrat signé valide. Une possibilité est donc, par exemple, de dire que si (s,A) a reçu une marque d'abandon, alors il est impossible pour (s,B) d'obtenir un contrat signé valide :

$$\begin{array}{l}\langle\langle\rangle\rangle\Box\ knows(q,(s,A),\mathrm{Sig}_t^-(\mathrm{Sig}_a^-(\langle c,\langle a,\langle b,\mathrm{abort}\rangle\rangle\rangle))) \\ \quad\Rightarrow \neg\langle\langle(s,B)\rangle\rangle\Diamond(corrupt(q,(s,A),honest)\wedge knows(q,(s,B),\mathrm{Sig}_a^-(c)))\end{array}$$

Bien entendu, on peut également voir cela au niveau des participants :

$$\begin{array}{l}\langle\langle\rangle\rangle\Box\ knows(q,a,\mathrm{Sig}_t^-(\mathrm{Sig}_a^-(\langle c,\langle a,\langle b,\mathrm{abort}\rangle\rangle\rangle))) \\ \quad\Rightarrow \neg\langle\langle b\rangle\rangle\Diamond(corrupt(q,a,honest)\wedge knows(q,b,\mathrm{Sig}_a^-(c)))\end{array}$$

Une forme plus forte consiste à dire que (s,A) possède une stratégie lui permettant d'empêcher les autres participants d'obtenir un contrat signé valide :

$$\begin{array}{l}\langle\langle\rangle\rangle\Box\ knows(q,(s,A),\mathrm{Sig}_t^-(\mathrm{Sig}_a^-(\langle c,\langle a,\langle b,\mathrm{abort}\rangle\rangle\rangle))) \\ \quad\Rightarrow \langle\langle(s,A)\rangle\rangle\Box(corrupt(q,(s,A),honest)\wedge\neg knows(q,(s,B),\mathrm{Sig}_a^-(c)))\end{array}$$

et au niveau des participants, cela donne :

$$\begin{array}{l}\langle\langle\rangle\rangle\Box\ knows(q,a,\mathrm{Sig}_t^-(\mathrm{Sig}_a^-(\langle c,\langle a,\langle b,\mathrm{abort}\rangle\rangle\rangle))) \\ \quad\Rightarrow \langle\langle a\rangle\rangle\Box(corrupt(q,a,honest)\wedge\neg knows(q,b,\mathrm{Sig}_a^-(c)))\end{array}$$

La troisième forme de *fairness* énonce que chaque participant honnête peut arriver au bout du protocole (sous l'hypothèse que les canaux de communication sont faiblement malhonnête, ce qui implique une collaboration de l'intrus). Pour l'agent (s,A), ceci donne :

$$\langle\langle(s,A),I\rangle\rangle\Diamond(corrupt(q,(s,A),honest)\wedge corrupt(q,I,honest)\wedge finish(q,(s,A)))$$

ce qu'on peut aussi énoncer au niveau des participants :

$$\langle\langle a,i\rangle\rangle\Diamond(corrupt(q,a,honest)\wedge corrupt(q,i,honest)\wedge finish(q,(s,A)))$$

6.4 *TTP accountability* et *abuse-freeness*

La propriété de *TTP accountability* (par rapport à un agent (s, A) par exemple) énonce que si (s, A) a été lésé à cause d'une mauvaise action de (s, T), alors il est possible de prouver que (s, T) a mal agit. A notre connaissance, la propriété *TTP accountability* n'a pas encore été formellement spécifiée en utilisant *ATL*. Cette définition constitue une première proposition. Tout d'abord mettons-nous d'accord sur ce qu'est une preuve d'un mauvais comportement de (s, T) en convenant qu'il s'agit d'un couple formé de deux messages contradictoires de (s, T), ici : une marque d'abandon et une signature obtenue par conversion. On dira donc que l'agent (s, A) peut prouver le mauvais comportement de T lorsqu'il est capable de construire ces deux messages :

$$proof(q, (s, A), (s, T)) = knows(q, (s, A), \langle \mathrm{Sig}_a^t(c), \mathrm{Sig}_t^-(\langle c, \langle a, \langle b, \mathrm{abort}\rangle\rangle\rangle)\rangle)$$

Dans une forme faible de cette propriété, on montre que si (s, T) est honnête, alors il est impossible pour (s, A) de prouver un mauvais comportement de (s, T) :

$$\langle\langle\rangle\rangle \Box corrupt(q, (s, T), honest) \Rightarrow \neg\langle\langle (s, A)\rangle\rangle \Diamond proof(q, (s, A), (s, T))$$

et, bien sur, cette propriété peut aussi s'énoncer au niveau des participants :

$$\langle\langle\rangle\rangle \Box corrupt(q, t, honest) \Rightarrow \neg\langle\langle a\rangle\rangle \Diamond proof(q, a, t)$$

Ensuite, il faut formaliser le fait pour un agent d'être lésé. On peut par exemple décréter que (s, A) est lésé si et seulement si (s, B) a obtenu le contrat signé et qu'il est impossible pour (s, A) d'en faire autant :

$$cheated(q, (s, A)) = knows(q, (s, B), \mathrm{Sig}_a^-(c)) \wedge \neg\langle\langle (s, A)\rangle\rangle \Diamond knows(q, (s, A), \mathrm{Sig}_b^-(c))$$

(mais on pourrait tout aussi bien imaginer utre chose, le mieux est donc de considérer cette notion comme un paramètre). Ceci nous permet alors de définir formellement une forme faible de *TTP accountability* :

$$\langle\langle\rangle\rangle \Box cheated(q, (s, A)) \Rightarrow \neg corrupt(q, (s, T), honest)$$

(*ie* si (s, A) est lésé, alors (s, T) ne s'est pas comporté honnêtement). On peut bien sur vouloir que le participant lésé puisse prouver le mauvais comportement de (s, T) :

$$\langle\langle\rangle\rangle \Box cheated(q, (s, A)) \Rightarrow \langle\langle (s, A)\rangle\rangle \Diamond proof(q, (s, A), (s, T))$$

Finalement, la propriété d'*abuse-freeness* (pour l'agent (s, A) par exemple) dit qu'il est impossible pour (s, A), a n'importe quel point du protocole, de prouver à un observateur extérieur qu'il a la possibilité de déterminer l'issue du contrat. Convenons qu'une telle preuve consiste en un couple constitué d'un contrat signé valide et d'une marque d'abandon :

$$proof_ext(q, (s, A)) = knows(q, (s, A), \langle \mathrm{Sig}_b^-(c), \mathrm{Sig}_t^-(\langle c, \langle a, \langle b, \mathrm{abort}\rangle\rangle\rangle)\rangle)$$

alors cette propriété peut être spécifiée de la manière suivante :

$$\langle\langle\rangle\rangle\Box\neg\langle\langle(s,A)\rangle\rangle\Diamond proof(q,(s,A))$$

et bien entendu la forme correspondante au niveau des participants est :

$$\langle\langle\rangle\rangle\Box\neg\langle\langle a\rangle\rangle\Diamond proof(q,a)$$

7 Conclusion et développements ultérieurs

On a proposé ici une modélisation, assez flexible, des protocoles de signatures de contrats dans le cadre des systèmes de transitions alternés et on a utilisé la logique *ATL* afin de modéliser les propriétés attendues de ces protocoles. Dans notre modélisation, nous avons pris en compte différents types possibles de comportements pour les participants du protocole, ainsi que pour les canaux de communications. Cette prise en compte se fait, non pas en faisant des hypothèses *a priori* et en construisant le modèle associé comme c'est l'usage, mais à l'intérieur même du protocole. On a également vu qu'il était possible de préciser les niveaux d'honnêté des participants attendus à l'intérieur des formules à vérifier.

Les points suivants mériteraient d'être développés. Tout d'abord une implémentation permettrait de faire la distinction entre les différentes formes données pour chaque propriété. Cela pourrait aussi permettre de vérifier que la modélisation des certaines propriétés (en particulier *TTP accountability* et *abuse-freeness*) n'est pas trop naïve, ou au contraire trop restrictive. Enfin, il semble aussi que la modélisation que l'on a fait en utilisant des *ATS* et *ATL* n'est pas aussi adéquate qu'on pourrait le croire. En effet, si l'on regarde la définition de ce qu'est une stratégie pour un agent, on constate qu'il n'est *a priori* pas exclu qu'un agent prenne des décisions en consultant l'état local des autres agents. Ceci est assez peu réaliste, dans le cadre des protocoles cryptographiques. Pour remédier à ce problème, il semble possible de se placer dans le cadre des systèmes de transitions alternés avec information incomplète ([1]).

Références

[1] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Lecture Notes in Computer Science*, 1536 :23–60, 1998.

[2] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 86–99, May 1998.

[3] R. Chadha, M. Kanovich, and A. Scedrov. Inductive methods and contract-signing protocols. In P. Samarati, editor, *Proceedings of the 8th ACM Conference on Computer and Communications Security*. ACM Press, Nov. 2001.

[4] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2) :198–208, 1983.

[5] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In M. Wiener, editor, *Advances in Cryptology – CRYPTO ' 99*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1999.

[6] N. Heintze, J. D. Tygar, J. Wing, and H. C. Wong. Model checking electronic commerce protocols. In *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 147–164, Oakland, California, Nov. 1996. USENIX.

[7] S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *CONCUR : 12th International Conference on Concurrency Theory*. LNCS, Springer-Verlag, 2001.

[8] V. Shmatikov and J. C. Mitchell. Analysis of abuse-free contract signing. *Lecture Notes in Computer Science*, 1962 :174–? ?, 2001.

[9] V. Shmatikov and J. C. Mitchell. Finite-state analysis of two contract signing protocols. *Theoretical Computer Science*, 2002. (To appear).