Numéro d'ordre 011 Année 2003

Version du 9 Avril 2010

THESE

préparée à

L'UNIVERSITE DE TECHNOLOGIE DE BELFORT-MONTBELIARD

pour obtenir le grade de

Docteur de l'Université de Technologie de Belfort-Montbéliard
et l'Université de Franche-Comté

spécialité AUTOMATIQUE ET INFORMATIQUE

par

Nathanaël COTTIN

intitulée

Contribution à la sécurisation des échanges électroniques en environnement réparti objet

soutenue publiquement le 15 décembre 2003

Composition du jury :

Président :	**Abderrafiâa KOUKAM**, Professeur, UTBM, Belfort
Rapporteurs :	**Pierre-Yves CUNIN**, Professeur, CICG, Grenoble
	Thierry DIVOUX, Professeur, Faculté des Sciences, Nancy
Examinateur :	**Jaafar GABER**, Maître de Conférences, UTBM, Belfort
Directeur de thèse :	**Abdellah EL MOUDNI**, Professeur, UTBM, Belfort
Co-encadrant :	**Maxime WACK**, Maître de Conférences, UTBM, Belfort
Invité :	**Thierry PIETTE-COUDOL**, Avocat, Paris

Remerciements

Le présent travail a été mené au sein du laboratoire Systèmes et Transports (SeT) de l'UTBM sous la direction de monsieur Abdellah EL MOUDNI, pour la partie scientifique, en collaboration avec la société GED Software, spécialisée dans la gestion électronique de documents (GED), présidée par monsieur Jean-Claude MONNIER, pour l'industrialisation des résultats scientifiques obtenus.

Je leur adresse ma profonde gratitude pour m'avoir permis d'intégrer leurs structures respectives dans le cadre de mon contrat CIFRE.

Je remercie vivement mon directeur de thèse, monsieur EL MOUDNI, Professeur à l'Université de Technologie de Belfort-Montbéliard (UTBM), ainsi que mon co-encadrant, monsieur Maxime WACK, Maître de Conférences, pour la qualité de leur encadrement. Tous deux m'ont initié à la recherche et m'ont guidé tout au long de l'élaboration de ce travail. Ils m'ont encouragé à m'impliquer dans leurs thématiques de recherche relatives à la sécurité des systèmes d'information.

Nous sommes particulièrement sensibles à l'honneur que nous fait monsieur le Professeur Abderrafiâa KOUKAM en acceptant de présider le jury.

Nous remercions chaleureusement messieurs les Professeurs Pierre-Yves CUNIN et Thierry DIVOUX pour avoir accepté la charge de travail qu'implique la lecture critique du présent mémoire. Leurs remarques pertinentes et conseils avisés nous ont amené à de nouvelles réflexions et perspectives de recherche.

Monsieur Jaafar GABER, Maître de Conférences à l'UTBM, nous a conseillé au cours de l'élaboration et la révision de ce travail. Nous lui exprimons notre profonde reconnaissance.

Les recherches menées étant en étroite relation avec les contextes juridiques européen et français, je remercie Maître Thierry PIETTE-COUDOL, président fondateur du comité IALTA France et avocat près la Cour d'Appel de Paris. Ses connaissances tant techniques que juridiques m'ont régulièrement aidé à appréhender la législation qui se construit peu à peu depuis 1999 autour de la signature électronique.

Je remercie également les enseignants et administratifs de l'UTBM ainsi que mes collègues de travail du groupe Khephren pour leur chaleureux accueil. Ils m'ont permis d'effectuer mes recherches dans une ambiance conviviale.

Monsieur Sofian AZZABI, doctorant à la faculté de Droit, des Sciences Economiques et de Gestion de Nancy II, m'a en outre été d'un immense secours et je l'en remercie à cet égard.

Enfin, mes parents, ma sœur, ma fiancée et ses parents m'ont supporté tout au long de ces trois années et ont volontiers relu mon mémoire avec l'œil du néophyte.

A mes parents

A ma soeur

A ma fiancée

Contexte général

A l'aube de l'explosion des échanges électroniques, que ce soit entre partenaires commerciaux (B2B) ou avec la participation des consommateurs (B2C), la sécurité des systèmes d'information revêt une importance capitale et sans cesse croissante. Non seulement cette sécurité doit être mise en place au sein des entreprises, elle doit également s'étendre aux communications entre commerçants et foyers, de plus en plus sujets aux attaques de personnes malveillantes.

Les environnements informatiques n'ont cessé de s'adapter à la fois aux contraintes qu'imposent les géants de l'informatique ainsi qu'aux besoins liés à la sécurité de l'information. Une protection physique apporte une réponse partielle à cette problématique de sécurité essentielle. Elle doit de fait être complétée par une sécurisation logique de l'information, par l'emploi de systèmes informatiques eux-mêmes sécurisés, fiables et ouverts aux évolutions futures.

Ce n'est qu'en prenant en compte l'information au plus tôt que la garantie de sa sécurité est acquise. Aucune faille au cours de son utilisation ne peut être tolérée afin de ne pas compromettre l'intégrité globale du système.

Le présent travail s'attache de fait à présenter un modèle de système informatique permettant de répondre aux contraintes logiques liées à l'utilisation de l'information numérique en amont de sa transmission et en aval de sa réception. L'objectif est de proposer une solution à la fiabilisation des échanges en garantissant l'intégrité de l'information reçue, l'authenticité du message véhiculé et la non répudiation de l'échange. En effet, dans ce dernier cas l'émetteur ne doit pouvoir nier être à l'origine de l'envoi de l'information et le récepteur ne doit être en mesure de feindre d'avoir effectivement reçu le message en se prévalant d'une quelconque erreur de transmission de l'information.

Contexte spécifique

Les systèmes d'information évolués bénéficient des technologies orientées objet aptes à gérer leur adaptation aux évolutions tant techniques qu'organisationnelles. Les architectures réparties objet offrent notamment une base solide et éprouvée pour concevoir de tels systèmes.

Le modèle proposé est par conséquent organisé autour des concepts préconisés par les systèmes répartis objet : l'abstraction des services proposés et des traitements réalisés, la réutilisation et la spécialisation des composants, l'interopérabilité entre environnements et plates-formes hétérogènes. Il repose essentiellement sur les concepts de l'environnement réparti objet CORBA (« Common Object Request Broker Architecture ») de l'OMG (« Object Management Group »). Cet environnement met en avant des besoins génériques, communs à l'ensemble des systèmes actuels et notamment la nécessaire mise en place d'un service de sécurité chargé de gérer la sécurisation des échanges.

Ce modèle est employé pour mettre en œuvre un ensemble complexe faisant appel à divers services liés à la reconnaissance juridique de la signature électronique. Cette dernière peut être considérée dans un premier temps comme l'équivalent de la signature manuscrite transposée au monde numérique. Il est en effet possible aujourd'hui de signer électroniquement des informations numériques, qu'elles soient numérisées par l'utilisation d'un scanner par exemple (image fidèle) ou dématérialisées, c'est-à-dire n'ayant qu'une existence numérique (image exacte).

Le déploiement de ce modèle pour concevoir un ensemble multi-tiers dans lequel cohabitent des services complémentaires permet d'attribuer puis conserver la valeur juridique des informations échangées entre les utilisateurs des systèmes de demain. Cette prise en compte d'un contexte juridique nouveau induit une réflexion en amont sur le format des échanges et des informations échangées. En effet, les systèmes informatiques doivent dès maintenant prendre en charge la directive européenne 99/93/CE [PEC 99] ainsi que les divers décrets et lois français [SEN 00] [SEN 01b] lors des échanges d'information revêtant un caractère juridique.

Pour cela, l'étude de la signature électronique et de ses représentations par l'informatique est une étape essentielle à la mise en oeuvre d'un modèle industrialisable car mettant en adéquation d'une part les contraintes techniques liées à la création et la conservation dans le temps des documents signés et d'autre part les besoins juridiques, notamment de fiabilité de l'information et des signatures électroniques.

Table des matières

Introduction générale

La mondialisation des échanges sous forme numérique s'accompagne d'une évolution des technologies de l'information qui ont dû s'adapter aux nouveaux besoins. En effet, les utilisateurs exigent aujourd'hui des protections juridiques et techniques lorsqu'ils transmettent des informations sur les réseaux publics.

Les applications ont peu à peu abandonné le modèle monolithique pour adopter des modèles construits autour de composants chargés de répondre à des services. De fait, les environnements et composants répartis faisant appel à des processus interagissant à distance, en particulier les environnements répartis objet, ont connu un essor considérable ces dernières années.

D'autre part, les besoins de protection, aussi bien immédiate qu'à long terme d'une information toujours plus volatile ont été sujets à de nombreuses recherches. La sécurité des systèmes d'information, faisant appel aux techniques de cryptographie et à la supervision des systèmes, est aujourd'hui un enjeu déterminant dans la pérennité et la fiabilité de l'information authentique.

Cette authenticité, fruit de la reconnaissance juridique de l'information numérique signée électroniquement, a contribué à l'essor de la signature électronique dans le monde du droit, à tel point que certaines formalités administratives (téléTV@, déclaration d'imposition, facture électronique) en font déjà un médium privilégié permettant d'établir la confiance sur Internet. Cet engouement mondial et français s'est d'ailleurs traduit par l'adoption récente d'un décret [SEN 03] relatif à la facture électronique d'une part et à l'active participation de groupes de travail isolés ou coordonnés par le Ministère des Finances.

En dernier lieu, les systèmes répartis de demain doivent, à la rencontre de ces technologies, proposer des solutions ouvertes, liées non seulement à la sécurité des échanges mais également à leur contexte juridique. C'est pourquoi nous présentons un modèle d'environnement réparti objet reposant sur la cryptographie à clé publique et la signature électronique d'une part et ouvert sur l'audit et la supervision des échanges d'autre part.

De fait, les échanges d'informations entre objets répartis sont intrinsèquement authentifiés et leur confidentialité préservée par le modèle suggéré. Un mécanisme de gestion des accès aux méthodes que mettent à disposition les objets répartis permet de déployer une stratégie de sécurité adaptée aux besoins des utilisateurs. En sus de la sécurité qu'apporte la signature électronique, un système d'audit et de surveillance tel que celui proposé par [COT 00] peut être intégré au modèle afin de conforter l'établissement de la preuve d'existence des échanges.

Le premier chapitre est consacré à la présentation générale de la signature électronique et la sécurité dans les systèmes répartis. Nous y développons particulièrement la gestion sécuritaire au sein de l'environnement réparti objet

CORBA de l'OMG [OMG 97b], qui servira de point d'ancrage à notre proposition de modèle.

Le second chapitre étudie le thème de la signature électronique telle que proposée dans les standards mis en œuvre aujourd'hui en faisant cependant abstraction des différentes technologies d'apposition des signatures (signature scannée, biométrique, par mot de passe notamment). Les réflexions qui en résultent nous mènent à la définition d'un nouveau format de signature électronique, plus adapté aux besoins mis en avant par les juristes, dont la validation repose non seulement sur les techniques traditionnelles mais également sur la notarisation électronique. Afin de permettre une validation au cours du temps de la signature électronique, la signature doit être datée de manière certaine tout en permettant de détecter les fraudes liées à l'anté- ou à la post-datation. Pour garantir l'exactitude de cette date, nous proposons de recourir à un protocole d'horodatage multiple de la signature. Les horodatages obtenus prennent part à la construction d'une signature « sécurisée ».

Le troisième chapitre prend en considération les protocoles et structures de données définies dans les deux précédents chapitres afin d'établir un modèle de système réparti objet reposant sur la signature électronique et répondant à la problématique de la sécurisation des échanges. Ce modèle suggère en effet un moyen de sécuriser et authentifier les communications entre objets répartis valable même dans le cas où un canal de communication non sécurisé (tel que l'Internet) est utilisé. L'emploi des formats de données et protocoles suggérés apporte aux utilisateurs l'assurance du respect du contexte juridique relatif à la signature électronique. Le modèle permet de plus de réaliser la surveillance des interactions entre objets, préalablement étudiées par [COT 00] et propose également une ouverture sur un système d'audit. Les structures des données relatives à la signature électronique tiennent une part importante dans la définition de ce modèle. En effet, elles servent à authentifier les objets participant aux échanges par le biais d'un format d'échange approprié.

Ce modèle est enfin intégré au sein d'une architecture répartie à grande échelle permettant de répondre à l'ensemble des services nécessaires à la reconnaissance de la validité d'une signature électronique par les juristes. Ce dernier chapitre traite de l'utilisation du modèle proposé pour la mise en œuvre d'un ensemble de tiers de confiance [IAL 98] qui proposent des services complémentaires permettant de gérer les documents signés électroniquement. Ils garantissent la mise en œuvre effective de la « Chaîne de Confiance », depuis la création et la signature de l'écrit sous forme numérique jusqu'à sa conservation sécurisée sur le long terme[1].

[1] Du moment que cet écrit n'a cessé de produire ses effets juridiques. Dans le cas contraire, la conservation *intelligible* [PIE 02a] revêt importance réduite.

CHAPITRE 1 : SECURITE DES SYSTEMES REPARTIS OBJET

« Pourquoi faire simple quand on peut faire compliqué ?! »

Les devises Shadok

Les systèmes répartis sont une composante essentielle dans le processus de développement et déploiement d'applications soumises à des contraintes d'hétérogénéité, d'interopérabilité et de sécurité. Actuellement, la sécurité repose à la fois sur le respect des politiques de sécurité au sein des entreprises ainsi que sur l'utilisation de procédés techniques tels que la cryptographie et la signature électronique. Après une succincte présentation de la cryptographie actuelle et de la signature électronique d'un point de vue juridique, nous étudierons les procédés techniques de sécurité dans les principaux systèmes répartis actuels. L'accent est mis sur l'architecture CORBA issue des travaux de l'OMG [OMG 97b].

Ce chapitre pose ainsi les bases nécessaires à l'étude plus approfondie de la signature électronique ainsi qu'à la définition d'un modèle réparti objet proposant un environnement de développement dont la sécurité et le respect des règles juridiques sont les composantes maîtresses.

1. Présentation des systèmes répartis objet

Le partage de l'information est un élément fondamental dès lors que les échanges d'informations numériques au niveau mondial se multiplient. Les systèmes répartis objets sont incontournables lorsqu'il s'agit de concevoir des applications faisant appel à diverses sources de données et de traitements éloignées géographiquement. En effet, les concepts de portabilité, interopérabilité, coopération et partage des ressources sont devenus les clés des applications informatiques [KRA 93].

Parmi les principaux « middlewares » [MOW 95] objet, ou environnements répartis basés sur l'objet, nous pouvons citer CORBA [GEI 99] de l'OMG, DCOM [TOG 99] de l'Open Group, OLE-2 [GAR 96] de Microsoft et JAVA RMI de Sun [GRO 01], tous devenus incontournables. Ces logiciels intermédiaires proposent un ensemble d'outils chargés de simplifier le développement des applications réparties objet en masquant les communications réseau.

Ce chapitre introduit cette notion de système réparti objet avant de présenter plus en détail l'architecture CORBA qui servira de base au système réparti proposé.

1.1. Approche générale

Nous désignons par « middleware », dans le cadre de l'informatique répartie, l'ensemble des couches logicielles permettant à des applications de communiquer à distance. En particulier, ces derniers s'attachent à masquer les couches réseau et transport ainsi que l'hétérogénéité des environnements [BAL 00].

Les systèmes répartis peuvent ainsi être considérés comme l'extension du modèle 3-tiers vers un modèle multi-tiers [ORF 95], plus apte à gérer la qualité de service (QoS) des applications ouvertes sur l'extérieur en offrant notamment des procédés de tolérance aux pannes et de répartition de charge [COT 00].

Un système réparti est ainsi composé d'un ensemble de processus qui s'exécutent sur des sites reliés par un réseau de communication. Le fonctionnement global du système repose sur la coopération entre processus par le biais de l'envoi et le traitement de messages selon le principe d'appel de procédure à distance (ou « Remote Procedure Calls », RPC) [GAR 96].

La technologie RPC permet à un processus d'appeler une procédure déclarée dans l'interface d'un processus distant tout en masquant les transmissions via le réseau de communication. Cette abstraction est rendue possible grâce à des souches client (« stub ») et serveur (« skeleton ») insérées entre les processus et le réseau (figure 1). L'appel de la procédure déclenche une succession d'actions qui aboutissent à la génération d'un résultat retourné au processus appelant.

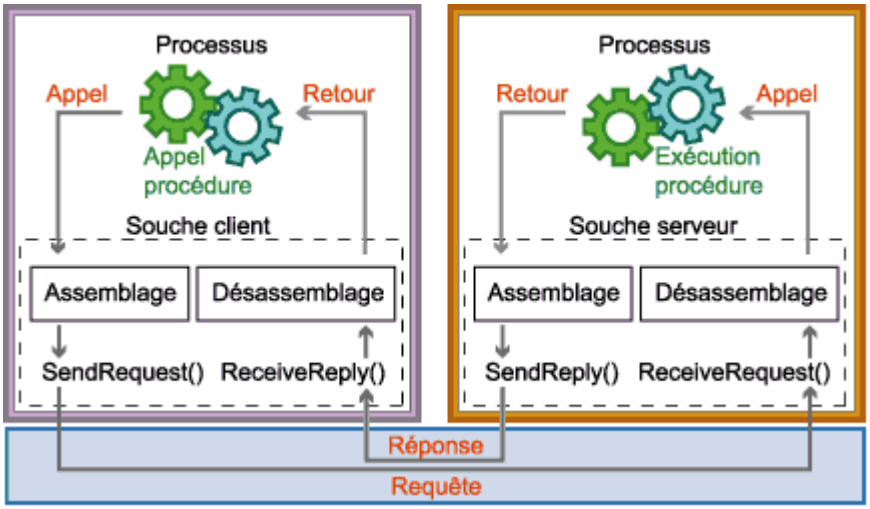

Figure 1 : fonctionnement du RPC

Selon les liens existant entre processus, on opère une distinction entre processus de type client, serveur et mixte [COT 00].

1.1.1. Processus client

Ces processus peuvent être vus comme les initiateurs et terminateurs des communications. Ils sont à l'origine des requêtes soumises au système réparti et reçoivent les réponses finales de la part du système. Ces processus ne disposent pas d'interface car ils ne peuvent être eux-mêmes appelés.

1.1.2. Processus serveur

Un processus de type serveur est chargé d'attendre et de répondre aux requêtes des processus client. Ils peuvent être dupliqués sur plusieurs localités voisines ou distantes afin de répondre au mieux aux clients (et ainsi maintenir la qualité de service – QoS – requise [MIC 03]) dès lors que le système global gère les accès concurrents et met en place de la répartition de charge. Chaque processus serveur publie son interface afin d'indiquer aux éventuels clients les méthodes qu'il supporte (figure 2).

Un même processus serveur peut publier plusieurs interfaces destinées à fournir différentes vues aux utilisateurs. Cette propriété est cependant peu supportée. Il suffit en effet de disposer d'un serveur par interface pour obtenir un fonctionnement équivalent à un unique serveur. De plus les accès concurrents aux services proposés sont facilités par l'interrogation de plusieurs serveurs.

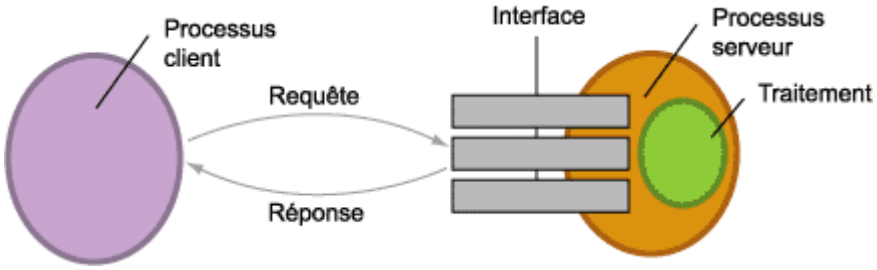

Figure 2 : appel d'un processus serveur par le biais de son interface

1.1.3. Processus mixte

Ces processus, désignés par le terme « délégateurs » par [COT 00], combinent les propriétés des processus client et serveur. Ils sont à l'origine de la coopération entre processus et peuvent être considérés comme des processus intermédiaires apportant une part des traitements nécessaires à la construction de la réponse à une requête.

Ils combinent par le fait à la fois une partie serveur chargée de répondre aux demandes émanant des clients et une partie client susceptible de faire appel à d'autres processus serveur (ou mixte).

1.2. Notion d'IDL

Les processus de type serveur peuvent être invoqués par le biais de leur interface. Cette dernière propose la liste des méthodes disponibles en indiquant en particulier leur signature (au sens de la programmation objet) afin que les processus client soient en mesure d'élaborer leurs appels correctement.

Ainsi, interface et traitements réalisés sont séparés. En effet, peu importe au client les traitements effectués en interne par le processus appelé, seul compte le résultat retourné. Généralement (RMI étant un cas particulier), l'interface doit être compréhensible indépendamment du langage de programmation dans le but de rendre possible les communications entre processus hétérogènes.

C'est pourquoi le recours à un langage dédié, appelé « Interface Description Language » ou IDL, est quasi inévitable. Chaque modèle réparti dispose donc de son propre langage IDL : Microsoft IDL pour DCOM et OMG IDL pour CORBA par exemple. L'architecture .NET [LOW 03] de Microsoft se situe en-dehors de notre étude puisqu'elle ne fait pas appel à la notion d'interface IDL.

L'interprétation (ou « projection ») des interfaces IDL permet la génération automatique des codes sources des souches client et serveur utilisées pour réaliser les appels entre objets répartis via le mécanisme RPC exposé précédemment.

1.3. Modèles DCOM, OLE-2 et RMI

Le « Distributed Component Object Model » (DCOM) est un environnement proposé par l'Open Group en 1992. Ce système repose sur l'architecture « Distributed Computing Environment » (DCE) [TOG 97], environnement de développement et d'exécution d'applications réparties ouvert à l'initiative de l'« Open System Foundation » (OSF) qui propose un modèle gérant des règles de sécurité, ordonnancement et synchronisation. Ces règles sont mises en œuvre par des services tels que le « service de temps » chargé de synchroniser les horloges locales et le « Distributed File System » (DFS) ou gestionnaire de fichiers à distance. Les fichiers manipulés par DFS sont connus globalement sur le réseau et accessibles via un mécanisme proche des noms intelligents (« monikers ») de l'architecture COM (« Component Object Model ») [GAR 96].

D'autres systèmes à objet répartis, commercialisés (par Microsoft notamment) ou libres, tels que Java RMI et CORBA, en sont inspirés.

La seconde version d'« Object Linking and Embedding » (OLE-2) [GAR 96] est une architecture répartie proposée par Microsoft. Elle s'appuie sur le modèle COM afin d'intégrer les fonctions de persistance d'objet, gestion des noms intelligents et transfert de données entre objets. OLE-2 octroie ainsi la possibilité de créer des documents composites dont les informations sont construites d'après plusieurs sources réparties.

Le modèle RMI, à l'initiative de Sun Microsystems, tire parti de la technologie Java. Il fait notamment usage du procédé de sérialisation d'objets Java. La sérialisation permet de construire une chaîne de caractères comprenant l'ensemble des valeurs définissant l'état courant de l'objet. Cette chaîne peut alors être transmise à un objet Java distant similaire qui restituera l'état de l'objet sérialisé.

> *Note : toutes ces technologies mettent en œuvre la notion d'interface IDL précédemment citée. Dans le cas de RMI, cette interface correspond à l'interface Java implémentée (au sens de la programmation java) par le serveur. Elle n'apparaît donc pas clairement comme interface IDL.*

1.4. Architecture CORBA

La coopération entre objets répartis et hétérogènes soulève plusieurs problèmes [GAR 96] tels que l'échange d'objet d'un environnement à un autre ou l'échange des références réseau des objets. D'autres problèmes référencés par [COT 00] se posent également.

Ces problèmes sont relatifs à :

- La gestion des références des objets.
- La localisation des objets sur le réseau.
- La gestion de la répartition de la charge, de la duplication et de la migration des objets.
- L'organisation des graphes d'appels entre objets et la détection des éventuels cycles d'appels menant à des blocages.

CORBA (« Common Object Request Broker Architecture ») [OMG 97b] [DAN 00] est un environnement réparti objet non propriétaire créé en 1989 par l'OMG (« Object Management Group »), un consortium regroupant aujourd'hui plus de 900 acteurs dont plusieurs grands comptes internationaux tels que Sun, 3Com, Unisys, Hewlett-Packard, Rational Software, Canon, Phillips, American Airlines, la NASA, ainsi que des universités (dont les universités fraçaises INRIA et LIFL, cette dernière est particulièrement active avec ses contributions liées à « CorbaScript »).

L'idée directrice de l'OMG est que la mise en œuvre de logiciels permettant la création d'applications réparties sans standards ni conventions est une utopie. L'OMG est donc un organisme de standardisation et de promotion de CORBA et non de développement. L'accent est mis sur l'interopérabilité de ces applications réparties qui doivent être capables de communiquer entre elles indépendamment de leur environnement, les mécanismes tels que les sockets IP et RMI ayant montré leurs limites [GEI 99] :

- Utilisation trop bas niveau : tel est le cas de la programmation par sockets qui se situe au niveau de la couche réseau du modèle ISO/OSI [ROL 93] (voir annexes).
- Trop spécialisé : RMI est utilisé uniquement pour développer des applications à l'aide du langage de programmation Java. DCOM est quant à lui spécifique aux plates-formes Windows.

D'autres travaux comparatifs, tels que ceux menés par [NAM 03] et [JUR 00], ont également montré les atouts de CORBA, notamment par rapport à RMI.

Les recherches menées par l'OMG ont abouti à la mise au point de standards pour la conception d'applications réparties, indépendantes non seulement du langage de programmation des objets, mais également de l'hétérogénéité des machines utilisées. Ces concepts fondamentaux sont développés dans l'OMA.

1.4.1. Modèle OMA

L'OMA (« Object Management Architecture ») indique les lignes directrices suivies par les spécifications de la norme. Il est composé d'objets clients et serveurs reliés entre eux par une couche de communication reposant sur le protocole TCP/IP.

Du côté serveur, l'OMA définit trois types d'objets serveur susceptibles d'être appelés par les clients (figure 3) :

- Les objets métier (que l'OMG qualifie de « vertical facilities ») proposent des objets spécifiques à un corps de métier tel que le médical ou la finance.
- Les services standard prennent en charge la gestion des autres objets. Nous présenterons les services les plus courants par la suite.
- Les objets applicatifs sont des composants réutilisables mais non spécifiques à un corps de métiers et dont l'objectif est très ciblé : un objet applicatif de signature électronique pourra par exemple être chargé de valider les signatures que tout objet client lui présente. Un objet de traçabilité pourra quant à lui affranchir le système des contraintes d'enregistrement et de contrôle des traces générées lors de l'exécution de certains objets.

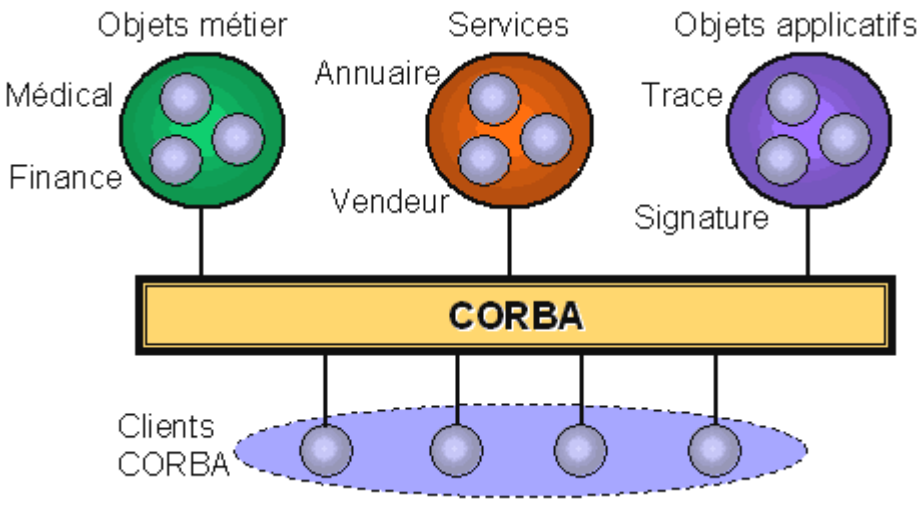

Figure 3 : architecture générale OMA de CORBA, exemples d'objets

On retrouve une fois encore la volonté, inhérente à la programmation orientée objet, de séparer interfaces et traitements étant donné que les objets définis par l'OMA mettent leurs services à disposition des clients par le biais de leur interface IDL. Les clients ne perçoivent pas les traitements réalisés sur les serveurs.

1.4.2. Au cœur de CORBA, l'ORB

CORBA repose sur un bus appelé ORB (« Object Request Broker »). Défini par l'OMG comme le noyau central de CORBA, celui-ci permet d'envoyer et recevoir des requêtes sur le réseau et de fait rend possible la communication entre des objets hétérogènes et distants [OMG 97b]. Il apparaît comme la généralisation de RPC au monde des objets.

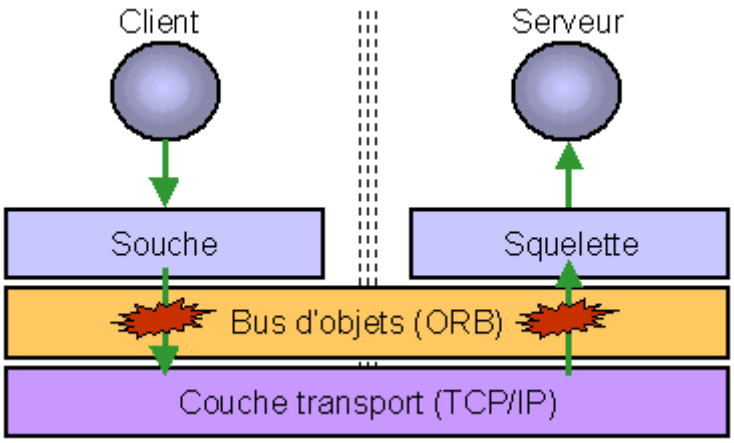

Figure 4 : assemblage et désassemblage d'une requête CORBA

Les objets dialoguent entre eux par l'intermédiaire de proxies client et serveur (dénommées respectivement « souches » et « squelettes ») qui leur masquent les communications à travers le réseau. Ces proxies assemblent (opération dite de « marshalling ») et désassemblent (« unmarshalling ») le contenu des requêtes transportées par le bus. Comme l'indique la figure 4, les objets, interconnectés à travers le bus CORBA, n'ont plus à se préoccuper des aspects de la communication entre des sites hétérogènes, aspects entièrement gérés par le bus [GEI 99].

1.4.3. Quelques services standards de CORBA

Les services standards [OMG 97a] que propose l'OMG permettent de standardiser la gestion du cycle de vie des objets : création, accès, destruction, déplacement.

Comptent parmi l'ensemble des services les plus utilisés par les concepteurs d'applications répondant à la norme CORBA :

- Le service annuaire ou de nommage (« naming service ») permet d'accéder aux objets serveur référencés. Il s'apparente aux pages blanches de l'annuaire téléphonique. La recherche s'effectue lorsque le client connaît l'espace de nommage (littéralement « namespace ») et l'interface supportée par l'objet serveur demandé.

- Le service vendeur (« trading service ») permet aux clients d'obtenir les références d'objets serveur en fonction de critères de comportement tel que le type d'interface supportée. Il peut être considéré comme un espace publicitaire pour les serveurs et fonctionne de manière similaire aux pages jaunes de l'annuaire téléphonique.

- Le service de temps (« time service ») permet de synchroniser les horloges avec une estimation de l'incertitude inférieure à un seuil donné. Ce service permet de gérer une horloge globale au sein des applications réparties.

- Le service de transaction (« transaction service ») propose des moyens de s'assurer de la complétude des traitements effectués et s'apparente au paradigme développé pour les transactions faisant appel à des bases de données.

- Le service cycle de vie (« life cycle service ») présente les interfaces nécessaires à la mise au point de la migration des objets serveur et mixte.

- Enfin, l'OMG prend depuis peu en considération la sécurité au sein de son environnement réparti avec la définition d'un service dédié à la sécurité. Le service sécurité (« security service ») propose ainsi une architecture globale permettant d'obtenir divers niveaux de sécurité lors de l'utilisation des objets serveur et mixte.

Par la suite, nous nous intéresserons plus particulièrement à la présentation critique du service sécurité de CORBA dans le but d'en proposer une alternative reposant sur la cryptographie pour s'assurer de la confidentialité des échanges d'une part, et sur la signature électronique afin d'authentifier les protagonistes impliqués dans ces échanges d'autre part.

2. Sécurité des systèmes répartis objet

Au-delà des moyens physiques et logiques mis en place au sein des entreprises et dirigés par un plan de sécurité, les systèmes répartis objet sont particulièrement sujets aux attaques de personnes mal intentionnées. Ces dernières peuvent souhaiter rendre le système inutilisable (c'est le cas des attaques par refus de service distribué ou « Distributed Denial of Service », DDoS) ou cherchent à percer la sécurité afin de s'introduire frauduleusement dans le système. Les principaux problèmes de sécurité relevant directement du fait des appels distants entre objets dans divers environnements (qui peuvent être hétérogènes) demeurent communs aux échanges tels que ceux rencontrés sur Internet. Il s'agit en particulier de l'usurpation d'identité (active ou passive), de l'obtention frauduleuse de droits d'accès par objet mixte interposé et du passage des murs pare-feu (firewalls) [KAE 00].

La sécurité peut être définie au niveau des couches 4 et 5 (respectivement « session » et « transport ») du modèle OSI [ROL 93] :

- L'étude de la sécurité au niveau des couches basses a permis de mettre en œuvre des protocoles réseau tels que IPSec [KEL 03] [KAE 00] et TLS [DIE 99]. D'autres projets se sont également intéressés à la sécurisation de la couche transport. Par exemple, le projet ARCADE (« Architecture de Contrôle Adaptative Des Environnements IP) [ACA 00], piloté par le LIP6 et l'INRIA, s'intéresse à l'Internet de demain en dressant un modèle de sécurité sur les réseaux IP.

- Dans le cadre des systèmes répartis, nous nous intéressons davantage à la sécurisation de l'information au niveau de la couche « transport » (se référer au modèle de sécurité OSI annexé), obtenue notamment grâce à deux technologies : la première est le recours à la cryptographie, qu'elle soit à clé secrète ou à clé publique ; la seconde est l'utilisation de la signature électronique.

2.1. Modèle de base

Un modèle sécuritaire de base est proposé par [JAW 01]. Il repose sur les interactions entre un fournisseur de ressources et un fournisseur de services de sécurité afin de fournir des services aux clients du système tout en prévenant des attaques de pirates telles que la corruption de données, l'accès non autorisé, la substitution d'information et le refus d'accès (« Denial of Service », DoS).

Nous avons modifié le modèle de base de [JAW 01] afin de tenir compte des particularités des objets répartis. Le modèle résultant de cette évolution est illustré par la figure 5 : les applications apparaissent comme composées d'un ensemble d'objets répartis qui interagissent pour fournir des services.

La gestion de la sécurité est assurée au niveau de chaque objet par un fournisseur de services de sécurité (FSS) en relation avec un fournisseur de ressources (données et objets répartis). Il peut aussi bien s'agir d'un FSS monolithique, offrant une gestion centralisée de la sécurité, qu'un FSS réparti dont chaque composante prend en charge un groupe d'objets ou un objet unique.

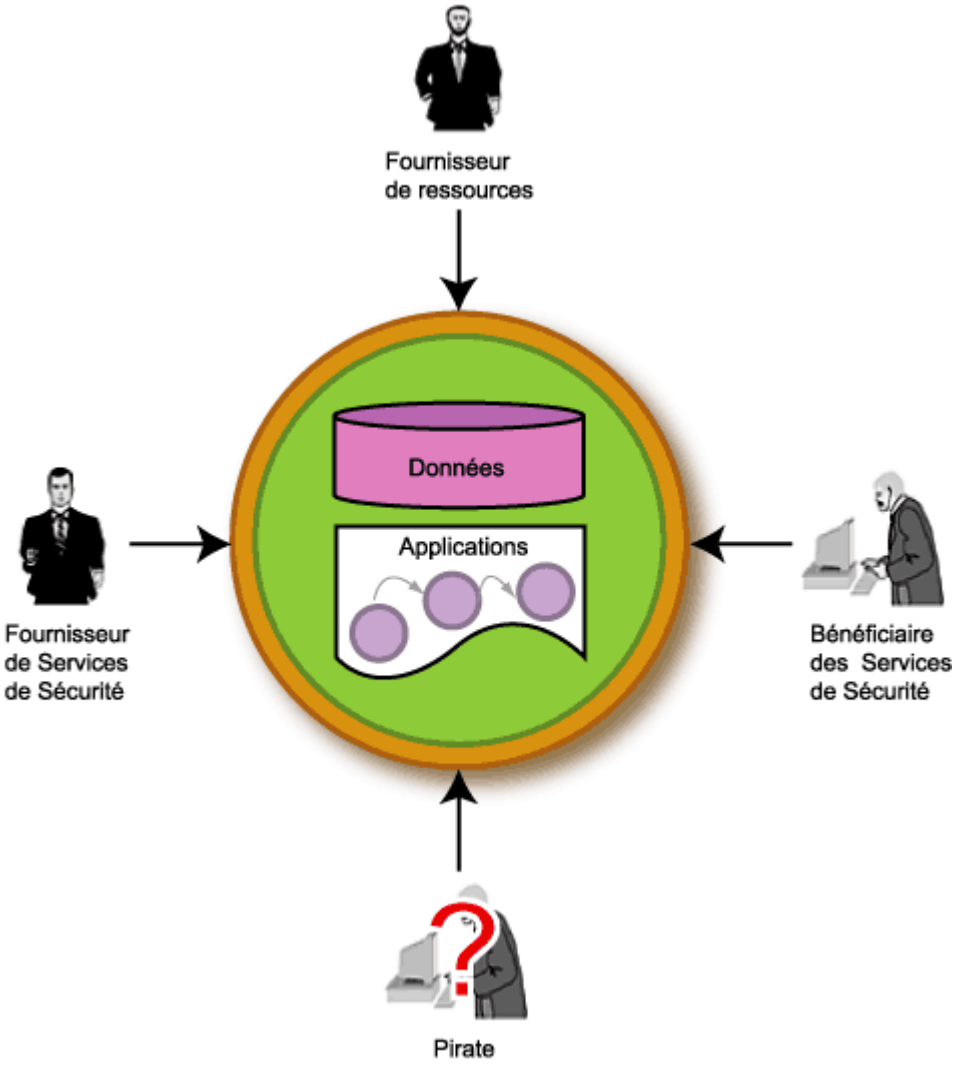

Figure 5 : modèle de sécurité dans les systèmes répartis

Nous nous intéressons précisément au fournisseur de services de sécurité (FSS). Ce dernier tente de protéger les ressources (données et applications) en contrant les attaques de pirates [JAW 01] qui peuvent être externes au système ou identifiés comme bénéficiaires des services de sécurité (BSS).

2.1.1. Services proposés par le FSS

Le FSS offre des services de protection tels que :

- La vérification de l'intégrité des messages échangés, des données et des applications : le bénéficiaire (ou client) doit pouvoir s'assurer que les informations et codes exécutables auxquels il accède sont exempts de manipulations de la part du pirate.

- La préservation de la confidentialité : le client dispose de moyens permettant de garantir qu'une tierce partie non autorisée (le pirate en particulier) ne puisse être en mesure de prendre connaissance des informations dès lors qu'il ne dispose pas d'une donnée secrète permettant de les obtenir.

- La permission d'accès permet au fournisseur de ressources (FR) de restreindre l'accès aux ressources qu'il met à disposition de ses clients. Le FSS prend alors en charge la validation des permissions.

- La vérification d'identité : d'une part, le client doit pouvoir être assuré de la provenance des ressources qu'il utilise ; d'autre part, le FR doit être en mesure d'identifier le client utilisateur de ses ressources.

- La preuve de non répudiation : le client possède la preuve que l'intégralité des informations qu'il a transmises au FR ont été reçues. Il en est de même pour le FR.

- La disponibilité du service : le client dispose d'un accès aux ressources fournies par le FR lorsqu'il le souhaite. De son côté, le FR doit s'assurer que les ressources demandées sont disponibles (en gérant éventuellement des sémaphores permettant l'accès à des ressources limitées).

- La preuve d'audit : le client peut prouver que les opérations demandées ont été réalisées par le système. Le FR peut faire de même dès lors qu'il repose sur une traçabilité sécurisée.

2.1.2. Contrôle d'accès du FSS

Le FSS peut mettre en œuvre un système de contrôle d'accès centralisé ou réparti faisant appel aux modes « push » ou « pull » [LAM 92].

L'utilisation du mode « pull » suppose que le FSS reçoit un identifiant distinctif du client et indique au FR si ce client dispose ou non du droit d'accès à la ressource demandée en se référant aux informations locales dont il dispose.

A contrario, le mode « push » suppose que le client présente des informations d'accès qui seront validées par le FSS. Dès lors que ces informations sont valides et

en adéquation avec les droits d'accès requis, le FR peut donner suite à la demande en autorisant l'accès à la ressource.

Les communications entre le client, le FSS et le FR doivent apporter la confiance nécessaire sachant que [NAG 98] :

- Les source et destination ont besoin de s'assurer de l'identité de leur interlocuteur.
- Les informations d'identité ou de droit d'accès doivent être authentiques.
- Les données échangées peuvent être confidentielles.

La sécurité traditionnelle des applications informatiques repose sur une authentification (faible) par le biais d'un couple formé d'un identifiant et d'un mot de passe donnant accès au système [GAR 96]. Les techniques actuelles permettant de répondre aux besoins de sécurité décrits auparavant reposent pour la plupart sur l'utilisation de procédés cryptographiques. Il est alors d'usage d'employer le terme d'authentification forte.

2.2. Présentation de la cryptographie

La cryptographie (du grec « Kruptos » signifiant cacher et de « Graphein », écrire) est la science de la dissimulation de messages de sorte que seuls certains initiés disposant d'une donnée secrète (appelée « trappe ») soient en mesure de prendre connaissance de leur contenu.

Cette science mathématique d'écriture et lecture de messages codés a une longue histoire puisque le premier « document » chiffré connu remonte au XVI^{ème} siècle avant Jésus-Christ. Un potier irakien y avait gravé sa recette secrète en supprimant certaines consonnes et en modifiant l'orthographe des mots.

La cryptographie moderne [SCH 01] est officiellement initiée en 1976 par Whitfield Diffie et Martin Hellman [DIF 76]. Il semblerait néanmoins que James Ellis en soit le fondateur en 1970 si l'on se réfère aux informations fournies par le CESG (« Communications Electronics Security Group ») du Royaume-Uni [JAW 01].

L'usage de la cryptographie s'est banalisé avec la démocratisation de l'Internet. Les systèmes informatiques actuels en font un emploi intensif lorsqu'il s'agit de proposer des techniques efficaces et fiables aptes à garantir :

- L'authentification réciproque des entités qui s'échangent des messages.
- L'intégrité des données échangées afin de détecter toute modification, en particulier au cours de leur transmission sur le réseau de communication.

- La confidentialité des informations échangées, qui assure que seules les parties disposant d'une donnée secrète supplémentaire appelée clé, sont en mesure de prendre connaissance de ces informations.

On retrouve ainsi des informations codées dans les cartes bancaires, les téléphones cellulaires, les échanges bancaires ou les valises diplomatiques.

L'objectif fondamental de la cryptographie est de permettre un échange d'informations entre deux personnes – ou programmes – au travers d'un canal de communication peu sûr tel que l'Internet. L'information que l'émetteur souhaite transmettre (ou texte clair) est rendue inintelligible par un procédé connu ou secret, faisant appel à une clé, avant d'être soumise au destinataire. La cryptographie veille à ce que seul le destinataire soit techniquement en mesure de décoder le message reçu et ainsi de prendre connaissance de l'information originelle. Notamment, un espion ne doit pouvoir disposer de suffisamment d'informations ou de puissance de calcul lui permettant de décoder aisément le message intercepté.

De manière formelle, un système cryptographique est de fait un quintuplet $SC = \{P, C, K, E, D\}$ composé des éléments suivants :

- P est un ensemble fini de textes clairs possibles. Bien que l'on imagine cet ensemble infini, les ouvrages traitant de la cryptographie préfèrent manipuler des ensembles finis (afin d'utiliser les propriétés relatives aux groupes et corps mathématiques).

- C est un ensemble fini de textes chiffrés (codés) possibles. La finitude de C doit être mise en relation avec la finitude des ensembles P et K.

- K est un ensemble fini de clés possibles. La clé est l'élément nécessaire au déchiffrement des messages chiffrés.

- $E : \{P, K\} \mapsto C$, définit une règle de chiffrement. A tout texte clair (élément de P) correspond un texte chiffré à l'aide d'une clé donnée.

- $D : \{C, K\} \mapsto P$, définit une règle de déchiffrement. D et E sont involutives, ce qui signifie que $DoE(x,k) = D(E(x,K)) = x, \ \forall x \in P, \forall k \in K$.

Selon le type des clés mises en jeu dans un échange à texte chiffré, on opère une distinction entre la cryptographie symétrique, où une seule clé est nécessaire, et la cryptographie asymétrique employant un ensemble composé de deux clés distinctes appelé « bi-clé ».

2.2.1. Cryptographie symétrique

Lorsque qu'une unique clé est utilisée pour à la fois chiffrer et déchiffrer les messages (figure 6), le terme « clé secrète » est employé. Cette branche de la cryptographie est connue sous le nom de cryptographie symétrique.

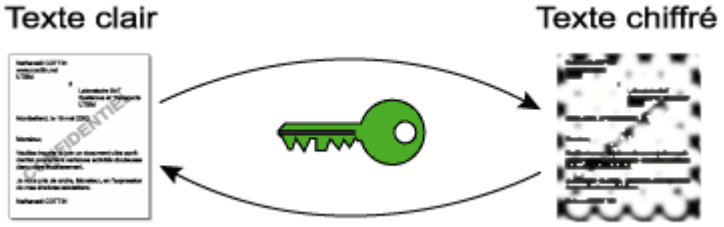

Figure 6 : principe de la cryptographie symétrique

Un exemple simple est le chiffrement par décalage utilisé par César. Les textes chiffrés sont obtenus en décalant de N rangs les lettres dans l'alphabet. Le décodage s'effectue à l'aide d'un décalage inverse.

Exemple 1 : le modèle de César, défini par $César = \{P, C, \{3\}, \succ, \prec\}$, se lit comme suit : tout texte clair (élément de P) est chiffré à l'aide de l'opération « $\succ$ » (décalage vers la droite, c'est à dire de « A » vers « Z ») et de la clé « 3 » (nombre de décalages). Ainsi le texte chiffré correspondant au texte clair « UTBM » est « XWEP ». Tout texte chiffré est décodé à l'aide de l'opération « $\prec$ » et de la clé « 3 ».

Cet exemple ancestral montre une caractéristique de la cryptographie mise en évidence par une question formulée comme suit : à quel chiffre correspond la lettre « Y » (en clair) ?

La lettre « Y » est chiffrée de manière intuitive en « B », ce qui suppose un raisonnement en « modulo ». La cryptographie, qu'elle soit ancienne ou actuelle, fait usage de l'arithmétique modulaire. Dans l'exemple proposé, les règles de chiffrement et déchiffrement sont appliquées « modulo 26 ».

Exemple 2 : un second cas particulier de ce type de chiffrement est le système ROT13 [SCH 01] qui repose sur la clé « 13 » et supporte une unique opération de décalage notée « $\succ$ ». En effet, les calculs étant soumis à un modulo, l'application successive de deux opérations de décalage permet de retrouver le texte en clair.

2.2.2. Cryptographie asymétrique

Le principal défaut des systèmes à clé secrète provient :

- De la difficulté à gérer les clés lorsqu'un grand nombre d'interlocuteurs différents sont impliqués dans des échanges secrets.
- De la nécessité d'utiliser un canal sûr pour distribuer la clé secrète à l'ensemble des parties impliquées dans un échange sécurisé donné.

L'obtention de la sécurité recherchée recourt ainsi à une technique concurrente appelée cryptographie asymétrique, où une paire de clés (« bi-clé ») est mise en jeu : une clé publique et une clé privée.

Les clés publique et privée sont utilisées conjointement aux règles respectives de chiffrement et de déchiffrement (figure 7). La clé publique est distribuée (diffusée) par le destinataire à tout émetteur souhaitant lui communiquer des informations sensibles. Le destinataire conserve toutefois le secret de sa clé privée.

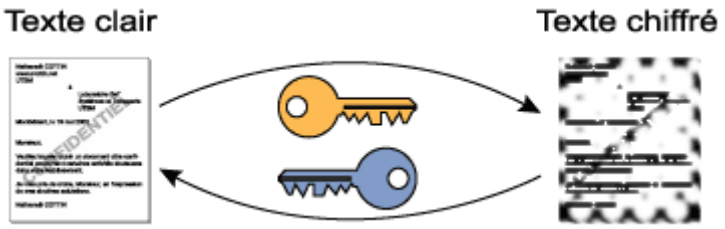

Figure 7 : principe de la cryptographie asymétrique

Exemple : la fonction mathématique $f:\{x,y\}\rightarrow x\times y, \forall x\in P, \forall y\in K$ peut être utlisée dans cette optique. Le système algébrique est alors défini par $S_3=\left\{P,C,\left\{3,\frac{1}{3}\right\},\{f\}\right\}$ si l'on s'intéresse uniquement aux clés $\left\{3,\frac{1}{3}\right\}$. De manière générale, l'ensemble des paires de clés envisageables est de la forme $\left\{k,\frac{1}{k}\right\}, \forall k\in Z^*$.

Soit le texte clair « MESSAGE ». Ce dernier peut être représenté par la succession des codes ASCII qui le composent, à savoir « 53 45 59 59 41 47 45 ». Le chiffrement à l'aide de la clé publique « 3 » donnerait alors « 159 135 177 177 123 141 135 ».

Bien que le message chiffré transmis semble indéchiffrable au premier abord pour qui ne connaît pas la clé de déchiffrement, nous présentons maintenant quelques exemples d'attaques possibles dont l'objectif est double : d'une part découvrir le texte en clair et d'autre part dévoiler la valeur de la clé de déchiffrement (clé privée).

2.2.3. Exemple d'étude cryptanalytique

Comment un intrus ayant pris connaissance du texte chiffré pourrait-il en déduire la clé de déchiffrement (privée) ? Etudier la vulnérabilité des procédés cryptographiques et exploiter leurs failles de sécurité, tant au niveau des algorithmes que des protocoles mis en œuvre, est du ressort de la cryptanalyse.

Nous allons nous intéresser à trois attaques à texte chiffré connu, différentes selon le degré d'informations dont dispose le cryptanalyste : l'attaque par recherche exhaustive, l'attaque par diviseur commun et enfin l'attaque par probabilité d'occurrence.

2.2.3.1. Attaque par recherche exhaustive

Une attaque par recherche exhaustive (ou « brute force ») est possible : l'attaquant essaie l'ensemble des clés possibles jusqu'à obtention d'un texte clair cohérent. Cette attaque fonctionne dans un délai raisonnable si la bi-clé utilisée est peu sûr. Ce type de recherche est souvent inefficace dans les systèmes cryptographiques modernes tels que RSA [KAL 98a], le temps mis pour découvrir la clé privée étant trop important et ce, même à l'échelle informatique, lorsque la taille des clés est suffisante. C'est pour cette raison que les cryptanalystes utilisent souvent d'autres méthodes fondées sur les informations dont ils disposent ou qu'ils peuvent déduire des messages chiffrés interceptés [MEL 01].

Exemple : le cryptanalyste procède par essai-erreur et teste l'ensemble des clés possibles de la forme $\frac{x}{y}$, $x \in N^*$, $y \in N^*$ jusqu'à obtention de la clé $\frac{1}{3}$.

2.2.3.2. Attaque par diviseur commun

Afin de procéder à cette attaque, le cryptanalyste pose pour hypothèse que la clé privée à découvrir est de la forme $\frac{1}{y}$ où y est un entier non nul.

De fait, chaque valeur entière du texte chiffré est divisible par y. Il lui suffit de calculer le plus grand diviseur commun (ou « pgcd ») de l'ensemble des valeurs chiffrées pour obtenir la borne supérieure de la clé. Il s'agit dans l'exemple précédent de la valeur « 3 ».

La clé recherchée étant un diviseur de ce pgcd (la valeur « 1 » étant à exclure), le pgcd donne dans ce cas directement la valeur de la clé.

2.2.3.3. Attaque par probabilité d'occurrence

Première version : un raisonnement autre que les deux attaques dévoilées ci-avant est toutefois envisageable, pour peu que le cryptanalyste sache que le message chiffré est textuel : il est fort probable que la lettre « A » corresponde au plus petit entier du message chiffré.

La valeur de la clé privée est alors immédiate puisque $\frac{135}{45} = 3$.

Cette astuce fonctionne du fait que le texte en clair contient effectivement la lettre « A ». Pour des messages plus longs (ou lorsque le cryptanalyste dispose de plusieurs messages chiffrés à l'aide d'une même clé), une attaque corrélée est possible.

Seconde version : cette attaque repose sur l'étude de la fréquence d'apparition des codes et permet de déduire la probabilité qu'une valeur chiffrée corresponde à une valeur en clair donnée. Le calcul de la clé en est ainsi simplifié.

Les lettres les plus usitées en français étant par exemple « A » et « E », un texte de plusieurs lignes chiffré donnera une fréquence d'apparition des résultats chiffrés de « A » et « E » plus grande que les autres lettres. Si le chiffre le plus fréquent est 135 alors le cryptanalyste pourra supposer qu'il correspond à la lettre « E ». Dans le cas où la clé correspondante ne permet pas de déduire le reste du message, la correspondance entre 135 et « A » sera évidente et le cryptanalyste retrouvera la clé privée 3.

> *La sécurité d'un système cryptographique est mesurée par la difficulté à briser le système en fonction de l'espace des clés possibles. Cette mesure fait appel à la notion d'entropie ou degré d'incertitude quant à la probabilité de découvrir la clé privée lorsqu'un sous-ensemble de l'ensemble des textes chiffrés est connu.*

2.3. Aperçu de la signature électronique

2.3.1. Définition informelle

Une signature manuscrite (« conventionnelle ») apposée sur un document papier engage la responsabilité du signataire dans le sens où :

- Il est possible de prouver que sa signature est authentique.
- La signature identifie le signataire, si tant est que le vérificateur dispose d'une signature témoin qu'il considère comme fiable.
- Le signataire ne peut revenir en arrière et invalider sa signature dès lors qu'une tierce partie en a pris connaissance et est en mesure de la produire.

Une signature électronique peut être vue comme l'équivalent numérique de la signature manuscrite, sous réserve que certaines conditions sont remplies. C'est pourquoi les signatures que les juristes nomment « sécurisées » [PIE 02b], car

répondant aux critères légaux, sont désignées techniquement par « signatures qualifiées ». Ces dernières, par opposition aux signatures sans valeur légale (« signatures simples ») [EES 99], font l'objet de notre étude, contrairement à la plupart des systèmes informatiques sécurisés qui ne prennent pas en compte la dimension juridique pouvant être acordée à une signature dérivée de la signature numérique.

2.3.2. Usage et garanties apportées

Comme indiqué précédemment, la signature électronique apporte des garanties fondamentales lors d'échanges de données numériques entre parties, que le canal de communication soit sécurisé ou non. Ces garanties comprennent l'authentification forte des parties, la preuve d'authenticité des informations échangées, l'identification certaine (sans ambiguïté) des signataires et la non répudiation des informations signées.

2.3.2.1. Authentification forte

Le droit français n'emploie pas le terme d'« authentification » et lui préfère « authenticité » qui englobe non seulement le signataire mais également l'acte signé. Toutefois, dans un contexte informatique, l'authentification forte fait référence aux moyens mis en œuvre afin d'établir l'identité d'une personne souhaitant accéder à un système informatique.

L'authentification traditionnelle fait appel à la notion d'identificateur (« login ») et mot de passe. La signature électronique, supposée créée par des moyens que le signataire conserve « sous contrôle » (dixit la loi française), tels qu'une carte à puce, un « token », une cd-carte ou tout logicel sécurisé et audité, apporte des garanties supplémentaires. C'est pourquoi l'on parle d'authentification forte.

L'authentification forte contribue ainsi à sécuriser le système d'information en amont par l'ouverture des accès à l'aide de données personnelles souvent contenues sur un support physique que seul le propriétaire est en mesure de présenter.

2.3.2.2. Preuve d'authenticité des données

L'authenticité des données apporte la confiance nécessaire dans le fait que toute altération d'un message, dès lors qu'il a été signé, est détectée. Il en résulte une incohérence entre le message et la signature qui lui a été apposée.

2.3.2.3. Identification certaine

L'identification est le procédé par lequel le destinataire vérifie l'identité du signataire. Selon le type de signature employé, la vérification d'identité se fera à différents degrés de confiance. Lorsque l'identité est incontestable car unique et

attestée par une tierce partie neutre en qui le destinataire a confiance, le terme d'identification certaine sera employé.

Authentification forte et identification certaine désignent deux terminologies distinctes mais complémentaires : la première se situe en amont de l'accès au système alors que la seconde permet de s'assurer de l'identité du signataire lorsque des informations irréfutables sont fournies.

Preuve d'authenticité des données et identification certaine sont généralement regroupées sous le terme générique d'authentification.

2.3.2.4. Non répudiation des informations signées

La non répudiation est utilisée en cas de contestation du message signé. Le signataire ne peut nier avoir apposé sa signature sur le message, sauf s'il parvient à prouver l'invalidité de sa signature (le plus souvent due à la perte ou la compromission de sa clé de signature).

2.3.3. Signature manuscrite et signature électronique

D'un point de vue technique, quelles différences peut-on mettre en avant afin d'appréhender l'intérêt de recourir à la signature électronique ?

Le tableau comparatif ci-dessous fait état des principales opérations réalisables à la fois à l'aide d'une signature manuscrite et d'une signature électronique, étant sous-entendu que la signature électronique répond au contexte législatif en vigueur dans les pays respectifs de son apposition et de sa validation.

Service demandé	Electronique	Manuscrite
Manifestation du consentement[2]	**OUI**	**OUI**
Valeur juridique (sous certaines conditions)	**OUI**	**OUI**
Reconnaissance en dehors des frontières	**OUI**	**OUI**
Evolution au cours du temps[3]	**OUI**	**OUI**
Possibilité de sur-signer et contre-signer	**OUI**	**OUI**

[2] Il s'agit ici du droit français. D'autres législations (le droit anglais par exemple) permettent de « viser » des documents. Cette notion se retrouve également dans la loi type de la CNUDCI [CNU 01]. A noter que l'article 1316-4 de [SEN 01] crée un amalgame entre signature et contrat.

[3] La différence est que l'évolution de la signature manuscrite est à l'initiative du signataire alors que les changements de signature électronique sont réguliers et dictés par l'évolution des technologies.

Service demandé	Electronique	Manuscrite
Identifie un signataire connu du destinataire	**OUI**	**OUI**
Reflète le caractère et la personnalité[4]	**NON**	**OUI**
Reste valide au cours du temps[5]	**NON**	**OUI**
Autorise la signature à distance	**OUI**	**NON**
Identifie un signataire inconnu du destinataire	**OUI**	**NON**
Un signataire dispose de plusieurs signatures[6]	**OUI**	**NON**
Vérifiée automatiquement par informatique	**OUI**	**NON**
Contrôlée objectivement par un tiers neutre	**OUI**	**NON**
Duplication possible	**NON**	**OUI**
Difficulté d'imitation	**OUI**	**NON**
Intégrité du contenu auquel elle se rapporte	**OUI**	**NON**
S'applique au multimédia (non juridique en soi)	**OUI**	**NON**

De plus, la signature électronique simplifie le paraphe de chaque page d'un document en ne le signant électroniquement qu'une seule fois.

Cependant, comme l'a noté Carl Ellison [ARI 98, p.12], on observe une différence fondamentale entre ces deux types de signatures (en dehors du fait que l'une est manuscrite et l'autre informatique) :

- La signature manuscrite est fortement corrélée à l'identité du signataire mais non spécifique à l'information signée : la signature est immuable.

[4] La signature manuscrite est un dessin dont le graphique est une création de l'individu, contrairement à une valeur générée par un ordinateur.

[5] En supposant qu'aucune disposition particulière de conservation ne soit mise en œuvre.

[6] Les signatures sont ici supposées sans lien direct permettant de les attribuer à un unique signataire (de telle sorte que l'équivalent informatique d'une analyse graphologique indique que les signatures ne proviennent certainement pas du même individu).

- La signature électronique (dans sa forme la plus simple) est peu liée au signataire mais en relation forte avec les informations signées. Ainsi, chaque signature électronique est propre au contenu signé, bien qu'elle soit apposée par un unique signataire.

De fait, certaines conditions doivent être remplies par la signature électronique afin de lui attribuer une valeur légale en terme de faire-valoir de preuve et une correspondance effective avec la signature manuscrite. Il s'agit d'obtenir à la fois une relation forte envers l'identité du signataire ainsi qu'un lien étroit avec les informations signées.

2.3.4. Valeur légale accordée à la signature électronique

Avant d'être un ensemble de données informatiques, la signature électronique trouve sa raison d'être dans la loi nationale reposant sur la directive européenne 99/93/CE [PEC 99]. Une signature électronique ne peut être juridiquement recevable que si elle se rapporte à un contenu produisant un effet juridique [PIE 01]. Son apposition doit en outre être conforme à certaines règles en vigueur. Ces règles tiennent compte de la sécurité mise en œuvre par l'émetteur de la signature, de l'environnement de signature et de la reconnaissance juridique des pièces justificatives. Les juristes emploient le terme de signature « sécurisée ».

A ce propos, l'article 1316-4 de la loi française du 13 mars 2000 [SEN 00] précise que « *l'écrit sous forme électronique est admis en preuve au même titre que l'écrit sur support papier, sous réserve que puisse être dûment identifiée la personne dont il émane et qu'il soit établi et conservé dans des conditions de nature à en garantir l'intégrité* ».

Cette loi est complétée par le décret n° 2001-272 du 30 mars 2001 modifié [SEN 01a] [SEN 01b] qui lui confère un contexte d'application. Ce dernier indique de quelle manière un dispositif de création et vérification de signatures électroniques doit être sécurisé et de quelle manière les PCSE et les certificats électroniques qu'ils délivrent doivent être qualifiés (procédure auprès du MINEFI).

Les textes juridiques évitent sciemment de faire intervenir une quelconque contrainte technologique dans leurs définitions et préconisations. Bien que puissent être envisagés une signature scannée, un tatouage ou encore une empreinte biométrique (empreinte digitale, rétinienne ou génétique – ADN – par exemple) [AZZ 03], seule l'utilisation conjointe d'une clé unique et d'un algorithme de hachage[7] sûr permet de s'assurer à la fois de l'authenticité et de l'intégrité de l'acte signé.

[7] Un algorithme de hachage (ou de calcul d'empreinte numérique) sûr permet de produire un résumé informatique unique pour une information source donnée. Les propriétés de ces algorithmes seront décrites dans la section consacrée à l'étude de la signature électronique.

Ainsi, la valeur juridique d'une signature électronique est déterminée aujourd'hui non seulement par les processus d'apposition et de vérification mais également par le degré de sécurité de l'environnement dans lequel la signature a été établie et la nature des pièces justifiant la précision de l'identification du signataire en rapport avec la criticité des informations de preuve requises [PIE 02b].

2.3.4.1. Environnement de création de la signature

On entend par environnement de création de signature :

- Le degré de sécurité du système d'exploitation utilisé, la possible compromission de la machine par un programme « étranger ».
- Le support matériel de la clé signature tel que la carte à puce, le e-button ou la cd-carte (objet du brevet international [OEB 02a]).
- Les pièces permettant de justifier de la valeur de la signature, comme un certificat numérique, une date certifiée ou les procédures de déverrouillage de la clé de signature (par mot de passe, empreinte digitale ou rétinienne par exemple).

2.3.4.2. Reconnaissance des pièces justificatives

Pour qu'un vérificateur (il s'agit en règle générale du destinataire) soit en mesure d'accepter une signature juridique comme valable, il est recommandé qu'un lien extrêmement fort entre le signataire et les informations afférentes à sa signature (la clé de vérification en particulier) soit attesté par une tierce partie de confiance. Ce lien peut être démontré par le biais d'un certificat numérique d'identité [KOH 78] [HOU 02] ou d'attributs [FAR 02], générés par une entité reconnue de confiance et appelée « Prestataire de Services de Certification Electronique » (PSCE) ou, *in extenso*, « Autorité de Certification » (AC) ou « Tiers de certification ».

Cette autorité peut être publique (une administration par exemple) ou privée. Elle est chargée de mettre en place des moyens techniques, tels qu'une « Infrastructure à Clé Publique » (ICP), permettant de gérer les certificats qu'elle délivre à ses clients.

Les textes législatifs [PEC 99] [SEN 01b] ne reconnaissent que les certificats numériques qualifiés, c'est pourquoi les systèmes informatiques mettant en œuvre la signature électronique doivent s'attacher à prendre en compte ce type de signature.

Il reste à définir la signature électronique d'un point de vue informatique, en termes de données manipulées, processus de création et de validation. C'est ce que propose la partie suivante, à l'issue de l'introduction à divers modèles de sécurité et notamment au modèle de sécurité CORBA proposé par l'OMG.

2.4. Quelques modèles de sécurité

Bien que nombre de modèles existent [JAE 01] (« Schematic Protection Model » [SAN 88] et « Domain and Type Enforcement » [BOE 85] par exemple), nous présentons brièvement JAAS, DSSA et SDM, trois modèles représentatifs des recherches concernant la sécurité dans les systèmes répartis avant d'aborder la sécurité mise en place au sein de l'architecture CORBA.

2.4.1. Modèle JAAS

Le service d'authentification et d'autorisation de Java (ou « Java Authentication and Authorization Service », JAAS) [JAW 01] permet de construire des applications Java sécurisées par le biais du modèle standard de Java 2. JAAS propose une extension de ce modèle en y ajoutant la possibilité d'authentifier les objets Java.

Ce modèle repose essentiellement sur la définition d'une classe Java intitulée « Subject » (sujet) offrant la possibilité d'assigner et de contrôler des références (ou attributs) d'authentification et de droits d'accès publics et privés. Les références publiques peuvent par exemple indiquer des certificats numériques et les références privées des clés de signature ou de chiffrement. Le sujet est en relation avec une ou plusieurs identités (« principals ») afin de produire l'identité adaptée à chaque demande.

2.4.1.1. Authentification de JAAS

L'authentification d'un objet java appelant (ou demandeur) nécessite la mise à disposition d'un fichier de configuration de la sécurité gérant ses droits d'accès. Celui-ci est utilisé par le module de connexion au niveau de l'objet appelé (ou fournisseur). L'authentification repose également sur la définition de comportements associés à l'interface Java « LoginModule ». Cette interface dispose des méthodes « login », « commit », « abort » et « logout » qui doivent être définies.

Chaque fournisseur JAAS étant supposé initialisé lors de sa création (construction), une session entre un demandeur et un fournisseur est établie par le biais de la méthode « login ».

Lors de la connexion, le fournisseur consulte dans un premier temps le fichier de configuration puis compare les informations de références requises avec celles présentées par le demandeur via la méthode « login ». Dans une second temps, le fournisseur appelle « commit » en cas de succès de la première phase d'authentification ou « abort » dans le cas contraire. Lorsque la seconde phase réussit, le demandeur peut demander au fournisseur le service souhaité dès lors qu'il y est autorisé. Ainsi JAAS apparaît comme une couche intermédiaire entre une demande de service et l'exécution de ce service.

La session se termine dès que le demandeur requiert l'exécution de la méthode « logout ».

2.4.1.2. Autorisation de JAAS

Une fois un demandeur authentifié, JAAS permet de limiter l'accès aux ressources dont il peut disposer. Les décisions d'accès ou de refus d'accès sont prises en fonction de l'identité du demandeur. Là encore, un fichier définissant les stratégies de sécurité à suivre doit être renseigné. Ce fichier obéit à la grammaire suivante :

```
grant codebase "<URL>", SignedBy "<list of names>"
    Principal [<class name>] "<name>"
    [, Principal, ...]
{
  permission <class name> ["<target name>"]
                          [, "<actions>"]
                          [, SignedBy "<list of names>"];
  [permission ...]
};
```

Cette grammaire, définie par le modèle de sécurité Java 2 et résumée par [NAG 98], permet de déclarer les permissions d'accès (aptitude à exécuter une action sur le système), rôles (ensemble de privilèges liés au statut de l'appelant) et signatures numériques requis afin d'être en mesure d'exécuter l'appel à un objet Java, souvent distant.

Par exemple, la déclaration suivante permet de rejeter l'accès en écriture portant sur le fichier « MyFile.txt » lorsque le demandeur porte une identité autre que « cottin » :

```
grant codebase "file:auth.jar",
    Principal ejava.jaas.MyPrincipal "cottin"
{
  permission java.io.FilePermission "MyFile.txt", "write";
};
```

JAAS est de fait le médium privilégié dès lors qu'il s'agit de gérer un ensemble d'objets requérant une gestion de droits d'accès. L'atout majeur de JAAS est qu'il repose sur plusieurs interfaces Java ouvertes sur différentes mises en œuvre de manière à conserver l'indépendance de l'environnement d'application par rapport aux techniques d'authentification et d'autorisation sous-jacentes.

2.4.2. Modèle DSSA

Le modèle DSSA (« Digital Distributed System Security Architecture ») [GAS 89] [GAS 90] permet de restreindre l'accès à des services par le biais de listes de contrôle d'accès (« Access Control Lists », ACL) [CLA 01]. Ces listes répertorient les accès autorisés en fonction des permissions accordées aux demandeurs. L'accès aux ressources protégées est autorisé lorsque le demandeur dispose des droits

suffisants en accord avec l'ACL. Afin de faciliter le contrôle d'accès, DSSA gère des groupes d'utilisateurs dont les droits d'accès sont similaires.

Ce modèle manipule également des rôles inclus dans des certificats de délégation. A un utilisateur donné sont associés un ou plusieurs rôles en fonction de son statut. Ce dernier choisit alors les rôles qui lui autorisent l'accès à un service et génère un certificat de délégation. Le contrôle d'accès n'est donc pas réalisé directement sur l'identité du demandeur mais sur les différents rôles que le demandeur est en mesure de produire.

2.4.3. Modèle SDM

Le model de sécurité SDM (« Secure Delegation Model ») [NAG 97] [NAG 98] repose sur les invocations d'objets à distance selon le procédé RMI que propose le langage Java. Il étend le modèle de sécurité Java 2. Alors que ce dernier offre la possibilité de signer numériquement des objets et de définir des permissions d'accès, SDM y introduit la notion de délégation de permissions d'accès par le biais de certificats de rôles, comparables aux certificats de délégation de DSSA.

Cette délégation autorise l'utilisation des permissions d'accès et des rôles d'un objet par un autre dès lors qu'un accord de délégation a été approuvé par les deux parties. La validité de la délégation peut être contrôlée directement par les objets appelés ou à l'aide d'un service externe d'administration.

2.5. Modèle de sécurité de CORBA

Le service de sécurité de CORBA, dont la spécification (« CORBA Security Service Specification », CSSS) [OMG 02] est actuellement disponible dans sa version 1.8, propose un ensemble de classes permettant de gérer la sécurité au niveau du middleware en faisant appel à des services applicatifs combinés à certaines modifications internes à l'ORB.

Cette spécification fait intervenir deux niveaux de sécurité organisés en différents composants [BAL 00] :

- La sécurité de niveau 1 présente des interfaces IDL de sécurité minimale pour les applications réparties dont la sécurité n'est pas prioritaire.
- La sécurité de niveau 2 propose des interfaces IDL plus fournies afin d'intégrer des politiques de droit d'accès au sein des applications dont la sécurité est primordiale.

Il apparaît au regard des propositions d'évolution de la sécurité CORBA que le modèle proposé est difficile à mettre en œuvre [JAW 01] : il fait appel aux intercepteurs [BAL 00] (ensembles d'instructions exécutées au moment de l'appel à distance et lorsque la réponse est retournée) ainsi qu'à plusieurs services complémentaires, parmi lesquels :

- Un service chargé d'attester l'existence des transactions. Ce service de non répudiation prévient toute tentative de négation de participation à une transaction par un quelconque acteur impliqué dans celle-ci.

- Un service d'autorisation d'accès pour le client : ce service définit les règles d'autorisation chez le client avant émission de la requête.

- Un service d'autorisation d'accès chez le serveur : détermine si un client dispose des droits d'accès suffisants pour honorer l'exécution de la méthode demandée.

- Un outil d'administration des privilèges et politiques de sécurité.

- La possibilité de réaliser des audits via un service dédié.

De plus, bien que CORBA 1.2 intègre les protocoles IIOP (« Internet Inter-ORB Protocol ») et UNO (« Universal Networked Objects »), l'interopérabilité entre middlewares CORBA fait actuellement défaut. En effet, cette architecture est sujette à interprétation par les fournisseurs de solutions logicielles : les règles de construction des références d'objets sont approximatives et les services proposés sont décrits en langage OMG IDL, ce qui ne définit pas une sémantique précise [GA 96]. L'ajout de protocoles de sécurité va contribuer à éloigner l'OMG de cet objectif.

Cette norme tient néanmoins compte des considérations de sécurité liées aux appels combinés d'objets mixte, telles que l'usurpation d'identité, la non répudiation et le transfert de droits d'accès (également connu sous le terme « délégation de privilèges »). Ce dernier point concerne l'utilisation d'un objet mixte (intermédiaire) ayant des droits suffisants dans le but de réaliser l'appel d'une méthode sur l'objet serveur cible alors qu'initialement l'objet demandeur ne dispose pas de tels droits.

2.6. Objectifs

Bien que les divers environnements précités disposent de procédés visant à accroître la sécurité des échanges tels que le modèle de sécurité que propose CORBA, ces derniers montrent leurs limitent lorsqu'il s'agit de conférer un contexte légal à ces mêmes échanges électroniques.

En particulier, les problèmes de datation certaine et de non répudiation des échanges sont souvent tenus à l'écart des spécifications de ces environnements. Pourtant, ces informations et processus sont nécessaires au juriste pour déterminer la recevabilité d'une preuve devant un tribunal et aux systèmes d'information pour décider des garanties duales apportées à la fois par un demandeur de service (client) et par l'exécuteur de ce service (serveur).

C'est pourquoi le présent travail aborde la modélisation d'un environnement réparti objet, ouvert et évolutif, reposant sur les principes majeurs édictés par la norme

CORBA tels que l'interopérabilité, la séparation des interfaces et des traitements sous-jacents ou encore la réutilisation des composants. Les prémisses de la légalisation des échanges sont mis en œuvre par :

- Le recours à de nouveaux formats de signature et de certificat électronique.
- La définition de protocoles et particulièrement un protocole d'horodatage multiple.
- La possibilité de joindre une connexion à un module chargé de la traçabilité des échanges entre objets.
- L'intégration de ces technologies à la constitution d'une « Chaîne de Confiance » (CdC).

Le prochain chapitre est ainsi consacré à la présentation approfondie et l'étude critique de la signature électronique. Cette etude s'accompagne de diverses propositions de format permettant de décrire la structure d'une telle signature. Ce format doit notamment offrir la possibilité de renfermer l'ensemble des informations nécessaires à la validation immédiate et *a posteriori* des signatures électroniques créées.

Le chapitre suivant intègre les résultats précédents au sein d'un modèle de système réparti dérivé de CORBA, en particulier pour réaliser des échanges authentifiés et non répudiables entre objets répartis.

Enfin, ce modèle prend part dans le dernier chapitre à la conception de la CdC chargée de fournir les services permettant d'apporter la sécurité nécessaire à la mise en pratique :

- Des protocoles de création et validation dans le temps des signatures électroniques.
- De la gestion des documents sous forme électronique et principalement des documents signés tout au long de leur cycle de vie (création, signature, archivage, restitution, destruction).

CHAPITRE 2 : ETUDE DE LA SIGNATURE ELECTRONIQUE

« Mieux vaut pomper même s'il ne se passe rien que risquer qu'il se passe quelque chose de pire en ne pompant pas »

Les devises Shadok

La signature électronique apporte une solution élégante aux problèmes d'une part de preuve d'authenticité des actes à valeur juridique et d'autre part de gestion des droits d'accès aux services applicatifs des systèmes d'information sécurisés, notamment ceux mis en œuvre par des objets répartis de type serveur ou mixte.

L'objectif est double : il s'agit d'une part d'étudier le fonctionnement de la signature électronique d'un point de vue technique et d'autre part de proposer un format de signature électronique sécurisée reposant sur les standards actuels de même que des protocoles de création et validation de signature. Ce format sera implanté au cœur du modèle objet réparti présenté dans le prochain chapitre.

Bien que nous nous intéressions à la signature électronique au sein des échanges en objets répartis, nous nous attacherons dans ce chapitre à proposer un format de signature tenant compte des contraintes non seulement techniques mais également juridiques de sorte qu'il soit adapté à son utilisation pour la création de documents signés électroniquement et qu'il contribue à leur conférer une reconnaissance légale.

1. Définition technique

Une première définition de la notion de signature électronique est donnée par l'article 2 de la directive 99/93/CE du parlement européen [PEC 99] : « *donnée sous forme électronique qui est jointe ou liée logiquement à d'autres données électroniques et qui sert de méthode d'authentification* ».

L'ISO 7498-2 apporte une définition plus technique, considérant la signature électronique comme « *un ensemble de données ajoutées à une unité de données, ou transformation cryptographique d'une unité de données, permettant à un destinataire de prouver la source et l'intégrité de cette unité en la protégeant contre la contrefaçon (par le destinataire par exemple)* ». Cette définition est corroborée par le rapport d'experts [EES 99] en charge de l'étude des moyens techniques permettant d'apporter les garanties exigées par la loi européenne [PEC 99]. Les solutions techniques considérées comme robustes sont décrites dans [NIS 00] telles que :

- Des considérations de taille de clés suffisamment grandes afin de prévenir des attaques courantes sur les clés (en particulier l'attaque de recherche exhaustive) [MEL 01].

- L'étude des algorithmes de génération de bi-clés basés sur des suites de nombres pseudo aléatoires [JAW 01] émises par des générateurs conformes à [EAS 94].

- La sélection d'algorithmes à clé secrète permettant de chiffrer des informations confidentielles, telles que les clés privées (ou de signature), à l'aide d'une phrase-clé (« passphrase »).

Le lien entre signature électronique et cryptographie à clé publique apparaît clairement. Là où la cryptographie montre ses limites (preuve d'authenticité, identification certaine, droits d'accès, etc.), la signature électronique apporte à la fois des réponses concrètes et ouvre des thèmes de recherche passionnants, tels que la preuve d'authenticité à long terme des archives signées ou encore la gestion de la délégation de droits associés aux signatures.

1.1. Rapport entre signature électronique et chiffrement

Dans quelle mesure la signature électronique est-elle apparentée à la cryptographie ? En effet, certaines similitudes peuvent être reconnues entre la signature électronique et la cryptographie à clé publique :

- Pour envoyer un message chiffré, l'émetteur utilise la clé publique du destinataire ainsi que la méthode de chiffrement. Le destinataire déchiffre alors le message reçu à l'aide de sa clé privée et de la méthode de déchiffrement.

- Pour signer un message, le signataire doit utiliser une information qui lui est propre afin que le destinataire puisse s'assurer de la provenance du message reçu. Justement, le signataire peut disposer d'une bi-clé et donc d'une clé privée qu'il conserve secrète. Il va donc utiliser sa propre clé privée pour réaliser l'opération de déchiffrement (sur un texte en clair !).

La directive européenne [PEC 99] propose d'ailleurs d'utiliser la cryptographie à clé publique lorsqu'elle définit les « *données afférentes à la création de la signature* » comme étant « *des données uniques, telles que des codes ou des clés cryptographiques privées, que le signataire utilise pour créer une signature électronique* ».

Signature et chiffement asymétrique font tous deux intervenir une bi-clé. La différence tient dans l'ordre dans lequel les clés sont employées et leur propriétaire. On parlera alors de clé de signature et clé de vérification en lieu et place des termes de clé privée et clé publique afin d'éviter toute confusion entre signature et chiffrement.

Cryptographie	Signature numérique
Utilisation de la clé publique du destinataire pour chiffrer	Utilisation de la clé de signature (privée) du signataire (émetteur) pour signer : la valeur résultante s'apparente au déchiffrement du texte en clair
Le destinataire utilise sa clé privée pour déchiffrer le message reçu	Le destinataire utilise la clé de vérification (publique) du signataire pour vérifier l'origine et l'intégrité du message reçu. Le texte en clair est alors obtenu par une opération de chiffrement

Les problèmes de génération de clés que symbolise la figure 8 (et de tirage aléatoire d'exposants premiers [DEL 00] entre eux dans le cas de RSA [KAL 98a]) à l'aide de générateurs de nombres aléatoires [JAW 01] demeurent d'actualité : l'objectif de choisir les exposants de telle sorte que la connaissance de la clé de vérification et d'un ensemble de signatures donne le moins d'indices possible aux cryptanalystes en quête de la découverte de la clé de signature correspondante.

Le choix des nombres aléatoires retenus est aujourd'hui déterminé par les moyens technologiques dont peut disposer un attaquant et notamment la puissance de calcul mise à sa disposition. De fait, ces nombres sont choisis suffisamment grands, de telle sorte que le délai nécessaire à la recherche de la clé de signature soit démesuré par rapport à la durée de vie attribuée aux clés. Ainsi, lorsque l'attaquant prend connaissance de la clé de signature, cette dernière est désuète et inutilisable. Le fonctionnement du générateur aléatoire ISAAC [JEN 96] est décrit en annexe.

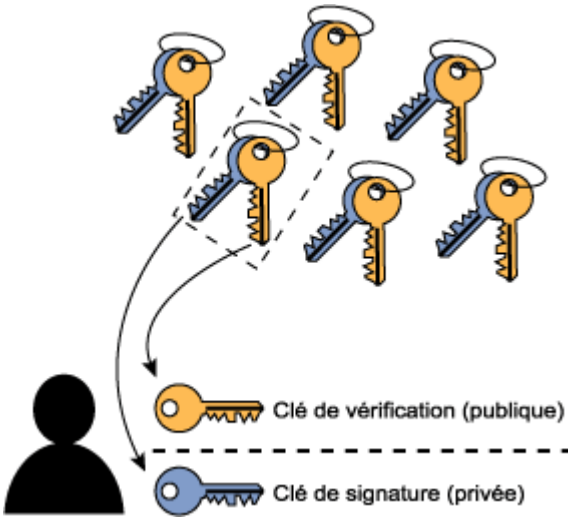

Figure 8 : tirage aléatoire d'une bi-clé

> *Rappel : une première intuition pourrait mener à penser qu'il suffit d'échanger la dénomination des clés, ce qui reviendrait à publier la clé privée et conserver secrète la clé publique. Cependant, comme dans le cas du système RSA présenté en annexe, la clé publique (vérification) contient moins d'informations que la clé privée (signature).*

1.2. Différents types de signatures électroniques

Comme nous l'avons vu dans le précédent chapitre, il existe différents types de signatures selon le degré de confiance et de fiabilité que le vérificateur peut leur accorder. La directive européenne [EES 99] opère une distinction entre signatures électroniques simples, avancées et sécurisées (figure 9).

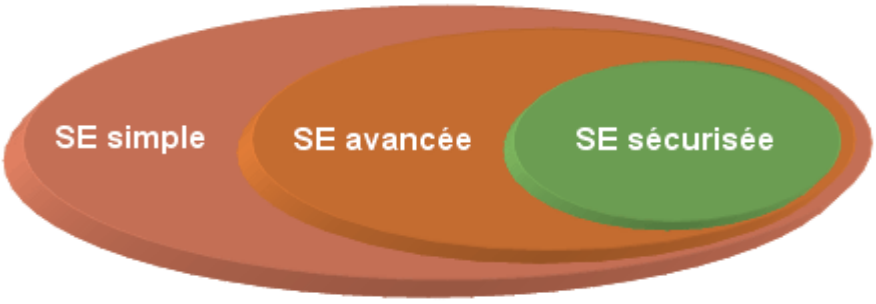

Figure 9 : signature simple, avancée et sécurisée

1.2.1. Signature électronique simple

La signature électronique simple est admissible dans le droit français (bien que refusée par la directive européenne) possède une valeur probante limitée. Le vérificateur ne dispose en effet que des informations essentielles à la validation de la signature : le certificat du signataire et la valeur numérique de la signature. La signature numérique permet de fait d'apporter la preuve de l'intégrité du contenu signé par rapport à la clé de vérification que renferme le certificat d'identité.

Il n'existe cependant aucune garantie relative à la validité des informations du certificat, telles que l'identité du signataire. C'est pourquoi il ne peut être mis en avant pour « *dûment identifier la personne dont [le contenu] émane* », comme le préconise la loi française [SEN 00].

1.2.2. Signature électronique avancée

Toute signature électronique avancée dispose d'une reconnaissance légale. La directive européenne [PEC 99] mentionne certaines exigences concernant la signature avancée qui doit ainsi :

- Etre liée de manière unique et non ambiguë au signataire : implicitement, deux titulaires différents ne doivent pouvoir partager la même bi-clé (dans le cas où la technologie cryptographique est utilisée). Des problèmes d'homonymie se posent inévitablement. Il appartient au vérificateur de s'assurer de l'identité (et éventuellement de l'habilitiation) du signataire.

- Permettre d'identifier le signataire : la directive ne précise pas que cette identification doit être critique. En effet, le recours au pseudonyme (ou identité d'emprunt oude fantaisie) peut être suffisant, par exemple dans le cas où l'acte juridique ne requérant pas de fournir une pièce d'état civil, le signataire ne souhaite pas divulguer son identité réelle à son interlocuteur.

- Etre créée par des moyens que le signataire peut garder sous son contrôle exclusif : le support de la signature est alors primordial. En effet, cette assertion implique que les données secrètes ne peuvent être dupliquées (sauf dans le cas d'une éventuelle négligence de la part du signataire).

- Etre liée aux données auxquelles elle se rapporte de telle sorte que toute modification ultérieure des données soit détectable : bien que des méthodes mathématiques existent, la technologie ne peut aujourd'hui assurer le signataire de la validation de sa signature sur le long terme (plusieurs dizaines d'années après la création de sa signature). Cet obstacle sera abordé par la suite.

De fait, cette signature ne peut cependant être vérifiée que sur le court terme (lors d'une transaction par exemple) et de fait perd sa valeur juridique dès lors qu'elle est

conservée en l'état sur le long terme [COT 03a]. Elle apporte néanmoins au vérificateur les informations nécessaires à sa validation immédiate.

1.2.3. Signature électronique sécurisée

La signature « sécurisée », que les techniciens nomment « qualifiée », reconnaît le consentement au contenu[8] [SEN 00]. Il s'agit d'une signature électronique avancée, reconnue dans le droit français, à laquelle s'ajoutent des informations susceptibles d'aider le vérificateur à valider la signature sur le long terme [COT 03a] en joignant notamment un jeton temporel d'horodatage au message signé. D'un point de vue technique, la signature peut être elle-même horodatée.

Le vérificateur est en sus assuré que le certificat du signataire et sa bi-clé de signature lui ont été remis dans un environnement sécurisé (au sens de la directive [PEC 99]) et que le signataire a utilisé un dispositif de création de signature fiable.

Le certificat permettant d'apposer une signature électronique qualifiée (« certificat qualifié ») répond également à certains critères, définis par décret [SEN 01b], dont le vérificateur doit prendre connaissance lors de la validation de la signature qualifiée.

Pour ce faire, le certificat qualifié mentionne soit :

- Un champ « certificatePolicies » [ETS 01] indiquant la référence à une politique de certification (PC). Cette dernière doit indiquer que les certificats délivrés sont qualifiés.

- Un champ « qCStatements » [SAN 01] qui, de par sa présence, indique que le certificat est qualifié.

1.3. Quelques normes de certificat numérique

Parmi les normes existantes, trois standards se font remarquer : le certificat PGP [MEL 01] [CAL 98], le certificat SPKI/SDSI [ELL 99a] [ELL 99b] et enfin le certificat X.509 [HOU 02].

Le standard PGP, présenté en annexe, repose sur une confiance mutuelle. Il ne peut être de fait reconnu comme modèle de confiance par les gouvernements.

[8] Bien que la première rédaction de [CNU 01] en 1997 mentionne l'engagement sur le contenu (art. 2.1 : « Une signature sous forme numérique intégrée, jointe ou liée logiquement à des données, utilisée par un signataire pour signifier son acceptation du contenu des données [..] »), le texte définitif n'en fait pas mention, à la demande de l'Angeleterre notamment. L'article 1316-4 de [SEN 00] indique que la signature « manifeste le consentement des parties […] ». De fait, la signature de l'une des parties induit le consentement de l'ensemble des parties concernées ?!

SDSI, également abordé en annexe, propose un modèle de confiance basé sur la clé de vérification (supposée unique) et non l'identité du détenteur de la bi-clé correspondante. Il s'attache avant tout à la définition de droits d'accès et non à l'identification certaine du détenteur de la bi-clé. [AUT 02] souligne de plus que le modèle SPKI (reposant sur SDSI) n'est pas implémenté actuellement dans les logiciels de signature électronique. De fait, SDSI n'est pas encore reconnu comme apte à répondre aux exigences juridiques. Cependant, quelques principes de ce modèle seront mis en œuvre dans le modèle de sécurité proposé, tels que les mécanismes de garde (« guardian ») et de droits d'accès à des ressources (« Access Control Lists ») [CLA 01].

Par contre, le certificat X.509 semble remplir les conditions requises par les textes juridiques. De plus, X.509 est le seul format de certificat d'identité mis en œuvre aujourd'hui dans les logiciels et infrastructures de confiance en conformité avec les exigences de la directive européenne et de la législation française.Nous nous intéressons par conséquent à ce standard, issu des travaux de l'organisme de standardisation Américain IETF (« Internet Engineering Task Force ») et particulièrement du groupe de travail PKIX.

1.4. Certificat numérique X.509

1.4.1. Aperçu général

Le certificat d'identité numérique X.509 [HOU 02], dit « de clé publique », définit les différents éléments nécessaires à l'approbation d'une clé de vérification (figure 10) alors que le vérificateur ne dispose d'aucune information préalable concernant l'identité du signataire. Afin d'être reconnu légalement, il doit être délivré par un PSCE accrédité mettant en œuvre une infrastructure à clé publique (ICP ou PKI) [AUT 02] [MBO 03]. Cette infrastructure, parfois controversée [ELL 00], obéit néanmoins à des règles précises de fonctionnement essentiellement dictées par les standards américains [ADA 99] et européens [ETS 02b].

L'ICP est dérivée d'un concept de [KOH 78] qui soulève le problème de la vérification du lien entre une personne et une clé publique via un certificat signé électroniquement par une tierce partie de confiance.

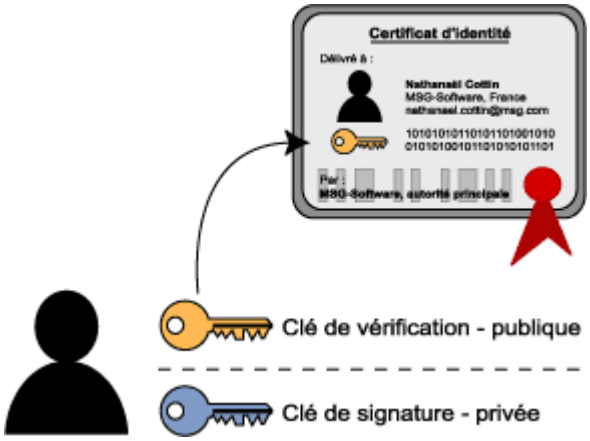

Figure 10 : vue d'ensemble d'un certificat X.509

1.4.1.1. Utilisation du certificat

En l'état (l'IETF a spécifié la version 3 du standard X.509 et élabore actuellement une version 4), le certificat numérique permet non seulement de s'assurer du lien entre une identité et une clé de vérification, mais également de vérifier que le message (ou le document) a été signé conformément à l'usage de la bi-clé spécifié.

Les usages de bi-clé les plus connus car les plus souvent usités sont :

- Le chiffrement de messages ou de clés.
- La signature de messages. Dans ce cas, cette option est incompatible avec le chiffrement de messages ou de clés, pour des raisons de sécurité évidents (le possesseur d'un tel certificat donnerait un nombre trop important d'indices concernant sa clé privée aux cryptanalystes) [MEL 01]. Un exemple d'attaque est présenté en annexe.
- La participation aux protocoles d'accord de clé (construction d'une clé secrète de session) tel que le protocole proposé par W. Diffie et M. Hellman et normalisé dans le standard PKCS#3 [RSA 93a]. Le chiffrement de clé doit être autorisé.

> *Note : d'autres usages sont possibles, tels que la signature des certificats ou des jetons d'horodatage [ADA 01]. Ils sont cependant réservés aux tiers de confiance [MEN 01], et au PSCE en particulier.*

Bien que le certificat renseigne le vérificateur sur l'usage supposé de la bi-clé certifiée, il ne donne cependant aucune information relative à l'environnement d'apposition de la signature ni aux politiques de sécurité requises.

1.4.1.2. Chemin de certification

Le certificat délivré par l'ICP du PSCE à l'utilisateur final est signé numériquement par le PSCE de telle sorte que la confiance dans le lien qui unit l'identité du signataire à sa clé publique soit égale à la confiance accordée au PSCE.

De fait, la signature du PSCE doit pouvoir être associée à l'identité du PSCE de manière non ambiguë. Le PSCE détient un certificat auto-émis, appelé certificat racine, qui sert de référence. Il est caractérisé principalement par le fait que la clé de vérification de la signature jointe au certificat est contenue dans celui-ci.

Soit un certificat numérique désigné par l'ensemble $\{I_1, K, V\}_{I_2, S_{K'}}$, sachant que :

- I_1 représente l'ensemble des informations relatives à l'identité du titulaire du certificat.
- K contient la clé publique de celui-ci combinée à des informations d'usage de clé.
- V indique la période de validité du certificat.
- I_2 permet de prendre connaissance de l'identité du PSCE émetteur du certificat.
- $S_{K'}$ fait référence à la signature numérique du PSCE émetteur, vérifiable à l'aide de la clé K'.

Alors un certificat racine s'exprime de la sorte : $\{I_{PSC}, K_{root}, V_{root}\}_{I_{PSC}, S_{K_{root}}}$.

> *Note : dans le cas d'une autorité intermédiaire, le certificat considéré comme racine est en réalité émis par un PSCE de niveau supérieur.*

Le PSCE peut alors soit émettre directement des certificats à ses clients à l'aide de sa clé de signature racine, soit créer un certificat intermédiaire, appelé certificat de distribution, qu'il utilise pour générer les certificats aux clients.

Cette dernière méthode, qui offre une plus grande souplesse de fonctionnement, peut être modélisée comme suit, étant supposé que la flèche indique l'émission d'un certificat, conformément à [AUT 02] :

$$\begin{cases} \{I_{PSC}, K_{root}, V_{root}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} \\ \{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{user}, K_{user}, V_{user}\}_{I_{dist}, S_{K_{dist}}} \end{cases}.$$

1.4.2. Définition technique

Le certificat de clé publique X.509v3 est composé d'informations permettant d'identifier son porteur [AUT 02] (également appelé détenteur), telles que son nom, son prénom, son adresse complète (personnelle ou professionnelle) incluant le code du pays, ses adresses de courriel.

Selon l'utilisation future du certificat, celui-ci peut être délivré à titre personnel ou professionnel. Dans ce dernier cas, le certificat pourra de plus inclure le nom de l'organisation (par exemple, l'entreprise, l'administration, etc.) et le service dont dépend le demandeur.

L'organisme émetteur du certificat est identifié de même. Il fut question d'inclure un nom distinctif (tel un code) permettant de le nommer sans ambiguité. Bien que toujours présent dans la norme, cet identifiant est aujourd'hui obsolète.

En sus des données de la clé publique du porteur, le certificat mentionne une période de validité sous la forme d'un intervalle borné par une date de début et une date de fin de validité. Bien que l'on puisse utiliser le certificat en dehors de sa période de validité pour signer ou chiffrer des messages, les outils conformes à la norme X.509 devront rejeter ces tentatives d'utilisation frauduleuses.

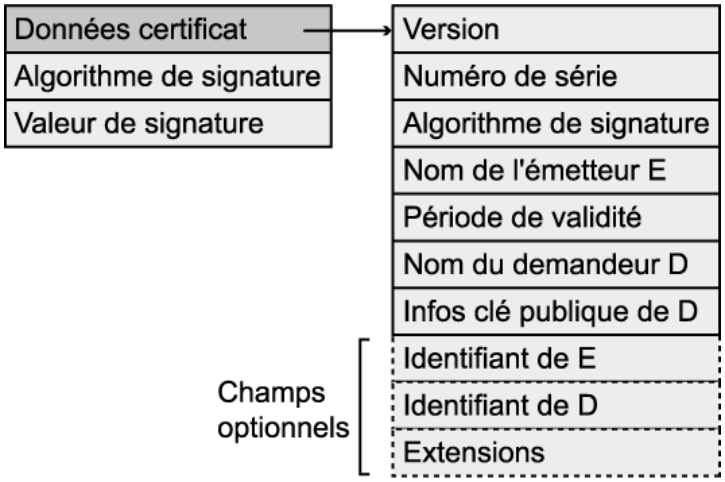

Figure 11 : structure interne d'un certificat X.509v3

Afin d'être ouvert à de futures évolutions tout en conservant une compatibilité ascendante, le certificat inclut un champ optionnel appelé « extensions » (figure 11). Une extension est un triplet $\{id, c, v\}$ tel que id caractérise le contenu de l'extension (par l'utilisation d'identificateurs d'objets normalisés), c renseigne sur la criticité de l'extension et v indique la valeur associée à l'extension. La criticité est définie comme suit [AUT 02] :

- Tout certificat doit être rejeté s'il mentionne une extension critique non reconnue par l'outil en charge d'utiliser ce certificat.
- Toute extension critique doit être prise en compte.
- Une extension non critique et reconnue doit être traitée.
- Une extension non critique et non reconnue n'est pas prise en considération.

> *Note : le terme porteur est considéré au sens large comme un individu, une machine ou un programme et se rapproche ainsi du terme anglo-saxon « principal ».*

1.5. Cycle de vie du certificat

Après avoir été généré, un certificat n'est généralement pas utilisable directement. En effet, une procédure de retrait du certificat auprès de l'autorité de certification doit aboutir avant que le certificat soit actif. Le certificat est mis en état suspendu tout au long de la procédure (figure 12). Bien souvent, cet état est « omis » par les politiques de certification (tant au niveau du MINEFI que de l'ADAE), même si la loi type de la CNUDCI [CNU 01] y fait référence (article 11).

Comme indiqué précédemment, chaque certificat X.509 n'est utilisable que durant l'intervalle défini par ses dates de validité. Lorsque la date de fin de validité est dépassée, ce dernier expire et doit être renouvelé. Le renouvellement implique la création d'un autre certificat et la génération d'une nouvelle bi-clé.

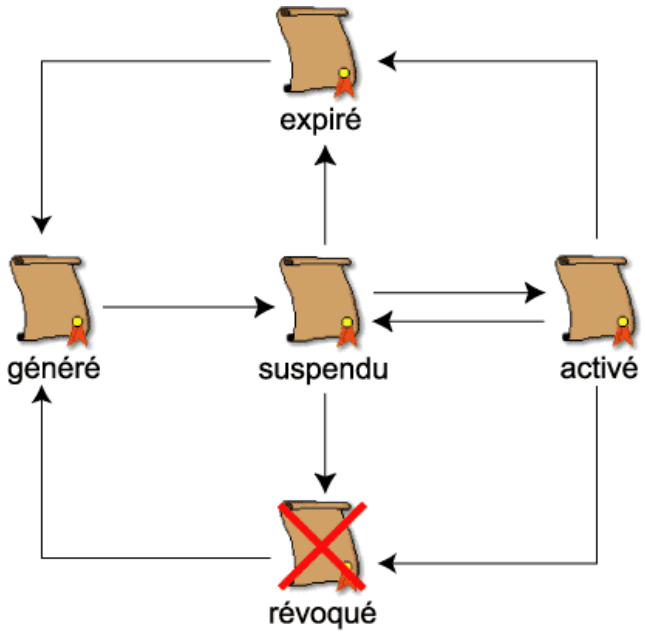

Figure 12 : cycle de vie d'un certificat X.509

Il se peut que la bi-clé soit compromise (une tierce personne ayant réussi à découvrir la clé privée) ou que son détenteur l'ait perdu. Dans ce cas, une procédure similaire à la mise en opposition d'une carte de crédit est entamée. A l'issue de cette procédure, le certificat (et par conséquent, le bi-clé) est révoqué par l'ICP du PSCE émetteur. Il est alors inutilisable. Un nouveau certificat (comprenant ou non une nouvelle bi-clé, selon la procédure de renouvellement en vigueur) doit être généré.

1.5.1. Révocation d'un utilisateur

La révocation se traduit par l'ajout de l'identifiant du certificat à la liste de révocation (ou « Certificate Revocation List », CRL). Cette dernière contient l'ensemble des certificats révoqués par une autorité de certification donnée entre deux intervalles de temps.

Il convient de noter à ce propos que seuls les certificats révoqués et suspendus sont inscrits dans les CRL. Ceci implique que tout certificat arrivé au terme de sa vie n'est retiré des CRL futures. Les problèmes engendrés par la présence des certificats suspendus dans les CRL menant à suggérer une modification du format de ces dernières sont traités par [COT 03a] et reprises en annexe.

1.5.2. Révocation d'une autorité

Il est également possible recourir à des listes de certificats d'autorités révoqués (« Authority Revocation List », ARL). Comparée à une CRL, une ARL n'a pas besoin d'être mise à jour régulièrement, étant donné qu'une autorité reconnue est supposée disposer d'un degré de sécurité de son système informatique en adéquation avec

ses responsabilités. Ainsi, une ARL a pour objectif de demeurer vide. Dès lors qu'une autorité compromet sa signature, une nouvelle entrée est ajoutée dans l'ARL. Les usagers sont tenus au courant par notification (qui peut être médiatique) de cette mise à jour.

Ce cycle de vie est primordial lorsqu'il s'agit de vérifier une signature plusieurs mois, voire plusieurs années après sa constitution. Le contrôle de révocation est également nécessaire lorsque le destinataire reçoit un message signé. Ces processus sont décrits dans les paragraphes suivants.

2. Utilisation de la signature électronique

Comme nous allons le montrer, il existe une différence fondamentale entre signature numérique et signature électronique [AUT 02] [COT 02a] [PIN 00]. En effet, la signature numérique ne s'intéresse qu'aux bits de la signature et à la bi-clé alors que la signature électronique offre un caractère légal puisqu'elle s'attache également à l'identité du signataire et au contrôle de la validité de son certificat numérique. Ces données doivent être consultables par le vérificateur afin qu'il puisse déterminer la fiabilité de la signature qui lui est présentée.

2.1. Utilisation de la signature numérique

2.1.1. Création d'une signature numérique

La création d'une signature numérique suppose que l'on soit en possession d'un bi-clé. Elle fait appel à la notion d'empreinte numérique. En effet, pour des raisons de performances, la signature ne s'applique pas directement à l'information à signer mais à son résumé numérique (empreinte).

2.1.1.1. Notion de résumé de message

Le résumé de message [MEN 01] [COT 02a], également appelé empreinte numérique (de l'anglais « hash »), permet de condenser une information numérique et ainsi créer une seconde information de telle sorte que ces deux informations sont liées logiquement.

Un résumé sécurisé implique les propriétés suivantes :

- Il est issu uniquement de l'information à signer M et d'un algorithme dit de « hachage » sécurisé p : $H = hash_p(M)$, lorsque $hash$ désigne une fonction de calcul d'empreinte. Il peut par exemple s'agir de $hash_{SHA-1}(M)$ ou de $hash_{MD5}(M)$. Une annexe présente sur ce propos quelques algorithmes de calcul d'empreinte.

- Il est de taille fixe : $hash : (0,1)^* \mapsto (0,1)^n$, $n > 0$. Cette taille varie selon l'algorithme utilisé. Pour des raisons de sécurité, la taille minimum recommandée par le « Digital Signature Standard » [NIS 00] est de 160 bits.

- Il ne renseigne pas sur le contenu de l'information source (la fonction de génération d'empreinte est non inversible).

- Il doit être fortement résistant aux collisions : deux sources différentes ne doivent produire un même résumé à l'aide du même algorithme de calcul.

Il s'agit d'un raisonnement informatique et non mathématique (l'espace des résultats est fini et de cardinalité 2^n).

Parmi les algorithmes les plus employés actuellement nous pouvons citer SHA-1 [NIS 95], MD5 [RIV 92] et RIPEMD-160 [DOB 96]. Le principe de fonctionnement de l'algorithme MD5 est proposé en annexe.

2.1.1.2. Intégration à la signature numérique

Soient un algorithme de calcul d'empreinte répondant aux critères précédemment cités ainsi qu'une bi-clé. La création d'une signature numérique s'effectue en deux étapes successives que présente la figure 13 :

1. Génération d'une empreinte numérique à l'aide d'un algorithme d'empreinte sécurisé. Cette empreinte sera par la suite qualifiée d'empreinte « originale ».

2. Transformation de cette empreinte à l'aide d'une clé de signature. Dans le cas d'une bi-clé RSA, cette étape s'apparente à un déchiffrement.

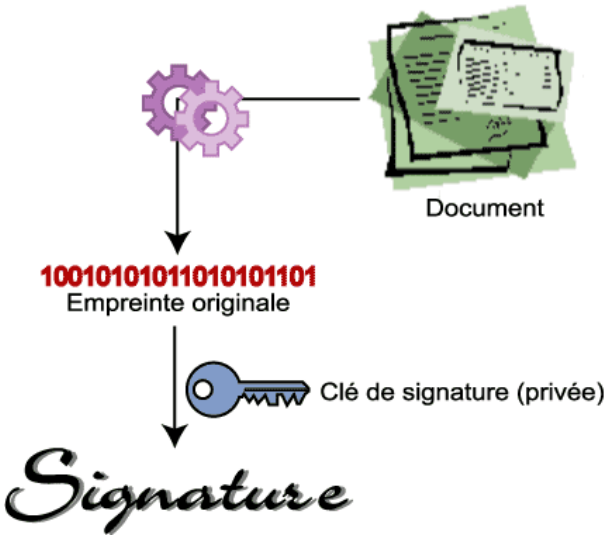

Figure 13 : création d'une signature numérique

Le résultat obtenu est un ensemble de bits dont la longueur dépend de l'algorithme d'empreinte et de la taille de la clé de signature utilisés. Cette valeur binaire peut être transmise au destinataire afin qu'il puisse vérifier l'authenticité de l'information reçue.

2.1.2. Vérification d'une signature numérique

Un individu (ou un outil informatique) ayant reçu une signature numérique procède à sa vérification. Pour ce faire, il doit, conformément à la figure 14 :

1. Entrer en possession de l'ensemble $\{M, DS, H, K\}$ composé de l'information signée M, de la signature numérique DS, de l'algorithme d'empreinte H utilisé par le signataire et de la clé de vérification (publique) K.

2. Déverrouiller l'empreinte « originale » en appliquant la clé de vérification sur l'empreinte codée. Si l'obtention de cette empreinte échoue, la clé de vérification employée ne correspond pas à la clé de signature (il ne s'agit pas de la bi-clé du signataire).

3. Générer une empreinte « témoin » à l'aide de l'algorithme indiqué par le signataire.

4. Effectuer la comparaison des empreintes qui doivent être égales pour que la signature numérique soit acceptée.

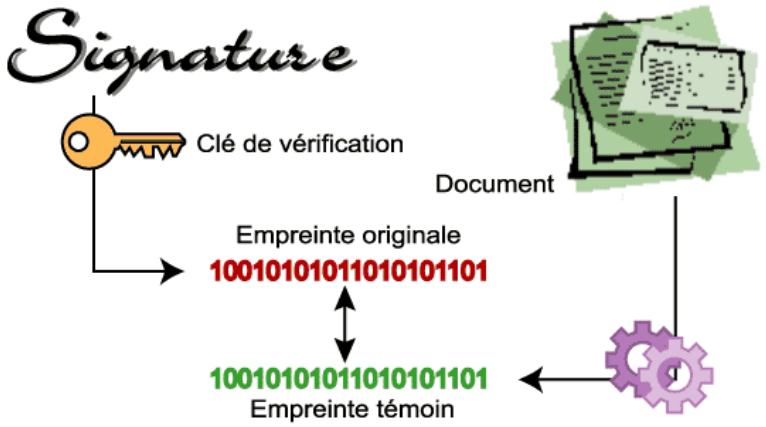

Figure 14 : vérification d'une signature numérique

Cette procédure ne constitue en réalité qu'une première dans la validation d'une signature électronique car aucun lien entre clé de vérification et signataire ne peut être apporté avec certitude par exemple.

> *Bien que règne la confusion entre « vérification » et « validation » de signature, nous employons « vérification » lorsqu'il s'agit d'une opération cryptographique (liée à la signature numérique) et*

« validation » lorsqu'interviennent d'autres paramètres, tel que le positionnement de la signature dans le temps [AUT 02]. La « validation » s'opère ainsi sur une signature électronique.

2.2. Utilisation de la signature électronique

2.2.1. Création d'une signature électronique

Le procédé de création d'une signature électronique est en tout point similaire à la création d'une signature numérique. Cependant, à l'ensemble des bits résultant du calcul de la signature numérique sont ajoutées des informations relatives à :

- L'identité du signataire. Il s'agit de son certificat numérique [HOU 02] [ARC 00].
- L'ensemble des politiques de signature [ROS 01] suivies lors du processus de création de la signature numérique. Le respect de ces politiques conforte le vérificateur dans l'acceptation ou le refus de la signature qui lui est soumise.
- La production de références permettant d'attribuer une validité au certificat, telles qu'un chemin valide de certification menant au certificat racine de l'autorité de certification émettrice.
- La nature de la transaction signée : il peut s'agir d'un paiement en ligne, d'un document juridique, d'un contrat, etc.
- La date de création de la signature numérique. Ce peut être une date locale (date revendiquée ou « claimed date ») ou une date attestée par une autorité d'horodatage (« certified date ») appelée valeur d'horodatage ou estampille temporelle [MEL 01].

Ces informations doivent permettre au vérificateur de contrôler que la signature a été conçue dans un environnement sécurisé et à l'aide de procédés fiables de nature à garantir son intégrité.

La date associée à la signature permet de la situer dans le temps. Cette information est nécessaire pour déterminer la validité de la signature à laquelle il se rapporte. Cette date peut être omise au profit de la datation du message signé.

Nous verrons que ce type de signature ne permet pas encore de répondre à l'ensemble des besoins qui peuvent être rencontrés, tels que la multi-signature [RIE 03b], la sur-signature, la contre-signature et la conservation de la validité sur le long terme des documents signés électroniquement.

2.2.2. Validation d'une signature électronique

Pour valider une signature électronique, le destinataire doit procéder à la vérification de la signature numérique. Lorsque cette étape est franchie, il doit s'assurer que :

- La nature de la transaction signée correspond effectivement à l'information reçue.
- Le certificat du signataire est valide au moment de la signature. Pour cela, le destinataire doit tenir compte de la date mentionnée dans la signature. Bien entendu, une date certifiée (horodatée) sera considérée comme incontestable, contrairement à une date locale. A ce propos, le destinataire est en droit de refuser la signature dans le cas où aucune date certifiée n'est mentionnée.
- Les politiques de signature attestent d'un environnement de création de la signature en accord avec le niveau de sécurité souhaité.
- La longueur de la clé de signature apporte une protection suffisante contre les attaques de clé de type recherche exhaustive.

> *Note : lorsqu'un horodatage est présent, le vérificateur doit aussi s'assurer de la validité du certificat de l'autorité d'horodatage avant d'être en mesure de valider la signature électronique qui lui est présentée. Les problèmes liés aux possibles attaques sur des informations horodatées mentionnées par [GUE 03] et [GON 90] ne font pas l'objet de la présente étude.*

Il est ainsi possible de définir quatre niveaux d'acceptation d'un message comportant une signature électronique :

- L'acceptation de la signature numérique : l'empreinte « originale » a été décodée puis comparée avec succès à l'empreinte « témoin ».
- L'acceptation de la clé de signature : la bi-clé du signataire est de longueur suffisant eu égard aux technologies de cryptanalyse.
- L'acceptation du certificat du signataire comme non réfutable.
- L'acceptation du message comme authentique en fonction des contraintes liées à l'environnement d'apposition de la signature conformément aux politiques de création applicables.

De fait, valider une signature électronique fait appel à une phase critique de validation de certificats électroniques (que ce soit celui du signataire ou ceux des éventuelles autorités d'horodatage).

2.2.3. Validation d'un certificat numérique

Pour qu'une signature électronique soit reconnue comme valide, le certificat numérique du signataire doit être vérifié. En particulier, il doit être prouvé qu'il n'était pas révoqué au moment de la génération de la signature.

La validation d'un certificat numérique à un instant donné demande deux étapes :

1. La vérification du chemin de certification.

2. La vérification de la validité du certificat qui ne doit être ni révoqué, ni suspendu au moment de la création de la signature.

2.2.3.1. Vérification du chemin de certification, CTL

Le chemin de certification (« certification path », CP) est une chaîne linéaire de certificats (sauf en cas de certification croisée [TRU 03] présentée en annexe). Il débute par un certificat auto-émis (le certificat racine de l'autorité de certification) et se termine par le certificat du signataire. Chaque certificat (hormis le certificat racine) est émis par son prédécesseur dans le chemin de certification.

Un chemin de certification peut ainsi être représenté algébriquement par la relation de récurrence suivante :

$$\begin{cases} CP_0 = sign(pub_0, priv_0) \\ CP_{i+1} = sign(pub_{i+1}, priv_i) \end{cases}.$$

Cette relation signifie qu'une clé de vérification donnée (pub_i) est certifiée par la signature de son émetteur (à l'aide de sa clé de signature $priv_{i-1}$). Ceci est également vérifié pour des certificats numériques. Dans ce cas, l'ensemble du certificat est certifié, et non uniquement la clé de vérification. Le chemin de certification est alors le suivant :

$$\{I_{PSC}, K_{root}, V_{root}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{user}, K_{user}, V_{user}\}_{I_{dist}, S_{K_{dist}}}.$$

Le vérificateur doit disposer d'une liste de certificats d'autorités qu'il reconnaît comme étant de confiance afin d'accepter ou rejeter le chemin de certification que lui indique le signataire. Cette liste porte le nom de « liste de certificats de confiance » (« Certificate Trusted List », CTL). Elle peut également porter le nom de magasin de racines de confiance (sous Windows en particulier). Ces autorités peuvent être « principales » (racines) ou « intermédiaires ».

Au cours de cette première phase, le vérificateur recherche un chemin de certification valide. Le chemin sélectionné débute par un certificat de confiance membre de sa CTL et aboutit au certificat du signataire. L'autorité ayant émis le certificat racine instaure la confiance entre le vérificateur et le signataire. Par exemple, le certificat $\{I_{PSC}, K_{root}, V_{root}\}_{I_{PSC}, S_{K_{root}}}$ faisant partie de sa CTL, le vérificateur peut accepter le chemin menant au certificat du signataire en procédant à la validation du certificat intermédiaire (1) puis du certificat final (2) :

$$\begin{cases} \{I_{PSC}, K_{root}, V_{root}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} & (1) \\ \{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{user}, K_{user}, V_{user}\}_{I_{dist}, S_{K_{dist}}} & (2) \end{cases}.$$

Bien entendu, chaque certificat mis en jeu dans le chemin de certification doit être vérifié et valide. La validité de chaque certificat est alors déterminée par la validité de son prédécesseur (mis à part le certificat racine) et par ses dates de validité. En effet, un certificat doit être considéré comme non valide lorsque de sa durée de validité est complètement ou partiellement en dehors de la durée de validité de son émetteur.

Ainsi, tout certificat utilisateur délivré par un PSCE selon la chaîne $\{I_{dist}, K_{dist}, V_{dist}\}_{I_{PSC}, S_{K_{root}}} \rightarrow \{I_{user}, K_{user}, V_{user}\}_{I_{dist}, S_{K_{dist}}}$ n'est recevable que si :

$$\begin{cases} V_{dist} = \{from_{dist}, to_{dist}\}, \quad from_{dist} < to_{dist} \\ V_{user} = \{from_{user}, to_{user}\}, \quad from_{user} < to_{user} \\ (from_{dist} \leq from_{user}) \wedge (to_{dist} \geq to_{user}) \\ \exists revoke_{dist} \rightarrow revoke_{dist} > from_{user} \quad (I) \end{cases}.$$

Le dernier point, marqué (I), indique l'impossibilité pour un PSCE de générer des certificats dès lors que son certificat de distribution est révoqué.

2.2.3.2. Vérification de la validité du certificat

Etant donné une date de signature (certifiée ou prétendue), le vérificateur doit s'assurer que le certificat dont il vient de vérifier le chemin de certification par le biais de sa CTL a été utilisé :

- Conformément à l'usage de clé mentionné dans le certificat.
- Alors que ledit certificat était actif, c'est à dire au cours de sa période de validité et alors qu'aucune notification de révocation ou de suspension n'avait été préalablement publiée.

Deux moyens standardisés sont mis à sa disposition par les autorités de certification : les listes de révocation [HOU 02] et le protocole en ligne OCSP [MYE 99]. Tous deux permettent de déterminer si le certificat est révoqué au moment

du contrôle et, dans une moindre mesure, s'il était révoqué à l'instant où la signature a été générée :

- La liste de révocation (« Certificate Revocation List », CRL), mise à jour régulièrement afin que le vérificateur dispose d'informations fiables. Le principal reproche que l'on peut attribuer aux CRL (ou aux CRL intermédiaires appelées delta-CRL [HOU 02]) est relatif au délai entre deux mises à jour. Une liste de révocation ne permet donc pas de connaître le statut exact d'un certificat dans le cas où il n'apparaît pas dans celle-ci. Bien que l'instant exact de signature soit bien souvent inutile lors de la constitution d'actes juridiques, les systèmes d'information doivent être en mesure de déterminer avec précision la validité des signatures.

- Le protocole de validation en ligne des certificats (« Online Certificate Status Protocol », OCSP) permet l'accès à une information en temps réel (ou avec un léger décalage dans le cas où les réponses sont groupées par l'algorithme de Merkle [LIP 99] [MER 80]). Le serveur mettant en œuvre ce protocole n'est pas limité par un délai de mise à jour des informations de statut. Une version améliorée de ce protocole est proposée en annexe.

La notion d'« instant de création signature » beaucoup utilisée par les standards, notamment [ETS 02b] [ETS 02c] [HOU 02] [ADA 99], fait appel à une autorité de confiance chargée de délivrer des horodatages certifiés. D'autres protocoles font appel à la notarisation électronique [ROO 99] [TRU 03] pour laquelle le statut du certificat du signataire est certifié par un notaire lors de chaque création de signature.

Nous allons discuter des attraits des horodatages et de leur intégration dans la signature électronique.

2.3. Horodatage sécurisé

2.3.1. Présentation

L'horodatage sécurisé [GTH 03] [ETS 02a] consiste à attribuer une date certaine à une donnée informatique. Le protocole d'horodatage « Time-Stamp Protocol » ou TSP [ADA 01], illustré par la figure 15, construit une date signée électroniquement à partir d'une empreinte numérique. Cette date prouve l'existence préalable de l'empreinte, et par conséquent l'existence de l'information utilisée pour la génération de cette empreinte dès lors que celle-ci a été créée à l'aide d'un algorithme sécurisé.

Ainsi, l'horodatage permet par exemple de protéger des œuvres numériques en apportant une preuve d'antériorité.

Un horodatage se distingue d'une valeur de date issue d'un système non certifié par le fait qu'il peut être délivré par une autorité neutre de confiance. L'horodatage émis par cette autorité est protégé par sa signature électronique et serait susceptible

de disposer d'une reconnaissance juridique, contrairement à toute date non certifiée qui peut être falsifiée.

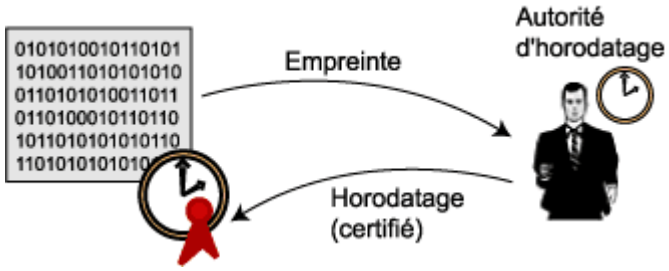

Figure 15 : demande d'horodatage d'une empreinte numérique

2.3.2. Horodatage d'une signature électronique

Un cas particulier concerne l'horodatage de l'empreinte numérique d'une signature électronique. Il est alors possible d'apporter la preuve que la signature a été apposée sur un contenu numérique avant la valeur de son horodatage [ADA 01].

Cette fonction est utilisée afin de valider les signatures électroniques dans le temps en renseignant le vérificateur sur le statut du certificat du signataire à la date indiquée par l'horodatage [ETS 02a].

Une notion annexe décrite dans [ADA 01] concerne la protection de la signature horodatée par la signature de l'autorité d'horodatage. En effet, par définition, toute modification de l'information horodatée entraîne l'invalidité de la signature numérique de l'autorité d'horodatage. Il est alors possible de faire reposer la sécurité de la signature horodatée sur la signature de l'autorité d'horodatage. L'utilisateur peut ainsi signer à l'aide de la bi-clé de faible taille, dès lors que l'horodatage est apposé dans un délai plus court que le délai nécessaire à la découverte de sa clé de signature (selon une attaque par recherche exhaustive en particulier).

2.3.3. Moment d'apposition d'un horodatage

Une question évoquée par le guide de l'horodatage sécurisé [GTH 03] concerne le moment le plus opportun pour horodater un message (signé ou non) ou une signature électronique. Il s'agit de déterminer la valeur ajoutée d'un horodatage précédent ou suivant la création d'une signature électronique.

Il ressort de l'étude [GTH 03] que signature et horodatage sont séparés (juridiquement et techniquement). En effet, signer puis dater le message signé est utilisé pour préparer un envoi électronique (pour un téléservice par exemple). Le message et sa signature sont ici d'importance égale. Dans le cas où le message est

d'abord daté puis signé, la signature revêt une importance prépondérante par rapport au message.

2.3.3.1. Horodatage d'un message

Joindre une date certaine à un message permet de majorer sa date de construction [ADA 01]. Bien qu'il soit possible d'horodater tout type de message (notamment non signé et sans valeur juridique), il est intéressant d'horodater des messages dont le contenu possède des effets juridiques.

Dans le cas où le message horodaté est signé, l'horodatage permet non seulement de protéger l'authenticité du message mais aussi de dater la signature qui lui est jointe.

2.3.3.2. Horodatage d'une signature

Comme indiqué précédemment, un horodatage apposé à un message préalablement signé est suffisant pour garantir l'authenticité globale du message et dater la signature. Cependant, certaines procédures (qui demandent par exemple la collecte de plusieurs signatures avant de procéder à l'horodatage du message signé) nécessitent l'individualisation des horodatages qui s'appliquent alors aux signatures. L'intérêt de l'horodatage final du message signé n'est pas remis en cause.

Un horodatage h_2 apposé sur une signature électronique permet de la situer dans le temps. Cette information est primordiale puisqu'elle permet de valider la signature électronique à laquelle l'horodatage se rapporte. En effet, si d_s désigne la date effective de création de la signature alors l'horodatage donne la relation $d_s < h_2$ pouvant se traduire par la relation suivante : « la date mentionnée dans l'horodatage d'une signature électronique majore l'instant de création de la signature dont il se rapporte ».

Cependant, cet horodatage n'indique pas si cette signature a été générée quelques secondes ou plusieurs mois avant la date du jeton d'horodatage. Ceci peut poser problème lorsqu'il s'agit d'apporter la preuve que le certificat du signataire n'était pas suspendu au moment de la génération de la signature. Ainsi, un horodatage h_1 apposé au contenu signé préalablement à la signature offre l'avantage de donner une borne inférieure prouvant que la signature a été générée après cette date : $h_1 < d_s$: « un horodatage apposé à un contenu devant être signé minore l'instant de création de la signature sur ce contenu horodaté ».

De fait, $d_s \in]h_1; h_2[$. La valeur de la signature peut donc être déterminée avec plus de précision qu'en utilisant h_2 seul.

Les valeurs des horodatages indiquent généralement des dates incluant des millisecondes, voire au-delà. Nous pouvons nous interroger sur la nécessité d'une

telle précision dans le cadre de la signature de contenus juridiques qui, sur format papier, se contentent de la date du jour et éventuellement de l'heure, non certifiées mais acceptées comme authentiques par les signataires.

> *Dans certains cas, des dates non certifiées peuvent être employées avec les risques d'erreur que nous avons évoqués précédemment. Par exemple, lors de la signature d'un contrat, les parties signataires peuvent s'entendre sur une date unique qui fera foi dès lors qu'elle aura été signée par chacune d'elles.*

Nous avons discuté de la nécessité de recourir à un double horodatage. Le processus est le suivant :

- Avant signature : le contenu à signer est horodaté (ou daté par un autre moyen tel qu'une date prétendue). L'ensemble composé du contenu et de la date (horodatée ou non) est alors signé. Il est techniquement possible de ne signer que l'horodatage puisque celui-ci est fortement lié au contenu horodaté, mais un processus homogène, applicable à fois pour un horodatage et une date non certifiée, est préférable. La date de génération de la signature apposée au contenu horodaté est ainsi minorée par la valeur de date de l'horodatage.

- Après signature : un second horodatage scelle la signature et apporte la preuve que la signature a été crée au préalable. Cet horodatage est donc un majorant de la date de génération de la signature.

L'instant de création de la signature est donc compris dans un intervalle défini par ces deux valeurs de dates. Plus cet intervalle est réduit, plus cet instant est déterminé avec précision.

2.4. Notarisation électronique

La notarisation électronique [ROO 99] [TRU 03] fait intervenir une autorité de confiance appelée notaire[9], que [ROL 93] décrit comme sûre ou aveugle selon que ce tiers prend connaissance ou non des messages qui lui sont envoyés. L'appellation « notaire », bien que controversée [AUT 02] car empruntée par les informaticiens au monde juridique, provient de la conservation par le notaire électronique de la trace

[9] Nous conservons ce terme anglo-saxon pour demeurer en accord avec la littérature scientifique, même si « contrôleur de signature » semble moins sujet à confusion pour les juristes, le notaire « traditionnel » s'intéressant plus au document et aux procédures qu'à la signature en elle-même.

de la demande de validation (et de la réponse fournie en retour) à des fins de non répudiation.

Lorsqu'une signature électronique lui est transmise, le notaire délivre des pièces justificatives authentiques qui attestent la validité de la signature, telles qu'un certificat de validité [TRU 03] (figure 16). Ces pièces peuvent ensuite être utilisées pour contrôler *a posteriori* la validité de la signature à laquelle elles se rapportent.

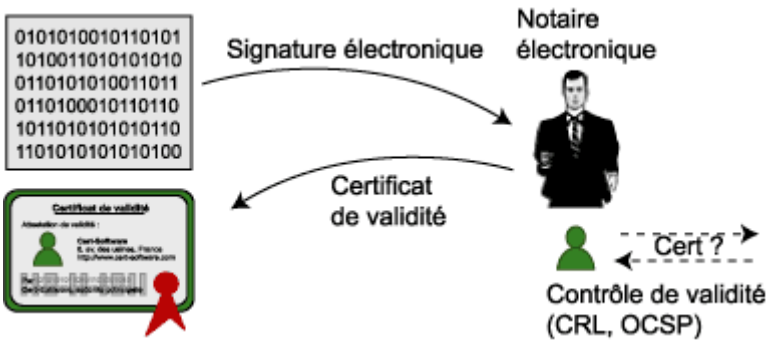

Figure 16 : notarisation à l'aide d'un certificat de validité

De manière semblable à l'horodatage, la notarisation d'une signature électronique protège cette dernière [COT 03b]. En effet, la sécurité de la signature notariée peut être déléguée au notaire électronique. Pour plus de sécurité, le notaire peut protéger sa propre signature en lui apposant un ou plusieurs horodatages sécurisés.

2.5. Visualisation de données signées électroniquement

La visualisation d'un document manuscrit permet d'appréhender le contenu signé de même que les signatures apposées. Il est relativement aisé de s'apercevoir d'un artifice ayant pour objet de transformer le contenu ou les signatures ou de masquer certaines informations.

L'étude menée par [COT 03e] montre qu'il est beaucoup moins évident de s'assurer d'une visualisation correcte d'un document électronique formatté, rendue nécessaire par le consentement au contenu. En effet, l'intelligibilité est liée à l'interprétation des informations binaires par un outil informatique de visualisation. Même si les processus de création et validation de signatures sont intègres et non remis en cause, il s'agit d'apporter la certitude :

- De signer ce que l'on « pense » signer, connu sous l'acronyme WYSIWYS (« What You See Is What You Sign ») [COT 03e] [MIS 04].

- De consulter un document préalablement signé de telle sorte que la visualisation (du document et des signatures) soit identique à celle lors de l'apposition des signatures [COT 03e].

Il est notamment possible de créer un document dont l'affichage est polymorphe (diffère en fonction de l'environnement de visualisation : modèle ou version du visualiseur, système d'exploitation ou date système par exemple).

3. Structure d'une signature électronique

Comme nous souhaitons que les échanges entre objets répartis soient signés électroniquement, se pose la question de structurer les champs d'une signature électronique afin de la joindre aux requêtes et réponses échangées.

3.1. Première approche : la signature numérique

Bien que nombre d'articles fassent référence à la signature électronique, peu d'entre eux font état d'un besoin de définir l'ensemble des éléments constitutifs de la signature électronique du point de vue informatique. Une vision impropre aboutit à la définition que l'on retrouve dans le standard X.509 : la signature est considérée comme étant la conjonction d'un algorithme de signature et d'une valeur sous forme de bits. L'algorithme indique en particulier l'algorithme que doit utiliser le vérificateur pour être en mesure de vérifier la signature.

Le langage « Abstract Syntax Notation One » (ASN.1) [DUB 00] est le langage de description des structures de données relatives à la signature électronique. Il est employé dans de nombreux protocoles du fait qu'il propose non seulement un formalisme mais également différents procédés d'encodage permettant de représenter informatiquement les structures de données sous forme de flux d'octets.

Par exemple, un certificat X.509 est défini en langage ASN.1 comme suit :

```
X509Certificate ::= SEQUENCE {
  tbsCertificate       TBSCertificate,
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING
}
```

Où « tbsCertificate » contient l'ensemble des informations relatives au détenteur du certificat (notamment à sa clé publique) et à l'autorité émettrice.

Le standard OCSP propose quant à lui les descriptions suivantes :

```
Signature ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING,
  certs   [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}
```

```
BasicOCSPResponse ::= SEQUENCE {
  ...
  signatureAlgorithm   AlgorithmIdentifier,
```

```
  signature            BIT STRING,
  certs   [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}
```

La confusion s'installe dès lors que les standards reconnus du groupe de travail PKIX de l'IETF, tels le « Cryptographic Message Syntax » (CMS) [HOU 99] ou encore [PIN 01b] représentent la valeur numérique de la signature sous la forme d'une succession d'octets (et non de bits).

```
SignerInfo ::= SEQUENCE {
  ...
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature          SignatureValue,
  ...
}

SignatureValue ::= OCTET STRING
```

Par la suite, nous considérerons la signature numérique comme composée des champs suivants :

```
Sign ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING
  -- Signature value
}

AlgorithmIdentifier ::= OBJECT IDENTIFIER
```

Note : la vérification d'une signature s'intéresse uniquement aux bits qui la composent alors que la validation tient compte de l'environnement de signature et des informations jointes qui permettent d'accepter un document comme authentique ou, au contraire, de le rejeter comme étant faux.

Lorsqu'un tel message est reçu, aucune information relative à la clé utilisée n'est disponible. Il faut donc soit joindre la clé de vérification au message, soit indiquer un autre moyen permettant de prendre connaissance de cette clé (une adresse, une base de données, un serveur de clé, etc.).

Lorsque cette clé est connue, le destinataire peut procéder à la vérification de la signature qui lui est présentée. Cette étape lui permet de valider et surtout rejeter la signature lorsque les bits qui la composent ne peuvent être déchiffrés par la clé de vérification. Cependant, une information fondamentale, relative au degré de confiance qu'il peut accorder à la fois à la clé et au lien avec le signataire, lui manque lorsque cette première étape a abouti. Une distinction s'opère donc entre signature numérique et signature électronique [PIN 00] [AUT 02].

3.2. Seconde approche : vers la signature électronique

Un intrus ne doit pouvoir transformer son identité en prenant celle d'une autre entité, puis envoyer des réponses erronées aux clients sans être détecté. La signature ne doit être ni fraudable, ni répudiable [PIN 01c]. Un client doit donc être en mesure d'apporter la preuve que le serveur contacté est intègre et que la clé de vérification qu'il utilise est bien celle du serveur concerné. Les informations contenues dans la signature doivent apporter une réponse (même partielle) à ces contraintes.

3.2.1. Première description

C'est pourquoi une signature électronique peut être considérée comme étant composée d'un algorithme (qui combine à la fois un algorithme de calcul d'empreinte et le type de bi-clé utilisé), d'une valeur de signature et d'informations relatives au signataire [COT 03f] :

```
ElectronicSign ::= SEQUENCE {
  signValue     Sign,
  signer        OCTET STRING
}

Sign ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING
}

AlgorithmIdentifier ::= OBJECT IDENTIFIER
```

Le signataire (« signer ») est représenté sous la forme d'un ensemble d'octets. Ce format générique permet de manipuler tout type de données, même s'il est dans le cas présent conçu pour recevoir en premier lieu un certificat numérique à la norme X.509. Ce dernier apporte la preuve du lien fort entre la clé de vérification et l'identité de son possesseur. Ce certificat doit être généré par un tiers dont le vérificateur peut avoir confiance pour être considéré comme valable.

D'autre part, la valeur de la signature numérique du champ « signature » est considérée comme un ensemble de bits (et non d'octets), ce qui suppose des règles

de codage strictes : en particulier, les bits supplémentaires doivent être systématiquement représentés par la même valeur binaire.

> *Note : le degré de confiance dans un certificat est en grande partie lié à la sécurité de son environnement de création. C'est pourquoi des structures spécialisées, appelées autorités de certification, sont accréditées par les gouvernements pour délivrer des certificats numériques. Ces autorités mettent en œuvre des Infrastructures à Clé Publique (ICP ou PKI) [AUT 02] et subissent des contrôles réguliers.*

3.2.2. Seconde description

Afin d'être plus complet, il convient d'ajouter le type de certificat contenu dans le champ « issuer » afin que le vérificateur puisse décoder le certificat. Cette possibilité n'est envisageable que si le type de certificat est lui-même normalisé (une valeur entière par exemple).

La structure de données résultante est alors de la forme :

```
ElectronicSign ::= SEQUENCE {
  signValue            Sign,
  signer               Certificate
}

Sign ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING
}

AlgorithmIdentifier ::= OBJECT IDENTIFIER

Certificate ::= SEQUENCE {
  certType             OBJECT IDENTIFIER,
  certValue   EXPLICIT ANY DEFINED BY certType
}
```

3.3. Prise en compte des politiques de signature

Comme nous l'avons vu, la valeur d'une signature électronique fait non seulement intervenir la validité de la signature numérique et du certificat du signataire, mais également la validité de l'environnement de signature présumé.

Il est par exemple essentiel de connaître l'outil de création de signature utilisé, le type de matériel ainsi qu'éventuellement le lieu dans lequel la signature a été construite afin de déterminer quel droit appliquer pour la défense du signataire [PIN 00].

Sur ce dernier point, la loi type de la CNUDCI [CNU 01] précise (article 12) que, « *pour déterminer si, ou dans quelle mesure, un certificat ou une signature électronique produit légalement ses effets, il n'est pas tenu compte :*

a) Du lieu dans lequel le certificat est émis ou la signature créée ou utilisée, ou

b) Du lieu dans lequel l'émetteur ou le signataire a son établissement. »

En particulier :

- « *Un certificat émis en dehors de [l'Etat adoptant] a les mêmes effets juridiques dans [l'Etat adoptant] qu'un certificat émis dans [l'Etat adoptant] à condition qu'il offre un niveau de fiabilité substentiellement équivalent* ».

- « *Une signature électronique créée ou utilisée en dehors de [l'Etat adoptant] a les mêmes effets juridiques dans [l'Etat adoptant] qu'une signature électronique créée ou utilisée dans [l'Etat adoptant] à condition qu'elle offre un niveau de fiabilité substentiellement équivalent* ».

Ces informations sont indiquées par les politiques de signature suivies lors de la génération de la signature dans son environnement. Une solution technique mais toutefois complexe est proposée par [ROS 01]. D'autre part, la structure informelle d'une politique générique comprend, selon [ETS 02b] :

- Un identifiant de la politique : cet identifiant permet de l'identifier sans ambiguïté au sein d'un espace de nommage donné.

- Une date d'émission.

- La signature de l'organisme émetteur de la politique, en général différent du PSCE émetteur du certificat du signataire. Il s'agit d'une autorité de confiance spécialisée dans la définition de politiques de sécurité.

- La période de validité de la politique, durée pendant laquelle la politique est reconnue comme valable lorsqu'elle est jointe à une signature électronique.

- Les termes légaux d'application de la politique.

Nous pouvons dériver de la structure de données mise en œuvre par [TRU 03] une structure plus complète reposant sur [ETS 02b] et représentant les différents champs précités :

```
Policies ::= SEQUENCE OF Policy

Policy ::= SEQUENCE {
-- Proposal derived from Trustronic "TT TD S01-1.2e"
  policyInfo     PolicyInfo,
  sign           ElectronicSign    OPTIONAL
  -- Signature of the policy issuing authority
}

PolicyInfo ::= SEQUENCE {
  policyId       OBJECT IDENTIFIER,
  reference      VisibleString,
  issuerName     GeneralName,
  issuingDate    GeneralizedTime,
  validity       Validity,
  legalTerms     VisibleString     OPTIONAL
}

Validity ::= SEQUENCE {
  validFrom  [0] GeneralizedTime   OPTIONAL,
  validTo    [1] GeneralizedTime   OPTIONAL
}
```

Dans le cas où la signature de l'autorité émettrice de la politique n'est pas présente, le vérificateur devra demander confirmation à cette autorité par le biais du champ « issuerName » en mentionnant l'identifiant de la politique. En réponse, l'autorité contactée indiquera la validité de cette politique et signera électroniquement sa réponse.

Un autre problème demeure : il concerne la validation a posteriori de la signature électronique. En effet, il semble intéressant de pouvoir apporter la preuve de l'authenticité d'une transaction en cas de contestation. Bien souvent cette contestation n'est pas immédiate, c'est à dire au moment de la transaction, mais intervient avec un temps de retard. Il est donc nécessaire soit de dater l'instant de signature de telle sorte que la preuve de la validité de la signature à la date indiquée soit irréfutable, soit de justifier d'une notarisation électronique.

3.4. Prise en compte de l'horodatage

Lorsqu'un message ou une signature électronique ne mentionne pas de date, ni prétendue ni certifiée par une autorité de confiance, le vérificateur ne dispose d'aucune information temporelle fiable. Il lui est donc difficile de valider cette signature électronique lorsque le certificat du signataire a expiré ou a été révoqué ou suspendu, sauf si une date prétendue lui suffit.

> *Note : cette mention de date importe cependant peu lorsque la vérification est immédiate et qu'aucune autre vérification postérieure est nécessaire (lors d'une transaction avec une instance administrative par exemple).*

3.4.1. Sécurisation de l'horodatage

La question qui survient alors est relative au « maillon le plus faible » [KAE 00] de la signature électronique : la date certifiée est protégée par la signature de l'autorité d'horodatage. Cependant, cette signature lui ayant été délivrée par un PSCE, l'horodatage devient invalide lorsque la clé de signature de l'une ou l'autre de ces autorités est compromise. Il est donc nécessaire d'employer des clés très robustes, tant au niveau du PSCE que de l'autorité d'horodatage.

[PIN 01a] souligne de plus que le recours à un horodatage sécurisé permet aux utilisateurs d'employer des clés de taille réduite, leurs signatures étant protégées par les différents horodatages. Cependant, les horodatages doivent être ni fraudables, ni répudiables. La non répudiation est supportée par l'utilisation de la signature électronique. Cependant, les valeurs d'horodatage ne doivent pouvoir être anti-datées ou post-datées. Les protocoles existants permettant de protéger les valeurs d'horodatage contre la fraude et développés en annexe, sont discutés dans [GUE 03].

3.4.2. Proposition d'un protocole d'horodatage multiple

Comme stipulé précédemment, l'horodatage permet de protéger les données dans le temps. Cependant, la signature de l'horodateur est seule garante de l'intégrité de la donnée et de sa datation, essentielle à la validation de la signature.

Un système sûr doit donc reposer sur deux horodatages ou plus afin que la compromission de la signature d'un horodateur soit compensée par l'intégrité des horodatages restants.

3.4.2.1. Première version du protocole

Le protocole défini par [PIN 01a] repose sur l'ajout à la signature électronique de plusieurs horodatages indépendants. Ce mode de fonctionnement est décrit par la figure 17.

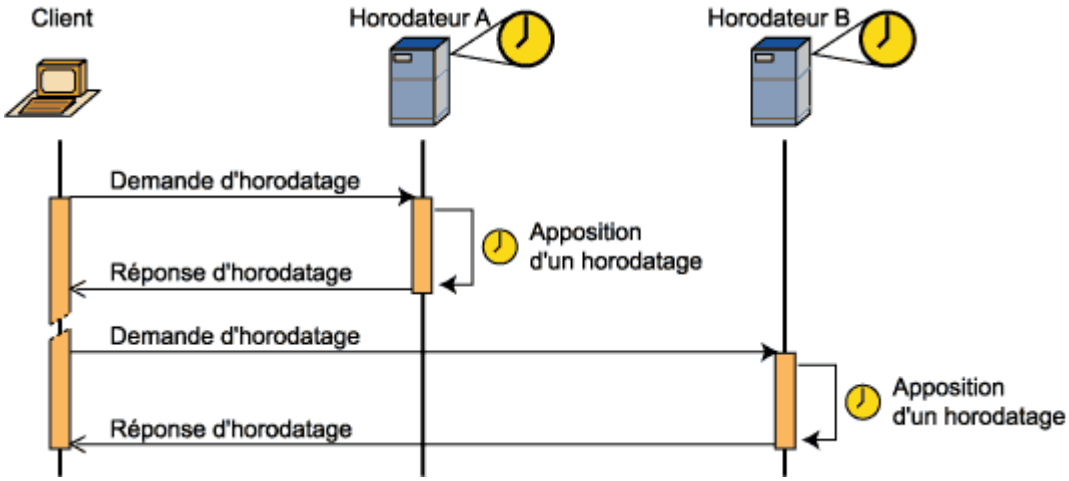

Figure 17 : protocole standard d'horodatages indépendants

Un horodatage multiple pourrait alors être défini comme suit :

```
MultipleTimeStamp ::= SEQUENCE OF TimeStamp

TimeStamp ::= SEQUENCE {
  token    TimeStampToken,
  -- defined by RFC3161
  sign     ElectronicSign
}
```

Deux problèmes subsistent cependant :

- Chaque jeton d'horodatage étant indépendant, une incohérence de date peut survenir lorsque plusieurs horodatages sur une signature électronique sont demandés alors que le certificat du signataire (demandeur) arrive à expiration ou est révoqué. Le premier horodatage peut alors indiquer que la signature est valide alors que les suivants apporteront une information contraire. Cette contradiction est viable tant que le premier horodatage est intègre. Cependant, dès lors qu'une attaque est reconnue et que la preuve de son invalidité est apportée, les horodatages restants indiquent que la signature est invalide.

- D'autre part, le demandeur (un objet client ou serveur dans le cas présent) effectue lui-même autant de demandes d'horodatage que requis par la politique de sécurité appliquée. Ainsi, l'objet demandeur doit supporter lui-même la charge induite par l'horodatage (construction de plusieurs requêtes, attente des réponses, validation des signatures apposées aux réponses). Cette surcharge peut être néfaste, en particulier lorsque le demandeur est un objet serveur.

3.4.2.2. Proposition d'une version sécurisée

En fait, disposer de plusieurs horodatages indépendants ne répond que partiellement au besoin. Il serait ainsi plus intéressant de ne disposer que d'un unique jeton et de demander une validation de ce jeton par plusieurs horodateurs, en fonction d'un délai maximum d'acceptation.

C'est ce que proposent des structures de données suivantes :

```
MultipleTimeStamps ::= SEQUENCE {
  sts          SignedTimeStamp,
  counter      ElectronicSigns      OPTIONAL
}

SignedTimeStamp ::= SEQUENCE {
  value        TimeStampValue,
  sign         ElectronicSign
}

TimeStampValue ::= SEQUENCE {
  policy  [0] TimeStampPolicies OPTIONAL,
  -- Time-stamping applied policies
  token        TimeStampToken
  -- Defined by RFC3161
}

ElectronicSigns ::= SEQUENCE OF ElectronicSign
```

Le jeton d'horodatage (« token ») est défini par un horodateur appelé « initiateur ». Ce dernier prend en charge la communication avec les autres horodateurs (« accréditeurs ») afin de collecter les signatures nécessaires. Ces signatures sont enregistrées via la liste de signatures conjointes « ElectronicSigns ».

Au lieu de N jetons d'horodatage certifiés par N signatures électroniques (N,N), ce procédé permet de n'utiliser qu'un seul jeton d'horodatage (1,N). De plus, le protocole de demande d'accréditation du jeton d'horodatage est transparent du point de vue du demandeur.

Il eût été possible de ne pas envoyer une structure « SignedTimeStamp » aux accréditeurs et uniquement un jeton. Dans ce cas, les requêtes auraient dû être signées par l'initiateur afin que les accréditeurs n'apposent pas leur signature sur une date non certifiée. L'envoi d'un jeton signé permet d'établir directement la confiance entre les horodateurs impliqués et l'initiateur.

Enfin, ce protocole supporte le fonctionnement par horodatages indépendants lorsqu'aucune accréditation n'est demandée.

3.4.2.3. Déroulement du protocole d'horodatage multiple

L'initiateur construit un jeton d'horodatage et le signe. Il envoie alors une demande d'accréditation à autant d'horodateurs que demandés (ce nombre ne peut être supérieur au nombre d'horodateurs reconnus). Ces derniers, appelés « accréditeurs », disposent d'un intervalle de temps limite pendant lequel ils approuvent (et donc contre-signent) la valeur d'horodatage que leur fournit l'initiateur. Au-delà de cet intervalle, les accréditeurs doivent refuser de se porter caution d'une date qu'ils jugent trop ancienne.

Le schéma de la figure 18 montre l'exemple d'un horodatage avec deux accréditations issues d'accréditeurs distincts.

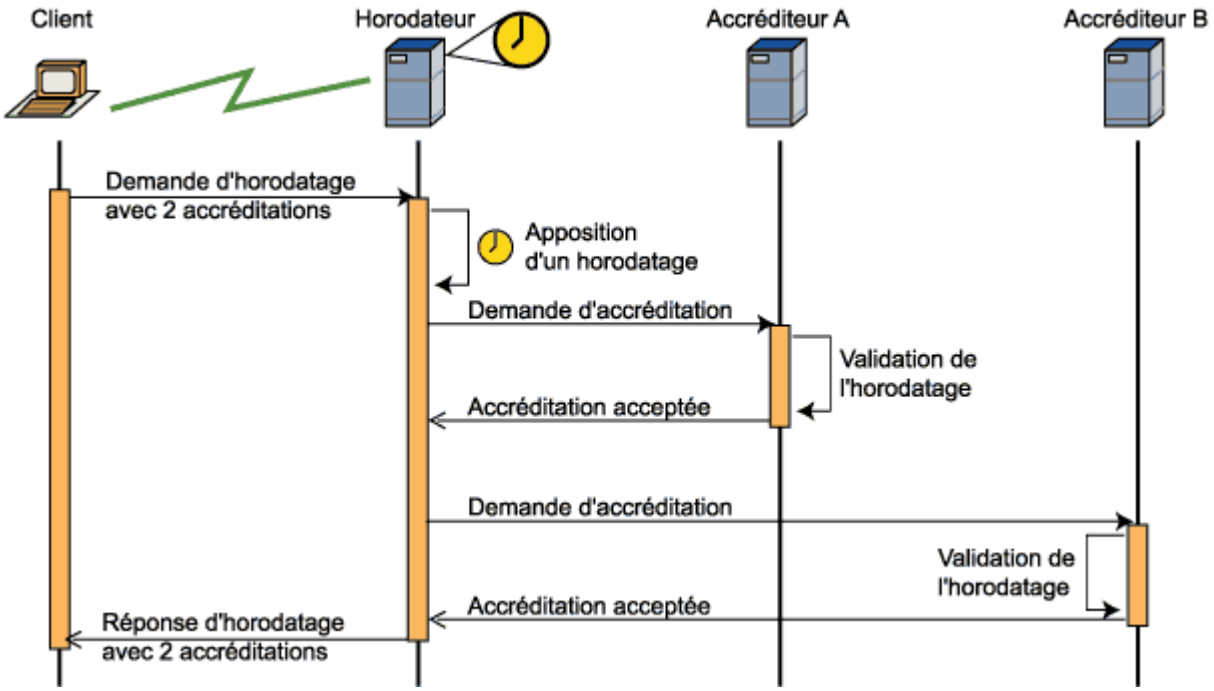

Figure 18 : protocole d'horodatage multiple

3.4.3. Prise en compte de la notarisation électronique

Nous suggérons d'inclure au format de signature électronique non seulement des informations de validation traditionnelles (telles que la consultation des listes de révocation ou l'interrogation d'un serveur OCSP) mais également une information de validité certifiée par un notaire électronique délivrant des certificats de validité [TRU 03].

Le certificat de validité, dérivé de [TRU 03], est organisé comme suit :

```
CertifiedSignatureValidity ::= SEQUENCE {
```

```
  version       INTEGER {v1(1)},
  validity      SignatureValidity,
  sign          Signature,
  -- Signature of the electronic notary
  revInfo   [0] RevocationInfo              OPTIONAL,
  extns     [1] EXPLICIT Extensions         OPTIONAL
}

SignatureValidity ::= SEQUENCE {
  sigStatus     ValidityInfo,
  ts            Dates                       OPTIONAL,
  -- Time-stamps created over "sigStatus"
  extns         Extensions                  OPTIONAL
}

ValidityInfo ::= SEQUENCE {
  sign          MessageImprint,
  -- Signature hash, from RFC3161 (TSP)
  status        ValidityStatus
}

ValidityStatus ::= PKIStatus
-- From RFC3161 (TSP)

Dates ::= SEQUENCE {
-- If present, must mention at least one field
  claimed   [0] GeneralizedTime             OPTIONAL,
  ts        [1] EXPLICIT MultipleTimeStamps OPTIONAL
}

Signature ::= ElectronicSign
-- To match our definition of the electronic signature

RevocationInfo ::= SEQUENCE {
  revDate       Time,
  -- Claimed revocation date, to verify status with CRL
  cor           CertificateOfRevocation
}

Time ::= CHOICE {
-- Defined in RFC3280, paragraph 4.1
  utcTime             UTCTime,
  generalTime         GeneralizedTime
}
```

Il fait appel à un certificat de révocation (« Certificate of Revocation », COR), utilisé pour faire opposition à une signature afin d'en limiter la portée juridique. Le COR est certifié par un notaire électronique de sorte que le signataire doit justifier sa demande de révocation de signature. Le format du COR sera défini par la suite.

3.5. Proposition d'un format signature électronique

Le format de signature électronique présenté ci-après intègre les informations relatives aux politiques de signature, à l'horodatage multiple et la notarisation électronique et prend en compte les remarques de [COT 03e] et [PIN 01c] concernant la création de signatures ni fraudables ni répudiables :

```
ElectronicSign ::= SEQUENCE {
  version       Version,
  signValue     Sign,
  issuer        CertificateAndPath,
  policies  [0] SignaturePolicies           OPTIONAL,
  atts      [1] SignatureAttributes         OPTIONAL,
  -- Claimed signature attributes
  attCert   [2] AttributeCertificates       OPTIONAL,
  -- Certified attribute certificates, from RFC3281
  date      [3] GeneralizedTime             OPTIONAL,
  -- Claimed date
  ts        [4] MultipleTimeStamps          OPTIONAL,
  cov       [5] CertifiedSignatureValidity  OPTIONAL,
  extns     [6] EXPLICIT Extensions         OPTIONAL
}

Version ::= INTEGER

Sign ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signatureValue       BIT STRING
}

AlgorithmIdentifier ::= OBJECT IDENTIFIER

CertificatePath ::= SEQUENCE OF Certificate
-- From signer's certificate to root issuing certificate

Certificate ::= SEQUENCE {
  certType               OBJECT IDENTIFIER,
  certValue     EXPLICIT ANY DEFINED BY certType
}

SignaturePolicies ::= Policies

SignatureAttributes ::= Attributes

AttributeCertificates ::= SEQUENCE OF AttributeCertificate
-- AttributeCertificate defined in RFC3281

Attributes ::= SEQUENCE OF Attribute

Attribute ::= SEQUENCE {
-- From RFC3280
```

```
  type      AttributeType,
  values    SET OF AttributeValue
  -- At least one value is required
}

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

Afin d'être conforme avec les recommandations de [PIN 01c], la valeur de la signature numérique (champ « signatureValue ») doit être construite à partir de l'ensemble formé par le contenu sous forme numérique d'une part et l'identifiant du certificat du signataire d'autre part. Cette sécurité est obtenue en modifiant le format de la signature électronique.

3.5.1. Processus d'apposition d'une signature électronique

Bien qu'étant sécurisé, le format proposé précédemment n'est que partiellement exploitable. En effet, il doit être combiné à un processus de création de signature apte à garantir la fiabilité de la signature (ni fraudable, ni répudiable). En effet, il ne doit être possible d'échanger des certificats (et par conséquents des signataires) tout en conservant la validité des signatures.

Par exemple, un attaquant ne doit être en mesure d'intégrer la clé de vérification du signataire original dans son propre certificat dans le but d'usurper son identité.

Pour cela, [PIN 01c] recommande que le numéro de série du certificat du signataire soit signé parallèlement à l'information numérique. Cependant, la signature de cet unique élément est insuffisante pour garantir sa fiabilité. Nous proposons non pas de signer l'identifiant du certificat du signataire mais l'ensemble de son certificat numérique. L'on dispose dès lors d'informations fiables notamment relatives à l'autorité émettrice ainsi qu'aux attributs et extensions mentionnées dans le certificat du signataire.

3.5.1.1. Première version

La signature d'un contenu numérique ne se suffit donc pas à elle-même. Par contre, il est souhaitable de signer une structure de la forme :

```
tbsData ::= SEQUENCE {
  tbsContent    ContentInfo,
  dates         Dates                       OPTIONAL,
  -- Time-stamps "ts" of "dates" created over "tbsContent"
  signerCert    Certificate
}
```

Le champ « tbsContent » (« to be signed content » ou contenu à signer) est défini comme conteneur générique d'une donnée représentée dont le format peut varier. [KAL 98b] le présente comme suit :

```
ContentInfo ::= SEQUENCE {
-- From RFC2315
  contentType   ContentType
  content   [0] EXPLICIT ANY
                  DEFINED BY contentType    OPTIONAL
}

ContentType ::= OBJECT IDENTIFIER
```

Le type de contenu (« content ») est indiqué par la présence d'un identifiant normalisé « contentType ». La lecture de la valeur de cet identifiant renseigne sur le format du contenu, extrait à partir du format non structuré « ANY » que propose la syntaxe ASN.1.

La structure temporaire « tbsData » (donnée à signer) est reconstituée lors de la vérification de la signature numérique afin :

- D'authentifier le contenu numérique signé.
- D'apporter une information temporelle par une date revendiquée ou certifiée par une autorité d'horodatage. Cette date, antérieure à la signature, permet de la situer dans le temps. La signature est supposée créée a posteriori.
- D'authentifier le certificat du signataire. Bien que chaque certificat soit identifié de manière unique par son numéro de série, le degré de confiance est accru lorsque l'ensemble du certificat est authentifié et non uniquement son numéro de série, car si un faussaire peut créer un certificat comportant un numéro de série identique, il ne sera pas en mesure d'imiter la signature de l'autorité de certification émettrice du certificat.

Ceci étant, un fraudeur peut encore substituer les informations complémentaires de la signature, telles que les politiques de signatures et le certificat d'attributs.

3.5.1.2. Version améliorée

L'ensemble des informations liées à la signature doivent être signées numériquement, à l'exception des seconds horodatages et certificats de validité, construits à partir de la valeur de la signature numérique résultante.

Les données à signer « tbsData » sont alors décrites comme suit :

```
tbsData ::= SEQUENCE {
```

```
  contentInfo  ContentInfo,
  signerInfo   SignerInfo
}

SignerInfo ::= SEQUENCE {
  signerCert    CertificatePath,
  policies  [0] SignaturePolicies          OPTIONAL,
  atts      [1] SignatureAttributes        OPTIONAL,
  -- Claimed signature attributes (no claimed date)
  attCert   [2] AttributeCertificate       OPTIONAL,
  -- Certified attribute certificate, from RFC3281
  dates     [3] Dates                      OPTIONAL,
  -- Claimed date and/or first time-stamps if present
  extns     [4] EXPLICIT Extensions        OPTIONAL
  -- Authenticated extensions
}
```

Il suit que la signature électronique peut être modifiée une dernière fois afin de tenir compte de la structure « SignerInfo » et ainsi faciliter la vérification en mettant directement cette structure à disposition du vérificateur.

3.5.2. Version définitive du format de signature électronique

La signature numérique devant protéger non seulement le contenu numérique signé mais également l'ensemble des informations liées à la signature, le format définitif de la signature électronique doit intégrer les données relatives au signataire ainsi qu'à l'environnement de signature. Ces informations sont décrites par la structure « SignerInfo ».

Le format résultant est alors le suivant :

```
ElectronicSign ::= SEQUENCE {
  version       Version,
  signValue     Sign,
  signerInfo    SignerInfo,
  ts        [0] MultipleTimeStamps         OPTIONAL,
  cov       [1] CertifiedSignatureValidity OPTIONAL,
  extns     [2] EXPLICIT Extensions        OPTIONAL
  -- Unauthenticated extensions
}
```

La valeur de la signature numérique, construite à partir de la structure « tbsData » précédemment mentionnée, scelle d'une part le contenu numérique et d'autre part les informations relatives au signataire.

De fait, ce format apporte des réponses aux questions soulevées par [PIN 01c] concernant la fraudabilité et la répudiabilité des signatures électroniques (consulter

les annexes relatives à ce propos). Les seules informations susceptibles d'être modifiées (ou supprimées) sont l'horodatage multiple, le certificat de validité et les éventuelles extensions non authentifiées. Ces informations pourront néanmoins être protégées par l'emploi de formats de contenus signés tels que ceux présentés en annexe. Il est également possible d'authentifier à nouveau ces informations en considérant la signature électronique comme source de génération d'une seconde signature numérique.

Le format définitif de la signature électronique est alors :

```
ElectronicSign ::= SEQUENCE {
  version             Version,
  signValue           Sign,
  signerInfo          SignerInfo,
  validationData  [0] SignedValidationData   OPTIONAL,
  envelopSigns    [1] ElectronicSigns        OPTIONAL,
  counterSigns    [2] ElectronicSigns        OPTIONAL,
  extns           [3] EXPLICIT Extensions    OPTIONAL
  -- Unauthenticated extensions
}

SignedValidationData ::= SEQUENCE {
  valData       ValidationData,
  valSignValue  Sign,
  -- Digital signature on ValidationData
  extns     [0] EXPLICIT Extensions         OPTIONAL
  -- Unauthenticated validation extensions
  -- if neither envelopSigns nor countersigns are present
}

ValidationData ::= SEQUENCE {
  ts        [0] MultipleTimeStamps          OPTIONAL,
  cov       [1] CertifiedSignatureValidity  OPTIONAL,
  extns     [2] EXPLICIT Extensions         OPTIONAL
  -- Authenticated validation extensions
}

ElectronicSigns ::= SEQUENCE OF ElectronicSign
-- List of signatures
```

De par les éléments qui le composent, ce format est aussi bien adapté à la signature électronique de documents sous forme numérique qu'à l'authentification des participants à une communication informatique. En particulier, l'utilisation de multiples sur-signatures (« envelopSigns ») et contre-signatures (« counterSigns ») autorise les échanges authentifiés entre plusieurs acteurs.

3.5.2.1. Format du certificat de révocation

Le certificat de révocation, inclus dans le certificat de validité (« cov »), peut alors prendre la forme suivante :

```
CertificateOfRevocation ::= SEQUENCE {
  version     INTEGER {v1(1)},
  corInfo     CorInfo,
  sign        Signature,
  -- Signature of "corInfo" by the electronic notary
  extns   [0] EXPLICIT Extensions OPTIONAL
}

CorInfo ::= SEQUENCE {
  SigValue
  certRevDate Time,
  -- Certified revocation date
  sigVal      Sign,
  sigInfo     SignerInfo,
  -- Data linked with the revoked signature
  revPolicies RevocationPolicies,
  -- Applied revocation Policies
  extns   [0] EXPLICIT Extensions OPTIONAL
}

RevocationPolicies ::= Policies
```

3.5.2.2. Sur-signature et contre-signature

La structure de signature électronique précédente prend également en compte la sur-signature (« envelopSigns ») et la contre-signature (« counterSigns ») sous la forme de listes ordonnées de signatures électroniques définies récursivement. La contre-signature se différencie de la sur-signature par le fait que le contenu signé n'est pas nécessairement visualisé [TRU 03]. C'est pourquoi nous les dissocions au sein de la signature électronique.

> *L'ordonnancement de cette liste (par l'utilisation de « SEQUENCE OF ») a été privilégié par rapport à un ensemble non ordonné (de type « SET OF ») afin de faciliter l'encodage ASN.1 puisque l'encodage selon les règles DER d'un tel ensemble nécessite un tri préalable sur les éléments qui le composent.*

Chaque contre-signature est apposée sur un élément de type « tbsData » dans lequel le contenu « contentInfo » correspond à la valeur du champ « signValue » de

la signature contre-signée. L'identifiant correspondant peut être identique à celui standardisé par [RSA 00], à savoir 1.2.840.113549.1.9.6.

Lorsqu'une même contre-signature s'applique à plusieurs signatures, il est envisageable de ne pas utiliser le champ « counterSigns » proposé et de contre-signer l'objet représentant la liste des signatures. C'est ce que propose le standard CMS.

3.5.3. Comparaison avec le format CMS

Le standard CMS [HOU 99] fait également état d'une signature électronique composée de plusieurs champs. Ce formalisme est utilisé par d'autres standards tel que [PIN 01b]. La signature électronique est intégrée au format CMS sous la dénomination « SignerInfo » :

```
SignerInfo ::= SEQUENCE {
  version            CMSVersion,
  sid                SignerIdentifier,
  digestAlgorithm    DigestAlgorithmIdentifier,
  signedAttrs    [0] IMPLICIT SignedAttributes   OPTIONAL,
  signatureAlgorithm SignatureAlgorithmIdentifier,
  signature          SignatureValue,
  unsignedAttrs  [1] IMPLICIT UnsignedAttributes OPTIONAL
}

SignerIdentifier ::= CHOICE {
  issuerAndSerialNumber     IssuerAndSerialNumber,
  subjectKeyIdentifier  [0] SubjectKeyIdentifier
}

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute

SignatureValue ::= OCTET STRING

UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute

Attribute ::= SEQUENCE {
  attrType           OBJECT IDENTIFIER,
  attrValues         SET OF AttributeValue
}

AttributeValue ::= ANY

SubjectKeyIdentifier ::= OCTET STRING
```

Une première distinction s'impose : le format CMS introduit les champs signés et la valeur de la signature au même niveau, alors que nous faisons appel à la signature du champ « SignerInfo » au sein de la structure « ElectronicSign ». Ceci

permet de ne pas recourir à des structures de données temporaires permettant d'aboutir au même résultat.

D'autre part, la signature proposée par CMS n'introduit pas directement des champs spécifiques à :

- L'horodatage, qu'il soit standard ou conforme à notre proposition d'horodatage multiple.
- La notarisation électronique.
- La sur-signature et la contre-signature.

Nous avons de plus proscrit l'utilisation du conteneur générique « ANY » pour proposer des champs spécialisés.

Enfin, les mises en garde de sécurité concernant la fraudabilité de la signature sont également prises en compte par le champ « sid » du format de signature CMS.

3.6. Intégration aux standards actuels

Il existe actuellement deux principaux standards faisant appel à la signature électronique : CMS [HOU 99] et ES-X [ETS 02b]. Nous avons délibérément omis les autres formats PKCS de RSA qui s'attachent à des points particuliers (PKCS#8 par exemple décrit un format d'enregistrement d'une clé privée protégée par mot de passe) ou sont sous-exploités (tel que le format PKCS#12 ou PFX utilisé pour stocker à la fois un certificat public, le chemin de certification, ainsi que la clé privée correspondante).

3.6.1. Intégration au standard CMS

Bien que le format de signature électronique proposé tienne compte intrinsèquement de la sur-signature et de la contre-signature, ces dernières ne peuvent être appliquées que sur une unique signature source. La sur-signature (ou contre-signature) d'un ensemble de signatures électroniques demande de fait autant de sur-signatures (respectivement contre-signatures). Le standard CMS propose de pallier à ce défaut en indiquant une possible hiérarchisation des signatures dérivée du format standard PKCS#7 [KAL 98b] et applicable aux sur-signatures.

Outre la définition des éléments d'une signature électronique, le format CMS décrit une structure, présentée par la figure 19, permettant de présenter des données signées et de les sur-signer ou contre-signer récursivement. De fait, une sur-signature ou contre-signature s'applique non pas à une signature parmi celles existantes mais directement à l'ensemble des signatures.

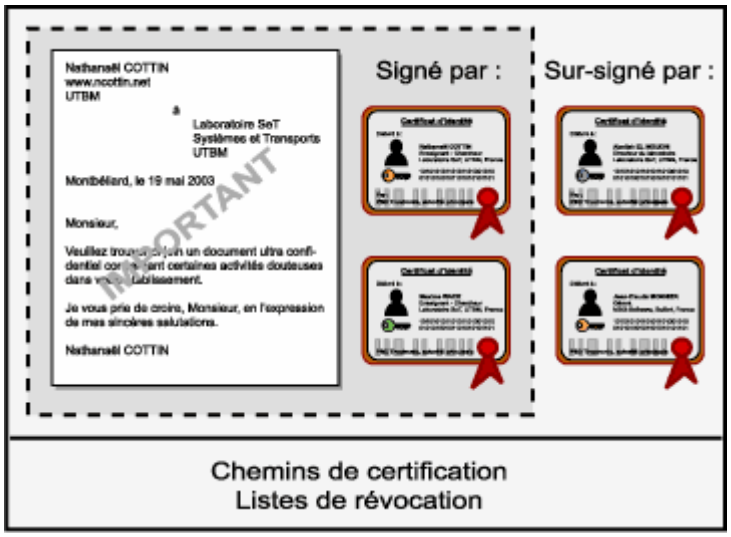

Figure 19 : sur-signatures au format CMS

Bien que CMS définisse plusieurs types de contenus (appelés « bags »), nous nous intéressons au contenu signé. Le format ASN.1 public est présenté ci-dessous :

```
SignedData ::= SEQUENCE {
  version           Version,
  digestAlgorithms  DigestAlgorithmIdentifiers,
  contentInfo       ContentInfo,
  -- May recursivemy enclose another SignedData bag
  certificates
    [0] IMPLICIT ExtendedCertificatesAndCertificates
        OPTIONAL,
  crls
    [1] IMPLICIT CertificateRevocationLists
        OPTIONAL,
  signerInfos       SignerInfos
}

Version ::= INTEGER

ContentInfo ::= SEQUENCE {
  contentType       ContentType,
  content
    [0] EXPLICIT ANY DEFINED BY contentType
        OPTIONAL
}

ContentType ::= OBJECT IDENTIFIER

SignerInfos ::= SET OF SignerInfo
-- From CMS
```

Une évolution envisageable peut conduire au formalisme suivant :

```
SignedData ::= SEQUENCE {
  version           Version,
  digestAlgorithms  DigestAlgorithmIdentifiers,
  contentInfo       ContentInfo,
  signatures
    [0] IMPLICIT ElectronicSigns
        OPTIONAL,
  crls
    [1] IMPLICIT CertificateRevocationLists
        OPTIONAL
}
```

L'avantage d'une telle structure réside dans la prise en compte du format de signature électronique proposé. Ce dernier manipule à la fois l'horodatage multiple, la notarisation électronique et la contre-signature. L'horodatage pourra être réalisé au moment de l'apposition de chaque signature ou lors de l'apposition d'une sur-signature ou d'une contre-signature (au cas où l'élément « counterSigns » de la signature électronique n'est pas utilisé). Il pourra alors s'appliquer, par transitivité, à l'ensemble des signatures de niveau inférieur.

Le format CMS inclut de plus les listes de révocation (CRL) associées. Nous les avons conservées afin de présenter au vérificateur les signatures et une méthode de vérification hors ligne. L'ensemble des CRL des PSCE impliqués doivent y être mentionnées afin que le vérificateur puisse déterminer la validité de toutes les signatures, aussi bien des signataires que des autorités, tels que les horodatages et certificats de validité délivrés par les notaires électroniques). Cependant, pour plus de sécurité, le vérificateur doit employer en priorité ses propres moyens de validation.

> *Note : le format CMS fournit à la fois les signatures électroniques et les moyens hors-ligne de les valider. Ce faisant, CMS devient juge et partie et par conséquent illégal puisque « nul ne peut se constituer sa propre preuve ». De fait, il est fort peu probable que l'ensemble des CRL jointes puissent être proposées comme éléments de preuve.*

En effet, l'ensemble des certificats et listes de révocations peuvent être créés par un individu malintentionné. Ainsi, les certificats apparaissent comme valides dès lors que le vérificateur utilise les « vraies-fausses » listes de révocation fournies.

D'autres formats de contenu comportant de multiples signatures sont suggérés en annexe.

3.6.2. Intégration au format ES-C

L'« European Telecommunications Standards Institute » (ETSI) suivi de l'IETF [PIN 01b] ont mis au point un format de données permettant une validation sur le long terme des signatures électroniques (au format CMS). Ce format, baptisé ES-C [ETS 02b], est représenté par un ensemble de couches (figure 20).

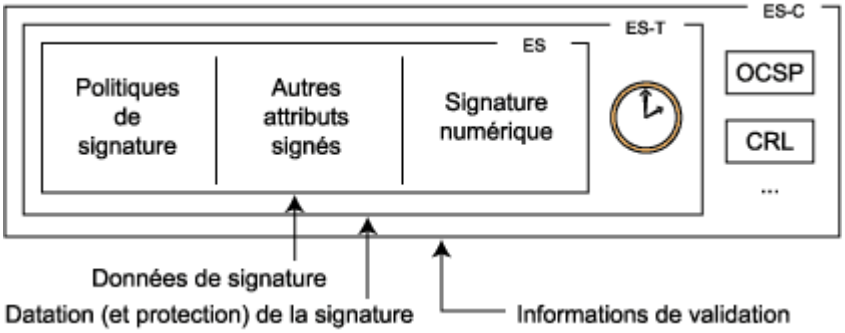

Figure 20 : formats standard ES, ES-T et ES-C

L'intégration de notre proposition à ES-C est aisée puisqu'il suffit de remplacer le champ CMS de la signature numérique par notre proposition de format de signature électronique.

Le format ES-C contient une signature numérique horodatée à laquelle sont associées des informations de validation telles que [PIN 01b] :

- Le certificat du signataire.
- L'ensemble des certificats permettant d'établir le chemin de certification complet jusqu'à la racine de certification.
- Des informations concernant la révocation des certificats sus-mentionnés. L'IETF préconise à ce propos l'utilisation de listes de révocation.

La signature ainsi remplacée n'est plus nécessairement horodatée et ne comporte plus de date revendiquée ni de politiques de signature. Ces éléments sont en effet pris en charge par les attributs de la signature électronique CMS (authentifiés et non authentifiés) appliquée aux formats ES, ES-T et ES-C de l'ETSI. Le principe de notarisation électronique via l'utilisation d'un certificat de validité vient alors avantageusement compléter l'horodatage de ES-T lorsque joint à notre format de signature électronique [COT 03a].

3.6.3. Méthode générale d'intégration

Suite à ces deux exemples d'intégration, nous pouvons dériver une méthode générique permettant d'intégrer notre proposition de format de signature à l'existant.

Le procédé d'intégration de déroule comme suit :

1. Une structure appelée « TBSData » (« to be signed data ») et regroupant l'ensemble des informations à signer doit être créée. Elle peut être directement issue d'une structure existante.

2. Cette structure est associée à notre signature électronique dans une seconde structure de la forme suivante :

```
GlobalData ::= SEQUENCE {
  tbsData       TBSData,
  signs     [0] ElectronicSigns      OPTIONAL,
  ...
  extns     [n] EXPLICIT Extensions  OPTIONAL
}
```

Ainsi, chaque signature électronique est construite à partir des informations contenues dans le champ « tbsData ».

3.7. Proposition d'un certificat à multiples clés publiques

Le standard X.509 intègre une unique clé publique (ou clé de vérification) avec un ensemble d'usages qui déterminent son contexte d'utilisation (signer des contenus numériques, chiffrer des données confidentielles, etc.).

Nous proposons néanmoins une alternative à la version 3 du format X.509 en y apportant deux améliorations :

- L'intégration de plusieurs clés publiques (ou clés de signature selon l'usage des clés retenu).

- L'ajout de la classe du certificat.

Le certificat obtenu porte le nom de « certificat à multiples clés publiques » (MPKC) [COT 03g].

3.7.1. Intégration de multiples clés publiques au certificat

Le standard X.509 permet le lier une identité et une clé publique unique dont le rôle est défini. Ainsi, un certificat ne doit combiner les usages de signature et de chiffrement. L'utilisateur doit alors disposer de deux certificats distincts.

De plus, la durée de validité d'un certificat est déterminée par la taille de sa clé publique, la clé du PSCE émetteur étant beaucoup plus sécurisée.

Intégrer plusieurs clés publiques permet ainsi :

- D'allonger la période de validité du certificat tout en limitant la compromission des clés.
- D'utiliser le même certificat à des fins de signature et de chiffrement.
- De manipuler des clés à usage unique [STI 01].

Les clés de signature (ou privées) correspondantes pourront être organisées en trousseau, chaque clé étant sécurisée à l'aide d'un mot de passe qui lui est propre. Le trousseau pourra lui-même être verrouillé par mot de passe et inscrit sur un support adapté, que le signataire conservera sous son contrôle exclusif.

3.7.2. Ajout de la classe du certificat

Chaque certificat dérive d'une classe normalisée par [ADA 99] représentée par une valeur entière :

- Les certificats de classe 1 sont destinés aux individus à titre personnel. Aucune vérification des informations fournies n'est opérée par le PSCE émetteur du certificat. Ils n'ont bien entendu aucune valeur juridique mais peuvent être utilisés par le courrier électronique sécurisé.
- Les certificats de classe 2 sont délivrés à titre personnel ou professionnel. Dans le cas d'un certificat professionnel, l'adresse de l'entreprise est mentionnée. Le rôle du demandeur au sein de l'entreprise peut de plus être précisé. Les informations fournies par le demandeur sont vérifiées par les différentes autorités d'enregistrement avant acceptation du dossier et génération du certificat. Ces certificats disposent d'une valeur juridique dans le cas où le PSCE est habilité et respecte une procédure (ou politique de certification) type mise en place par le gouvernement (telle que celle du MINEFI [MEF 99]). Bien que le demandeur d'un certificat de classe 2 fournisse des informations administratives, la création de son certificat ne requiert pas sa présence physique.
- Les certificats de classe 3 sont destinés avant tout à des serveurs informatiques impliqués dans des échanges sécurisés et/ou authentifiés. Ils mentionnent en particulier le nom du domaine Internet (DNS) dont ils dépendent. Ces certificats peuvent également être distribués à des personnes. Dans ce cas, leur présence physique est requise lors de la demande du certificat. Un mandataire peut également être désigné afin d'effectuer une demande par procuration [GTG 04].

Cependant, à l'heure actuelle, deux certificats personnels de classe 1 et 2 ne peuvent être différenciés par le biais des informations qu'ils renferment. Le degré de confiance attribué à un certificat de classe 1 ne pouvant être comparé à celui accordé à un certificat de classe 2, il est important que cette distinction puisse s'opérer. Pour cela, la méthode traditionnelle employée par les PSCE est de signer différemment le certificat délivré selon sa classe à l'aide de différents certificats de distribution (ou éventuellement différentes racines).

3.7.3. Vue d'ensemble du format final

Les structures proposées ne font pas appel aux extensions du certificat bien que cette solution soit envisageable, la clé publique originale étant considérée comme la première clé devant être utilisée.

Tout certificat à multiples clés publiques, tel que le schématise la figure 21, peut être représenté algébriquement par l'ensemble $\{I,\{K_1,...,K_n\},V\}_{I_{issuer},S_{K_{issuer}}}$, où $\{K_1,...,K_n\}$ désigne l'ensemble des clés publiques attribuées au titulaire du certificat, chaque clé contenant les informations qui suivent :

- Valeur de la clé publique.
- Type de clé et algorithme de signature ou de chiffrement.
- Période de validité de la clé et de son usage. En effet, cette proposition permet de définir un nombre maximum d'utilisations de la clé en sus de sa période de validité. Il est alors possible de définir des clés à usage unique lorsque ce nombre vaut 1.
- Usage requis pour cette clé, conformément aux usages définis par le standard X.509 [HOU 02].

L'indication de la validité de la bi-clé est requise, cependant les champs qui la composent sont optionnels. Un champ doit cependant être renseigné parmi la période de validité de la clé et le nombre d'utilisations. Par défaut, la période de validité est similaire à celle du certificat et aucune restriction concernant le nombre d'utilisations de cette clé n'est imposée.

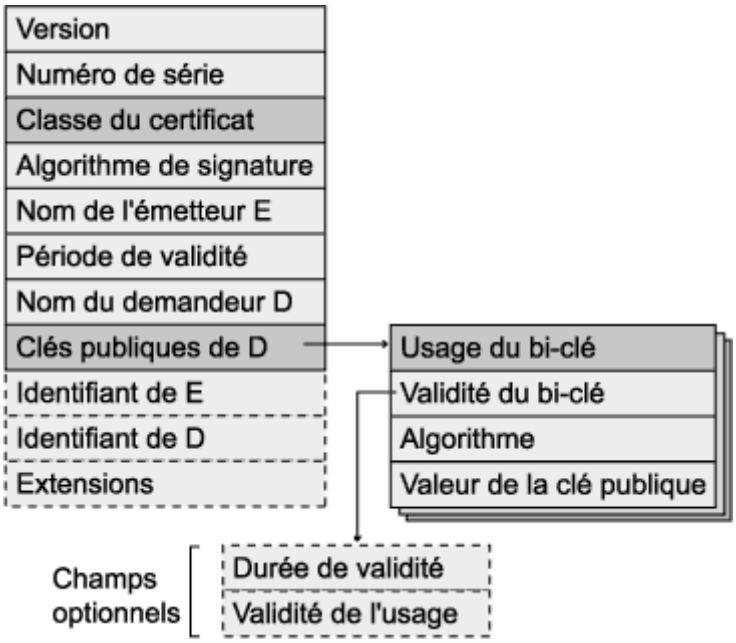

Figure 21 : proposition d'un certificat X.509 à multiples clés publiques

Ce format précise que chaque clé est indépendante et dispose de son propre usage. L'utilisateur peut ainsi disposer de plusieurs clés de signature, utilisables simultanément ou à tour de rôle en fonction de leur durée de validité. Dès lors, un unique certificat permet de chiffrer et signer des messages tout en limitant la compromission des clés.

3.7.4. Formalisation

La définition ASN.1 du MPKC est dérivée du format X.509 :

```
MPKC ::= SEQUENCE {
-- Derived from X509Certificate in RFC3280
  tbsCertificate      TBSCertificate,
  signatureAlgorithm  AlgorithmIdentifier,
  signature           BIT STRING
}

TBSCertificate ::= SEQUENCE {
  version         [0] EXPLICIT Version          DEFAULT v1,
  serialNumber        CertificateSerialNumber,
  certClass           CertificateClass,
  signature           AlgorithmIdentifier,
  issuer              Name,
  validity            Validity,
  subject             Name,
  subjectKeys         SubjectKeypairsInfo,
  issuerUniqueID  [1] IMPLICIT UniqueIdentifier OPTIONAL,
  subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
```

```
  extensions      [3] EXPLICIT Extensions       OPTIONAL
}

CertificateClass ::= SEQUENCE {
-- Class of the certificate:
-- standardized classes are 1, 2 and 3
  classId             INTEGER DEFAULT {1},
  classExtns          EXPLICIT Extensions       OPTIONAL
}

SubjectKeypairsInfo ::= SEQUENCE OF KeypairInfo

KeypairInfo ::= SEQUENCE {
  usage               KeyUsage,
  validity            KeypairValidity           OPTIONAL,
  algorithm           AlgorithmIdentifier,
  publicKey           BIT STRING
}

KeyUsage ::= BIT STRING {
-- Defined in RFC2459 paragraph 4.2.1.3
  digitalSignature    (0),
  nonRepudiation      (1),
  keyEncipherment     (2),
  dataEncipherment    (3),
  keyAgreement        (4),
  keyCertSign         (5),
  cRLSign             (6),
  encipherOnly        (7),
  decipherOnly        (8)
}

KeypairValidity ::= SEQUENCE {
  timeValidity        KeypairTimeValidity       OPTIONAL,
  usageValidity       KeypairUsageValidity      OPTIONAL
}

KeypairTimeValidity ::= TimeValidity

TimeValidity ::= SEQUENCE {
-- Derived from RFC2459, paragraph 4.2.1.4
  notBefore       [0] Time                      OPTIONAL,
  notAfter        [1] Time                      OPTIONAL
}

KeypairUsageValidity ::= INTEGER
-- Maximum number of key usages before
-- key expiration (digital signature only)
```

3.7.5. Validité des clés publiques

Le PSCE attribue plusieurs clés publiques à un unique certificat. Il doit veiller à ce que le certificat soit utilisable à tout moment. Ainsi chaque clé dispose de sa propre période de validité.

Le PSCE peut générer plusieurs clés pour une période de validité donnée. Dans ce cas, il doit tenir compte d'une période de recouvrement (« overlapping period ») durant laquelle une clé arrive à expiration et l'autre clé débute sa période de validité (figure 22). Cette période permet d'éviter l'usage de clés (de signature en particulier) en fin de vie.

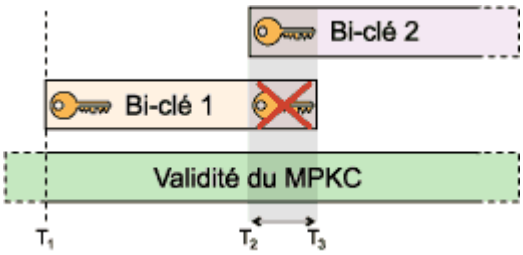

Figure 22 : période de recouvrement de clés

Par mesure de sécurité, la durée de la période de recouvrement doit être supérieure à la durée de validité de la liste de certificats révoqués afin de limiter le recours aux listes de certificats révoqués intermédiaires (delta-CRL) [HOU 02].

3.7.6. Cycle de vie du certificat

3.7.6.1. Suspension du certificat

Bien qu'une suspension de clé soit envisageable, il est préférable de suspendre l'intégralité du certificat. Durant la période de suspension, les bi-clés qui le composent ne peuvent être utilisées. Ils le seront à nouveau à l'issue de la période de suspension, à l'exception des clés ayant expiré.

3.7.6.2. Expiration des clés

Toute clé expirée (ou éventuellement non encore active) ne peut être utilisée. Pour cela, le vérificateur doit contrôler la période de validité de la clé utilisée. Il pourra soit se baser sur son horloge locale (synchronisée ou non) ou demander l'heure courante à une autorité d'horodatage selon les degrés de sécurité et de précision requis.

3.7.6.3. Révocation des clés

Toute clé active ou suspendue peut être révoquée dès lors que le titulaire du certificat suppose sa compromission. La révocation s'effectue en indiquant soit la valeur de la clé publique, la valeur de l'empreinte de cette clé ou encore l'indice au sein du certificat. En retour, le PSCE opère l'ajout de la clé dans la liste de révocation en ajoutant les informations $\{id,\{p,i\}\}$ où id désigne l'identifiant du certificat, p l'empreinte de la clé révoquée et i son indice parmi la liste des clés que renferme le certificat.

Bien que cet indice ne soit pas indispensable (l'empreinte numérique étant unique pour une clé donnée), il permet au vérificateur de vérifier directement l'empreinte sans devoir procéder par essais successifs sur les différentes clés que renferme le certificat.

> *Note : dans ce cas, il apparaît judicieux de gérer une liste de clés révoquées (Key Revocation List, KRL) en lieu et place de la liste de certificats révoqués (CRL) dont seuls les identifiants des certificats mis en opposition sont mentionnés. Un exemple de format de KRL est proposé en annexe.*

3.7.6.4. Révocation du certificat

Le certificat à multiples clés publiques peut être révoqué par son détenteur. Dans ce cas, il perd l'usage de l'ensemble des clés qu'il renferme. Toute signature réalisée à l'aide d'une clé non expirée devra être considérée comme invalide par le vérificateur. Les émetteurs de message confidentiels devront rejeter toute clé de chiffrement contenue dans ce certificat.

3.7.6.5. Expiration du certificat

Comme tout autre certificat, le certificat à multiples clés publiques est sujet à expiration. Dans ce cas, un éventuel renouvellement demandera la génération d'autant de bi-clés que de clés publiques contenues dans le certificat ayant expiré. Les usages et durées de validité des nouvelles clés seront identiques aux clés précédentes.

3.7.7. Incidences sur la signature électronique

3.7.7.1. Format de la signature électronique

Le fait que le certificat dispose de plusieurs clés publiques mène à fournir au vérificateur d'une signature le moyen de connaître la clé de vérification devant être utilisée (lorsque le certificat compte plus d'une clé de vérification valide).

Bien que le vérificateur puisse utiliser une par une les clés de signature du signataire jusqu'à vérification de sa signature numérique, la surcharge de calcul induite conduit à penser que :

- Le format de la signature indique intrinsèquement l'indice de la clé de vérification à utiliser et éventuellement l'empreinte numérique de cette clé afin d'éviter toute confusion.

- Ces informations sont jointes à la politique de création de signature soumise au vérificateur. Dans ce cas, l'algorithme d'empreinte utilisé peut être associé à un paramètre de la politique suivie.

3.7.7.2. Contrôle de la validité de la signature

Lorsqu'un MPKC est utilisé pour créer une signature électronique, la validation du certificat global peut ne pas donner d'informations pertinentes sur l'état de la clé de vérification. Comme indiqué précédemment, l'usage d'une liste de clés révoquées (KRL) est donc nécessaire puisque la CRL n'indique que les identifiants des certificats révoqués et non les identifiants (indice et valeur d'empreinte) des clés q'un MPKC renferme.

De même, une variante au protocole de validation en ligne OCSP [MYE 99] doit être fournie afin de valider une clé (ou un ensemble de clés) parmi celles disponibles au sein du certificat présenté au vérificateur.

Pour cela, la structure d'identification « ReqCert » [MYE 01] du certificat dont le vérificateur souhaite connaître le statut peut être modifiée en « ReqMPKC » comme suit :

```
ReqMPKC ::= SEQUENCE {
  reqCert           ReqCert,
  keyId             KeyIdentifier              OPTIONAL
}

ReqCert ::= CHOICE {
  certID            CertID,
  issuerSerial      [0] IssuerandSerialNumber,
  pKCert            [1] Certificate,
  name              [2] GeneralName,
  certHash          [3] EXPLICIT MessageImprint
```

```
  -- Hash of the certificate, MODIFIED FROM OCSP v2 draft
}

KeyIdentifier ::= SEQUENCE {
  keyIndex           INTEGER,
  keyHash            EXPLICIT MessageImprint   OPTIONAL
  -- From RFC3161: represents the public key hash value
}

Certificate ::= SEQUENCE {
-- Different from OCSP v2 standard structure
  certType           OBJECT IDENTIFIER,
  certValue          EXPLICIT ANY DEFINED BY certType
}
```

Cette structure remplace les champs « reqCert » originaux mentionnés dans les requêtes et réponses OCSP afin de supporter à la fois les certificats traditionnels (X.509) et les MPKC.

CHAPITRE 3 : PROPOSITION D'UN MODELE SECURISE

« En essayant continuellement, on finit par réussir. Donc : plus ça rate, plus on a de chances que ça marche »

Les devises Shadok

L'étude de la sécurité dans les systèmes répartis objet connus a mis en évidence les besoins de gérer la sécurité « au plus tôt », c'est à dire au sein même du système puisque les échanges doivent tenir compte des impératifs de sécurité des transactions. Ce chapitre propose une extension aux modèles répartis en apportant une gestion bas niveau de la confidentialité et de l'authentification forte et apporte un moyen de supervision et d'audit des échanges entre objets répartis.

L'intérêt de définir un tel modèle prend son essence dans les implications techniques et juridiques du recours conjoint à la cryptographie et à la signature électronique. Ce modèle, ouvert aux évolutions futures liées à la création et la validation des signatures électroniques, affranchit les développeurs de la gestion technique de l'authentification et de la confidentialité des échanges. Il permet de plus de gérer des droits d'accès conformément aux politiques de sécurité mises en place pour chaque objet dont l'accès est contrôlé.

1. Introduction

Etant donné les problématiques de sécurité auxquelles doit faire face un système réparti, que ce soit l'authentification des acteurs (parties), la protection de l'intégrité des données, la surveillance des échanges (dans un objectif de traçabilité certifiée), un recours conjoint aux solutions complémentaires de cryptage et signature s'avère nécessaire.

Il apparaît primordial que le middleware de déploiement des objets répartis prenne intrinsèquement en charge la sécurité des communications et le respect des règles de sécurité en vigueur.

Contrairement aux environnements répartis actuels, tels que CORBA, RMI ou DCOM, l'approche retenue consiste d'une part à utiliser le modèle « push » distribué [LAM 92] et d'autre part à intégrer le cryptage et la signature électronique directement au niveau de la couche middleware (couche virtuelle située au-dessus de la couche présentation du modèle ISO/OSI). Cette décision est justifiée par l'intégration des standards relatifs à la signature électronique reposant sur X.509 [HOU 02] ainsi que par l'utilisation des techniques de cryptage et de signature électronique au sein de l'ORB.

Ainsi, le modèle proposé, inspiré des principes RPC objet de CORBA tels que la notion de proxy (procuration) client et serveur ou la consultation de références via un service annuaire et de l'accès réparti aux ressources de SDSI (présenté en annexe), intègre des réceptacles permettant de mettre en œuvre des modules ad-hoc répondant aux problèmes :

- Du cryptage des informations échangées.
- De la construction et validation des signatures électroniques apposées aux données.
- De l'autorisation d'accès aux méthodes proposées par les objets serveur.
- De l'audit par le biais de la traçabilité et la supervision des échanges.
- De l'archivage des données échangées.

Les modules sont utilisés par le middleware par le biais de leurs interfaces. Ainsi, les solutions informatiques peuvent évoluer sans remettre en cause le fonctionnement global de la sécurité du système. Ceci permet en particulier au système d'évoluer en fonction des nouveaux standards disponibles sur le marché.

2. Vue d'ensemble

2.1. Modèle en couches

Le modèle présenté reprend les concepts de proxy client (« souche ») et serveur (« squelette ») des RPC et de CORBA. Le bus de transport des requêtes (« ORB ») se voit adjoindre le préfixe « S- » (« secure ») pour indiquer qu'il est sécurisé par l'utilisation conjointe de la cryptographie asymétrique et la signature électronique.

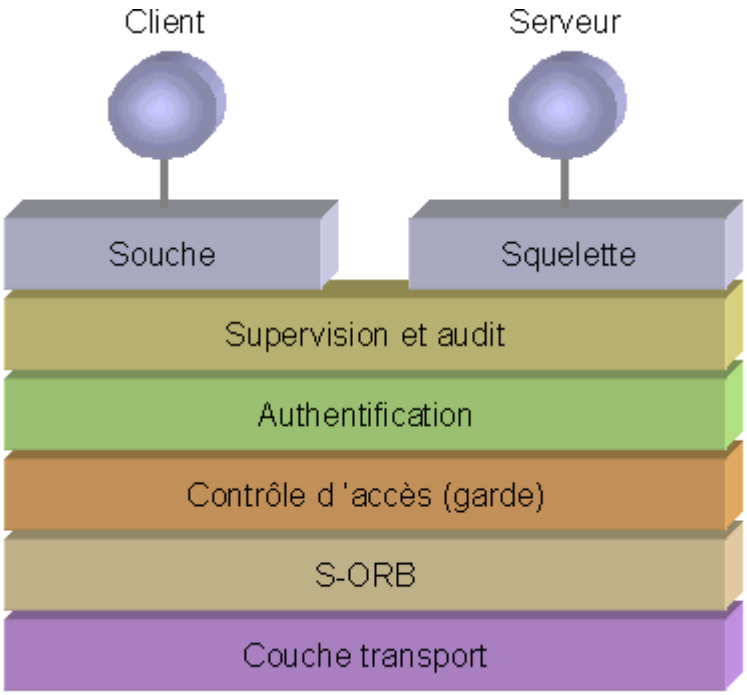

Figure 23 : organisation du modèle réparti object sécurisé

Trois modules (ou couches) s'insèrent entre les proxies RPC et le bus (figure 23) :

- Une couche de supervision qui permet de collecter des informations relatives aux appels entre objets et à divers indices de charge [COT 00].
- Une couche d'authentification met en œuvre la création et la vérification des signatures électroniques des objets prenant part aux échanges.
- Une couche de vérification des droits d'accès, appelée lorsque l'authentification a réussi. Ce module de droit d'accès intégré à l'ORB peut jouer le rôle de gardien des ressources mises en œuvre par l'objet serveur, de manière similaire au garde de SDSI [CLA 01] et fait appel à une liste de contrôle d'accès (« Access Control List », ACL) [CLA 01] locale ou

hébergées par un service de sécurité. L'objectif de ce module est de n'autoriser l'accès à la méthode demandée que dans le cas où le client détient les droits d'accès requis tels que mentionnés dans l'ACL.

Ce modèle permet dès lors d'échanger des informations entre objets de manière sécurisée et éventuellement dans un cadre légal, lorsque le module d'authentification supporte la signature électronique « sécurisée » (au sens juridique).

Les modules intégrés au modèle étant spécifiques à chaque objet, il est dès lors envisageable de dupliquer un même objet serveur tout en déclarant différents comportements d'authentification et de contrôle d'accès.

Note : le service annuaire, le plus couramment utilisé par CORBA, permet de découvrir les références (adresses) des objets serveur à contacter. En tant qu'objet serveur sécurisé, ce service doit de même faire l'objet d'une évolution de manière à intégrer le modèle présenté. De fait, les références des objets serveur ne seront accessibles qu'aux objets client disposant de droits d'accès minimaux. Ce service pourra de plus supporter des algorithmes de répartition d'invocation [COT 00] afin de limiter la charge des objets serveur contactés.

2.2. Organisation interne

Nous présentons quelques diagrammes de classes relatifs à la mise en œuvre de la gestion de la sécurité, de la supervision et de l'audit par le middleware. Nous nous inspirons pour cela du formalisme de [LAI 00] relatif à la représentation d'objets Java sous forme de diagrammes UML (« Unified Modelling Language »). Les erreurs pouvant être générées par les méthodes proposées sont indiquées au niveau des classes par des commentaires UML.

Le bus est composé d'une partie cliente et d'une partie serveur. Toutes deux sont dérivées d'une classe générale proposant des fonctionnalités communes appelée « GenericObject » et modélisée par la figure 24. En particulier, cette classe prend en charge la gestion des références d'objets serveur ou mixte et notamment l'enregistrement de références. Ces références (ou IOR, « Interoperable Object Reference ») permettent aux objets clients d'accéder aux informations d'IP et de numéro de port des objets serveur ou mixte. Ces références doivent être normalisées afin que les clients soient en mesure d'accéder aux informations d'accès aux objets de type serveur ou mixte qu'ils souhaitent contacter. Un exemple de format est indiqué en annexe.

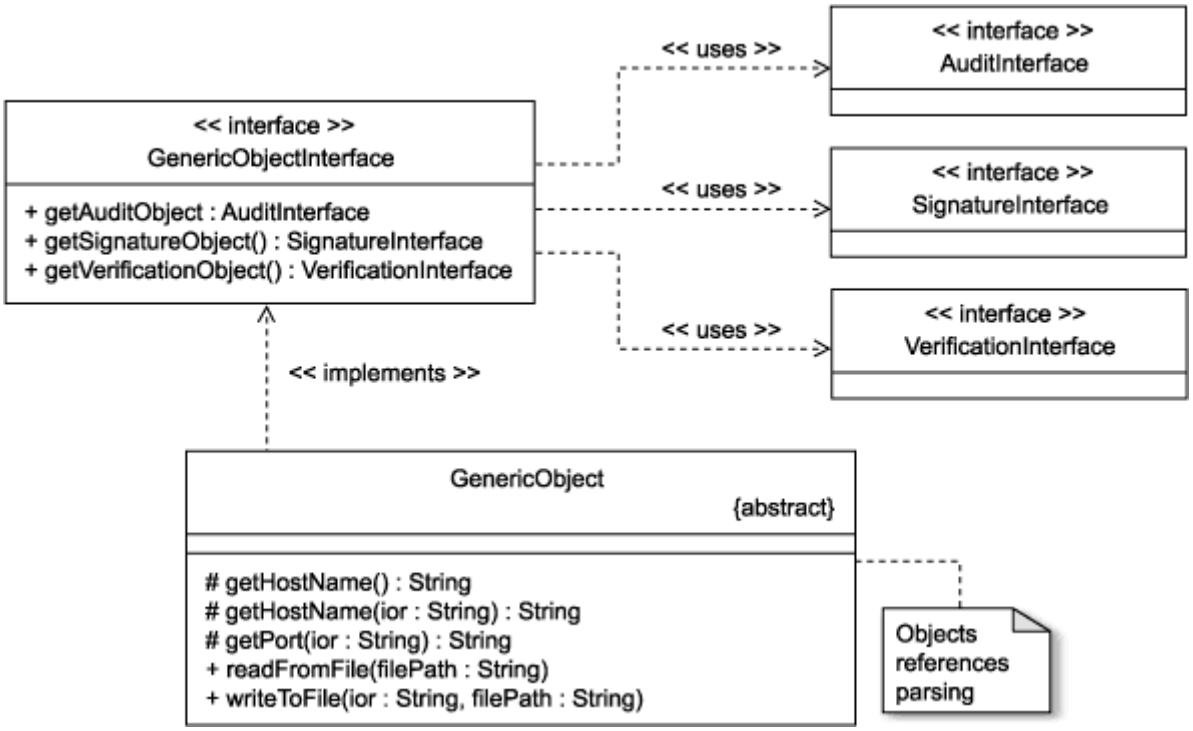

Figure 24 : schéma de la classe générique

Le module d'audit est chargé de collecter des informations relatives aux appels entre objets. Il est commun aux objets client et serveur puisque les informations sont généralement enregistrées localement.

Il en est de même des modules de signature et vérification. Leur séparation est un choix de conception justifié par le fait qu'ils peuvent faire appel à des technologies différentes et peuvent ainsi évoluer indépendamment : la génération d'une signature à l'aide d'un mot de passe peut de fait être remplacée par une empreinte digitale sans que le processus de vérification en soit affecté.

La classe principale « GenericObject » est spécialisée en une partie cliente « GenericClient » et une partie serveur « GenericServer », présentées par la figure 25, qui seront détaillées par la suite. Bien que ces dernières partagent les mêmes modules de signature, vérification et audit, chacune d'elles dispose de son propre module de supervision. En effet, le système de supervision doit être en mesure de distinguer les informations selon qu'elles proviennent de l'objet client, à l'initiative d'une requête, ou de l'objet serveur chargé de répondre à cette requête.

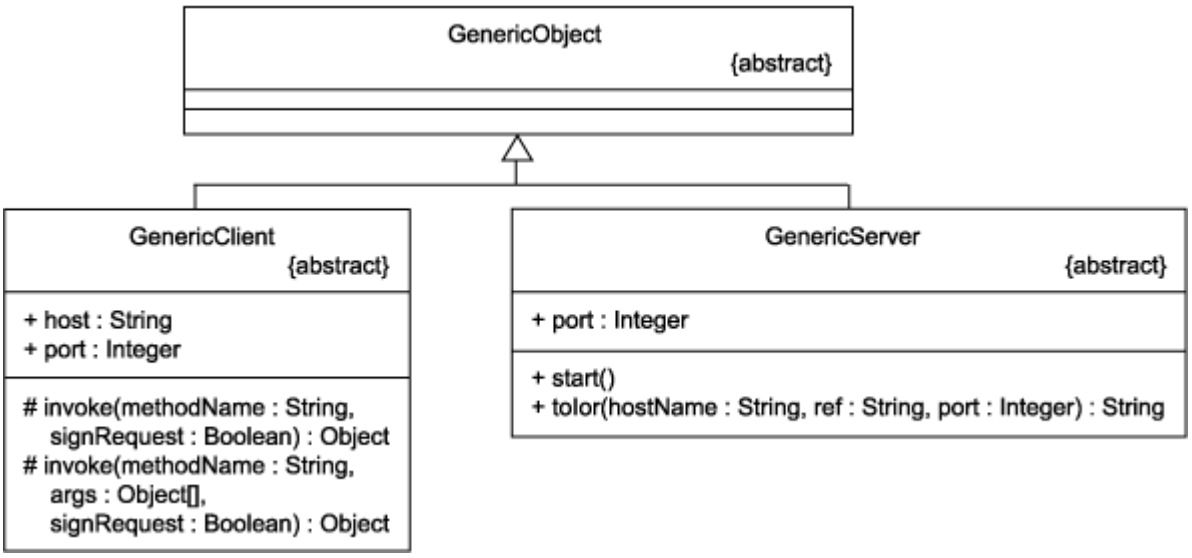

Figure 25 : schéma des classes des parties client et serveur du bus

Les interfaces de signature, vérification et audit seront détaillées lors de la présentation des classes composant les objets serveurs.

2.2.1. Côté client

Le bus S-ORB du côté client prend en charge la redirection des appels locaux à la souche (« stub ») vers l'objet serveur (ou mixte) distant.

2.2.1.1. Client, interface de supervision

L'interface de supervision « ClientSupervisionInterface » est intégrée au modèle au sein des méthodes « invoke() », comme l'indique le diagramme de la figure 26. A chaque appel d'une méthode située sur un objet réparti (local ou distant), le modèle exécute l'une des deux méthodes « sendCall() » selon que la méthode demandée contient ou non des arguments.

Ces méthodes ne génèrent aucune erreur afin que le déroulement des échanges ne soit pas perturbé par des erreurs liées à la supervision.

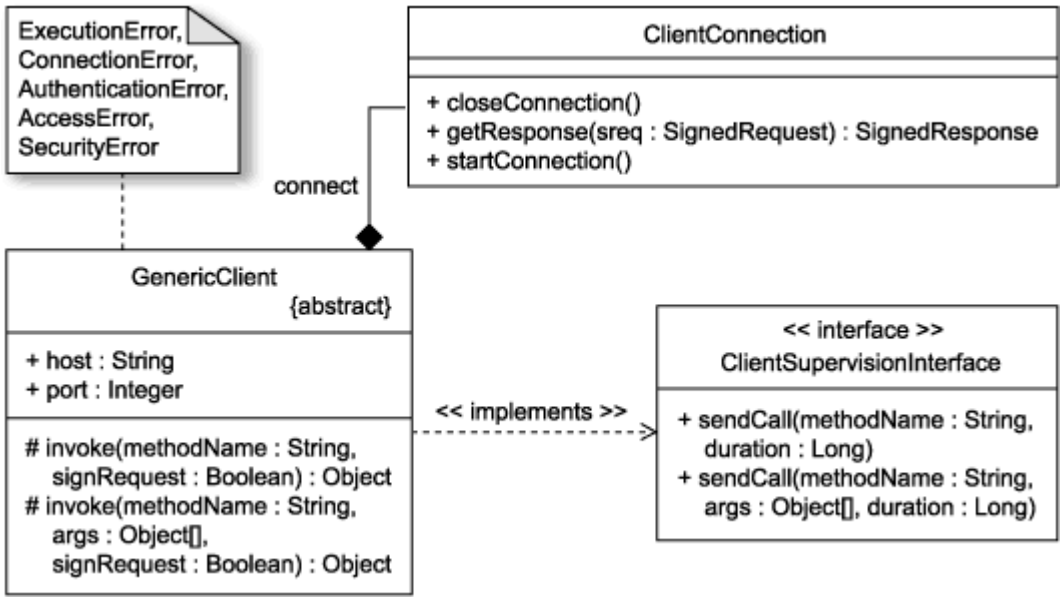

Figure 26 : schéma des classes du client

Enfin, la durée de l'appel réalisé par le client, de la création de sa requête au retour d'une réponse fournie par l'objet serveur ou mixte, est calculée en interne par le modèle. Le système de supervision est ainsi déchargé de la gestion des horloges.

2.2.1.2. Proxy client

Le proxy client ou « stub » est représenté par la classe abstraite « GenericClient » (figure 27). Il permet de se connecter à un objet serveur ou mixte et de réaliser l'appel à une méthode déclarée dans le contrat S-IDL de cet objet.

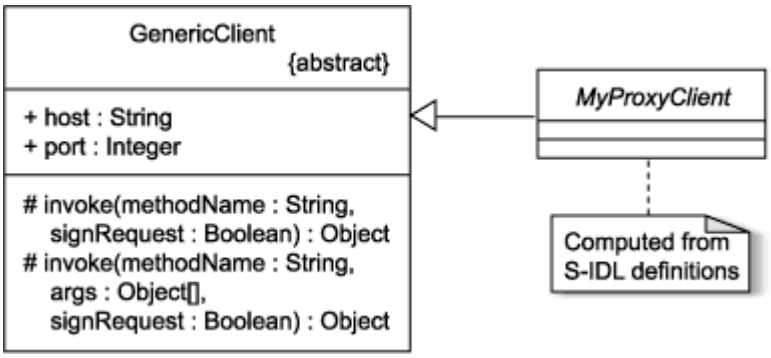

Figure 27 : schéma des classes du proxy client

Les méthodes « invoke() » prennent en charge les opérations liées à la souche cliente : génération et empaquetage de la requête, attente de la réponse (en mode synchrone) et restitution du résultat de l'appel. Les différents message d'erreurs en

provenance de l'objet serveur (ou mixte) contacté et divulgués au client sont indiqués par la figure 26.

2.2.2. Côté serveur

2.2.2.1. Serveur, interface de supervision

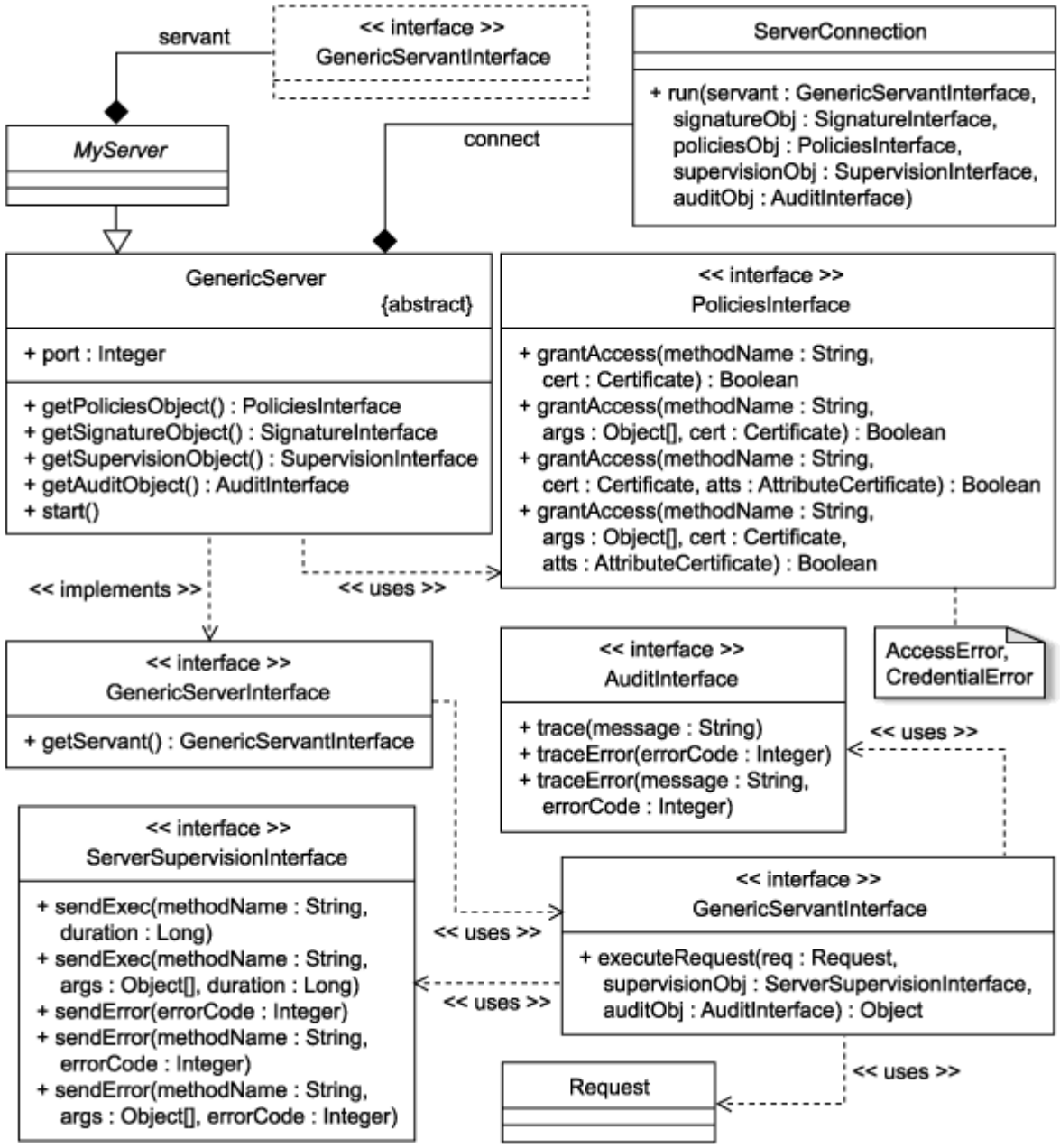

Figure 28 : schéma des classes du serveur

La méthode « grantAccess() » de l'interface « PoliciesInterface » décrite dans la figure 28 spécifique aux objets serveur effectue la liaison entre la signature de la requête reçue par un objet serveur ou mixte et sa liste de contrôle d'accès (ACL). La valeur de retour indique si l'accès a été accepté ou refusé et conditionne l'exécution du traitement associé à la requête.

Le module d'audit réalise, par le biais de l'interface « AuditInterface », les opérations liées à la traçabilité des opérations. Cette traçabilité revêt une importance fondamentale lorsqu'il s'agit de fournir des éléments de preuve de l'existence des échanges et des traitements effectués. Ce module d'audit peut ainsi faire appel à un système évolué mettant lui-même en œuvre les concepts relatifs à l'utilisation de la signature électronique.

L'objectif du module de supervision, intégré au système à l'aide l'interface « ServerSupervisionInterface » est de notifier les échanges d'un point de vue externe. Les informations enregistrées par le système de supervision, tant au niveau client que serveur, plus volatiles que les informations d'audit, se destinent à l'amélioration de la QoS générale du système réparti et non à une quelconque preuve juridique. Le système de supervision connecté peut faire appel à des mécanismes évolués, tels que ceux débattus par [PAL 01].

2.2.2.2. Proxy serveur

Le proxy serveur ou « squelette » reçoit les requêtes des clients et prend en charge leur exécution par le biais de la méthode « executeRequest » de la figure 29. Le résultat générique obtenu est alors transmis au serveur qui va tout d'abord construire la réponse, puis la signer et la chiffrer, avant de la transmettre au client.

L'exécution de la requête transmet également des informations de supervision et d'audit aux systèmes concernés. Une gestion interne de la durée d'exécution de la requête peut être proposée afin de transmettre le délai d'exécution au système de supervision employé [COT 00].

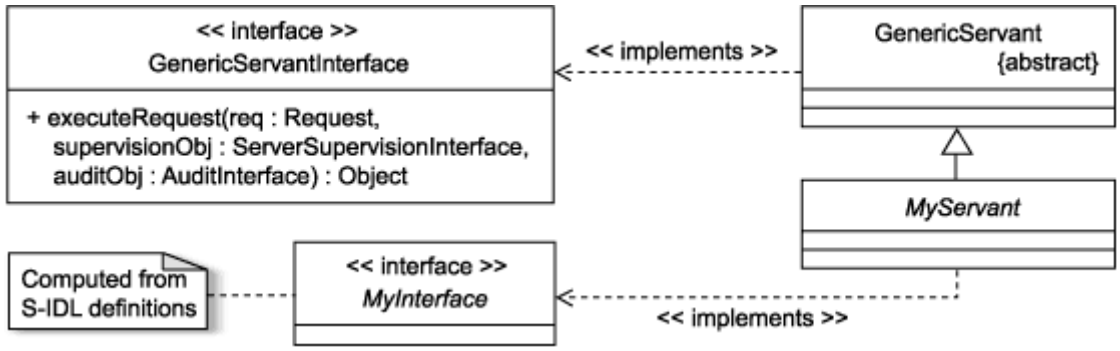

Figure 29 : schéma des classes du proxy serveur et interfaces

Il se peut que le proxy serveur ne soit en mesure d'exécuter la requête. Dans ce cas, un résultat nul est renvoyé au serveur. Ce dernier crée alors une réponse mentionnant une erreur qu'il retourne au client.

2.2.3. Interface de signature et CTL

L'interface de signature « signatureInterface » de la figure 30 fournit les méthodes nécessaires à la création et à la vérification des signature numériques. Elle fait appel à une liste de certificats de confiance (« Certificate Trusted List », CTL) comprenant les certificats des différentes autorités reconnues par les objets répartis. Bien entendu, chaque objet peut disposer se sa propre CTL puisque le bus S-ORB est un composant local, spécifique à chaque objet.

La création d'une signature numérique sur un contenu quelconque fait appel à la génération d'une empreinte numérique. L'algorithme employé est indiqué par la méthode « getSignatureAlgorithmOID() ». Connaissant cet algorithme, la clé privée est utilisée pour générer une signature lors d'un appel à l'une des méthodes « sign ».

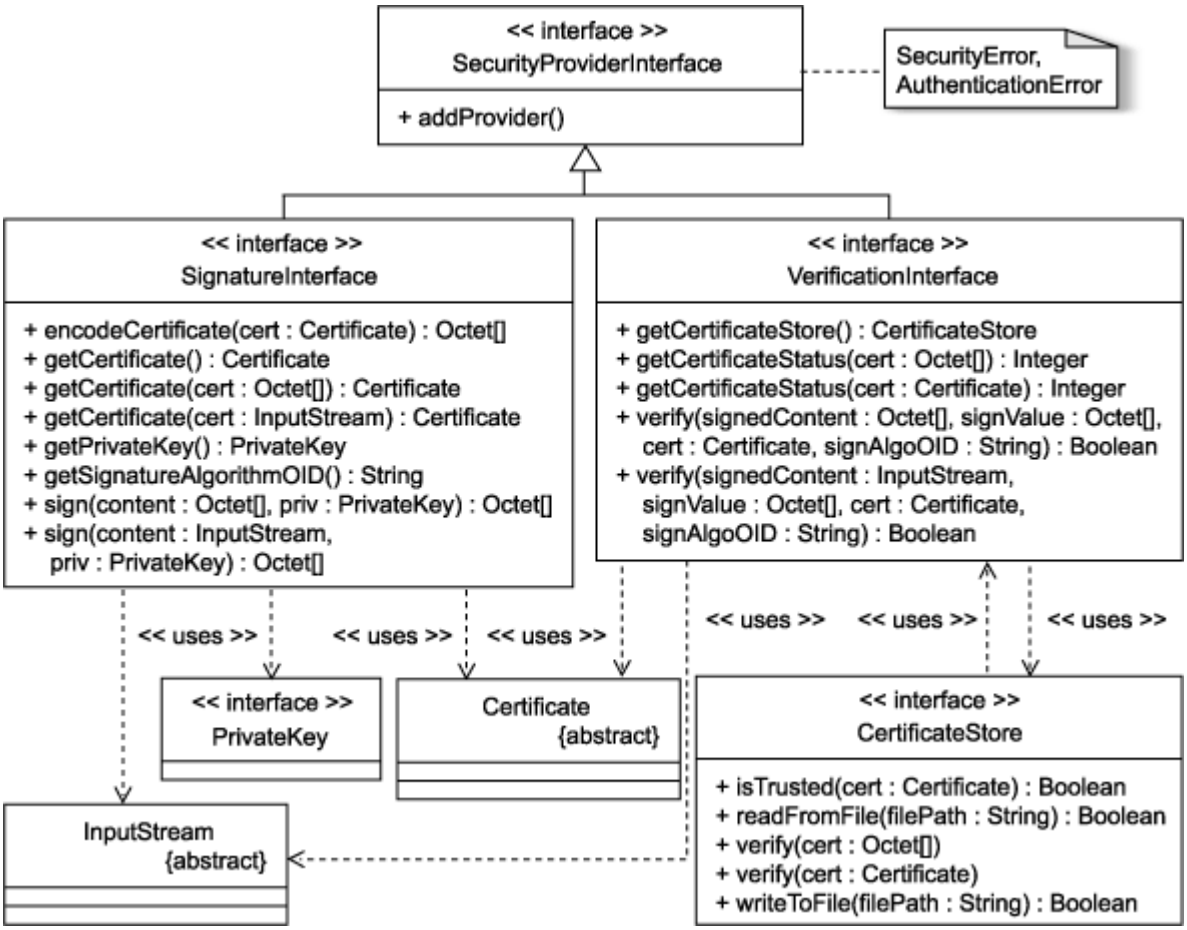

Figure 30 : schéma des classes de signature et CTL

Le processus de vérification demande d'une part la vérification de la signature numérique mais également la vérification du certificat du signataire. C'est pourquoi les méthodes de vérification et notamment « validateCertificate() » font appel à une ou plusieurs CTL désignées par l'interface « CertificateStore » ou « magasin de certificats » (de confiance). Le magasin, lui-même signé électroniquement, peut être authentifié par l'exécution d'une méthode « verify() » en indiquant le certificat de l'autorité émettrice. Ce dernier peut de plus être validé en utilisant les méthodes proposées par l'interface de vérification et notamment les méthodes « getCertificateStatus() ».

Il est de plus possible de vérifier la non compromission de la clé privée du signataire dès lors que le processus de création des signatures introduit au moins un horodatage à l'aide les protocoles mentionnés précédemment (multi-horodatages indépendants ou horodatages multiples).

Cependant, la vérification du statut du certificat et particulièrement de sa révocation n'est pas prise en charge lors de la création d'une signature par l'interface de signature. Le contrôle du statut du certificat du signataire est laissé à l'appréciation du développeur et de la solution technique mise en œuvre : cette dernière peut par exemple recourir à une CRL ou interroger un serveur en ligne tel un serveur OCSP. Les étapes nécessaires à la validation du certificat du signataire de la requête, décrites dans le second chapitre, doivent être mises en œuvre lors de l'appel aux méthodes « verify() ». Le recours à un service de validation apporte une solution aisée à mettre en œuvre.

2.2.4. Proposition d'un service de validation

Afin de limiter la charge de traitement liée à la validation systématique des certificats lors des échanges sécurisés ainsi que les interrogations en ligne (appel à un serveur OCSP par exemple), il est possible d'envisager le recours à un service local partagé par l'ensemble des objets présents sur une machine donnée. Ce service de validation de certificat (SVC) met à la disposition des objets répartis locaux un espace qu'ils peuvent interroger afin de connaître la validité d'un certificat donné. Deux cas peuvent se produire :

- Le certificat demandé n'est pas répertorié par le service de validation local : dans ce cas, ce dernier prend en charge l'interrogation de la liste de révocation locale ou d'un serveur en ligne afin de déterminer le statut de validité du certificat demandé.

- Le service a déjà opéré la validation du certificat :

  - Lorsque la date de la dernière validation est récente (inférieure à un délai d'expiration maximum), la valeur de statut courante est retournée.

  - Dans le cas contraire, le service de validation procède à une mise à jours de la valeur de statut du certificat.

Non seulement le SVC offre l'avantage d'externaliser la validation des certificats et ainsi harmoniser les procédés de validation, il permet en outre de mieux gérer les communications vers les serveurs distants lorsque les objets répartis souhaitent valider les certificats qu'ils reçoivent lors de leurs échanges.

Le SVC peut être avantageusement contacté au sein des méthodes « getCertificateStatus() » que propose l'interface de vérification de signature. Il peut offrir un service étendu de validation du chemin de certification en faisant appel à la méthode « getCertificateStore() » de l'objet demandeur afin de décharger ce dernier de cette vérification.

3. Gestion des contrôles d'accès

Le modèle présenté permet de définir des restrictions d'accès aux méthodes des objets serveur. Il est souhaitable que ces règles soient publiquement (ou éventuellement après une première authentification) accessibles afin que le client puisse déterminer s'il dispose des droits requis avant l'appel de l'objet serveur.

Une première méthode consiste à faire appel à un fichier de stratégie de sécurité comparable à la mise en œuvre de la sécurité en Java [JAW 01] : les restrictions concernent en premier lieu l'exécution de code Java (applets en particulier) et l'accès à des dossiers situés sur le poste local. Par défaut, toute opération est interdite. La création de ce fichier texte ouvre certains accès par le biais d'un triplet comportant :

- Le type de permission demandée : exécution de code, accès à un fichier ou un dossier, etc.
- L'objet cible de la permission : une ressource Internet ou URL, un chemin menant au fichier ou au dossier, un nom de domaine, etc.
- L'ensemble des actions autorisées : lecture, écriture en particulier.

Le principal inconvénient du recours à un fichier de stratégie de sécurité provient de la nécessaire installation de ce fichier sur le poste client. De fait, la mise en place d'une stratégie de sécurité globale et évolutive est compromise.

Une seconde proposition envisagée consiste à faire appel à un service (au sens CORBA) intermédiaire en possession d'un annuaire des objets et de leurs restrictions d'accès tel que défini par la méthode « pull ». Le service annuaire semble donc indiqué pour remplir cette tâche. Tout objet serveur sécurisé s'identifie et enregistre ses restrictions d'accès auprès de ce service. Ce dernier répond aux clients désireux de prendre connaissance des droits d'accès d'une méthode d'un objet serveur sécurisé identifié. Cette approche peut néanmoins poser des problèmes de surcharge du service concerné. Des règles d'accès réparties sont alors préférables.

Ainsi, une troisième solution est de faire appel à un contrôle d'accès réparti sous la forme d'une liste de droits d'accès (« Access Control List », ACL) locale à chaque objet serveur ou mixte (ou au middleware sous-jacent). C'est pourquoi l'approche retenue consiste à les mettre à disposition des clients en premier lieu par le biais du contrat IDL du serveur. Ces règles pourront être affinées (voire modifiées) en temps réel par un service de gestion des droits d'accès (« credentials ») centralisé. Parallèlement, différentes versions du contrat IDL reflétant ces modifications pourront être soumises aux clients par un mécanisme de rappel (ou « callback ») tel que mis en œuvre dans CORBA. D'autre part, la génération automatique des proxies client et serveur s'en trouve facilitée ; en effet, les méthodes proxies issues de la compilation de l'IDL définissent directement l'ensemble des erreurs susceptibles d'être rencontrées.

3.1. Evolution du contrat IDL : le S-IDL

Le contrat IDL (« Interface Description Language ») permet de mettre à disposition des clients les diverses méthodes mises en œuvre par un serveur réparti. Ainsi le client est tenu au courant de la syntaxe, des paramètres et du retour éventuel des appels qu'il souhaite effectuer.

Jusqu'à présent, les contrats IDL ne permettent pas de restreindre l'accès à un objet ou une méthode. De fait, toute méthode déclarée dans l'IDL d'un objet est publique et peut être appelée dès lors que la référence de cet objet est connue du client.

Dans un contexte de sécurité où l'accès aux méthodes est contrôlé, le client doit pouvoir connaître le plus tôt possible les restrictions qu'impose le serveur. Ainsi, le contrat IDL semble adapté à cet effet. Ce contrat « sécurisé » S-IDL (« secure-IDL ») doit alors non seulement indiquer la liste des méthodes publiques, pouvant être invoquées à distance, mais également informer le client des éventuelles règles de sécurité qui s'y rapportent.

L'introduction de ces règles est réalisée par l'ajout de nouveaux mots-clés pour former le contrat IDL sécurisé S-IDL :

Mot-clé	Signification
AUTHENTICATED	Une politique de sécurité basée sur la signature électronique s'applique au serveur et/ou au client
SECURE	Lors de la phase d'échange des clés (SSL [LAN 01] par exemple), les règles définies s'appliquent. Les informations échangées par la suite sont alors confidentielles entre les interlocuteurs
CLIENT	Les règles définies concernent la partie cliente
SERVER	Les règles définies concernent la partie serveur
BOTH	Indique que les règles décrites s'appliquent à la fois au client et au serveur
NONE	L'ensemble des règles de contrôle prédéfinies (au niveau de la définition de l'interface par exemple) ne s'appliquent pas à l'entité dont elles se rapportent (client ou serveur)
TIME_STAMPED	Un horodatage certifié est requis. Le nombre de certificats d'horodatage requis peut également être indiqué dans le cas où plus d'un horodatage est demandé

Les autres éléments lexicaux de cette grammaire reposent sur les définitions grammaticales de l'OMG IDL : modules, interfaces et signatures de méthodes.

> *Note : la conjonction du chiffrement et de la signature électronique est possible dans le cas où authentification et confidentialité sont requises. La signature intervient alors en amont du chiffrement du fait d'une part que le signataire doit prendre connaissance du contenu signé [PEC 99] et d'autre part que la conservation du contenu et de ses signatures devient problématique dès lors qu'une clé de déchiffrement doit être également conservée.*

3.1.1. Proposition de grammaire

La grammaire générale d'écriture d'une méthode en S-IDL supportant des restrictions d'accès est la suivante :

```
<METHOD> ::=
  <ReturnedResult> <methodName> "(" <param> ")"
  <AUTHENTICATED>
  <SECURE> ";"

<AUTHENTICATED> ::=
"AUTHENTICATED {"
      <AuthenticationRules>
"}"
<AUTHENTICATED> ::=

<SECURE> ::=
  "SECURE {"
    <SecurityProtocols>
  "}"
<SECURE> ::=
  "SECURE {"
    <SecurityProtocols>
    <ISSUERS>
  "}"
<SECURE> ::=

<AuthenticationRules> ::=
  <AuthenticationRule> ","
  <AuthenticationRules>
<AuthenticationRules> ::= <AuthenticationRule>

<SecurityProtocols> ::=
  <SecurityProtocol> ","
```

```
  <SecurityProtocols>
<SecurityProtocols> ::= <SecurityProtocol>

<AuthenticationRule> ::=
  <SRC> "=>" <RuleValues>

<RuleValues> ::= <RuleValue> "," <RuleValues>
<RuleValues> ::= <RuleValue>

<RuleValue> ::= <ISSUERS>
<RuleValue> ::= <IPS> -- Allows to restrict IP addresses
<RuleValue> ::= "NONE"
<RuleValue> ::= <GENERAL_NAME>
<RuleValue> ::= <Attributes>
<RuleValue> ::= "TIME_STAMPED"
<RuleValue> ::= "TIME_STAMPED(" <NB> ")"

<SRC> ::= "CLIENT"
<SRC> ::= "SERVER"
<SRC> ::= "BOTH"

<ISSUERS> ::= "ISSUER_ID IN" "[" <ISSUER_IDS> "]"

<ISSUER_IDS> ::= <GENERAL_NAME> "," <ISSUER_IDS>
<ISSUER_IDS> ::= <GENERAL_NAME>
<ISSUER_IDS> ::= "SN=" <NUMBER>
                 -- Issuer serial number

<IPS> ::= "IP IN" "[" <IP_LIST> "]"

<IP_LIST> ::= <IP_ADDRESS> "," <IP_LIST>
<IP_LIST> ::= <IP_ADDRESS>

<IP_ADDRESS> ::=
  <NUMBER> "." <NUMBER> "." <NUMBER> "." <NUMBER>

<NUMBER> ::= <NB> <NUM>

<NUM> ::= "0" .. "9" <NUM>
<NUM> ::=

<GENERAL_NAME> ::= <FIELD> "=" <VALUE>
<GENERAL_NAME> ::=
  <FIELD> "=" <VALUE> ","
  <GENERAL_NAME>

<Attributes> ::= "ATTRIBUTES IN" "[" <Attrs> "]"
<Attributes> ::=

<Attrs> ::= <AttVal> "]" "[" <Attrs>
<Attrs> ::= <AttVal>
```

```
<AttVal> ::= "REQUIRED" <OID>
<AttVal> ::= <OID>

<OID> ::= <NUMBER> "." <NUMBER> "." <OID_NEXT>
-- Must represent an Object Identifier

<OID_NEXT> ::= <NUMBER> "." <OID_NEXT>
<OID_NEXT> ::=

<FIELD> ::= <X509_FIELD>
<FIELD> ::= STRING  -- For extra field ids

<X509_FIELD> ::= "CN"  -- Common Name
<X509_FIELD> ::= "O"   -- Organization
<X509_FIELD> ::= "OU"  -- Organization Unit
<X509_FIELD> ::= "P"   -- Postal address
<X509_FIELD> ::= "E"   -- Email address
<X509_FIELD> ::= "R"   -- Rule within the organisation
<X509_FIELD> ::= "S"   -- State
<X509_FIELD> ::= "L"   -- Locality
<X509_FIELD> ::= "C"   -- Country

<VALUE> ::= <STRING1>

<STRING1> ::= <CHAR> <STRING>

<STRING> ::= <CHAR> <STRING>
<STRING> ::=

<CHAR> ::= -- Character type not defined in this grammar

<NB> ::= NUMBER  -- Must be greater than or equal to 1

<SecurityProtocol> ::= "NONE"
<SecurityProtocol> ::= "SSLv1"
<SecurityProtocol> ::= "SSLv2"
<SecurityProtocol> ::= "TLS"      -- Similar to SSLv3
<SecurityProtocol> ::= "DH"       -- Diffie-Hellman
<SecurityProtocol> ::= <STRING1>  -- Any other protocol

<param> ::= -- Described in CORBA IDL reference
```

Il apparaît que chaque méthode peut être soumise à des contraintes de droit d'accès et de confidentialité via les sections « AUTHENTICATED » et « SECURE ».

L'objet lui-même peut de plus être soumis à des règles de sécurité. Une syntaxe similaire s'applique alors non plus à chaque méthode mais à l'interface :

```
<INTERFACE> ::= "interface" <STRING1>
 <AUTHENTICATED>
```

```
 <SECURE>
 "{" <METHODS> "}"

<METHODS> ::=
  METHOD
  METHODS
<METHODS> ::=
```

Cette grammaire S-IDL ne définit pas de règles s'appliquant aux modules. Les modules ont en effet un rôle d'organisation du code avant tout.

3.1.2. S-IDL, objets signés et interopérabilité

Chaque méthode déclarée au sein de l'interface obéit alors aux règles de sécurité de l'interface. Ces règles peuvent être considérées comme des règles par défaut puisque les méthodes peuvent les compléter (ou invalider certaines règles) lors de leur description.

De fait, l'étude de cette grammaire permet de définir une granularité de droits d'accès et de chiffrement à la fois au niveau de l'interface, auquel cas les directives de sécurité s'appliquent à l'ensemble des méthodes qui la composent, ainsi qu'au niveau de chaque méthode d'un objet serveur. Ainsi, un même serveur peut gérer l 'appel de ses méthodes de manière différente en appliquant diverses politiques d'appel.

Le recours au contrat S-IDL permet ainsi d'obtenir une granularité au niveau de l'appel des méthodes alors que le concept d'objet signé (« SignedObject ») défini en Java [JAW 01] s'intéresse uniquement à la signature d'objets sérialisables, c'est à dire pouvant être représentés sous forme de flux de données (tel une structure de données ou un programme). De plus, la signature de tels objets n'est valide qu'au sein d'infrastructures mises en œuvre en langage Java. Ainsi, l'interopérabilité entre plates-formes et langages hétérogènes ne peut être conservée dès lors que le langage Java est imposé.

Enfin, le recours à des techniques telles que le référentiel d'interfaces (IREP, « Interface Repository ») et rappels de CORBA peut être envisagé afin d'assouplir les mises à jour des contrats S-IDL.

3.1.3. Exemple d'application

Voici un exemple d'utilisation de la grammaire proposée. L'objet serveur considéré est appelé « MyObjectInterface ». Il définit une unique méthode, « myMethod », faisant appel d'une part à une communication sécurisée et d'autre part à l'authentification du client par le biais des informations contenues dans son certificat.

3.1.3.1. Contrat S-IDL

```
interface MyObjectInterface
 AUTHENTICATED {
   CLIENT  =>  C="FR"
   SERVER  =>  TIME_STAMPED(2)
 }
 SECURE {
   SSLv2, TLS
 }
{
 void myMethod(in boolean param)
   AUTHENTICATED {
     CLIENT  =>  O="UTBM", L="BELFORT",
     CLIENT  =>  O="MSG-SOFTWARE", L="BELFORT",
     CLIENT  =>  ISSUER IN [O="CERT-SOFTWARE"],
     SERVER  =>  NONE
   };
}
```

3.1.3.2. Analyse de l'exemple

L'objet serveur supporte une méthode appelée « myMethod ». Cette méthode est régie par une politique de sécurité définie par l'interface « MyObjectInterface » comme suit :

- Chaque appel (émanant du client) doit être signé (de par la présence du mot-clé « AUTHENTICATED ») puis chiffré à l'aide d'un protocole parmi ceux supportés par le serveur : Secure Socket Layer ou Transport Layer Security (« SSLv2, TLS »). Cette opération de chiffrement est la dernière phase permet de conserver la confidentialité de l'envoi au serveur. Le protocole utilisé pour l'envoi chiffré doit être supporté par le serveur.

- Toute requête doit émaner d'un client résidant en France (« FR »).

- Par défaut, le serveur ajoute deux valeurs d'horodatage (« TIME_STAMPED(2) ») à ses réponses puis chiffre la transmission au client (car l'interface est de type « SECURE »). Il convient de noter que le chiffrement de la réponse est la dernière étape avant envoi.

Au niveau de la méthode, les règles suivantes s'appliquent :

- Le client est soit un membre de l'UTBM (« O="UTBM" »), soit un employé de la société MSG-SOFTWARE, toutes deux situées à BELFORT (« L="BELFORT" »). Son certificat d'identité joint à sa signature doit émaner de l'autorité de certification CERT-SOFTWARE (« ISSUER IN [O="CERT-SOFTWARE"] ») pour que sa signature soit acceptée.

- Les réponses du serveur ne sont contraintes par aucune politique de sécurité (elles ne sont ni signées ni horodatées) : les règles définies au

niveau des méthodes s'appliquent en priorité par rapport aux règles de l'interface (« NONE »). Par contre, les réponses demeurent chiffrées.

3.2. Via un service de sécurité indépendant

Lorsque les politiques de sécurité sont trop complexes ou évoluent au cours du temps de manière régulière, un procédé de découverte dynamique doit être supporté. Ainsi, un service de sécurité trouve sa place au sien du système réparti. Comparable à un service annuaire, il permet au serveur d'enregistrer les règles de sécurité qu'il applique et au client de découvrir ces règles. Ce service est nécessaire lorsque le système réparti fait appel à des services standard tels que le service annuaire ou le service vendeur. Les interfaces de ces derniers sont donc rédigées en IDL traditionnel. La gestion des accès clients à ces services sera alors prise en charge par le service de sécurité.

Un modèle de notification peut également s'intégrer à ce service afin de notifier les clients lors d'une mise à jour des règles de sécurité.

Un problème de persistance des règles peut se poser afin que le serveur n'ait pas à soumettre ses règles d'appel au service de sécurité lors de chaque redémarrage de ce dernier.

Ce service de sécurité pourra enfin faire appel à un modèle équivalent à celui du service sécurité de CORBA [OMG 02]. Dans ce cas, le service annuaire indiquera non pas directement les droits d'accès requis par les objets serveur enregistrés mais plutôt un lien vers le service de sécurité concerné.

3.3. S-IDL et certificat à multiples clés publiques

Bien que X.509 soit le format d'identification privilégié des objets répartis sécurisés, l'utilisation du certificat à multiples clés publiques (MPKC) s'intègre dans cette logique sécuritaire.

En effet, si l'on considère que chaque objet souhaite à la fois signer et chiffrer ses communications, il doit disposer d'au moins deux certificats distincts (en référence aux règles de sécurité énoncées lors de la présentation du certificat X.509 ainsi que par [MEL 01]). Il est alors possible d'utiliser des MPKC contenant chacun une clé de chiffrement (utilisée à des fins de confidentialité) et une clé de vérification de signature (employée pour vérifier une signature électronique) issues de bi-clés différents.

Le nombre d'entrées du référentiel des certificats mis à disposition des objets se trouve de fait divisé par deux. De plus, les objets peuvent contrôler plus facilement les chiffrements et signatures puisqu'une seule vérification du chemin de certification est nécessaire.

4. Format des échanges

Une fois le contrat S-IDL rédigé, le middleware doit proposer des structures de données standardisées afin de transmettre les informations des requêtes et réponses entre objets répartis.

Les structures des requêtes et réponses proposées prennent en considération la signature électronique et non le cryptage car, comme vu précédemment, la phase de cryptage agit comme une protection externe des informations échangées entre les objets répartis.

4.1. Schéma général

Les communications entre objets client et serveur utilisent le paradigme RPC d'envoi de message par le biais des deux méthodes « sendMessage() » de la classe abstraite « CommonConnection », définies au sein de la figure 31.

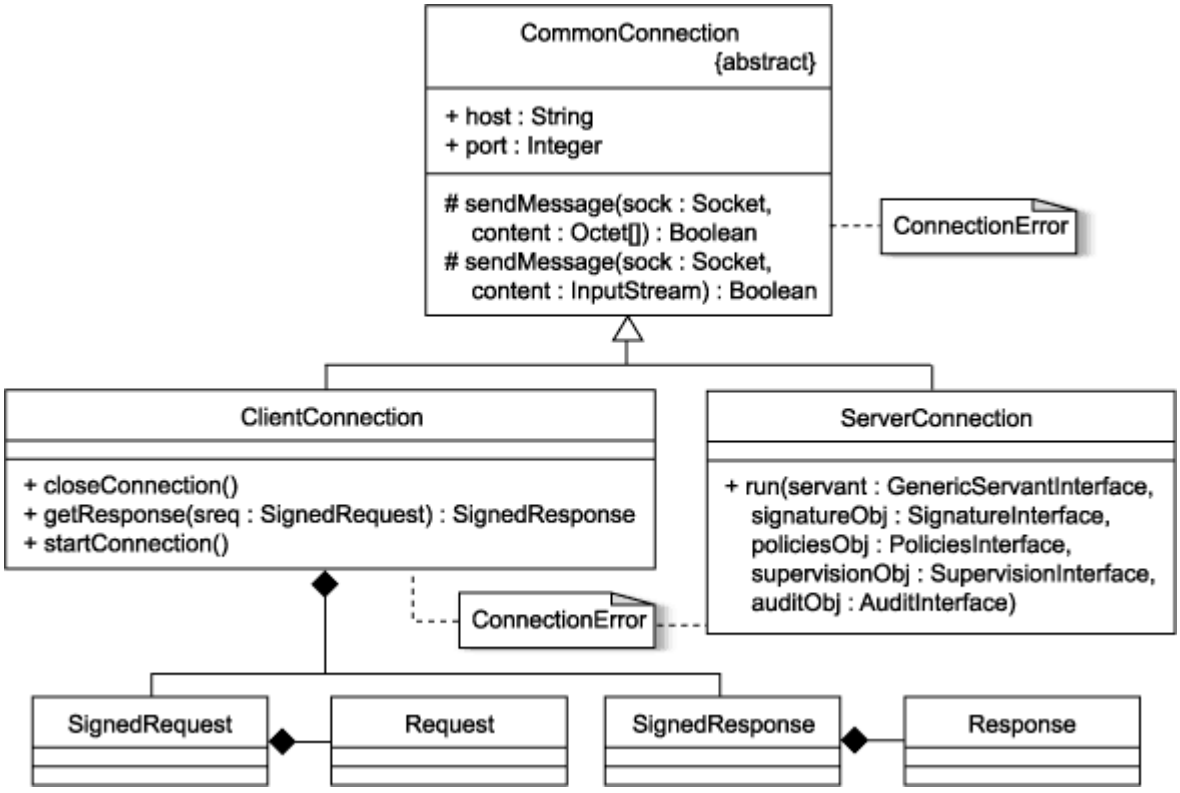

Figure 31 : schéma des classes de connexion des objets répartis

Le serveur met à jour ses informations d'hôte et de port de communication avant d'exécuter la méthode « run() ». Cette dernière prépare le serveur à recevoir des appels en provenance des clients et met à jour les différents éléments de signature,

vérification, traçabilité, audit et gestion des politiques d'accès aux méthodes S-IDL. Le middleware prend intrinsèquement en charge la création des réponses lors de la réception de demandes formulées par les clients.

Le client débute sa session en appelant la méthode « startConnection() ». Suite à cette invocation, il peut demander au serveur contacté la réalisation d'un service proposé dans son contrat S-IDL par le biais de la méthode « getResponse() ». Il termine sa session et clôt la connexion avec le serveur à l'aide de la méthode « closeConnection() ».

Ce schéma permet à un client de se connecter à un ou n serveurs simultanément, sachant que la réalisation des communications peut faire partie de processus exécutés en parallèles (« threads »).

4.2. Proposition de format

Le format de description choisi est l'ASN.1 du fait qu'il est le format par défaut de présentation des structures de données relatives à la signature électronique. De plus, il permet une abstraction du langage de programmation et assure ainsi l'intégration des requêtes et réponses au sein d'environnements hétérogènes.

> *Le langage XML [CHA 01] aurait pu être utilisé. Cependant, il souffre de deux défauts majeurs : d'une part, la syntaxe XML décrivant la signature électronique [EAS 02] a peu d'utilité actuellement du fait que l'ensemble des structures de données standards sont décrites en ASN.1 ; d'autre part, l'encodage XML est verbeux comparé aux encodages DER et PER du langage ASN.1 [DUB 00], plus compacts.*

4.2.1. Requête

La requête est mise au point par le client qui indique au serveur la méthode qu'il souhaite appeler (en indiquant éventuellement les paramètres d'appel). Cette requête est encapsulée dans une structure comprenant également la signature électronique du client selon le format ASN.1 suivant :

```
SignedRequest ::= SEQUENCE {
  req           Request,
  signs     [0] AccessSigns           OPTIONAL,
  session   [1] SessionToken          OPTIONAL,
  extns     [2] EXPLICIT Extensions   OPTIONAL
}
```

```
Request ::= SEQUENCE {
  refKey        OCTET STRING          OPTIONAL,
  -- Key mentioned by the server reference
  -- to avoid "old" reference usage by requester
  ip            RelativeIdentifier    OPTIONAL,
  -- Requester's IP address (if required by server)
  nonce         INTEGER               OPTIONAL,
  -- Used to prevent client from server replay attacks
  call          MethodCall
}

MethodCall ::= SEQUENCE {
  name          VisibleString,
  -- Name of the method called
  args          Arguments             OPTIONAL
  -- Arguments values
}

Arguments ::= SEQUENCE OF Argument

Argument ::= SEQUENCE  {
  id            OBJECT IDENTIFIER,
  multiple      BOOLEAN       DEFAULT FALSE,
  -- Simple or array of values
  value     [0] ANY DEFINED BY id     OPTIONAL
  -- Is a sequence of values in case multiple is true
}

AccessSigns ::= CHOICE {
  signatures    ElectronicSigns,
  delegations   DelegationCerts
}
```

Un entier issu d'un tirage aléatoire (ou pseudo-aléatoire) appelé « nonce » (littéralement « entier n unique ») peut être indiqué par l'émetteur de la requête. Cet entier permet de limiter les attaques de type « réplication » (ou « replay »). Celles-ci consistent à enregistrer une information transmise sur le réseau de communication (il s'agit dans le cas présent d'une réponse émanant du serveur contacté) et de la transmettre à nouveau.

La valeur aléatoire que génère le client est utilisée par le serveur lorsqu'il construit sa réponse. Ainsi, le client est assuré, lorsque la réponse est authentifiée, que les informations reçues correspondent effectivement à sa requête.

Un jeton de session, nommé « session », peut être généré par le serveur contacté afin de ne pas procéder systématiquement à la vérification de l'authenticité de la requête et des droits d'accès que le client est en mesure de fournir.

La requête fait enfin appel à un champ « signs » de type « AccessSigns » qui permet d'accéder à un service distant uniquement lorsqu'une conjonction de droits d'accès est fournie au serveur. Il peut s'agir soit de la présence simultanée de plusieurs signataires, soit de la mention d'un ensemble de certificats de délégationde droits d'accès.

4.2.1.1. Présentation du jeton de session

Le jeton de session qui compose la requête signée (champ « session » de la structure « SignedRequest ») permet au serveur émetteur d'éviter de vérifier le statut de validité du certificat d'un objet client (demandeur) lorsqu'il a déjà procédé à cette vérification auparavant.

Son utilisation est particulièrement intéressante lorsqu'un client effectue un grand nombre de traitements auprès du même serveur et que ce dernier procède à la vérification en ligne (via un service de type OCSP par exemple) du statut du certificat. Le recours au certificat de session permet, au cours d'un intervalle de temps déterminé (période de validité), de s'affranchir de la validation du certificat.

Voici un exemple de format de jeton de session, décrit selon la syntaxe ASN.1, intégrant non seulement le statut des différents signataires de la requête (et en particulier le certificat du demandeur) mais également la liste des méthodes auxquelles ces derniers ont accès du fait de l'utilisation conjointe de leurs droits d'accès :

```
SessionToken ::= SEQUENCE {
  data          SessionInfo,
  sign          ElectronicSign
  -- Server's signature over "data"
}

SessionInfo ::= SEQUENCE {
  nonce         INTEGER,
  certs         CertificatesOrIds,
  -- Requester's certificates or identifiers
  status        INTEGER,
  methodIds [0] MethodIdentifiers    OPTIONAL,
  validity  [1] EXPLICIT Validity    OPTIONAL
}

CertificatesOrIds ::= SEQUENCE OF CertificateOrId

CertificateOrId ::= CHOICE {
  cert          Certificate,
  certId        INTEGER
  -- Identifier of the user's certificate
}

MethodIdentifiers ::= SEQUENCE OF MethodIdentifier
```

```
MethodIdentifier ::= CHOICE {
  methodIndex   INTEGER,
  method        Method
}

Method ::= SEQUENCE {
  name          VisibleString,
  argTypes      ArgumentTypes        OPTIONAL
}

ArgumentTypes ::= SEQUENCE OF ArgumentType

ArgumentType := ANY
```

Le jeton de session est fourni par le serveur contacté. Ainsi, lors du premier échange, le serveur peut choisir de retourner un jeton. Ce dernier sera alors joint aux prochaines requêtes du client afin de simplifier la vérification de son certificat et de ses droits d'accès. Ainsi, le serveur vérifie sa propre signature apposée sur le jeton et valide l'accès lorsque :

- Le jeton est valide : le serveur vérifie d'une part sa propre signature afin d'attester de l'authenticité du jeton de session et d'autre part que le jeton est en cours de validité via l'information portée par le champ « validity ».
- Le certificat du client (ou éventuellement son identifiant, bien que cette information apporte moins de garanties car un fraudeur peut générer un certificat portant le même identifiant) correspond au certificat authentifié par le serveur au sein du jeton.
- La valeur de statut indique que ledit certificat est valide.
- La méthode souhaitée est exécutable par ce client qui dispose des droits d'accès requis.

Au-delà de la période mentionnée par le champ « validity », le jeton expire et le serveur doit procéder à nouveau à l'authentification du client et éventuellement générer un autre jeton.

Le format proposé offre la possibilité d'inscrire la liste des méthodes accessibles via le champ « methodIds ». Dans ce cas, le serveur peut s'affranchir de contrôler l'accès à l'aide de son ACL. Cependant, le jeton de session peut omettre d'indiquer les méthodes accessibles (de par l'optionalité du champ « methodIds »). Dans ce cas, le serveur doit consulter son ACL afin de déterminer l'accès à la méthode demandée.

La sécurité du jeton de session est garantie à la fois par la signature de l'objet serveur émetteur ainsi que par le fait qu'il est attaché à un client donné par le biais

de son certificat (« cert ») et d'une valeur entière pseudo aléatoire (« nonce »). De fait, chaque jeton est authentique et spécifique au client.

Sa réémission suite à expiration permet de maintenir les informations de statut suffisamment à jour afin que le serveur dispose de données fiables, tant au niveau du statut de validité du certificat du demandeur que de ses droits d'accès.

> *Note : le format du jeton de session proposé est donné à titre indicatif et peut ultérieurement être soumis à modifications lors d'une étude plus approfondie des protocoles liés à son utilisation.*

4.2.1.2. Délégation de droits et certificat de délégation

Nous retrouvons le format général mentionné lors de l'étude de la signature électronique. La possibilité offerte par la signature conjointe de plusieurs demandeurs permet d'accéder à des services requérant la conjonction des droits d'accès associés aux signataires.

[NAG 98] distingue deux formes de délégation faisant intervenir un client, un objet intermédiaire et l'objet cible : la délégation simple et la délégation en cascade [SOL 88].

La délégation simple utilise uniquement les droits d'accès de l'objet intermédiaire dès lors que celui-ci accepte de traiter la demande du client. L'objet cible ne reçoit alors que des informations relatives à l'objet intermédiaire.

La délégation en cascade utilise la conjonction des droits d'accès du client et de l'objet intermédiaire pour que l'objet cible autorise l'exécution de la requête formulée par l'objet intermédiaire suite à la demande initiale émanant du client.

Il apparaît au regard de ces définitions que la délégation simple est un cas particulier de délégation en cascade. La délégation en cascade peut être présentée aux différents objets de type mixte jusqu'à l'objet serveur final à l'aide d'un certificat de délégation pouvant prendre la forme suivante (format indicatif de délégation en cascade, sujet à modification suite à une étude plus poussée) :

```
DelegationCerts ::= SEQUENCE OF DelegationCert

DelegationCert ::= SEQUENCE {
  delegationInfo   DelegationCertInfo,
  issuerSignValue  ElectronicSignature
  -- Signature of the delegator over "delegationInfo"
}

DelegationCertInfo ::= SEQUENCE {
```

```
  nonce             INTEGER,
  receiverCert      Certificate,
  -- Must correspond to the requester's certificate
  delegated     [0] DelegationCerts      OPTIONAL,
  -- Recursive definition to allow cascading delegations
  path          [1] DelegationPath       OPTIONAL,
  validity      [2] Validity,
  credentials       Credentials
}

DelegationPath ::= SEQUENCE {
  depth             INTEGER              OPTIONAL,
  auths             Certificates         OPTIONAL
}

Credentials ::= SEQUENCE (1..MAX) OF Credential

Credential ::= SEQUENCE {
  value             OBJET IDENTIFIER,
  -- Identifier of the delegated credential
  extns         [0] EXPLICIT Extensions  OPTIONAL
}

Certificates ::= SEQUENCE OF Certificate
```

Les certificats de délégation (« DelegationCerts ») peuvent être joints aux champs étendus authentifiés d'une signature électronique (champ « extns » au sein de la structure « SignerInfo ») particulière de la requête signée. Nous avons délibérément choisi de ne pas construire un champ spécifique du fait que la délégation des droits d'accès est spécifique aux systèmes d'information alors que le format de signature électronique proposé est conçu dans un cadre plus général.

La liste des droits délégués est ici représentée par une suite d'identifiants. Afin d'être évolutive et de fait ouverte à de futurs formats, nous y avons ajouté des extensions optionnelles. Les droits mentionnés doivent être soit en possession directe du délégateur, soit nouvellement acquis et indiqués dans la liste des certificats délégués « delegated ».

Chaque certificat de délégation dispose d'une durée de vie bornée par l'intervalle « validity ». Dans le cas où le certificat de délégation utilise une délégation en chaîne (le champ « delegated » est non vide), la durée de validité de ce dernier devra être comprise dans l'intervalle formé par la conjonction des durées de validité des certificats préalablement délégués. Il peut ainsi arriver que la délégation soit impossible du fait de la non intersection des divers intervalles de validité.

Bien que le certificat de délégation dispose d'une période de validité, il est parfois nécessaire de limiter la délégation à une seule utilisation en empêchant que l'objet délégué transmette ses nouveaux droits à des objets tiers. Pour cela, l'emploi d'un

chemin de délégation (« path ») fixant le nombre maximum de délégations en chaîne est recommandé. Un protocole de confirmation de délégation peut enfin être mis en place entre un objet récepteur d'une requête signée dont les droits d'accès sont délégués et l'objet source de la délégation (délégateur) afin qu'il confirme celle-ci.

4.2.2. Réponse

La réponse est générée par l'objet serveur contacté. Elle contient soit le résultat attendu par le client, soit un statut d'erreur. De manière semblable à la construction d'une requête, la réponse est englobée dans un format de données contenant la signature électronique du serveur en sus de la valeur de retour :

```
SignedResponse ::= SEQUENCE {
  resp        Response,
  signs   [0] ElectronicSigns       OPTIONAL,
  session [1] SessionToken          OPTIONAL,
  extns   [2] EXPLICIT Extensions   OPTIONAL
}

Response ::= SEQUENCE {
  nonce       INTEGER               OPTIONAL,
  -- If present, must correspond to requester's nonce
  status      ResponseStatus,
  returnVal   ReturnedValue         OPTIONAL
}

ReturnedValue ::= Argument
```

Le statut de la réponse indique si l'exécution de la méthode souhaitée a été réalisée ou si une erreur est survenue. Cette erreur peut provenir aussi bien de l'authentification du demandeur que de la génération d'une erreur d'exécution.

4.2.2.1. Statuts de réponse

L'ensemble des statuts du champ « status » au sein de la réponse renvoyée au demandeur est décrit par une valeur entière parmi les suivantes :

```
ResponseStatus ::= INTEGER {
-- To be possibly modified with new status values
  granted                           (0),
  grantedWithMods                   (1),
  untrustedCertificate              (2),
  denied                            (3),
  waiting                           (4),
  revocationWarning                 (5),
  revocationNotification            (6),
  invalidRequestFormat              (7),
```

```
  unsupportedCertificateFormat      (8),
  unknownMethod                     (9),
  invalidMethod                     (10),
  methodExecutionError              (11),
  invalidNumberOfArguments          (12),
  invalidArgumentType               (13),
  invalidArgumentValue              (14),
  connectionError                   (15),
  preInterceptionError              (16),
  postInterceptionError             (17),
  unavailable                       (18),
  invalidSessionToken               (19),
  sessionExpirationError            (20),
  unsupportedDelegationCertificate  (21),
  unsupportedDelegationPath         (22),
  delegationExpirationError         (23),
  invalidDelegatedCredentials       (24),
  invalidServerKey                  (25),
  migrating                         (26),
  locked                            (27)
}
```

4.2.2.2. Discussion

Le statut « granted » indique que la requête du client a été correctement traitée par le serveur. Sa réponse est alors jointe par le biais du champ « returnVal ». Ce dernier est déclaré comme optionnel : il n'est en effet pas renseigné en cas d'errur d'authentification ou d'exécution, lorsque le serveur ne souhaite pas retourner de valeur. Le cas particulier d'une méthode ne retournant aucun résultat (« procédure ») implique que le serveur retourne au client une valeur nulle qui indique que la méthode a bien été exécutée. Le champ « returnVal » est donc présent dans ce cas.

> *Note : le statut de retour « grantedWithMods » défini par le protocole TSP [ADA 01] signifie que le serveur a effectivement honoré la requête du client. Cependant, les informations nécessaires à la stricte application de la politique de sécurité n'ont pu être collectées. Il peut par exemple s'agir d'une requête que le serveur est dans l'impossibilité d'horodater pour cause d'erreur de communication avec l'autorité d'horodatage concernée. Dans ce cas, le serveur peut renvoyer une réponse partielle au client qui acceptera ou non une réponse non horodatée.*

Le champ « nonce » de la réponse permet au client de s'assurer que la réponse retournée est actuelle et correspond effectivement à sa requête. Il lui suffit pour cela de comparer la valeur initiale de ce champ inclus dans la requête à la valeur retournée par le serveur. Ces deux valeurs doivent êre identiques. Le recours à un tirage aléatoire peut être complété par des horodatages lorsque requête et réponse sont authentifiées afin de lutter plus efficacement contre les attaques par réplication.

Les divers statuts de réponse intègrent *in extenso* les valeurs « migrating » et « locked » utilisés au cours du processus de migration des objets. Lorsqu'un objet *O* a reçu un ordre de déplacement d'une machine M_1 (source) vers une machine M_2 (destination) :

- Le statut « migrating » est retourné par *O* initial (sur la machine source) afin d'indiquer à l'objet client appelant qu'il a entamé sa migration. Sa nouvelle référence peut être jointe au cas où *O* initial attende la fin de sa migration vers M_2 pour retourner sa réponse au client. Ce dernier devra alors réitérer sa demande auprès de *O* final (sur la machine destination).

- Le statut « locked » est renvoyé par *O* final sur la machine destination tant que le processus de migration n'est pas achevé et que *O* initial n'a pas été détruit.

Ce principe de migration est un exemple d'illustration de l'intérêt des statuts de réponse « migrating » et « locked » au sein du modèle.

Bien que le format de la réponse intègre la notion de signature multiple, la seule présence de la signature de l'objet serveur (ou mixte) contacté sera en règle générale suffisante.

L'horodatage des échanges n'est actuellement pas pris directement en charge par ce modèle, par contre le format de signature électronique présenté auparavant intègre cette notion. L'intérêt d'horodater un échange non sécurisé étant limité, nous pouvons nous reposer sur le format de la signature électronique proposé.

Enfin, il apparaît qu'aucune solution permettant de limiter l'usage de clés compromises n'est proposée. C'est pourquoi nous allons maintenant élaborer des moyens techniques permettant de s'assurer de la non compromission des clés (de signature notamment) utilisées.

5. Garanties de non compromission de clé

Bien que les clés utilisées pour générer des signatures électroniques soient supposées inviolables, personne n'est à l'abri d'une erreur humaine ayant pour effet une possible compromission de clé par une tierce partie malveillante. Dans ce cas, le soupçon de compromission aboutit à la révocation du certificat et par conséquent au possible rejet des informations signées.

Il est cependant possible que le détenteur d'un certificat de signature n'ait pas conscience de la compromission de sa clé de signature. Il continue alors à faire usage de cette clé sans se douter qu'une tierce personne est également en mesure de créer des signatures authentiques sous son identité.

Ce problème se retrouve de fait dans les échanges entre les objets qui composent un système réparti. C'est pourquoi nous suggérons un protocole d'ordonnancement des signatures faisant appel à une autorité de confiance dédiée ou faisant partie des PSCE. Celle-ci peut être considérée comme l'extension du notaire électronique décrit par [ANS 01].

5.1. Principe de fonctionnement du protocole

Le protocole proposé, baptisé Notarisation Avancée (« Advanced Notarization »), agit de manière préventive en empêchant quiconque ne dispose pas d'une information secrète de valider l'apposition de sa signature électronique. En contrepartie, son utilisation nécessite une connexion sécurisée au notaire électronique supportant ce protocole.

La notarisation avancée repose sur l'utilisation d'une valeur initiale secrète (ou IV, « Initial Value ») que le client doit fournir afin d'autoriser la validation de sa signature par le notaire contacté. Un indice d'ordonnancement des signatures peut également être employé afin de renforcer la sécurité de la demande. Le protocole est conçu de telle sorte que la valeur secrète ne soit divulguée directement : les informations fournies par le client sont combinées à des éléments aléatoires propres à la transaction, renouvelés à chaque nouvelle demande de signature. De fait, seuls le PSCE, le notaire électronique et le titulaire du support physique de la bi-clé de signature sont en mesure de connaître cette valeur initiale secrète. Ainsi, un attaquant ne peut en prendre connaissance qu'en ayant recours à un détournement physique du support matériel de la signature électronique du client (ou une intrusion au sein du système informatique du PSCE ou du notaire électronique).

De plus, le protocole fait appel à des moyens « externes » aux échanges transactionnels entre le notaire et son client. Le notaire peut envoyer un accusé de validation au signataire par courriel ou SMS en fonction des informations de contact mentionnées dans le certificat du client ou indiquées lors de l'enregistrement du client. Ce dernier dispose alors d'un délai de rétractation pendant lequel il peut faire

opposition à sa signature, dépendant des attributs joints à la signature. Passé ce délai, la signature est considérée comme ayant un possible effet juridique.

5.2. Déroulement du protocole

Le protocole se décline en deux variantes selon que le support de la signature est réinscriptible (certaines cartes à puce ou le e-button par exemple) ou non (la cd-carte par exemple). L'inconvénient d'un support non réinscriptible est que l'information nécessaire à la validation de la signature ne peut évoluer au cours du temps.

Dans les deux cas, le protocole demande une phase préliminaire au cours de laquelle l'utilisateur titulaire de la bi-clé de signature reçoit une valeur initiale secrète permettant de demander la validation de ses propres signatures. Cette valeur initiale est fournie par le PSCE et ajoutée aux informations de signature lorsque le support physique de la bi-clé le permet.

5.2.1. Avec support de signature non réinscriptible

La valeur IV est fixe. Elle s'apparente dès lors à un numéro de série, propre au support physique de la signature. Bien qu'il puisse s'agir du numéro de série du certificat, nous rejetons cette solution car cette information doit demeurer secrète.

5.2.1.1. Modélisation

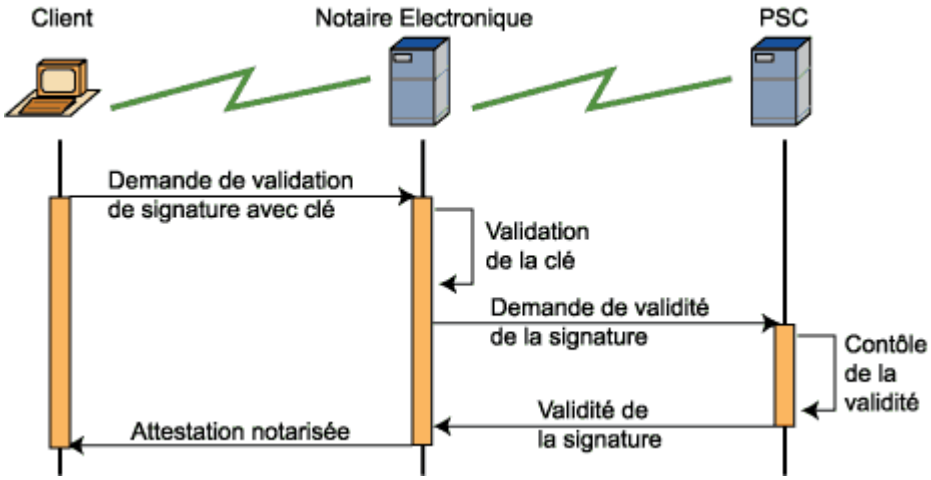

Figure 32 : protocole de notarisation avancée, première version

Ce protocole de notarisation avancée avec support non réinscriptible illustré par la figure 32 se déroule comme suit :

1. Le protocole débute par l'établissement d'une session sécurisée entre le client et son notaire électronique. Cette première phase permet la transmission au client d'un identifiant de transaction ID par le notaire. Cet identifiant peut prendre la forme d'une valeur numérique aléatoire.

2. Le client effectue sa demande. Pour ce faire, il transmet au notaire sa signature électronique ainsi que l'empreinte numérique $hash_p(\{IV, ID\})$. L'emploi de l'empreinte numérique permet de ne pas divulguer l'information secrète IV. Le fait d'y ajouter l'identifiant de la transaction évite la possibilité d'effectuer une attaque par réplication (« replay ») [GON 90] puisque cet identifiant est spécifique à la transaction.

3. Le notaire vérifie que l'empreinte $hash_p(IV, d, i)$ fournie correspond à l'empreinte de la structure constituée de la valeur IV, et de l'identifiant de la transaction ID.

4. Le notaire procède ensuite à l'obtention d'une valeur de validité du certificat du signataire inclus dans sa signature électronique. Cette démarche peut être réalisée localement par le biais d'une liste de certificats révoqués ou à distance via un protocole en ligne auprès du PSCE émetteur du certificat (ou de toute autre autorité accréditée et apte à retourner l'information de validité demandée).

5. Enfin, le notaire appose sa propre signature électronique, horodatée de préférence, à l'attestation notariée NI qu'il remet au demandeur. Cette attestation comprend la valeur de validité certifiée par la signature du notaire et peut prendre la forme d'un certificat de validité [TRU 03]. Il incrémente de plus l'indice de signature afin de conserver la cohérence entre les deux versions du présent protocole.

5.2.1.2. Formalisation du protocole

Ce protocole de notarisation avancée avec support non réinscriptible se déroule comme suit : peut être formalisé à l'aide de la notation utilisée par [ROO 99], étant supposé que le client dispose de la valeur ID fournie par le notaire lors de l'établissement de la transaction sécurisée :

1. $C \xrightarrow{1} N : Sign_K(\{sig, hash_p(\{IV, ID\})\})$ où sig représente la signature électronique devant être notariée.

2. N vérifie l'empreinte numérique $hash_p(IV, ID)$.

3. N obtient une valeur de validité du certificat du signataire, localement ou par une interrogation en ligne :

$\begin{cases} N \xrightarrow{2} CRL : id_{user} \\ N \xrightarrow{2} PSC : id_{user} \end{cases}$ où id_{user} fait référence à l'identifiant du certificat à valider.

4. N appose sa propre signature, horodatée de préférence, à l'attestation notariée NI qu'il remet au demandeur :

$$N \xrightarrow{3} C : \{\{NI\}, Sign_{K_{not}}(\{sig, NI, ID\})\}.$$

5.2.2. Avec support de signature réinscriptible

La valeur IV évolue au cours des demandes de validation. Par exemple, cette valeur peut être déduite de la valeur courante et de l'identifiant ID de la transaction fourni par le notaire selon la procédure suivante :

$$\begin{cases} IV_O = IV \\ IV_{i+1} = hash_p(\{IV_i, ID\}) \end{cases}.$$

Chacun peut ainsi calculer cette nouvelle valeur sans avoir à la transmettre sur le réseau.

5.2.2.1. Modélisation

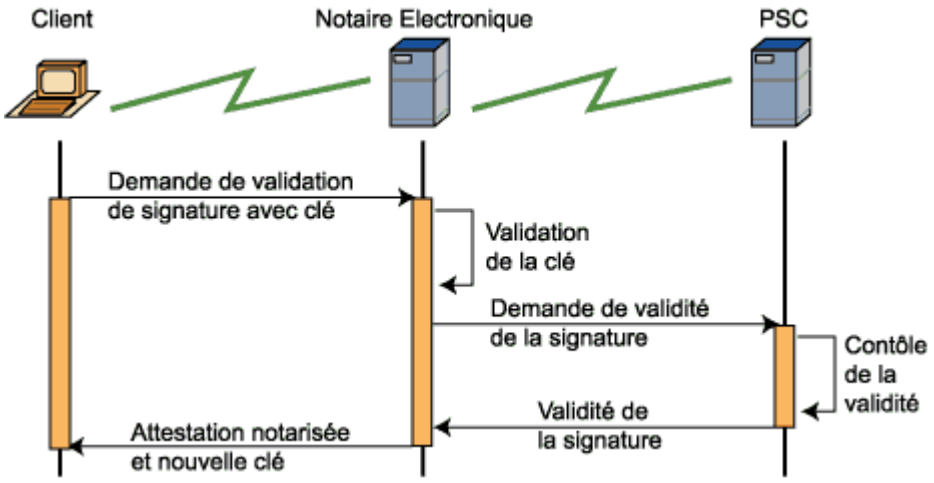

Figure 33 : protocole de notarisation avancée, seconde version

Cette seconde version du protocole, décrite par la figure 33, se déroule comme suit :

1. De même que précédemment, le protocole débute par l'établissement d'une session sécurisée entre le client et son notaire électronique. Cette première phase permet la transmission au client d'un identifiant de transaction ID par le notaire. Cet identifiant peut prendre la forme d'une valeur numérique aléatoire issue d'un PRNG.

2. L'utilisateur demande la validation de sa signature. Il transmet à son notaire sa signature électronique ainsi que l'empreinte numérique $hash_p(\{IV, ID\})$ calculée à partir de la valeur IV courante. L'empreinte est validée par le notaire en fonction de son historique et de la dernière empreinte communiquée.

3. L'empreinte étant acceptée, le notaire vérifie la validité du certificat mentionné dans la signature du client, à l'aide d'une CRL locale ou d'un protocole en ligne.

4. Le statut de validité du certificat (et par conséquent de la signature) est enfin retourné à l'utilisateur, sous la forme d'une attestation de validité notariée NI (certificat de validité par exemple). Cette dernière peut être horodatée afin d'acquérir une valeur probante au cours du temps. Une nouvelle valeur secrète est alors construite. Elle sera utilisée par le client lors de sa prochaine demande.

> *Remarque de sécurité : la clé utilisée est formée par l'empreinte numérique résultant de l'indice de la signature précédente (le nombre de signatures déjà validées) ainsi que d'une valeur aléatoire. Un intrus ne peut donc deviner la valeur de la prochaine clé lorsqu'une valeur est connue.*

Enfin, le vérificateur du statut de validité de la signature doit se connecter au notaire électronique afin que ce dernier apporte la confirmation de la validité de la demande. Le notaire peut alors retourner un statut d'erreur dans le cas où l'utilisateur a révoqué son certificat et annulé la dernière demande (au cas où sa signature a été usurpée).

5.2.2.2. Formalisation du protocole

Cette version du protocole est identique à la version précédente. Cependant, une nouvelle valeur initiale est construite à l'issue de chaque demande : $IV_{n+1} = f(IV_n, ID)$, f pouvant par exemple être une fonction de calcul d'empreinte numérique.

5.3. Gestion de signatures notariées

5.3.1. Révocation

Lorsque l'utilisateur soupçonne la compromission de sa clé de signature, il doit demander la révocation de sa bi-clé auprès du PSCE émetteur de son certificat. Le protocole proposé permet de plus de révoquer les dernières signatures apposées dont le délai de carence n'est pas dépassé.

Cette procédure est mise en application par le signataire légitime lorsque :

- Il reçoit un accusé de validation de la part d'un notaire électronique alors qu'il n'a demandé aucune signature.
- Il demande à valider sa signature et le notaire électronique refuse la validation pour motif de duplication de la demande (même indice de signature, même clé de validation, mais signature différente).
- Il fait usage de son droit de rétractation, en fonction du type de document signé.

La révocation ne peut cependant intervenir dès lors que le délai légal, défini contractuellement avec le notaire électronique, est dépassé.

5.3.2. Validation

Le vérificateur soumis à une signature électronique notariée à l'aide du protocole proposé doit non seulement procéder à la vérification de l'authenticité de la signature mais également à la validité des informations relatives à la validité de la signature.

Deux cas peuvent se produire, sachant que le notaire indique que la signature est valide :

- La signature n'a été révoquée : dans ce cas, elle est considérée comme valide. Le vérificateur peut alors établir son statut de validité à l'aide des informations fournies par le notaire électronique ou par le biais d'une validation ad hoc (à l'aide d'une liste de certificats révoqués ou d'une connexion en ligne par exemple).
- La signature a été révoquée : le vérificateur peut la rejeter (ou l'accepter !).

Pour établir la preuve de révocation de la signature, le vérificateur doit demander confirmation au notaire électronique ou utiliser les informations de révocations éventuellement inclues dans la signature à l'aide du champ « revocationInfo ».

CHAPITRE 4 : EXEMPLE D'UTILISATION DU MODELE

« Je dis des choses tellement intelligentes que, le plus souvent, je ne comprends pas ce que je dis »

Les devises Shadok

Après avoir étudié un modèle intégrant les concepts de système réparti, cryptage, signature électronique, surveillance applicative et audit, nous proposons de montrer l'intérêt d'un tel outil pour déployer une architecture répartie homogène couvrant l'ensemble des services relatifs à la gestion de la signature électronique. L'ensemble des services permettent de rendre possible la création de documents numériques (numérisés ou dématérialisés) signés électroniquement et de leur conférer le même poids juridique que leurs équivalents sous forme papier.

Bien que l'ensemble des autorités impliquées dans la mise en œuvre du modèle sont présentées, seules les interfaces S-IDL du PSCE seront proposées en annexe. Une modélisation sous forme de cas d'utilisation exprimée en langage UML est néanmoins fournie afin de faciliter la déclaration des interfaces S-IDL de l'ensemble des entités du système réparti évoqué dans ce chapitre.

1. Présentation de la Chaîne de Confiance

1.1. Aperçu général

L'architecture répartie présentée par la figure 34 repose sur EDCI, « Electronic Data Certification Infrastructure », proposée et partiellement mise en œuvre par [TRU 03]. Elle répond à l'ensemble des besoins en matière d'utilisation de la signature électronique. Cette architecture générale fait intervenir un ensemble de tiers de confiance [IAL 98] [MEN 01] chargés de répondre à des besoins spécifiques liés à l'utilisation et la démocratisation de la signature électronique auprès du grand public.

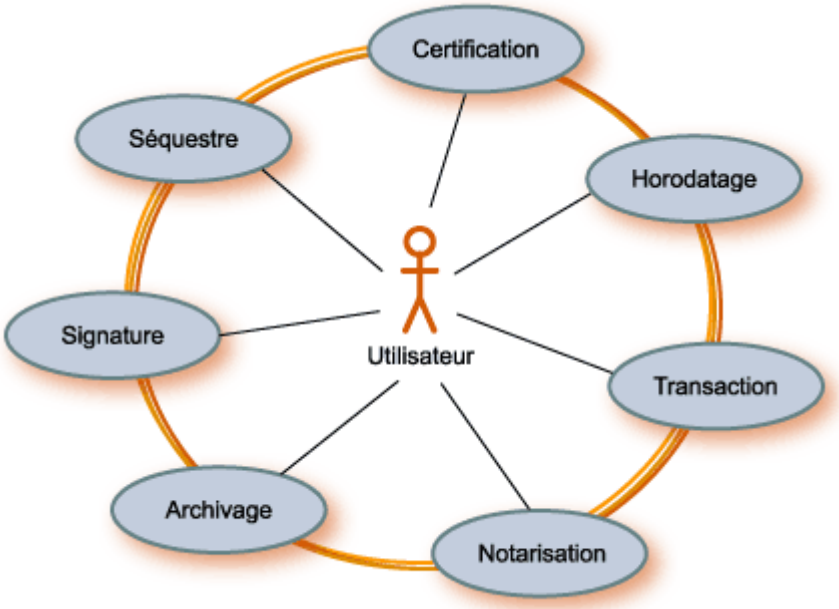

Figure 34 : acteurs de la Chaîne de Confiance

L'utilisation conjointe des services proposés par ces tiers définit une « Chaîne de Confiance » (CdC), depuis la génération des certificats par un PSCE jusqu'à la conservation sécurisée sur le long terme des documents ayant produit leurs effets juridiques (inactifs). Ces tiers de confiance sont reconnus par les instances juridiques comme dignes de confiance. Il serait souhaitable qu'un organisme ayant reçu un soutien gouvernemental soit chargé de veiller au respect des procédures et normes standard, à l'image de Agence Nationale de Certification Electronique (ANCE) tunisienne. Cet organisme pourrait délivrer des labels qui attestent du sérieux des tiers concernés dorénavant autorisés à exercer leur métier légalement.

1.2. Définition de la Chaîne de Confiance

La figure 35 présente un exemple organisation de la Chaîne de Confiance. Celle-ci désigne les interactions entre les tiers de confiance qui apportent, ensemble, une sécurité optimale permettant la constitution de l'écrit sous forme numérique ayant une valeur juridique car signé électroniquement conformément aux recommandations locales du pays concerné ou dépendantes des réglementations mises en place par la communauté européenne, sa transmission par le biais un canal non sécurisé, sa validation, son archivage sur le long terme et enfin sa restitution.

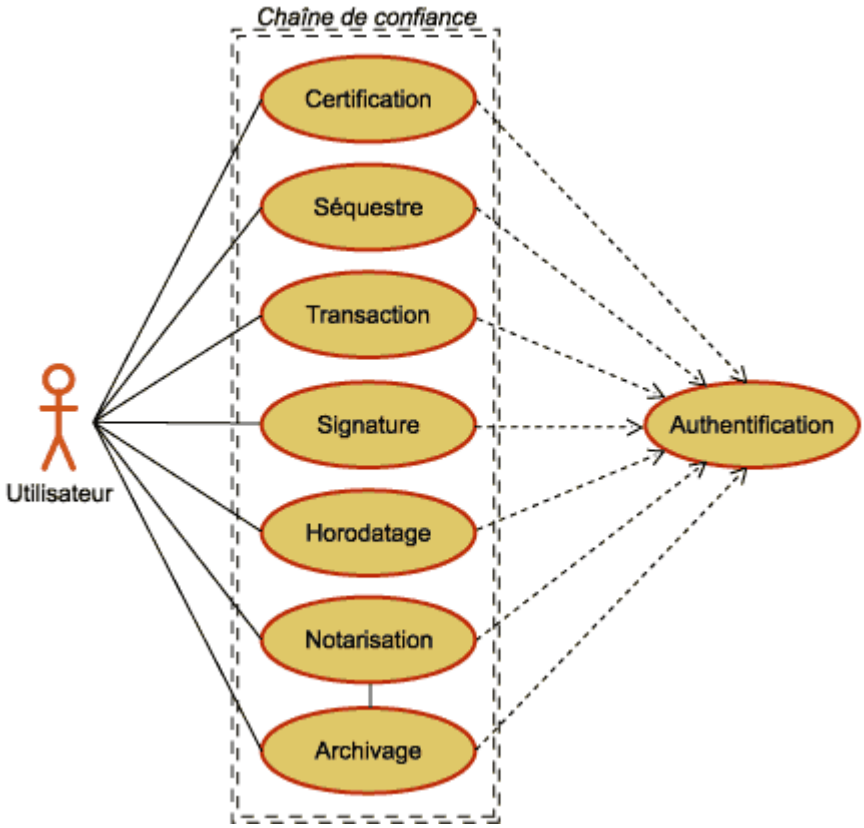

Figure 35 : organisation de la Chaîne de Confiance

Note : une confusion courante consiste à réduire la Chaîne de Confiance à un chemin de certification fourni par une autorité de certification.

2. Etude des autorités de la Chaîne de Confiance

Cette Chaîne de Confiance fait appel aux autorités suivantes :

- Une autorité de certification (employé ici en lieu et place de PSCE).
- Une autorité de séquestre.
- Une autorité de signature.
- Une autorité d'horodatage.
- Une autorité de transaction.
- Un notaire électronique, en relation avec une autorité d'archivage.
- Une autorité d'archivage (ou PSA, « Prestataire de Services d'Archivage » pour le compte d'un notaire électronique).

2.1. Autorité de certification

Le PSCE met en œuvre une Infrastructure à Clé Publique (ICP ou PKI) [ADA 99] [KAE 00] [AUT 02]. Cette architecture pyramidale, évoquée par la figure 36, distribue la confiance par le biais de certificats numériques dont il certifie l'intégrité et l'exactitude en leur apposant sa signature électronique.

[TRU 03] propose une extension à l'architecture standard [ADA 99] : l'ICP est dirigée par une autorité racine de certification (« AC racine ») chargée de créer les certificats numériques. Les informations contenues dans les certificats générés sont confiées à l'autorité racine par une autorité principale d'enregistrement (« AE principale »), elle-même en relation avec des autorités locales d'enregistrement (« ALE »).

La mise à disposition des informations de statut des certificats (listes rouges telles que CRL et OCSP) et d'annuaire de consultation (listes blanches de type LDAP [BOE 99]) est assurée par l'autorité principale d'enregistrement à l'aide des données fournies par l'autorité racine de certification.

Les certificats et bi-clés générés par l'autorité de certification principale (« AC principale ») sont distribués aux clients sur un support physique de type cd-carte et protégés par mot de passe, conformément au brevet [OEB 02a].

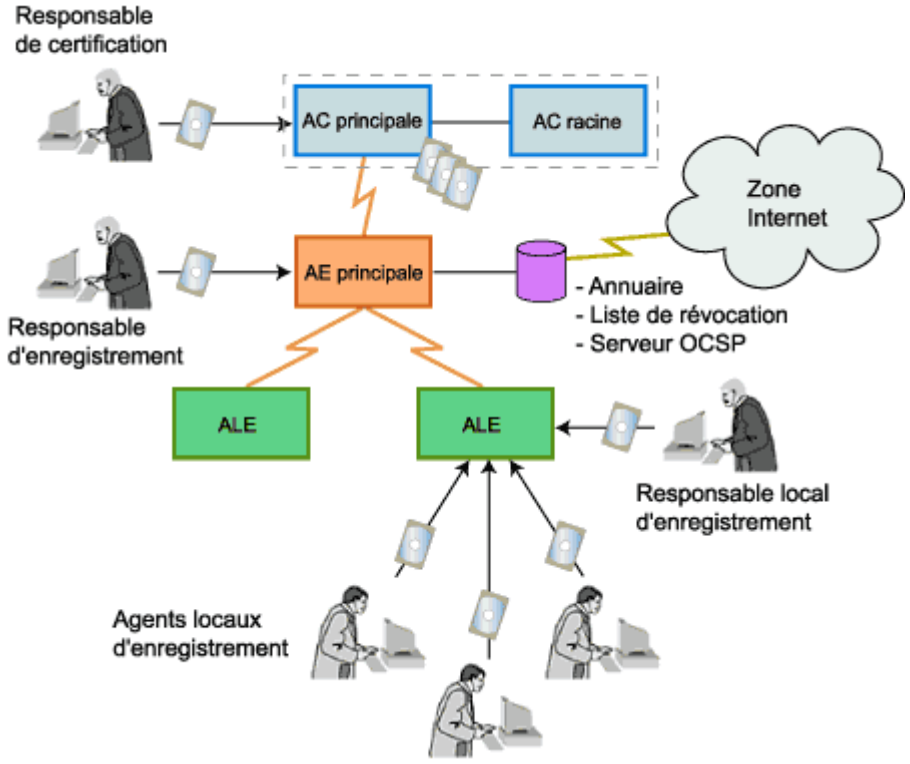

Figure 36 : architecture étendue d'une ICP

2.1.1. Services proposés

Les services dispensés par le PSCE sont les suivants [KAE 00] :

- L'enregistrement des informations relatives au demandeur d'un certificat numérique.
- La création d'une bi-clé pour le compte du demandeur : la bi-clé peut être générée localement par le demandeur ou au sein de l'ICP.
- La certification, opération de création d'un certificat numérique à partir des renseignements fournis par le demandeur aux autorités d'enregistrement locales.
- L'activation, suspension, révocation ou ré-émission d'un certificat préalablement créé et dont le PSCE est racine.
- L'accès aux informations publiques concernant l'identité d'un signataire.
- Le contrôle de la validité d'une signature à un instant donné par le biais d'un annuaire, d'une liste de révocation ou d'un service en ligne (de type OCSP).

- La co-certification, ou certification croisée, permet à différentes autorités de certification de s'attribuer une confiance mutuelle (une reconnaissance unidirectionnelle est également envisageable).

- La récupération d'une bi-clé perdue ou endommagée : l'architecture étudiée ici fait appel à une autorité extérieure appelée autorité de séquestre.

Deux services annexes peuvent être intégrés. Il s'agit de la notarisation électronique et de l'horodatage. L'architecture étudiée propose d'associer le premier service à l'autorité de signature et le second à une autorité d'horodatage indépendante du PSCE.

2.1.2. Modélisation d'une autorité locale d'enregistrement

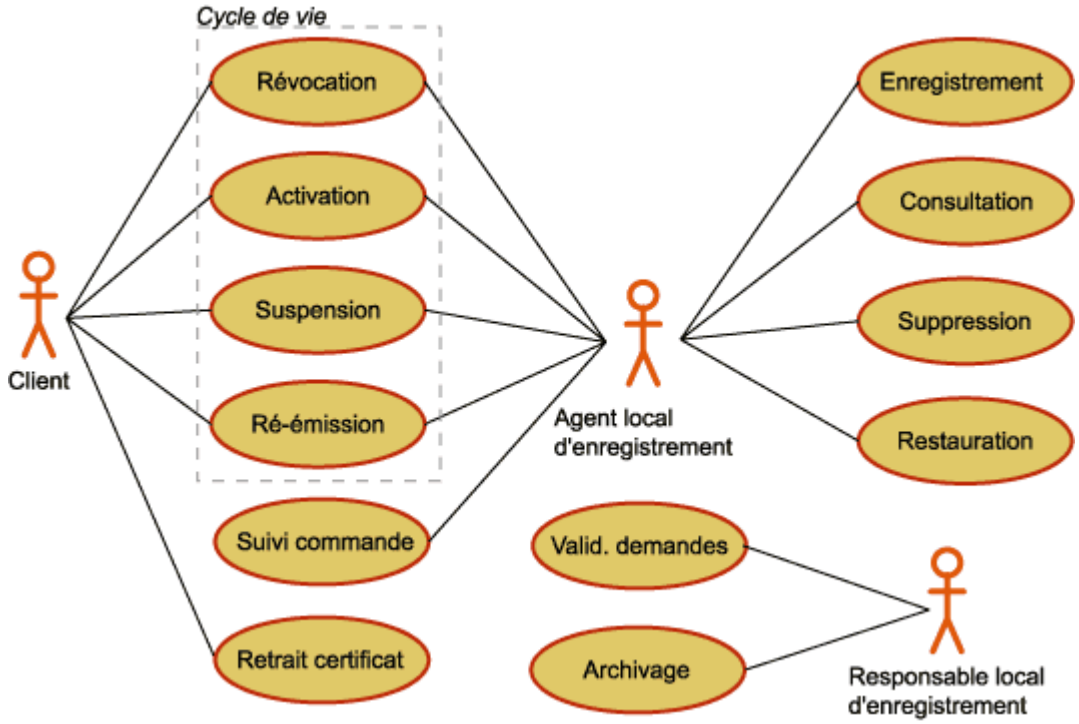

Figure 37 : cas d'utilisation d'une ALE

L'ALE est un guichet en relation directe avec le client. Nous avons délibérément omis dans la figure 37 le lien entre le client et le suivi de la commande car le processus de mise à disposition du support physique de la signature est contrôlé par l'agent local d'enregistrement qui prévient le demandeur lors de la réception de la signature demandée.

2.1.3. Modélisation de l'autorité principale d'enregistrement

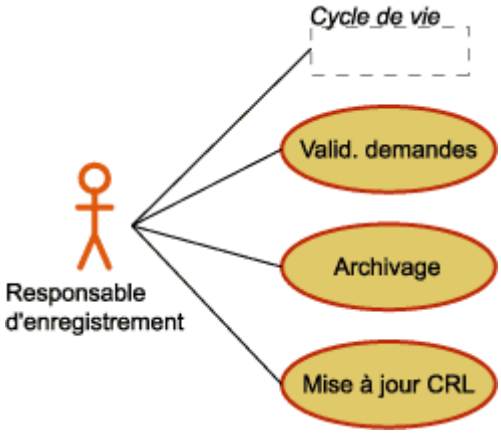

Figure 38 : cas d'utilisation de l'AE principale

Un état supplémentaire n'est pas mentionné dans le schéma de la figure 38 : il s'agit de la mise à disposition du grand public d'un site Internet de promotion de la signature électronique et d'aide à l'utilisation des services proposés par les autorités locales.

2.1.4. Modélisation de l'autorité de certification principale

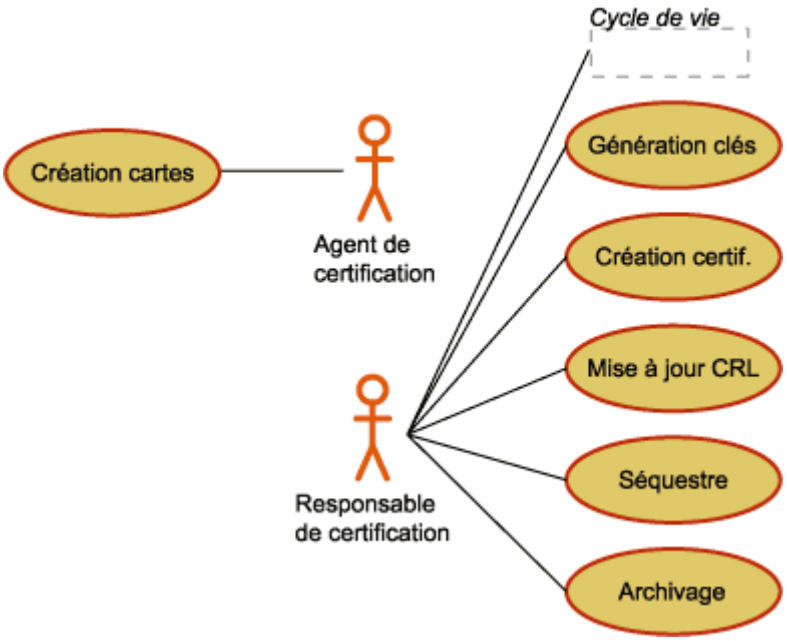

Figure 39 : cas d'utilisation de l'AC principale

Le module de création des cartes qu'indique la figure 39 fait référence à l'apposition des éléments permettant de créer et vérifier les signatures électroniques sur un support physique remis au demandeur.

2.2. Autorité de séquestre

L'autorité de séquestre [AUT 02] est un organisme agréé par le Premier Ministre. Son rôle est de conserver les clés secrètes ou privées des utilisateurs mises en œuvre à des fins de confidentialité (chiffrement) et de les remettre aux autorités judiciaires et de sécurité.

Cette autorité, qui peut être autonome ou faisant partie intégrante d'un PSCE [MCC 96], peut en outre répondre au problème de la perte des clés de chiffrement par les utilisateurs qui ne sont plus en mesure de prendre connaissance des informations confidentielles leur appartenant ou qui leur sont destinées. Cette incapacité peut être due :

- A la perte ou la détérioration du support contenant la clé de déchiffrement.
- A la perte de la phrase secrète permettant d'activer la clé de déchiffrement (en général chiffrée par le biais d'une clé symétrique ou d'un algorithme d'authentification de type 3-DES [KAE 00] et stockée sous un format issu de PKCS#5 [RSA 99] tel que PKCS#8 [RSA 93b]).

La cryptographie actuelle apporte des moyens techniques aptes à garantir la confidentialité absolue des informations durant plusieurs années. Ce dernier doit donc se prémunir contre l'éventuelle impossibilité de prendre connaissance des informations qu'il a lui-même chiffrées (par perte ou détérioration de sa clé de déchiffrement notamment).

L'autorité de séquestre est controversée [FOU 99] car elle permet l'ingérence du gouvernement lorsque sa demande est motivée (en cas de poursuites judiciaires par exemple). Certaines correspondances considérées comme illégales par le Service Central de Sécurité des Systèmes d'Information (SCSSI) ou la Commission Supérieure du Service Public des Postes et Télécommunications (CSSPPT) peuvent motiver le recours à l'autorité de séquestre.

Enfin, les clés de chiffrement peuvent être mises à disposition des entreprises lorsque leurs politiques de sécurité (notamment relatives aux courriels protégés) permettent de prendre de connaissance des messages chiffrés émis par les employés.

Néanmoins, ce recours soulève trois interrogations :

- Les fraudeurs peuvent générer eux-mêmes leurs bi-clés car ils disposent de moyens informatiques le leur permettant. Le problème de l'utilisation de

clés non émises par des autorités de certification ayant reçu l'agrément de l'Etat est donc posé.

- La restitution des bi-clés enregistrées fait appel à une procédure critique d'identification du demandeur ou de l'administration compétente.

- Que la génération des bi-clés s'effectue sur le poste local de l'utilisateur ou au sein de l'autorité de certification, cette dernière doit recevoir un certificat d'enregistrement de chaque bi-clé émise en provenance d'un tiers de séquestre.

Cette réglementation fort complexe devrait être en partie libéralisée au printemps 2004 par l'adoption d'une loi relative à l'économie numérique.

En sus des problèmes juridiques et ceux relevant de la Commission Nationale Informatique et Liberté (CNIL), des contraintes techniques contribuent à alimenter la controverse sur l'existence d'un tel tiers de confiance [ADE 97]. Il existe néanmoins des travaux, tel que le « Key Recovery Demonstration Project » (KRDP) [NIS 96a], chargés de mettre en œuvre une architecture de séquestre.

2.2.1. Services proposés

Les principaux services que propose l'autorité de séquestre des clés sont :

- L'enregistrement des clés de chiffrement (et non de signature) dont la taille dépasse 40 bits. Cet enregistrement est une obligation légale (en France) sauf lorsqu'une dérogation est accordée.

- La restitution (ou recouvrement) des clés à leurs propriétaires légitimes ou aux instances gouvernementales autorisées via un médiateur.

- L'envoi de certificats d'enregistrement aux différents PSCE.

2.2.2. Modélisation

Le modèle UML proposé par la figure 40 repose sur les critères d'évaluation du NIST [NIS 96b]. Les clés de chiffrement sont enregistrées par les clients de l'autorité. Un certificat d'enregistrement prouvant l'enregistrement, signé électroniquement par l'autorité de séquestre, est remis au client à l'issue de la procédure d'enregistrement. Cependant, la restitution d'une clé peut être à l'initiative du client mais également d'un médiateur. Ce dernier officie dans les cas suivants :

- Le client n'est plus en mesure de demander la clé séquestrée (incapacité physique ou intellectuelle).

- Le client refuse de confier sa clé à une administration telle que le SCSSI alors que sa requête est motivée.

Exemple d'utilisation du modèle

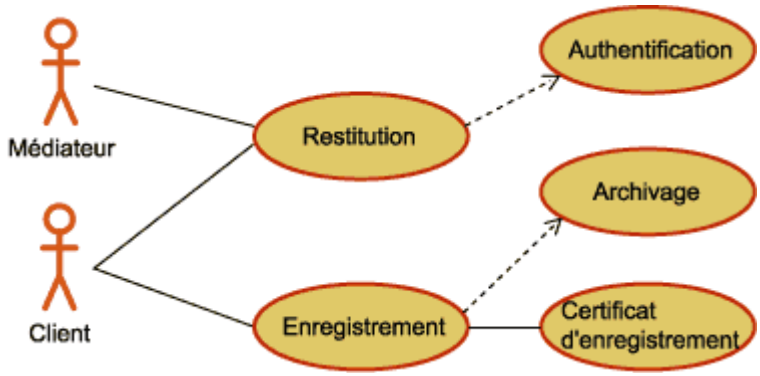

Figure 40 : cas d'utilisation d'une autorité de séquestre

Note : le terme « médiateur » est ici préféré à « mandataire » car il implique une conciliation entre le client réfractaire et l'administration mandatée pour prendre connaissance de sa clé de chiffrement.

2.3. Autorité de signature

L'autorité de signature supervise la multi-signature de documents alors que les parties signataires ne sont pas présentes physiquement. Il peut être considéré comme l'extension d'un tiers d'échanges, tel qu'un opératuer postal, qui sert de relais aux courriers envoyés en recommandé. Dans ce cas, l'autorité de signature propose un service de signature associé à un envoi en recommandé avec accusé de réception à des fins de non répudiation.

2.3.1. Services proposés

Il incombe à ce tiers de confiance d'organiser le processus de multi-signature de document. Il permet de s'assurer que tous les signataires signent effectivement le même document et utilise les services d'une autorité d'archivage sécurisé pour réaliser ses opérations de traçabilité utiles à la preuve de non répudiation des échanges.

Pour ce faire, il supervise :

- La vérification puis l'ajout de chaque signature sur le document. En particulier, la validité des signatures est attestée dès lors que ce tiers de confiance agit également en tant que notaire électronique. La supervision du processus d'ajout des signatures permet de plus de valider le respect de la hiérarchie des signataires selon un plan de multi-signature contrôlée [LGI 02] préalablement établi. Ce plan permet d'associer un niveau de

signature en fonction des attributs des signataires et de rejeter les signataires non référencés.

- Le chiffrement des transmissions destinées aux différents signataires.
- Les procédés de relance lorsque manquent des signatures.
- La constitution du document final en apposant sa propre signature. Cette étape de clôture assure l'impossibilité de modification ultérieure du document ou des signatures.
- La génération et l'archivage sécurisé des traces liées au processus de multi-signature.

Il propose de plus la possibilité d'envoi en recommandé en offrant un espace de récupération des documents. Les émetteurs d'envois en recommandé utilisent cet espace pour soumettre leurs courriers afin que les destinataires puissent les récupérer. L'autorité de signature prend non seulement en charge les restrictions d'accès aux courriers stockés mais également les relances aux destinataires ainsi que les attestations d'envoi et de distribution des courriers.

Ces principes ont fait l'objet d'un dépôt de brevet international [OEB 01a].

2.3.2. Modélisation

La modélisation des cas d'utilisation met en valeur les deux principaux services mis en œuvre par cette autorité (figure 41) :

- La supervision des processus de multi-signature (contrôlée ou non). La multi-signature contrôlée suit les recommandations énoncées par le rédacteur du plan de contrôle.
- L'envoi en recommandé électronique avec stockage temporaire en interne ou chez une autorité d'archivage selon le service demandé par l'initiateur.

Exemple d'utilisation du modèle

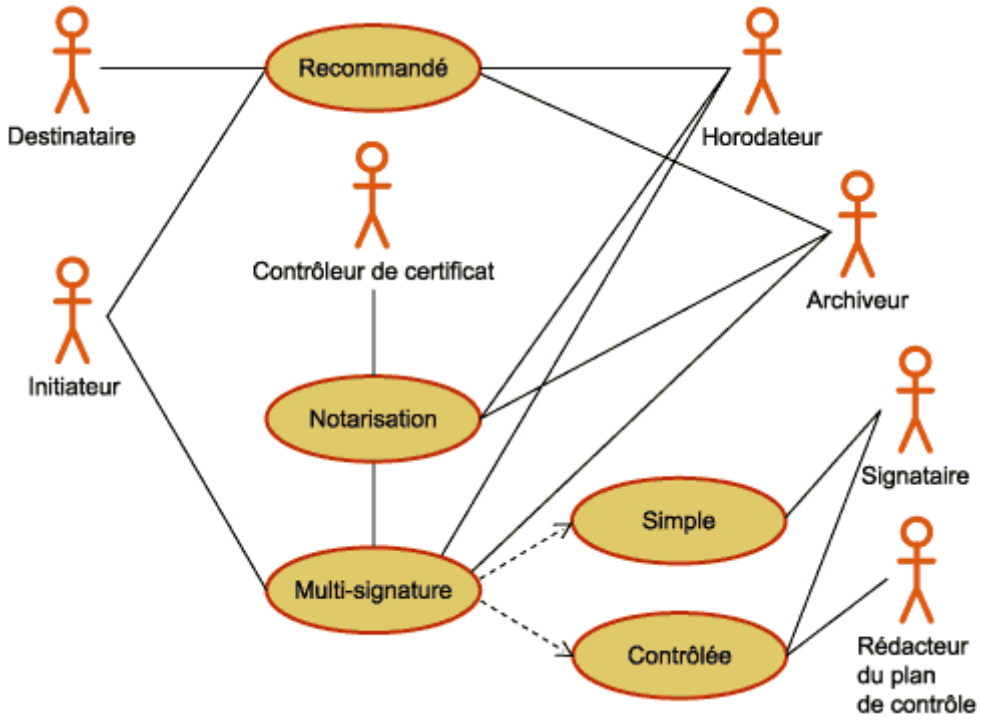

Figure 41 : cas d'utilisation de l'autorité de signature

2.4. Autorité d'horodatage

L'autorité d'horodatage [AUT 02] est une entité interne ou externe au PSCE. Elle garantit une datation certifiée par le biais de sa signature électronique. Cette date certifiée, indissociable de l'information à laquelle elle se rattache, apporte la preuve de l'existence préalable de cette information.

L'horodatage doit satisfaire à deux impératifs [AUT 02] :

- Il doit s'appuyer sur une source de temps sécurisée et reconnue comme fiable [GTH 03]. Il peut par exemple s'agir du système GPS, précis à $1\mu s$, ou du protocole de temps synchronisé « Simple Network Time Protocol » (SNTP) à précision variable [MIL 96].

- Il doit s'associer de manière irrévocable au contenu horodaté (document, transaction, etc.). Cette propriété est réalisée par le biais de la signature électronique de l'autorité d'horodatage permettant de sceller la date sécurisée et la donnée à laquelle elle se réfère.

Dans le cas où l'horodatage est un service complémentaire assuré par un PSCE, la question de l'horodatage des éléments de traçabilité du PSCE liés à la gestion des certificats numériques se pose. Il est en effet envisageable de considérer les traces comme sans effet juridique dès lors que les valeurs d'horodatage sont délivrées en interne. Il est donc souhaitable de considérer l'autorité d'horodatage comme une

entité à part entière, indépendante du PSCE, et non comme un module proposé par ce dernier.

2.4.1. Services proposés

L'autorité d'horodatage propose pour unique service de joindre une date à une empreinte numérique. La combinaison de l'empreinte à horodater et de la date fournie par l'autorité d'horodatage est appelée jeton d'horodatage [ADA 01]. Le jeton est scellé suite à l'apposition de la signature électronique de cette autorité. Lorsque ce jeton est lié à une date revendiquée (« claimed date »), il peut être utilisé par le vérificateur pour valider cette dernière comme date probable.

L'horodatage est un élément essentiel dans la preuve d'authenticité d'un contenu, que ce soit un document à effet juridique, lors d'un échange ou pour permettre de déterminer la validité d'une signature au cours du temps. C'est pourquoi le protocole TSP [ADA 01] fournit un jeton d'horodatage à précision variable : les signatures de documents nécessitent généralement une datation de moindre précision (la date du jour par exemple) que les transactions informatiques et téléservices (faisant usage de la notion de « cachet de La Poste »). Pour éviter toute fraude, un service de publication des valeurs d'horodatage peut être utilisé, conformément aux études citées par [GUE 03] et mentionnées en annexe.

2.4.2. Modélisation

Le module d'accréditation proposé par la figure 42 permet de mettre en œuvre le protocole d'horodatage multiple. L'horodateur est considéré comme l'initiateur de la valeur d'horodatage qui sera par la suite validée par un ou plusieurs horodateurs.

Un module de traçabilité est explicitement mentionné dans le diagramme d'utilisation car celle-ci est primordiale lorsqu'il s'agit d'apporter la preuve a posteriori de la réalité d'un horodatage. Ce module peut s'accompagner d'une publication des valeurs d'horodatages générées selon les protocoles de sécurisation de l'horodatage mis en œuvre, dont certains sont décrits en annexe [GUE 03].

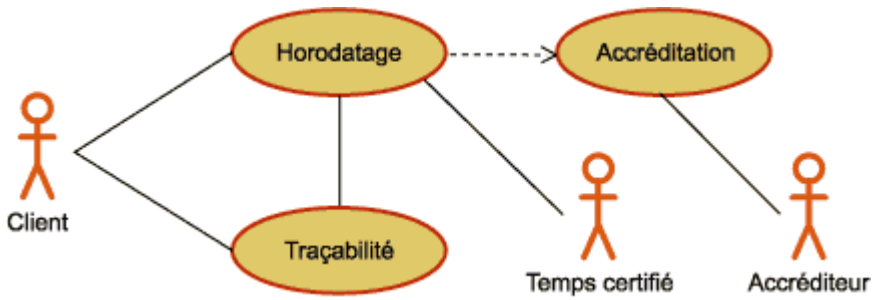

Figure 42 : cas d'utilisation de l'autorité d'horodatage

2.5. Autorité de transaction

Les internautes demeurent réticents lorsqu'il s'agit de réaliser des transactions électroniques sur Internet. En effet, les parties en présence s'accordent sur une transaction alors qu'elles ne se connaissent pas. Ils refusent souvent de mentionner des informations personnelles nécessaires à tout achat en ligne (en particulier, leur numéro de carte bancaire). Le client n'est pas certain que le marchand ne va pas utiliser frauduleusement les informations fournies. En contrepartie, ce dernier n'est pas assuré qu'il s'agit effectivement du client et non d'un individu en possession d'une carte dérobée ou d'une « yes-card » permettant des paiements illicites.

Bien que la législation assimile un achat en ligne sur Internet à une vente par correspondance, il est encore aujourd'hui difficile d'apporter la preuve irréfutable de l'existence d'une commande et de son paiement pour un montant fixé lors de l'achat.

La signature électronique apporte quelques éléments de réponse, tels que l'identification certaine des parties, la preuve d'audit [JAW 01] ou la preuve de non répudiation. Cette dernière protège avant tout le marchand qui peut d'une part être assuré de l'identité du client et d'autre part apporter la preuve de la réalité de la transaction. Cependant, le recours unique à la signature électronique ne protège pas le client d'une erreur, involontaire ou frauduleuse, imputable au marchand.

L'autorité de transaction [TRU 03] est un tiers de confiance dont le rôle principal est de superviser les transactions et d'archiver les éléments de preuve pour le compte des parties. Ce tiers concerne les organismes bancaires en priorité.

2.5.1. Services proposés

L'autorité de transaction propose une manière efficace de garantir à la fois :

- L'authenticité et l'intégrité des transactions.
- Une traçabilité légale de chaque transaction pour preuve de non répudiation.
- La confidentialité des informations personnelles concernant le client.
- Le lien entre les banques des clients et des marchands.
- L'assurance de ne divulguer aucune information confidentielle ou relative au compte bancaire du client (numéro de carte bancaire notamment). Le client est par le fait assuré que le vendeur ne peut réitérer la demande de débit sans son accord préalable.

Ce triple objectif est réalisé notamment grâce à l'intégration d'un procédé de labellisation des commerçants [TRU 03] et à la mise en place d'un paiement par monnaie électronique [COT 02b].

Un autre procédé faisant directement intervenir les banques des parties respectives a été envisagé par [MED 02]. Il se décompose en cinq phases :

1. Le commerçant transmet un ordre de paiement vers le tiers de transaction.
2. Le client transmet au tiers de transaction ses informations personnelles (numéro de carte de crédit, date d'expiration, nom du porteur éventuellement).
3. L'autorité de transaction autorise ou interdit le paiement.
4. Le paiement étant autorisé, l'autorité de transaction demande un ordre de paiement au client qui lui renvoie un numéro d'autorisation.
5. L'autorité de transaction débite alors le compte bancaire du client et crédite le compte du commerçant.

2.5.2. Modélisation

Nous nous intéressons précisément à une autorité de transaction faisant appel à un compte virtuel crédité par de la monnaie électronique [COT 02b], comme le souligne le cas d'utilisation « Monnaie électronique » de la figure 43.

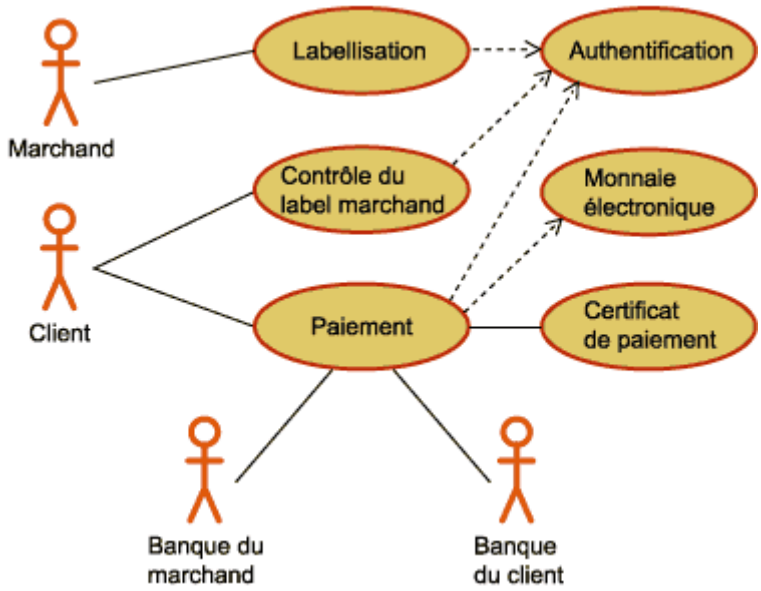

Figure 43 : cas d'utilisation de l'autorité de transaction

2.6. Autorité d'archivage

Les documents numériques signés électroniquement dans des conditions conformes à leur reconnaissance juridique doivent être archivés, de la même manière que le sont à l'heure actuelle les documents sur support papier [WAC 02], lorsqu'ils ont produit leurs effets juridiques.

L'objectif premier de l'autorité d'archivage est conserver « en l'état » les contenus qui lui sont confiés, en prenant en charge l'évolution des supports de stockage [GTA 00]. Elle garantit que le contenu restitué est identique au contenu déposé, et ce indépendamment du temps écoulé entre le dépôt et la restitution. Les textes normatifs et notamment les normes NF Z42-013 [OSI 99] et ISO 15489 ainsi que le « Guide de l'archivage électronique sécurisé » [GTA 00] attestent de la complexité de mise en œuvre de ce tiers de confiance.

La conservation de contenus signés électroniquement sur le long terme pose non seulement des problèmes juridiques [CAP 97] [PIE 02a] mais également des problèmes techniques spécifiques à leur volatilité, tels que :

- Le support physique des archives doit évoluer au cours du temps tout en conservant la garantie d'authenticité suite au transfert de support. Cette garantie est apportée par des audits ainsi que par la reconnaissance des procédés et la traçabilité (à valeur légale) des opérations effectuées sur les archives. Cette évolution est un problème d'actualité, comme le souligne un technicien de chez SONY [FOX 03] : « *Avant de voir disparaître toute possibilité de relire vos films cinéma […] et vos enregistrements sur cassette vidéo […], faites-les transférer sur DVD [..]. Ce transfert permet en outre de relire des documents qui, bientôt, ne trouveront plus de lecteur en état de marche* ».

- Le format du contenu signé actuel peut être désuet dans plusieurs années et de fait devenir illisible.

- Le format des informations relatives aux signatures (politiques de signature, format des certificats, méthodes d'obtention des signatures numériques, procédés de validation, etc.) peut également ne plus être reconnu dans quelques années si l'on se réfère à la vitesse d'évolution des technologies de l'information.

- Le document signé doit non seulement rester lisible, mais aussi intelligible [PIE 02a] et les signatures conserver leur validité au cours du temps.

Ainsi, nous proposons que l'autorité d'archivage prenne en charge uniquement l'évolution du support. Les autres questions précédemment soulevées peuvent être résolues en l'associant à un notaire électronique indépendant. L'archiveur agit alors en tant que prestataire de services d'archivage (PSA). Cette combinaison, apte à garantir la pérennité des archives sur le long terme, est envisagée par [COT 03c].

Les principes qui y sont décrits ont fait l'objet d'un dépôt de brevet international [OEB 01b].

2.6.1. Services proposés

L'autorité d'archivage dispense, au sein de la Chaîne de Confiance, le service de base dit de stockage passif [GTA 00]. Il peut en outre proposer les services suivants :

- L'enregistrement sécurisé des écrits numériques. Par exemple, la sécurité peut provenir des moyens traditionnels (mur pare-feu, zone démilitarisée, cryptage des archives, etc.). Elle peut également s'envisager sous la forme d'une architecture répartie faisant appel à plusieurs unités de stockage. L'archive peut être scindée en morceaux selon un principe dérivé d'un algorithme de partage de secret [KUL 00], chacun des morceaux étant enregistré sur une unité distincte.

- L'évolution des supports physiques (conservation active [GTA 00]).

- La mise à jour des contenus, dès lors qu'ils ne sont pas liés logiquement à une signature électronique attachée ou détachée, c'est à dire enregistrée avec le contenu (un seul fichier) ou séparément (la signature est enregistrée à part).

- La mise à disposition des contenus archivés.

- La consultation des écrits archivés par les personnes autorisées ou les administrations mandatées. Les problèmes liés à la consultation des archives non notariées alors que leurs propriétaires originels ne sont en mesure de réaliser la demande ne sont pas traités.

- La destruction, mise à jour, ainsi que la restauration des archives.

En particulier, l'archiveur peut être en relation avec un PSCE afin de conserver les éléments d'époque nécessaires à l'établissement, a posteriori, des signatures électroniques telles que les listes de révocations, politiques de signature standards et pratiques de certification. L'archiveur apporte ici une preuve informatique du vieillissement de ces éléments de preuve, comparable au jaunissement du papier, reposant sur leurs informations de traçabilité [COT 03c].

2.6.2. Modélisation

Le schéma de l'autorité d'archivage (figure 44) est centré sur le stockage des contenus (signés ou non) sur le long terme. Ainsi, au module de stockage logique (stockage réparti par exemple) qui permet de conserver l'accessibilité des contenus en ligne est couplé un module de stockage physique réalisant l'archivage.

L'archivage met en avant des procédés de mise à jour des supports physiques de stockage (CD-ROMS, DVD-ROMS, etc.) afin de conserver la possibilité de lecture des contenus. La lisibilité des contenus n'est cependant pas du ressort de cette autorité mais plutôt du notaire électronique qui considère l'archiveur comme un prestataire de services d'archivage (PSA).

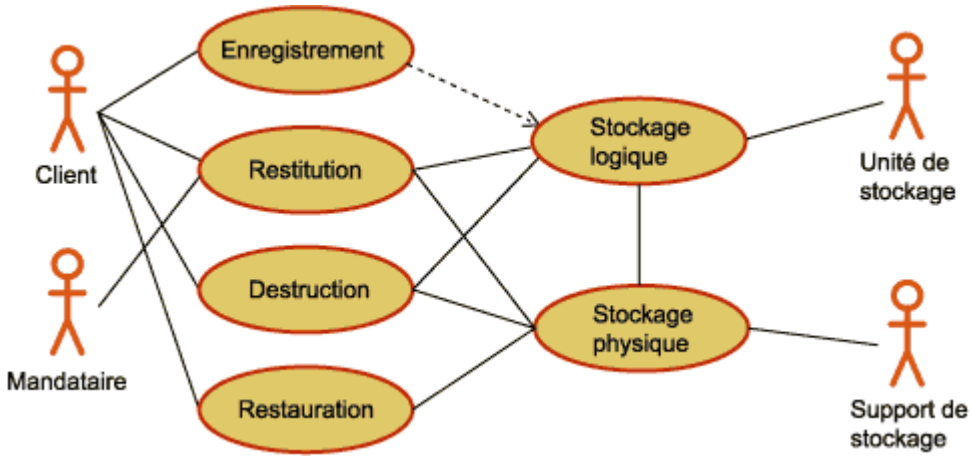

Figure 44 : cas d'utilisation de l'autorité d'archivage

2.7. Notaire électronique

Le notaire électronique[10] (ou « contrôleur de signature ») officie en relation avec un PSA et peut être lui-même une autorité de certification [ROO 99].

Le service de notarisation électronique joint une information de validité à la signature juridique qui lui est présentée [ROO 99]. La valeur de validité est attribuée en relation avec le PSCE émetteur du certificat du signataire ou avec tout autre tiers de confiance apte à fournir une réponse certifiée quant au statut de validité dudit certificat. A cette valeur de validité peut être apposée une date certifiée par une autorité d'horodatage afin de permettre une vérification dans le temps de la signature du notaire.

Cependant, en sus du besoin de sécurité de l'archivage, des contraintes supplémentaires se posent lorsqu'il s'agit de conserver un écrit numérique sous une forme intelligible pendant plusieurs décennies. Parmi celles-ci :

- L'écrit numérique doit être conservé de manière à apporter la preuve de son authenticité au cours du temps.

[10] Il s'agit ici d'un abus de langage, dérivé du terme anglo-saxon « electronic notary » qui ne joue en aucun cas le rôle du notaire que nous connaissons en France.

- L'écrit numérique doit rester lisible, et ce, bien que les technologies de l'information évoluent rapidement.

- Les signatures apposées sur l'écrit numérique doivent être vérifiables. Cela implique que les structures de données et les algorithmes d'aujourd'hui puissent être reproduits demain et que les données relatives à la détermination du statut des signatures électroniques soient elles aussi conservées.

- Le notaire assure que le format global de l'écrit numérique et des signatures électroniques associées est pérenne. Il n'assure cependant pas la pérennité du support physique de stockage : ce service est proposé par le PSA. Seuls les contenus non signés étant pérennisés par le PSA, le notaire électronique pourra archiver séparément le contenu non signé d'une part et son attestation de validité des signatures originales d'autre part [COT 03c]. Cette attestation, signée par le notaire, est conservée intègre par le PSA. Le notaire prendra en outre soin de conserver un lien logique entre le contenu non signé et son attestation.

Bien que des solutions techniques à la notarisation électronique (en relation avec à la conservation de la validité des signatures au cours du temps) existent, leur implantation se heurte à l'évolution de la profession qui désapprouve son utilisation dans le cadre de la signature authentique. Celle-ci requiert en effet la présence physique du signataire au moment de la signature de l'acte notarié sous forme papier.

2.7.1. Services proposés

Le notaire électronique est un organe clé de la Chaîne de Confiance. Il fournit les services suivants :

- La notarisation électronique des contenus soumis. Cette notarisation peut se traduire par la création d'un ensemble d'informations organisées ou libres attestant de la validité des signatures apposées sur le contenu [COT 03c].

- L'enregistrement sécurisé des contenus notariés par le biais d'un PSA.

- La mise à disposition des contenus notariés et archivés ainsi que des attestations.

- La prise en charge de l'évolution de certains formats de documents selon une procédure certifiée garantissant la conformité du document par rapport au document original. Cette procédure d'évolution de format doit s'accompagner de la conservation de la validité des signatures d'origine. Archiveur et notaire électroniques jouent ici un rôle complémentaire : le

notaire garanti l'authenticité des signatures et formats logiques alors que l'archiveur assure la pérennité des supports physiques.

- La création sous forme standard (et pérenne pour l'autorité d'archivage) de l'ensemble des informations pertinentes contenues dans l'archive originelle. Les informations extraites sont « notariées », conservées et liées logiquement à l'écrit sous forme électronique auquel elles se rapportent [COT 03c].

- La consultation des écrits archivés par les personnes autorisées. L'autorisation est déterminée par le notaire électronique. Le processus de délégation des autorisations ne fait pas l'objet de la présente étude.

- La destruction ainsi que la restauration des archives.

Les protocoles et structures de données relatives au système de notarisation électronique proposé sont détaillés dans [COT 03c].

2.7.2. Modélisation

De manière similaire à l'autorité d'archivage, le notaire électronique fournit des services liés à la gestion des contenus notariés présentés par le schéma de la figure 45. Le module de notarisation, le plus important, répond au problème de pérennité logique (pérennité de format et de validité à long terme des signatures) de l'écrit sous forme électronique évoqué précédemment. La liaison avec une autorité d'archivage est ainsi nécessaire afin d'aboutir à une pérennité à la fois logique et physique des documents notariés.

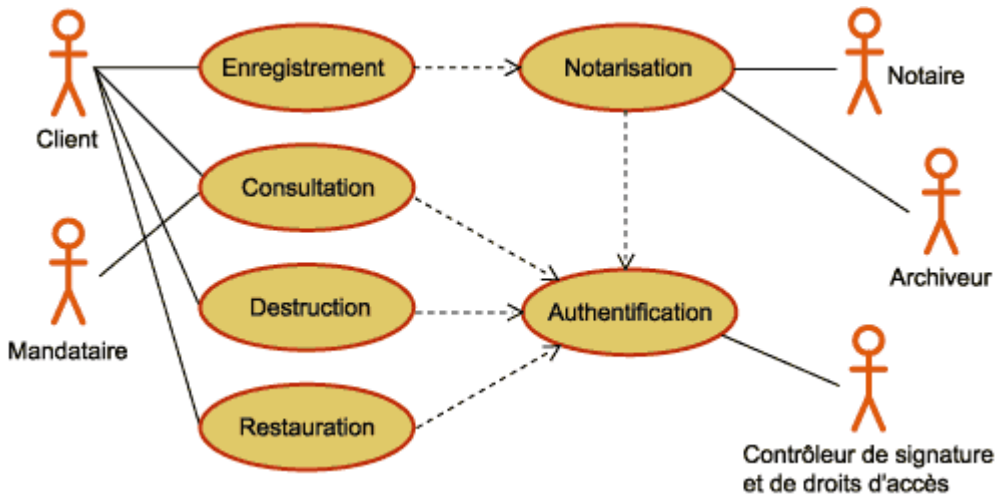

Figure 45 : cas d'utilisation du notaire électronique

Conclusion générale et perspectives

Nous avons présenté quelques éléments techniques et organisationnels permettant d'intégrer la signature électronique dans un nouvel environnement réparti objet ayant une dimension technique et juridique. Cet environnement a été partiellement mis en œuvre par un prototype qui nous a permis de réaliser quelques uns des tiers impliqués dans la Chaîne de Confiance.

Les réflexions de fond menées sur la signature électronique ont mis en avant de nouvelles propositions de formats et protocoles liés à sa reconnaissance juridique. Nous avons ainsi présenté un format de signature électronique dans lequel s'intègre un certificat à multiples clés publiques associé à des protocoles d'horodatage multiple et de non compromission de clé.

Le format de signature retenu a ensuite été intégré au sein des échanges dans un modèle d'environnement réparti objet chargé de sécuriser les échanges entre objets répartis. Ce modèle est dérivé de l'architecture CORBA de l'OMG, standard de-facto lorsqu'il s'agit de concevoir des applications réparties faisant appel aux concepts mis en avant par la programmation orientée objet. Là où CORBA fait appel à un service externe de sécurité, l'intérêt du modèle est d'intégrer la signature électronique au sein de l'environnement en proposant notamment des interfaces de signature et de validation et de mettre à disposition des objets client un contrat IDL étendu baptisé « secured-IDL » ou S-IDL. Cette évolution permet notamment d'intégrer les restrictions d'accès aux services mis en œuvres et l'authentification du demandeur d'un service.

Ce modèle propose un environnement réparti dans lequel les échanges peuvent être signés et chiffrés. La signature apporte les garanties d'authentification de l'émetteur et d'authenticité de l'information transmise. Elle s'inscrit dans un cadre juridique dès lors que les environnements et procédés de signature et validation sont conformes aux recommandations législatives. Le chiffrement permet de sécuriser les échanges dans la mesure où seul les destinataires d'une information confidentielle sont en mesure d'accéder à son contenu.

L'intégration de systèmes d'audit et de surveillance au sein du modèle octroie non seulement la possibilité de gérer les objets répartis déployés mais propose également une solution aux attaques de type déni de service (DoS) en permettant la mise en quarantaine d'objets soumis à de telles attaques dès lors que le système de surveillance permet d'effectuer des duplications et migrations entre objets [COT 00] [MAR 03] reposant sur la reconstruction des objets par l'intermédiaire de fabriques (ou usines) d'objets [GAR 96].

Le prototype réalisé à partir de ce modèle d'environnement réparti, et notamment des diagrammes UML proposés, a mis en évidence la pertinence des structures de données et protocoles proposés. La modularité de ce prototype introduite par les interfaces de signature, vérification, audit et supervision que présente le modèle en fait un outil évolutif apte à s'adapter à l'évolution des technologies en matière de

sécurité et de signature électronique. Ce prototype intègre non seulement l'ensemble des interfaces requises par le modèle mais définit également une interface d'intercepteurs [BAL 00] supplémentaire permettant d'intercepter les requêtes avant leur envoi et les réponses avant leur traitement.

Cependant, l'informatique ne peut se substituer à un jugement humain, c'est pourquoi, comme le souligne [COT 03a], la décision finale de recevabilité d'une information, quelle que soit sa forme ou son support, est laissée à l'appréciation du juge. Tout environnement réparti dérivé du modèle présenté apporte une expertise et des garanties procédurales permettant au juriste de prononcer objectivement la recevabilité pour preuve d'une information numérique.

Au regard du travail effectué et des expérimentations menées, de nombreuses perspectives de recherche restent à explorer.

Tout d'abord, un compilateur du langage S-IDL demande à être spécifié. Cette étude a été délibérément omise : en effet, les règles de compilation sont intimement liées à l'architecture mise en œuvre en fonction des diagrammes présentés. La transformation des structures de données S-IDL dans le langage de programmation considéré puis en ASN.1 s'inscrit dans la compilation des contrats S-IDL.

En l'état, le prototype n'intègre pas encore la gestion des jetons de sessions utilisés par les objets de type serveur afin de limiter les vérifications de validité des certificats et des droits d'accès associés. Les propositions discutées seront validées lorsque le prototype les intègrera.

La problématique de délégation de droits lors de l'appel à plusieurs objets de type serveur ou mixte pour réaliser un service demandé par le client nécessite un approfondissement. Cette thématique est commune à l'ensemble des environnements répartis sécurisés et a été étudiée par [NAG 98]. Les recherches menées par le groupe de travail GT-GA [GTG 04] de l'association IALTA France, rendues publiques depuis peu, attestent de l'actualité de la gestion des attributs des porteurs de certificats. Ces attributs peuvent être utilisés afin d'accéder à des services proposés par les systèmes informatiques. L'intégration de la délégation de droits au sein du prototype suppose cependant la mise en place de protocoles d'accord permettant la génération de ces jetons. Les objets souhaitant acquérir des droits délégués doivent prendre contact avec les objets qui d'une part disposent de ces droits et d'autre part sont enclins à les déléguer.

D'autres travaux complémentaires liés à la multi-signature contrôlée [LGI 02] sont en cours. Cette dernière permet de définir l'ordre et les rôles requis lors de la signature d'un document faisant intervenir un ou plusieurs signataires.

Plusieurs solutions techniques répondant aux problèmes de conservation sur le long terme et de pérennité de l'écrit sous forme électronique ont été proposées par [CAP 97], [PIN 01b] et [COT 03e]. Néanmoins, ces solutions ne peuvent être éprouvées aujourd'hui puisqu'elles répondent chacune à une projection dans l'avenir. Or, l'évolution des techniques et l'abandon de solutions actuelles ne peuvent être

prédites. La problématique de conservation et les possibilités de relecture et preuve d'authenticité dans le temps sont ainsi ouvertes.

La conservation des contenus numériques par le biais d'autorités d'archivage est de fait un thème en plein essor. Ces archiveurs légaux, de par leur statut juridique et leur responsabilité, doivent faire appel à des architectures matérielles et logicielles évoluées qui doivent s'assurer de la *visibilité* et de l'*intelligibilité* [PIE 02a] de leurs archives[11]. Nous envisageons par exemple de réaliser un archivage réparti faisant appel à des stratégies de répartition telles que le secret partagé [KUL 00] [ZIM 03] (présenté en annexe) appliqué à des contenus numériques quelconques.

Outre les problèmes d'adaptation de la Qualité de Service globale du modèle réparti proposé [MIC 03], il apparaît que ce dernier peut être amélioré en tenant compte par exemple du passage d'objets répartis par valeur (« Object By Value », OBV) tel que défini depuis la version 2.3 de la norme CORBA. De plus, nous ne nous sommes intéressés ni aux appels asynchrones entre objets ni aux rappels d'objets client (« callbacks » serveurs). Il est également intéressant de faire évoluer ce prototype afin de mettre en œuvre des objets répartis pouvant supporter plusieurs interfaces [GAR 96] afin de simplifier la conception et l'interprétation des droits d'accès. Les tâches relatives à la construction des souches client et serveur ainsi que la rédaction des contrats S-IDL peuvent faire l'objet d'une extension au prototype actuel.

De plus, les échanges entre objets répartis sont réalisés par le biais d'un encodage ASN.1 reprenant textuellement les structures de données proposées. Les tendances actuelles évoluent vers les services web (« Web Services »), le standard de représentation des données XML [CHA 01] et le protocole de communication SOAP (« Simple Object Access Protocol »). En effet, les éléments structuraux représentés ici en ASN.1 peuvent être exprimés en XML car ces deux langages permettent de représenter des données formatées. Il existe de plus des conventions de traduction entre ASN.1 et XML [IMA 00]. Se pencher sur l'intégration de ces technologies au sein du modèle proposé peut apporter de nouvelles perspectives mettant en jeu les avantages de la signature électronique au format XML [EAS 02] et particulièrement la possibilité de ne signer qu'une partie d'un contenu numérique.

Enfin, bien que la plupart des protocoles prenant par au modèle proposé ont été mis en œuvre dans le cadre d'un prototype multi-tiers, il est souhaitable que des preuves formelles soient apportées. Une modélisation par réseaux de Petri [MUR 89] [RIE 03b] associée à une validation par preuve de programme [WOO 96] [MIT 98] est une étape nécessaire à la finalisation du travail concernant les protocoles d'échange et notamment d'horodatage multiple.

---

[11] La visibilité ne pose pas de problème *a priori*, à moins que les archives ne soient chiffrées. L'intelligibilité de l'information ne peut être acquise que dans le cas où son format est pérenne (une utopie dans le monde de l'informatique) ou si divers traitements permettent de restituer une image fidèle et compréhensible de l'archive originale.

Lexique

ARL (« Authority Revocation List ») : une liste d'autorités révoquées fonctionne à l'identique d'une CRL traditionnelle à la différence que seuls les identifiants des certificats d'autorités de confiance y sont inscrits.

ASN.1 : notation abstraite (« Abstract Syntax Notation 1 ») normalisée par l'OSI en 1994. Ce langage permet de décrire textuellement des structures de données informatiques. Il s'accompagne de règles d'encodage et décodage permettant de transmettre les valeurs associées aux structures décrites entre différents systèmes hétérogènes.

Authentification : procédé par lequel un système informatique mesure le degré de fiabilité d'une signature électronique afin de valider un acte juridique sous forme numérique ou de contrôler l'accès d'un utilisateur ou d'un programme. Les informations liées à la signature de ce dernier peuvent revêtir un caractère juridique dès lors que la signature a été générée dans un contexte conforme à la directive européenne et à l'aide d'un dispositif sécurisé de création que l'utilisateur garde sous son contrôle exclusif. Ce type d'authentification apporte plus de garanties d'authenticité que le traditionnel recours à un identifiant et un mot de passe (identification faible).

Bi-clé : *(n. f.)* ensemble formé par une clé privée et la clé publique correspondante.

Certificat numérique (de clé publique) : pièce d'identité informatique permettant de prouver l'appartenance d'une clé publique (ou de vérification) à une entité donnée (une personne, un programme ou un serveur). [HOU 99]

Chaîne de confiance : suite de procédés impliquant des tiers de confiance, de la construction à la conservation des documents signés électroniquement.

Clé privée (ou clé de signature) : clé servant à apposer une signature numérique. Chaque clé privée est unique. Elle est conservée secrète par son propriétaire. Toute clé privée peut ou non être activée à l'aide d'un mot de passe.

Clé publique (ou clé de vérification) : clé unique liée à une clé privée. A une clé publique donnée correspond une unique clé privée et inversement. Chaque clé publique permet de vérifier une signature créée à l'aide de la clé privée correspondante.

Clé de session : clé unique, souvent temporaire, générée lors d'un échange transactionnel entre deux parties afin de préserver la confidentialité de l'échange.

Chiffrer (ou crypter) : utiliser une clé de chiffrement, généralement une clé publique ou une clé de session, afin de rendre incompréhensible la lecture des informations chiffrées à toute personne ou programme ne disposant pas de la

clé de déchiffrement (la clé privée correspondant à la clé publique utilisée pour le cryptage dans le premier cas, ou la clé de session dans le second).

Classe (de certificat) : les certificats (et clés associées) peuvent être délivrés par les PSCE sur différents supports et générés à partir de données d'identification d'une fiabilité variable. Un vérificateur doit être en mesure de déterminé le degré de confiance qu'il peut attribuer au certificat qui lui est présenté. Cette information lui est communiquée par le biais de la classe du certificat, souvent déterminée par le certificat du PSCE émetteur. Un PSCE disposera ainsi de différentes racines de certification prenant en charge des classes de certificats différentes, pouvant aller de classe 1 (le moins sécurisé) à classe 3. Certains PSCE définissent par ailleurs des classes 3+, 4 et 4+.

Collègue : « Quelqu'un qui fait le même boulot que moi en un peu moins bien » (Conrad Party).

Co-signer : action d'apposer sur un contenu numérique une signature électronique au même niveau que la précédente signature.

Contenu numérisé : contenu issu de l'acquisition numérique (numérisation) d'un contenu original sous forme papier.

Contenu dématérialisé : contenu n'ayant qu'une existence informatique, tel que les messages EDI.

Contre-signer : action d'apposer une signature électronique au-dessus d'une ou plusieurs autres signatures électroniques sans avoir pris connaissance du contenu numérique signé par les précédents signataires.

CRL (« Certificate Revocation List ») : une liste de certificats révoqués contient les identifiants de l'ensemble des certificats clients mis en opposition au cours d'une période donnée dans le passé. Ces listes sont utilisées lors de la validation des signatures électroniques et de l'établissement de connexions sécurisées.

Document numérique (ou *contenu numérique*) : concerne toute donnée informatique composée de bits. Il peut aussi bien s'agir d'un document MS-Office que d'une image ou d'un courriel.

Ecrit (sous forme) électronique : désigne un contenu numérique ayant une valeur juridique (selon l'article 1316-1 de [SEN 00]). Cette valeur est attribuée au contenu numérique de par sa signification et en fonction des différentes signatures électroniques qui lui sont apposées.

Empreinte numérique (résumé de message ou hash) : fonction mathématique non réversible qui permet de compresser un document de taille quelconque en une donnée de taille fixe. Il est pratiquement impossible de retrouver le document original en ayant pour seule connaissance l'empreinte dudit document. Les

algorithmes d'empreinte les plus utilisés actuellement sont SHA-1 [NIS 95] et MD5 [RIV 92]. D'autres algorithmes sont en cours de standardisation, tels que RIPEMD-160 [DOB 96].

Image exacte : l'image exacte d'un document est une duplication de ce document qui n'altère ni son support, ni son format, ni son contenu d'origine. Ce terme d'« image exacte » est en réalité un abus de langage puisque toute copie physique (une photocopie par exemple) finit par dénaturer le contenu lorsque cette opération est réalisée en chaîne. Ce problème peut, dans une moindre mesure, se retrouver dans le monde numérique dès lors qu'une erreur de copie d'un document dématérialisé n'est pas détectée.

Image fidèle : l'image fidèle d'un document consiste à appliquer une transformation du format de ce dernier sans toutefois en altérer le contenu. Il peut par exemple s'agir de la dématérialisation d'un document sous forme papier afin de pouvoir le manipuler par les systèmes d'information tels que la GED (Gestion Electronique de Documents). Il peut aussi s'agir d'une mise à jour permettant de conserver la lisibilité du document avec un visualiseur plus récent.

Interface : ensemble des opérations qui composent un objet. L'interface décrit l'ensemble des signatures des opérations qu'un client peut invoquer sur un objet [GAR 96] de type serveur.

ICP : une Infrastructure à Clé Publique est une architecture répartie, normalisée notamment par le RFC 2510 [ADA 99], qui met en œuvre différentes agences. Ces dernière sont organisées de manière à enregistrer les demandes de signature électronique (ou de certificats de sécurité permettant d'initier des communications sécurisées) et de générer physiquement les supports remis aux clients. L'ICP gère ensuite le cycle de vie des supports et principalement leur révocation. Cette architecture est déployée au sein des PSCE.

IDL (« Interface Description Language ») : les systèmes répartis orientés objet font en règle générale appel à la mise à disposition des méthodes qu'ils supportent (ou interface) auprès des clients (objets appelants). L'hétérogénéité des langages de mise en œuvre des objets a amené la communauté scientifique à promouvoir l'utilisation de langages indépendant des langages de programmation. Ainsi, clients et serveurs disposent d'un langage commun de description des interfaces : l'IDL.

IP (« Internetwork Protocol ») : ce protocole situé au niveau de la couche réseau du modèle ISO/OSI assure l'acheminement physique des trames réseaux de type TCP.

Middleware : le médiateur désigne, dans le cadre de l'informatique répartie, l'ensemble des couches logicielles permettant à des applications de communiquer à distance. Le *middleware* assure les dialogues entre clients et serveurs souvent hétérogènes. [GAR 96]

Moniker (nom intelligent) : nom d'objet, éventuellement composé, persistant, permettant d'implémenter un lien, avec capacité d'interprétation par des algorithmes de recherche pour chaque nom le composant. [GAR 96]

Objet : associe données et traitements dans une même entité, désignée par un identifiant immuable, en ne laissant visible que l'interface de l'objet. [GAR 96]

OCSP (« Online Certificate Status Protocol ») : ce protocole fait intervenir une autorité de confiance (un tiers certificateur par exemple) dont le rôle est de renseigner en temps réel sur le statut des certificats d'une ICP donnée. Il correspond à la consultation en ligne des CRL (sans toutefois souffrir de leur délai de mise à jour).

PC : une Politique de Certification décrit en détail les procédés mis en œuvre par les autorités de certification soumises au respect des règles dictées dans ce document. Le ministère de l'économie, des finances et de l'industrie (MINEFI) a ainsi mis en place sa propre politique de certification, baptisée PC-type. Cette politique sert de base au référencement de certaines familles de certificats. La version 3.1 de cette politique est susceptible d'être étendue aux autres administrations par le biais de l'ADAE.

PKC : « Public Key Certificate », voir *Certificat numérique*.

PKCS (« Public Key Cryptographic Standards ») : ensemble de standards de cryptographie *de facto* proposés par RSA Data Security imposés à la communauté scientifique. Ces standards prennent en compte l'évolution des technologies, de la création de messages chiffrés (PKCS#1 en 1993) jusqu'à l'intégration de jetons cryptographiques au sein des cartes à puce (PKCS#15 en 2000).

PKI : « Public Key Infrastructure », voir *ICP*.

PSCE : un Prestataire de Services de Certification Electronique (ou Prestataire de Services de Certification – PSC – selon [PEC 99]) met en œuvre une ICP afin de délivrer, conformément à une politique de certification (PC), des certificats numériques et des clés de signatures (ou de déchiffrement).

RMI (« Remote Method Invocation ») : Sun Microsystems, à l'origine de cette technologie, utilise les avantages de portabilité inhérents au langage Java afin de proposer des moyens permettant à deux entités Java de s'échanger des informations dans le cadre de l'informatique répartie.

RPC (« Remote Procedure Call ») : l'appel de procédure à distance est une technique permettant d'appeler une procédure distante de la même manière qu'une procédure locale, en rendant transparents les messages échangés sur le réseau ainsi que les assemblages et désassemblages de paramètres [GAR 96].

Signature électronique : ensemble d'informations numériques comprenant notamment une signature numérique ainsi que des données permettant l'identification du signataire et éventuellement le contrôle a posteriori de la validité de son certificat et de la transaction signée.

Signature numérique : résultat de l'application d'un traitement cryptographique sur une empreinte numérique à l'aide d'une clé de signature.

Sur-signer : action d'apposer une signature électronique au-dessus d'une ou plusieurs autres signatures électroniques. La sur-signature impose que le signataire ait pris connaissance (visuellement) du contenu numérique signé par les précédents signataires.

SSL : le protocole « Secure Socket Layer » permet d'établir une communication sécurisée sur un canal public (tel qu'Internet). Les communications sont sécurisées par l'emploi d'une clé symétrique (ou clé de session) utilisée à la fois pour chiffrer et déchiffrer les communications, source et destinataire des échanges étant les seuls à détenir cette clé.

TCP (« Transport Control Protocol ») : situé au-dessus de la couche IP, ce protocole met en œuvre de moyens de désassemblage et réassemblage des trames circulant sur le réseau de communication et de désignation des interlocuteurs par le biais de leur adresse.

TLS : le « Transport Layer Security » est le nom attribué à la version 3 du protocole SSL.

TSP : ce protocole d'horodatage (« Time-Stamp Protocol »), normalisé par le RFC 3161 [ADA 01], décrit les échanges de données entre un client, souhaitant dater de manière certaine et incontestable un contenu numérique (un document par exemple), et un prestataire d'horodatage appelé tiers horodateur chargé d'apposer une date authentique, attestée par sa signature électronique, sur l'empreinte numérique de ce contenu.

UML : l'« Unified Modeling Language » [FOW 01] est un langage de modélisation objet qui unifie les méthodes de Booch, Rumbaugh (OMT) et Jacobson. UML fait partie des standards adoptés par l'OMG [OMG 97b].

Validation d'une signature électronique : désigne l'ensemble des étapes nécessaires à l'approbation ou au refus d'une signature électronique en aval de la vérification de la signature numérique correspondante.

Vérification d'une signature numérique : utilisation d'une clé de vérification permettant de déterminer si la valeur numérique d'une signature a été créée à l'aide de la clé de signature correspondante. L'empreinte numérique obtenue doit pouvoir être recréée à l'aide du contenu signé et d'un algorithme de calcul d'empreinte afin de contrôler la relation entre la signature et le contenu signé.

Annexes

Les annexes suivantes proposent d'étudier plus en détail certains éléments mentionnés dans cet ouvrage et notamment :

1. Sécurité dans le modèle ISO/OSI

1.1. Présentation de l'organisation en couches

Le modèle de référence ISO/OSI [ISO 88] [ROL 93] propose une normalisation des architectures réseau. Il définit ainsi 7 couches ayant chacune un rôle particulier à jouer :

- La couche *application* (couche 7) offre aux applicatifs les moyens d'accéder à l'environnement OSI et « fournit nécessairement tous les services OSI directement utilisables par des processus d'application ».

- La couche *présentation* (couche 6) « se charge de la représentation des informations que des entités d'application se communiquent, ou auxquelles elles se réfèrent au cours de leur dialogue ».

- La couche *session* (couche 5), qui nous intéresse précisément, propose « aux entités de présentation coopérantes les moyens nécessaires pour organiser et synchroniser leur dialogue et pour gérer leur échange de données ».

- La couche *transport* (couche 4), également traitée dans ce mémoire, « assure un transfert de données transparent entre entités de session en les déchargeant complètement des détails d'exécution d'un transfert de données fiable ». La mise en œuvre la plus couramment utilisée est le « Transport Control Protocol » (TCP).

- La couche *réseau* (couche 3) assure l'acheminement des données jusqu'à la destination désirée. L'« Internetwork Protocol » (IP) en est l'illustration.

- La couche *liaison de données* (couche 3) « fournit les moyens fonctionnels et procéduraux nécessaires à l'établissement, au maintien et à la libération des connexions de liaisons de données entre entités du réseau ». Elle « détecte et corrige, dans la mesure du possible, les erreurs pouvant se produire dans la couche physique ».

- La couche *physique* (couche 1) « fournit les moyens mécaniques, électriques, fonctionnels et procéduraux nécessaires à l'activation, au maintien et à la désactivation des connexions physiques destinées à la transmission de bits entre deux entités de liaison de données ».

1.2. Organisation des fonctions de sécurité

Ce modèle suggère de plus des fonctions de sécurité au sein des différentes couches qui composent le modèle d'architecture réseau normalisée. Parmi ces fonctions nous pouvons citer la confidentialité et l'intégrité des informations transmises sur le réseau, le contrôle d'accès, l'authentification des participants aux communications et la non répudiation des échanges, comme l'illustre la figure 46.

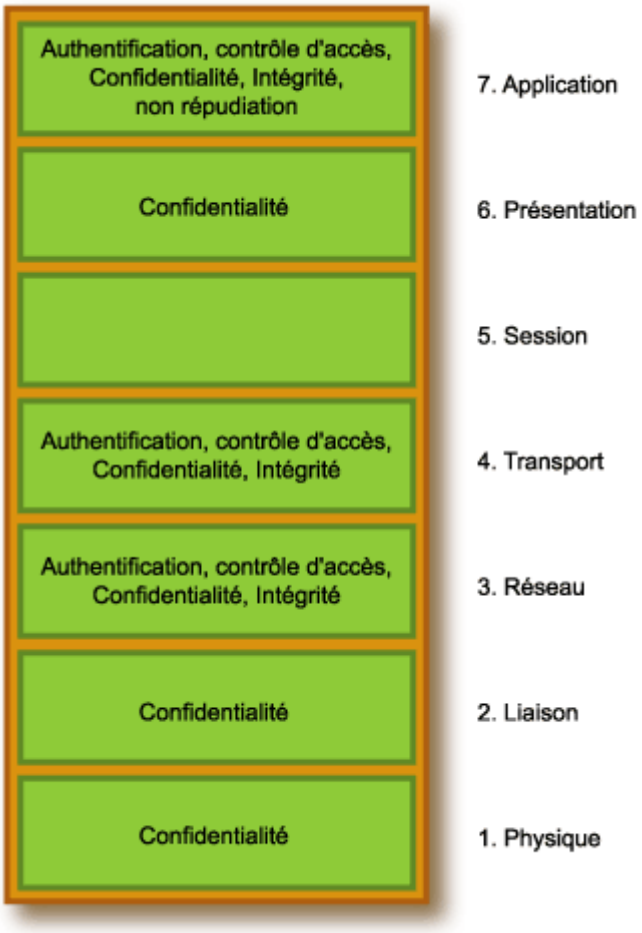

Figure 46 : fonctions de sécurité du modèle ISO/OSI

Il est à noter que la non répudiation est proposée uniquement au niveau de la couche application, ce qui sous-entend que cette fonction est mise en œuvre par les applicatifs.

2. Quelques algorithmes d'empreinte numérique

2.1. MD5

L'empreinte numérique MD5 (« Message Digest 5 ») [RIV 92] fait partie des standards reconnus par le DSS [NIS 00]. Le processus de calcul d'une valeur d'empreinte à partir d'un message source sous forme numérique est le suivant [FLO 97] :

1. Le message initial est décomposé en blocs de 512 bits : $B_0 ... B_n$.

2. Pour chaque bloc B_i, $i \in \{0..n\}$ faire
   Pour j de 0 à 15 faire
   $B_i^j \leftarrow$ le $(j+1)^{\text{ième}}$ sous-bloc de 32 bits de B_i
   Pour k de 0 à 3 faire
   $F_k(a,b,c,d,B_i^j,s,t)$
   fait
   fait
   fait

Etant données a,b,c,d,s,t constantes et F fonction telle que :

$$\boxed{F_k(a,b,c,d,m,s,t) \rightarrow a = b + ((a = f_k(b,c,d) + m + t) \prec s)}$$

où $\prec s$ désigne un décalage bit à bit de s positions vers la gauche et f est définie par :

$$\begin{cases} f_0(x,y,z) = F(x,y,z) = (x \wedge y) \vee (\neg x \wedge z) \\ f_1(x,y,z) = G(x,y,z) = (x \wedge z) \vee (y \wedge \neg z) \\ f_2(x,y,z) = H(x,y,z) = x \oplus y \oplus z \\ f_3(x,y,z) = I(x,y,z) = y \oplus (x \vee \neg z) \end{cases}$$

2.2. RIPEMD-160

RIPEMD-160 est l'une des méthodes de calcul d'empreinte numérique les plus rapides (elle devance les algorithmes MD4, MD5, SHA-1 et RIPEMD-128). Comme son nom l'indique, RIPEMD-160 permet de générer une empreinte numérique sur 160 bits à partir d'une source d'informations numériques quelconque. Cette méthode est supposée plus résistante aux collisions que MD5 [BOE 94].

Son fonctionnement est décrit par l'algorithme suivant [PRE 97] :

1. Le message initial est décomposé en t blocs de 512 bits : $\aleph_i, \forall i \in [0..t[$

2. Pour i de 0 à $t-1$ faire

$A \leftarrow h_0, \quad B \leftarrow h_1, \quad C \leftarrow h_2, \quad D \leftarrow h_3, \quad E \leftarrow h_4$

$A' \leftarrow h_0, \quad B' \leftarrow h_1, \quad C' \leftarrow h_2, \quad D' \leftarrow h_3, \quad E' \leftarrow h_4$

Pour j de 0 à 79 faire

$T \leftarrow rol_{s(j)}\left(A \otimes f(j,B,C,D) \otimes \aleph_i[r(j)] \otimes K(j)\right) \otimes E$

$A \leftarrow E, \quad E \leftarrow D, \quad D \leftarrow rol_{10}(C), \quad C \leftarrow B, \quad B \leftarrow T$

$T \leftarrow rol_{s'(j)}\left(A' \otimes f(79-j,B',C',D') \otimes \aleph_i[r'(j)] \otimes K'(j)\right) \otimes E'$

$A' \leftarrow E', \quad E' \leftarrow D', \quad D' \leftarrow rol_{10}(C'), \quad C' \leftarrow B', \quad B' \leftarrow T$

fait

$T \leftarrow h_1 \otimes C \otimes D'$

$h_1 \leftarrow h_2 \otimes D \otimes E'$

$h_2 \leftarrow h_3 \otimes E \otimes A'$

$h_3 \leftarrow h_4 \otimes A \otimes B'$

$h_4 \leftarrow h_0 \otimes B \otimes C'$

$h_0 \leftarrow T$

fait

Sachant que $h_i, i \in [0..4]$ ont pour valeurs hexadécimales initiales :

$$\begin{cases} h_0 = 67452301 & h_1 = EFCDAB89 \\ h_2 = 98BADCFE & h_3 = 10325476 \\ h_4 = C3D2E1F0 \end{cases}$$

et que K et K' sont deux fonctions discontinues à valeurs hexadécimales telles que :

$$\begin{cases} K(j) = 00000000, & \forall j \in [0..15] \\ K(j) = 5A827999, & \forall j \in [16..31] \\ K(j) = 6ED9EBA1, & \forall j \in [32..47] \\ K(j) = 8F1BBCDC, & \forall j \in [48..63] \\ K(j) = A953FD4E, & \forall j \in [64..79] \end{cases} \quad \text{et} \quad \begin{cases} K'(j) = 50A28BE6, & \forall j \in [0..15] \\ K'(j) = 5C4DD124, & \forall j \in [16..31] \\ K'(j) = 6D703EF3, & \forall j \in [32..47] \\ K'(j) = 7A6D76E9, & \forall j \in [48..63] \\ K'(j) = 000000, & \forall j \in [64..79] \end{cases}$$

Le symbole $\otimes$ représente une addition de module 2^{32} et rol_s une rotation gauche cyclique de s bits (équivalente à $<<<s$).

De plus, f est une fonction non linéaire manipulant un ensemble de bits définie par :

$$\begin{cases} f(j,x,y,z) = x \oplus y \oplus z, & \forall j \in [0..15] \\ f(j,x,y,z) = (x \wedge y) \vee (\neg x \wedge z), & \forall j \in [16..31] \\ f(j,x,y,z) = (x \vee \neg y) \oplus z, & \forall j \in [32..47] \\ f(j,x,y,z) = (x \wedge z) \vee (y \wedge \neg z), & \forall j \in [48..63] \\ f(j,x,y,z) = x \oplus (y \vee \neg z), & \forall j \in [64..79] \end{cases}$$

Enfin, r, r', s et s' sont constituées d'un ensemble de constantes définies pour tout $j \in [0..79]$ dans l'intervalle $[0..15]$ par :

$$\begin{cases} r(0..15) = 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 \\ r(16..31) = 7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8 \\ r(32..47) = 3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12 \\ r(48..63) = 1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2 \\ r(64,79) = 4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13 \end{cases}$$

$$\begin{cases} r'(0..15) = 5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12 \\ r'(16..31) = 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2 \\ r'(32..47) = 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13 \\ r'(48..63) = 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14 \\ r'(64,79) = 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11 \end{cases}$$

$$\begin{cases} s(0..15) = 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8 \\ s(16..31) = 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12 \\ s(32..47) = 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5 \\ s(48..63) = 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12 \\ s(64,79) = 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6 \end{cases}$$

$$\begin{cases} s'(0..15) = 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6 \\ s'(16..31) = 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11 \\ s'(32..47) = 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5 \\ s'(48..63) = 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8 \\ s'(64,79) = 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11 \end{cases}$$

3. Principe de la certification croisée

La certification croisée permet à un PSCE (A) de reconnaître un autre PSCE (B) comme digne de confiance. Pour ce faire, un certificat contenant la clé de vérification de (B), appelé « forward » [AUT 02], est signé par (A). Tout certificat issu par (B) est alors vérifiable en utilisant soit le chemin de certification menant à l'autorité (B) ou à l'autorité (A). Ce procédé est illustré par la figure ci-après.

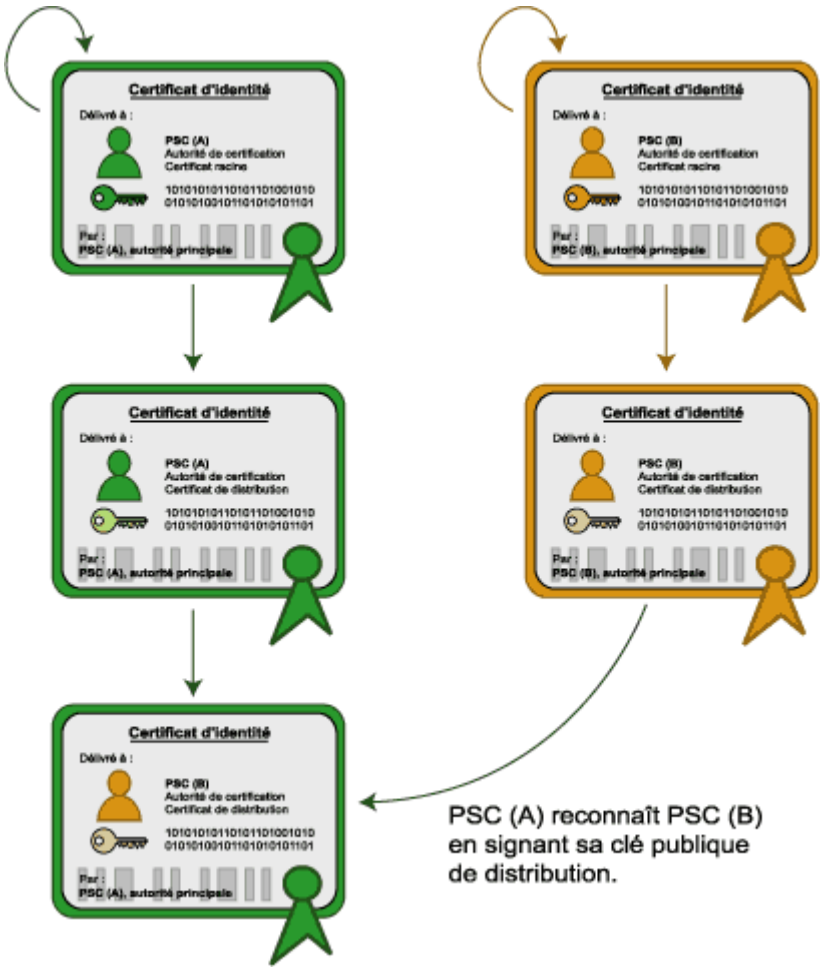

Figure 47 : exemple de certification croisée unilatérale

Etant donnés les certificats de distribution $\{I_A, K_A, V_A\}_{I_A, S_{K_A}}$ et $\{I_B, K_B, V_B\}_{I_B, S_{K_B}}$ des PSCE respectifs (A) et (B), la certification croisée prise pour exemple peut être formalisée comme suit :

$$\boxed{\{I_A, K_A, V_A\}_{I_A, S_{K_A}} \rightarrow \{I_B, K_B, V_B\}_{I_A, S_{K_A}}}.$$

De fait, tout certificat $\{I,K,V\}_{I_B,S_{K_B}}$ issu par le PSCE (B) et signé à l'aide de sa clé privée de distribution est reconnu comme valide par le PSCE (A) car inclu dans le chemin de certification menant à la fois à la racine de (B) et à celle de (A) :

$$\begin{cases} \{I_B,K_B,V_B\}_{I_B,S_{K_B}} \rightarrow \{I,K,V\}_{I_B,S_{K_B}} \\ \{I_A,K_A,V_A\}_{I_A,S_{K_A}} \rightarrow \{I_B,K_B,V_B\}_{I_A,S_{K_A}} \rightarrow \{I,K,V\}_{I_B,S_{K_B}} \end{cases}.$$

Lorsqu'un seul PSCE participe à la certification croisée, celle-ci est qualifiée d'unilatérale. Dans le cas où chacun des PSCE reconnaît son homologue et le certifie par le biais d'une certification croisée, on emploie alors le terme de certification croisée bilatérale.

> *Note : la certification croisée peut s'effectuer également au niveau des racines de certification et non à un niveau supérieur [AUT 02]. La certification croisée d'une bi-clé de distribution de certificats octroie la possibilité de n'accepter que la partie de la PC se rapportant à cette bi-clé (dans le cas où une bi-clé de distribution donnée valide des certificats d'une même classe).*

4. Substitution d'identité dans une signature électronique

Bien que le certificat électronique soit sécurisé par la signature électronique du PSCE émetteur, le format de la signature électronique et le procédé de création des signatures doivent prendre en compte la possible substitution d'identité. Cette substitution peut être [PIN 01c] :

- Un autre certificat du même signataire mais contenant des rôles différents.
- Le certificat d'un autre utilisateur, certifié par une autre autorité de certification. Ceci est possible dès lors que le PSCE émetteur ne génère pas lui-même la bi-clé mais délocalise sa création sur le poste du client.

Dans tous les cas, le certificat de substitution doit contenir la clé de vérification originale afin que la signature puisse être vérifiée, particulièrement dans le second cas où le fraudeur ne dispose pas de la clé de signature.

La signature numérique doit donc être construite non seulement à partir d'une information source (tel qu'un document numérique) mais également à l'aide du certificat du signataire : ce peut être l'identifiant du certificat, comme le propose [PIN 01c], ou l'ensemble du certificat.

> *Note : comme indiqué, la génération de la bi-clé sur le système de l'utilisateur pose un problème dès lors qu'il lui est possible d'envoyer une requête de certification contenant la clé publique d'un autre utilisateur. Il est donc recommandé de conserver le processus de génération de la bi-clé au sein du PSCE.*

Il est donc primordial de veiller à ce que la signature électronique n'octroie la possibilité d'impersonnaliser un signataire et de se substituer à lui.

5. Modèle PGP

Les infrastructures fondées sur le système PGP (« Pretty Good Privacy ») [CAL 98] n'emploient pas de PSCE central : la confiance n'est pas attribuée à un certificat de manière pyramidale en validant un chemin de certification menant à un certificat racine.

Dans le système PGP, chaque certificat est auto-émis par son détenteur qui atteste de la validité des informations qu'il renferme. Ce certificat est alors soumis à un nombre important de personnes qui le contre-signent lorsqu'elles estiment pouvoir lui faire confiance en mentionnant cependant le degré de confiance (indiqué sous forme de pourcentage par exemple).

Toute personne recevant ce certificat détermine le degré de confiance qu'elle peut lui accorder en vérifiant les signatures des contre-signataires. Dès lors qu'elle reconnaît l'un des contre-signataires comme digne de confiance, elle peut accepter le certificat.

C'est pourquoi, a contrario de X.509, PGP est qualifié de « système de confiance mutuelle ». Au contrôle centralisé de l'infrastructure X.509 s'oppose le contrôle décentralisé de PGP. Les gouvernements souhaitant pouvoir contrôler de manière certaine (et être en mesure de déterminer les responsabilités de chacun lorsqu'un faux est mis en circulation), le système PGP a été abandonné au profit du standard X.509 de l'IETF.

6. Modèle SDSI

SDSI (« Simple Distributed Security Infrastructure ») ou « sudsy » [RIV 96] [CLA 01] est une infrastructure basée sur la gestion de clés publiques et proposée comme alternative à X.509 [HOU 02]. Contrairement à l'ICP traditionnelle qui définit un espace de noms unique, SDSI attribue une portée locale à l'identité des titulaires de certificats.

SDSI a pour principal concept l'identification par le biais de la clé publique et non à l'aide de l'identité de son possesseur. Ainsi, une même identité (une personne) peut disposer d'un jeu de clés actives au même moment. Chaque clé publique est associée à l'identité par un certificat de nom. En l'état, X.509 n'octroie pas cette possibilité sans le recours à :

- Un certificat à multiples clés publiques (MPKC).
- La définition de plusieurs pseudonymes, chacun d'eux disposant de son propre certificat, dans le cas où le MPKC n'est pas utilisé.

Un autre concept avancé par SDSI est l'utilisation de listes de contrôle d'accès (« Access Control Lists », ACL) et de certificats d'autorisation permettant de prendre des décisions d'approbation ou de rejet des requêtes formulées par les clients souhaitant accéder à des ressources protégées. Ces notions sont issues des travaux de Butler Lampson et al. [LAM 92]. Ces derniers présentent un modèle d'authentification dans les systèmes distribués, représenté par la figure ci-dessous.

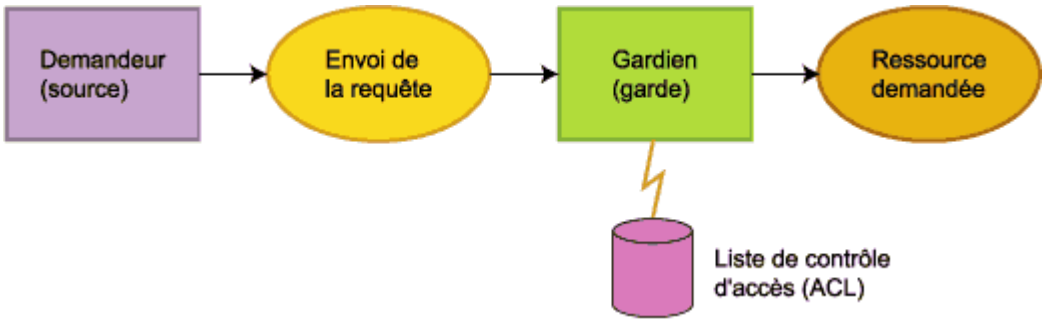

Figure 48 : modèle de contrôle d'accès

L'acceptation d'une requête est analysée par un gardien :

- A l'aide de son ACL locale (lorsqu'il s'agit d'un LOG ou « list oriented guardian »). Les droits d'accès sont alors en la possession exclusive du gardien.
- Ou lors de la présentation d'un ticket d'accès par le client (TOG ou « ticket-oriented guardian »). Dans le modèle X.509, le ticket peut être représenté par un certificat d'attributs [FAR 02] généré par le gardien et relatif à l'accès à un unique objet donné.

Enfin, SDSI repose sur la notion de groupe. Un groupe est constitué d'un ensemble d'identités. Il est géré par un propriétaire qui dispose de la possibilité d'ajouter ou de supprimer des identités dans son groupe. Ce propriétaire peut par exemple être un gardien. De fait, l'accès à l'une de ses ressources peut être accordé dès lors que le demandeur fait partie du groupe.

7. Quelques algorithmes à clé publique

7.1. Accord de clé Diffie-Hellman

Cet algorithme [RES 99] permet à deux correspondants de s'échanger une clé secrète (symétrique) s sur un canal de communication peu sécurisé. Les transmissions suivantes pourront alors être chiffrées à l'aide de cette clé, souvent qualifiée de clé de session, en référence à sa durée de vie.

L'algorithme est le suivant :

1. L'initiateur choisit trois nombres, p un nombre premier, x_1 un exposant aléatoire et g une base.

2. Il calcule alors $y_1 = g^{x_1} \bmod p$ et rend public le triplet $\{g, p, y_1\}$ qui constitue sa clé publique tout en conservant la valeur x_1 secrète (clé privée). La base doit être choisie de sorte que divers exposants produisent des valeurs différentes de y_1.

3. Le correspondant choisit de même un exposant aléatoire x_2 qu'il conserve secret.

4. Il calcule ensuite $y_2 = g^{x_2} \bmod p$, g et p étant les valeurs reçues. Sa clé publique est de fait $\{g, p, y_2\}$.

5. La clé symétrique s_1, calculée par l'initiateur, vaut $s_1 = (y_2)^{x_1} \bmod p$.

6. La clé s_2, caculée par son correspondant, vaut $s_2 = (y_1)^{x_2} \bmod p$.

Comme, $(y_2)^{x_1} \bmod p = (g^{x_2})^{x_1} \bmod p = g^{x_1 \times x_2} \bmod p = (g^{x_1})^{x_2} \bmod p = (y_1)^{x_2} \bmod p$, il suit que $s_1 = s_2 = s$.

> *Initiateur et correspondant partagent ainsi la même clé secrète s sans que celle-ci ait été transmise sur le réseau de communication.*

7.2. Algorithme RSA

Le système le plus employé actuellement fait appel d'une part aux mathématiques modulaires et d'autre part aux nombres premiers. Présenté en 1978, il s'agit du système RSA (Rivest – Shamir – Adleman) décrit dans le standard PKCS#1

[KAL 98a]. Dans ce système, une paire de clés (clé publique et clé privée) est tirée au hasard dans l'ensemble des clés possibles selon l'algorithme suivant :

```
e ← tirage aléatoire d'un grand nombre entier
p ← entier tel que pgcd((p-1),e) = 1
q ← entier tel que pgcd((q-1),e) = 1 et (q <> p)
n ← p*q ("modulus")
d ← entier tel que (d*e-1) est divisible par (p-1) et (q-1)
k ← entier tel que 2^8(k-1) ≤ n < 2^8k
```

La clé publique résultante est composée des éléments $\{n, e\}$ appelés modulus et exposant public. La clé privée contient quant à elle l'ensemble des éléments, à savoir $\{n, e, d, p, q\}$.

Note : la sécurité du système repose sur la difficulté de factoriser le nombre n. Plus les nombres p et q sont grands, plus le problème de factorisation est complexe. Un modulus n de 512 bits, factorisé en un peu moins de 4 mois par 290 machines à la fin de l'été 1999, doit par conséquent être considéré comme inadapté au regard des besoins actuels.

Un texte clair M sera alors chiffré comme suit : $C = M^e \bmod n$.

Le déchiffrement du texte chiffré C se fera en calculant : $M = C^d \bmod n$. En effet,

$$C^d \bmod n = \left(M^e \bmod n\right)^d \bmod n = M^{e \times d} \bmod n = M^{(p-1)\times(d-1)+1} \bmod n = M.$$

La dernière égalité repose sur le théorème d'Euler : ce dernier a démontré que $M^{(p-1)*(q-1)} \bmod n = 1$ lorsque (1) $n = p \times q$ et que (2) p et q sont premiers entre eux.

Note : le tirage de la composante e doit être aussi aléatoire que possible. En effet, l'informatique ne permet pas de générer des nombres réellement aléatoires mais plutôt pseudo aléatoires à partir d'une valeur initiale appelée germe ou semence. On parle alors de PRNG (« Pseudo Random Number Generators ») [JAW 01] ou PRBG (« Pseudo

Random Bit Generators »). L'exemple d'ISAAC [JEN 96] est proposé en annexe 12.

Le système RSA est actuellement le seul système cryptographique qui permette de chiffrer des messages indifféremment à l'aide de ses clés publiques ou privées. Il apporte donc des solutions techniques à la confidentialité des messages ainsi qu'aux besoins de signature électronique.

Voici un exemple simplifié, issu de [DEL 00], de chiffrement RSA : étant donnée une clé privée $\{n,e,d,p,q\}$, posons $p=47$ et $q=71$. Ainsi, $n=pq=3337$ et $r=(p-1)(q-1)=3220$. On choisit alors $e=79$, premier avec r à l'aide de l'algorithme d'Euclide pour le calcul du plus grand commun dénominateur (« PGCD ») entre e et r. On obtient également $d=1019$ puisque $ed=80501=25\times3220+1=1 \bmod 3220$.

Si m est un message tel que $m=6\,882\,326\,879\,666\,683$. L'application de l'algorithme RSA suppose que l'on découpe m en suites de 3 chiffres : $m_1=688$, $m_2=232$, $m_3=687$, $m_4=966$, $m_5=668$ et $m_6=3$.

Chaque élément m_i est chiffré en c_i en l'élevant à l'exposant e modulo n : $c_i=m_i^e \bmod n$. De fait, $c_1=688^{79} \bmod 3337=1570$. De même, $c_2=2756$, $c_3=2091$, $c_4=2276$, $c_5=2423$ et $c_6=158$. Le message chiffré m' est alors donné par $\{c_1,c_2,c_3,c_4,c_5,c_6\}$.

Le déchiffrement de m' est réalisé en opérant pour chaque c_i le calcul $m_i=c_i^d \bmod n$. Il résulte que $m_1=1570^{1019} \bmod 3337=688$. Nous retrouvons bien le message original m.

7.3. Algorithme DSA

Contrairement à l'algorithme RSA présenté ci-avant, le « digital Signature Algorithm » (DSA) du « National Institute of Standards and Technology » (NIST) permet de générer uniquement des signatures numériques reposant sur l'algorithme SHA-1 [NIS 00].

DSA repose sur les éléments de calcul suivants [JAW 01] :

```
p ← tirage aléatoire d'un nombre premier, multiple de 64
bits compris entre 512 et 1024 bits (512 ≤ |p| < 1024)
q ← nombre premier tel que |q| = 160 et (q%(p-1)) = 0
h ← nombre entier aléatoire tel que 1 < h < p-1
```

```
g ← h^((p−1)/q) mod p
x ← nombre entier aléatoire tel que 0 < x < q
y ← g^x mod p
```

La bi-clé est composée des valeurs publiques $\{p, q, g\}$ et de la valeur privée x. Une signature numérique DSA se calcule alors à partir du message M comme suit :

1. $H = hash_{SHA-1}(M)$ désigne l'empreinte numérique du message selon l'algorithme SHA-1 [NIS 00].
2. $k = rand(0, q)$ est un entier aléatoire tel que $0 < k < q$.
3. $r = (g^k \bmod p) \bmod q$.
4. $s = (k^{-1}(M + xr)) \bmod q$.

La signature est formée par la paire $\{r, s\}$.

Le vérificateur en possession de M et de la signature $\{r, s\}$ associée procède à la vérification de cette dernière de la manière suivante (étant supposé que $\{r, s\} \in]0; q[^2$) :

1. $H' = hash_{SHA-1}(M)$.
2. $w = s^{-1} \bmod p$.
3. $u_1 = (Mw) \bmod q$.
4. $u_2 = (rw) \bmod q$.
5. $v = ((g^{u_1} y^{u_2}) \bmod p) \bmod q$.

La signature est vérifiée lorsque $v = r$.

7.4. Protocole SSL

SSL (« Secure Socket Layer ») est un protocole à négociation (« handshake » ou poignée de main SSL), développé par la société Netscape afin de sécuriser les transactions sur Internet.

SSL permet :

- L'authentification de la machine serveur.

- L'authentification de la machine client (phase optionnelle).

- Le chiffrement des transmissions dès lors qu'une clé de session (symétrique) est convenue entre le client et le serveur.

Ce protocole est intégré dans tous les butineurs qui supportent les versions 1, 2 et 3 du protocole. Jusqu'à la version 3, le chiffrement était basé uniquement sur l'algorithme à clé publique RSA. La version 3.1 de SSL appelée TLS (« Transport Layer Security ») n'impose pas de méthode de chiffrement spécifique.

Le déroulement du protocole tel que décrit par [LAN 01] s'effectue en quatre étapes :

- Une phase d'initialisation des paramètres de sécurité au cours de laquelle le client envoie au serveur contacté les algorithmes de chiffrement qu'il supporte. En retour, le serveur lui communique un numéro de session, la stratégie de chiffrement et l'algorithme de compression choisis (supportés par les deux parties).

- Une phase d'authentification du serveur et d'échange des clés. Le serveur commence par envoyer son certificat numérique ainsi que la chaîne de certification émettrice nécessaire à sa validation. Le client vérifie alors que le certificat reçu est valide.

- Débute alors la phase d'authentification du client et d'échange de la clé de session. Au cas où le serveur demande l'authentification du client, ce dernier lui communique son certificat numérique. Le certificat étant accepté par le serveur, le client procède à la génération de la clé de session qu'il chiffre à l'aide de la clé publique du serveur puis signe avec sa clé privée. La clé est alors transmise au serveur.

- La dernière phase consiste en l'approbation de la clé de session par le serveur et le renvoi de son empreinte numérique signée afin que le client valide à son tour la bonne réception de la clé par le serveur.

8. Attaques au système RSA

Les attaques de type « force brute » au cryptosystème RSA consistent à factoriser la clé publique. Il existe cependant d'autres attaques à RSA lorsque certaines propriétés sont réunies [MEN 01] :

- Chiffrement répété d'un même message (attaque de Simmons et attaque de Hastad).

- Utilisation d'un exposant privé (*d*) de taille réduite (Wiener et De Weger).

- Utilisation d'un exposant public (*e*) de taille réduite (Hastad d'une part, Boneh et Durfee d'autre part).

Bien souvent, le système RSA n'est pas directement en cause, mais plutôt le protocole utilisé. Voici quelques exemples de failles.

8.1. Attaque de Davida et Dennings

Nous présentons une attaque rendue possible lorsqu'une même bi-clé RSA est utilisée par un système de réponse automatisé à la fois pour chiffrer et signer des messages. Un intrus peut s'immiscer dans une conversation entre deux correspondants pour enfreindre la confidentialité des messages échangés. Ce scénario est inspiré de [MEL 01].

Soient deux correspondants (A) et (B). L'intrus sera appelé (I). Chacun dispose d'une bi-clé RSA : $\{pub_A, priv_A\}$, $\{pub_B, priv_B\}$ et $\{pub_I, priv_I\}$.

(A) souhaite envoyer un message M à la fois confidentiel et authentifié à (B). Pour ce faire, il signe son message (en utilisant sa clé de signature – privée) puis chiffre le résultat à l'aide de la clé publique de (B) pour obtenir $C = Crypt(Sign(M, priv_A), pub_B)$.

(B) ayant reçu le message confidentiel C, le déchiffre à l'aide de sa clé privée $priv_B$ puis authentifie le message original M grâce à pub_A. Il retourne alors à (A) le message M, cette fois signé avec $priv_B$ et chiffré à l'aide de pub_A : $Crypt(Sign(C, priv_B), pub_A) = Crypt(Sign(M, priv_B), pub_A)$. Le message M est toujours chiffré lorsqu'il transite sur le réseau.

L'intrus, ayant écouté cette conversion, signe puis chiffre le message C envoyé à (B) par (A) : $Crypt(Sign(C, priv_I), pub_B)$. Si le système de (B) provoque des réponses automatiques, ce dernier va retourner à (I) le message $Crypt(Sign(C, priv_B), pub_I)$.

Dès lors, (I) dispose des informations suffisantes permettant de découvrir M.

En effet, le message d'acquittement renvoyé par (B) permet de supprimer le chiffrement original que (A) avait utilisé lors de l'envoi du message à (B) car :

$$\begin{aligned} &Crypt(Sign(C, priv_B), pub_I) \\ &= Crypt(Sign(Crypt(Sign(M, priv_A), pub_B), priv_B), pub_I) \\ &= Crypt(Sign(M, priv_A), pub_I) \end{aligned}$$

Il suffit à (I) de déchiffrer ce message pour obtenir $Sign(M, priv_A)$. Ayant pris connaissance de la clé publique de (A), l'intrus peut alors découvrir le message original M.

Cette attaque est rendue possible par la succession des failles de sécurité suivantes :

- (B) utilise une unique bi-clé à des fins de chiffrement et de signature.
- (B) retourne le message reçu en tant qu'accusé de réception, sans y apporter aucune modification.
- (B) ne vérifie pas la correspondance des clés et la signification du message déchiffré.

Afin de se protéger contre cette attaque, il est recommandé :

- De combiner RSA pour la confidentialité des messages transmis et un autre algorithme lors de l'apposition des signatures (DSA par exemple).
- D'utiliser deux paires de clés RSA différentes : l'une pour chiffrer et l'autre pour signer.
- D'utiliser un protocole reconnu (tel que SSL [LAN 01]) pour chiffrer les transmissions et de n'utiliser qu'une paire de clés RSA pour signer les messages.

Il faut donc veiller, lors de la conception d'un protocole cryptographique, à ne pas signer puis renvoyer les données que l'on reçoit sans les avoir modifiées au préalable.

8.2. Attaque de Hastad

En 1985, Hastad propose une attaque reposant sur l'utilisation d'un faible exposant de chiffrement baptisée « attaque par diffusion ». En effet, la rapidité du chiffrement d'un message tient non seulement compte de la taille des données à chiffrer mais également de la longueur de l'exposant de chiffrement. Un chifffrement efficace (en terme de rapidité) suppose donc l'utilisation d'un exposant public de petite taille.

Soit un module RSA robuste actuellement, de 1024 (ou même 2048) bits. Si l'on chiffre un message M à l'aide d'un exposant public petit (inférieur à 10 par exemple) $e = 3$ à l'intention de m destinataires disposant chacun de son propre module $n_i, i \in \{1..m\}$. Il est raisonnable de penser que les différents modules sont premiers entre eux.

Les m messages chiffrés sont représentés algébriquement ainsi :

$$\begin{cases} c_1 = M^e \bmod(n_1) \\ \ldots \\ c_{m1} = M^e \bmod(n_m) \end{cases}.$$

En appliquant le théorème des restes chinois [DEL 00] aux textes chiffrés c_i obtenus, il suit que le produit C des textes chiffrés s'exprime comme suit : $C = \prod_{i=1}^{m} c_i = M^m \bmod(n_1 \times n_2 \times n_3)$.

Dans le cas où $M^m < n_1 \times n_2 \times n_3$, $C = M^m \bmod(n_1 \times n_2 \times n_3) = M^m$. Ceci est d'autant plus probable que l'exposant e est petit. Le message initial M peut de fait être déduit en calculant la racine $m^{\text{ième}}$ de C : $M = \sqrt[m]{C}$.

8.3. Attaque de Simmons

Cette attaque dite de « module partagé » suppose l'emploi d'exposants publics e_1 et e_2 distincts (et premiers entre eux) et d'un module unique n. Ainsi, un message M transmis de manière sécurisée à l'aide du système RSA est créé avec :

$$\begin{cases} c_1 = M^{e_1} \bmod(n) \\ c_2 = M^{e_2} \bmod(n) \end{cases}.$$

Le théorème de Bezout affirme qu'étant donnés deux entiers a et b tels que $PGCD(a,b) = 1$, alors il existe un couple d'entiers (u,v) tel que $a \times u + b \times v = 1$.

Dans cet exemple, $PGCD(e_1, e_2) = 1$: il existe de fait deux entiers u et v tels que $e_1 \times u + e_2 \times v = 1$. Il découle que :

$M = M^1 = M^{e_1 \times u + e_2 \times v} = \left(M^{e_1}\right)^u \times \left(M^{e_2}\right)^v = (c_1)^u \times (c_2)^v \bmod(n)$. Le message initial M peut de fait être retrouvé à partir des textes chiffrés, du module et des entiers u et v.

9. Exemple de format de KRL

9.1. Besoins

Au standard X.509 est joint un format définissant les composants d'une liste de certificats révoqués (« Certificate Revocation List », CRL). Notre proposition de liste de clés révoquées (« Key Revocation List », KRL) répond à trois besoins :

- Le certificat peut contenir plus d'une clé publique.

- Un individu malintentionné ayant réussi à impersonnaliser le certificat d'un autre utilisateur ne doit pouvoir continuer à signer à l'aide de la bi-clé du détenteur légitime.

- Le signataire ne doit être en mesure de révoquer son certificat dans un premier temps puis de le « réactiver » de manière détournée par le biais de la génération d'un second certificat comprenant la même clé publique.

En l'état, la liste de certificats révoqués, qui permet l'ajout d'identifiants de certificats non utilisables, ne permet pas de répondre à ces besoins.

9.2. Proposition de format

Le format de la KRL repose en grande partie sur la seconde version du format de la liste de révocation tel que présenté par l'IETF [HOU 02] :

```
CertificationList ::= SEQUENCE {
  tbsKeyList          TBSKeyList,
  signature           ElectronicSign
}

TBSKeyList ::= SEQUENCE {
  version           Version DEFAULT v1(0),
  signature         AlgorithmIdentifier,
  issuer            Name,
  crlNumber         INTEGER             OPTIONAL,
  thisUpdate        Time,
  nextUpdate        Time                OPTIONAL,
  revokedKeys       RevokedKeys,
  crlExtensions [0] EXPLICIT Extensions OPTIONAL
}

RevokedKeys ::= SEQUENCE OF RevokedKey

RevokedKey ::= SEQUENCE {
  userKeyId         KeyId,
  userCertId        CertificateSerialNumber,
  revocationDate    Time,
  revocationReason  RevocationReason    OPTIONAL,
  keyExtensions     Extensions          OPTIONAL
}

KeyId ::= CHOICE {
  keyImprint        MessageImprint,
  -- Digest of the revoked key
  keyIndex          INTEGER
  -- Index within certificate
}

RevocationReason ::= INTEGER {
```

```
  -- Revocation reasons to be further defined
  unspecified             (0),
  keyCompromise           (1),
  cessationOfOperation    (5)
}

Time ::= CHOICE {
  utcTime           UTCTime,
  generalTime       GeneralizedTime
}

Version ::= INTEGER

CertificateSerialNumber ::= INTEGER
```

10. Modification du protocole OCSP

10.1. Format standard

Le standard de validation en ligne des certificats (« Online Certificate Status Protocol », OCSP) [MYE 99] fait appel à des requêtes et réponses en liaison avec un tiers de confiance.

Les requêtes sont de la forme :

```
OCSPRequest ::= SEQUENCE {
  tbsRequest              TBSRequest,
  optionalSignature   [0] EXPLICIT Signature OPTIONAL
}

TBSRequest ::= SEQUENCE {
  version             [0] EXPLICIT Version DEFAULT v1,
  requestorName       [1] EXPLICIT GeneralName OPTIONAL,
  requestList             SEQUENCE OF Request,
  requestExtensions   [2] EXPLICIT Extensions OPTIONAL
}

Signature ::= SEQUENCE {
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING,
  certs   [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}

Version ::= INTEGER  {  v1(0) }

Request ::= SEQUENCE {
  reqCert                    CertID,
  singleRequestExtensions [0] EXPLICIT Extensions OPTIONAL
```

```
}
```

Les réponses fournies par les serveurs OCSP sont définies par :

```
OCSPResponse ::= SEQUENCE {
   responseStatus         OCSPResponseStatus,
   responseBytes      [0] EXPLICIT ResponseBytes OPTIONAL
}

OCSPResponseStatus ::= ENUMERATED {
  successful        (0), --Response has valid confirmations
  malformedRequest  (1), --Illegal confirmation request
  internalError     (2), --Internal error in issuer
  tryLater          (3), --Try again later
  --(4) is not used
  sigRequired       (5), --Must sign the request
  unauthorized      (6)  --Request unauthorized
}

ResponseBytes ::= SEQUENCE {
  responseType         OBJECT IDENTIFIER,
  response             OCTET STRING
}

BasicOCSPResponse ::= SEQUENCE {
  tbsResponseData      ResponseData,
  signatureAlgorithm   AlgorithmIdentifier,
  signature            BIT STRING,
  --The value for signature SHALL be computed
  --on the hash of the DER encoding ResponseData
  certs   [0] EXPLICIT SEQUENCE OF Certificate OPTIONAL
}

ResponseData ::= SEQUENCE {
  version              [0] EXPLICIT Version DEFAULT v1,
  responderID              ResponderID,
  producedAt               GeneralizedTime,
  responses                SEQUENCE OF SingleResponse,
  responseExtensions   [1] EXPLICIT Extensions OPTIONAL
}

ResponderID ::= CHOICE {
  byName               [1] Name,
  byKey                [2] KeyHash
}

KeyHash ::= OCTET STRING
--SHA-1 hash of responder's public key
--(excluding the tag and length fields)
```

```
SingleResponse ::= SEQUENCE {
  certID                       CertID,
  certStatus                   CertStatus,
  thisUpdate                   GeneralizedTime,
  nextUpdate         [0] EXPLICIT GeneralizedTime OPTIONAL,
  singleExtensions   [1] EXPLICIT Extensions      OPTIONAL
}

CertStatus ::= CHOICE {
  good                   [0] IMPLICIT NULL,
  revoked                [1] IMPLICIT RevokedInfo,
  unknown                [2] IMPLICIT UnknownInfo
}

RevokedInfo ::= SEQUENCE {
  revocationTime             GeneralizedTime,
  revocationReason   [0] EXPLICIT CRLReason       OPTIONAL
}

UnknownInfo ::= NULL
--This can be replaced with an enumeration
```

10.2. Proposition d'évolution du format standard

Faisant suite à [COT 03a], nous suggérons de tenir compte de l'intervalle permettant de situer la signature dans le temps avec plus de précision qu'un unique horodatage. Le format de la requête de validation est de fait modifié comme suit :

```
Request ::= SEQUENCE {
  reqCert                     CertID,
  timeInterval             [0] EXPLICIT TimeInterval
  singleRequestExtensions  [1] EXPLICIT Extensions OPTIONAL
}

TimeInterval ::= Validity
```

Lorsque le champ « timeInterval » est omis, le serveur OCSP considère la requête comme une requête standard. Dans le cas contraire, ce dernier doit prendre en compte les informations de date fournies. Lorsque les deux dates sont incohérentes (la seconde est antérieure à la première) ou que l'une d'entre elles indique une date future, la requête est rejetée. Lorsqu'une unique date est indiquée et qu'il s'agit de la date « validFrom » alors la date courante est prise pour valeur de la date « validTo ». Dans le cas où seule la date « validTo » est mentionnée alors la date de début de validité du certificat du signataire est attribuée au champ « validFrom ».

Lorsque le certificat du signataire n'a pas subi de modification de statut pendant la période mentionnée, la valeur de statut est retournée. Au contraire, si le statut du

certificat a été modifié, la réponse indiquant la validité du certificat entre les deux instants mentionnés dans la requête peut indiquer plusieurs valeurs de statut. Bien qu'il soit possible de les énumérer au sein de la réponse en associant une valeur de statut à un intervalle de temps, nous préférons retourner une valeur d'erreur indiquant que la demande doit être affinée. Le format de la réponse n'a donc pas lieu d'être modifié par rapport au standard. Cependant, la valeur « multipleStatus » est ajoutée :

```
OCSPResponseStatus ::= ENUMERATED {
  successful         (0), --Response has valid confirmations
  malformedRequest   (1), --Illegal confirmation request
  internalError      (2), --Internal error in issuer
  tryLater           (3), --Try again later
  --(4) is not used
  sigRequired        (5), --Must sign the request
  unauthorized       (6), --Request unauthorized
  multipleStatus     (7)  --Certificate has different status
}
```

11. Modification du format des CRL

11.1. Format standard

Les listes de certificats révoqués (ou CRL) contiennent l'ensemble des certificats révoqués et suspendus non expirés au moment de leur création. Elles sont signées par l'autorité de certification émettrice et régulièrement mises à jour par cette dernière.

Le format standard courant des CRL est décrit en langage ASN.1 dans [HOU 02] :

```
CertificateList ::= SEQUENCE {
  tbsCertList          TBSCertList,
  signatureAlgorithm   AlgorithmIdentifier,
  signatureValue       BIT STRING
}

TBSCertList ::= SEQUENCE {
  version              Version OPTIONAL,
  -- if present, MUST be v2
  signature            AlgorithmIdentifier,
  issuer               Name,
  thisUpdate           Time,
  nextUpdate           Time OPTIONAL,
  revokedCertificates  SEQUENCE OF SEQUENCE  {
    userCertificate       CertificateSerialNumber,
    revocationDate        Time,
    crlEntryExtensions    Extensions              OPTIONAL
```

```
    -- if present, MUST be v2
  }  OPTIONAL,
  crlExtensions   [0]  EXPLICIT Extensions        OPTIONAL
  -- if present, MUST be v2
}

CRLNumber ::= INTEGER (0..MAX)
```

11.2. Suggestion d'évolution

Il est alors difficile pour un programme de déterminer si le certificat présenté est définitivement ou temporairement révoqué. Pour cela, nous utilisons deux listes distinctes de certificats selon qu'ils sont révoqués ou suspendus [COT 03a] :

```
TBSCertList ::= SEQUENCE {
  version               Version OPTIONAL,
  -- if present, MUST be v3
  signature              AlgorithmIdentifier,
  issuer                 Name,
  thisUpdate             Time,
  nextUpdate             Time                     OPTIONAL,
  revokedCertificates
    [0] SEQUENCE OF rosCertificate                OPTIONAL,
  suspendedCertificates
    [1] SEQUENCE OF rosCertificate                OPTIONAL,
  crlExtensions      [2] EXPLICIT Extensions      OPTIONAL
  -- if present, MUST be v2 or v3
}

rosCertificate ::= SEQUENCE {
-- revoked of suspended certificate
  userCertificate        CertificateSerialNumber,
  rosDate                Time,
  crlEntryExtensions     Extensions               OPTIONAL
  -- if present, MUST be v2 or v3
}
```

12. Générateur de nombres aléatoires ISAAC

Les PRNG (« Pseudo Random Number Generators ») [JAW 01] sécurisés proposent des moyens algorithmiques et techniques de tirages de valeurs qui paraissent aléatoires. L'objectif est de générer des tirages les moins prévisibles possibles pour une entité extérieure cherchant à « deviner » des cycles. L'informatique n'offrant pas de nombres « authentiquement » aléatoires, le recours à des nombres pseudo aléatoires est nécessaire.

12.1. Aperçu général

Le PRNG ISAAC (« Indirection, Shift, Accumulate, Add, Count ») génère des cycles capables de produire jusqu'à 2^{8295} tirages aléatoires. Un observateur externe ne peut deviner le $n^{\text{ième}}$ tirage, et ce même en ayant connaissance des $(n-1)$ tirages précédents.

En effet, chaque tour de l'algorithme produit 256 nombres de 32 ou 64 bits formant l'état courant d'ISAAC. Le tour suivant produit à nouveau 256 nombres issus de transformations de l'état courant, de décalages à gauche et à droite des nombres précédents et d'opérations binaires (xor, « OU exclusif »). Le résultat fourni à l'utilisateur est un tableau « rsl » obtenu en effectuant un ensemble d'opérations d'après l'état courant.

[JEN 96] propose des implémentations d'ISAAC dans les langages C, C++ et Java.

12.2. Variables globales

Les différentes versions de mise en œuvre de l'algorithme ISAAC font appel à des variables globales, parmi lesquelles :

```
rsl: tableau [0..255] d'entiers (sur 32 ou 64 bits)
mem: tableau [0..255] d'entiers (sur 32 ou 64 bits)
acc: entier (sur 32 ou 64 bits)
```

- « rsl » est le résultat d'un appel à l'algorithme ISAAC. Il contient 256 valeurs aléatoires.
- « mem » conserve l'état interne (état courant) utilisé pour calculer « rsl ».
- « acc » est un accumulateur chargé de compter les appels à l'algorithme. Il est ajouté à la valeur du dernier nombre produit avant un nouveau tour. Il a l'avantage d'allonger le cycle de génération des nombres.

12.3. Algorithme

La fonction principale « isaac » consiste en l'appel répété de la procédure « nextRand ». Celle-ci produit un nombre aléatoire, stocké à l'indice i au sein du tableau global « rsl » :

```
fonction isaac()
i: entier
```

```
début
  pour i de 0 à 255 faire
    nextRand(i)
  fait
  retourner (une copie de) rsl
fin isaac

procédure nextRand(entier i)
début
  acc ← permut(i, acc) + m[(i+128)%256]
  x ← m[i]
  m[i] ← m[part(x, 2, 10)] + acc + rand
  rand ← m[part(m[i], 10, 18)] + x
  rsl[i] ← rand
fin nextRand
```

La fonction « permut » appelée par « nextRand » consiste en l'appel d'une fonction de décalage parmi quatre décalages possibles en fonction de l'indice i du nombre aléatoire généré :

```
fonction permut(entier i, entier acc): entier
  p: entier
  m:  entier
début
  m ← i mod 4
  si (m = 0) alors
    p ← dag(acc, 13)
  sinon si (m = 1) alors
    p ← dad(acc, 6)
  sinon si (m = 2) alors
    p ← dag(acc, 2)
  sinon
    p ← dad(acc, 16)
  finSi
  retourner (acc xor p)
fin permut
```

La fonction « part » permet d'extraire une plage de bits parmi les bits qui composent un entier, sachant que si x désigne une information représentée sous forme binaire, alors $x|y$ désigne le bit de x situé à l'indice y :

```
fonction part(entier val, entier from, entier to): entier
  // L'indice 'from' est compris,
  // L'indice 'to' est exclu
  // Les erreurs d'indice ne sont pas prises en compte
  res, i: entier
```

```
début
  pour i de 0 à (from-to-1) faire
    res|i ← val|(i+from)
  fait
  retourner res
fin part
```

13. Protection de l'horodatage et attaques

Voici quelques protocoles faisant appel à une autorité de confiance et permettant de sécuriser les horodatages [GUE 03] délivrés par celle-ci afin de détecter toute fraude de datation (anti- ou post-datation). Nous supposons que la phase préliminaire d'initialisation par un premier horodatage a déjà eu lieu.

13.1. Par horodatage en cascade

Dans ce modèle, chaque horodatage ne porte pas directement sur la combinaison de l'identité du demandeur et de l'empreinte à horodater mais sur la conjonction de ces deux éléments et de l'horodatage précédent. De fait, une chaîne d'horodatages est créée. La validation d'un horodatage nécessite de consulter également l'horodatage précédent et l'horodatage suivant.

En effet, toute insertion frauduleuse dans la chaîne des horodatages est possible au cas où le vérificateur tient compte uniquement de l'horodatage précédent. Si l'on poursuit cette idée, la fraude peut reposer sur plusieurs chaînons consécutifs,voire sur l'ensemble de la chaîne. Ainsi, le chaînage devient une solution très coûteuse en temps de vérification.

13.2. Par publication des valeurs générées

Pour remédier au problème de constitution d'une fausse chaîne d'horodatages dans laquelle est inséré l'horodatage frauduleux, il est possible de publier régulièrement les valeurs d'horodatages dans un référentiel indépendant de l'autorité d'horodatage émettrice.

Ainsi, la constitution d'une chaîne frauduleuse est limitée par le dernier horodatage publié. Ceci n'empêche cependant pas la constitution d'une fausse chaîne dans le cas où le vérificateur ne prend pas en compte les horodatages publics à venir pour vérifier l'horodatage courant.

D'autre part, cette solution repose sur au moins deux autorités : l'autorité d'horodatage et l'autorité de séquestre des valeurs publiées. Ceci alourdit considérablement le processus de vérification de l'horodatage.

13.3. Par arbres d'horodatages

Plutôt que de constituer un chaînage linéaire des horodatages, il est possible de les organiser en arbres n-aires (l'arbre binaire semble approprié) où chaque feuille est un horodatage et chaque sommet l'horodatage de l'empreinte des sous-arbres. Les valeurs des sommets principaux sont publiées régulièrement afin de limiter la construction d'arbres frauduleux.

Il est également possible de combiner l'arbre d'horodatages avec un horodatage en cascade de sorte que chaque chaînon fasse référence au sommet d'un sous-arbre. Cette chaîne est alors régulièrement publiée.

13.4. Par confiance répartie

La confiance répartie opère un fonctionnement similaire au « web of trust » de PGP. Un horodatage est proposé par le demandeur et accepté (ou rejeté) par un grand nombre d'entités.

Cette idée est issue des travaux de [STU 91] où l'horodatage est soumis à suffisamment d'entités (utilisateurs) tirées au sort de telle sorte que le demandeur ne peut toutes les corrompre. Comme le souligne [GUE 03], « *Le fait que les entités doivent être disponibles, avoir un temps de réponse court et le fait qu'elles peuvent être dérangées à tout moment limitent l'efficacité de cette proposition* ».

13.5. Par demandes multiples indépendantes

Afin de limiter l'impact d'une éventuelle compromission de clé, l'IETF adopte une solution basée sur la demande de plusieurs horodatages à des autorités d'horodatage distinctes faisant appel à des chaînes de certification dont les certificats racine sont différents [PIN 01a]. Ces horodatages sont présentés au vérificateur qui souhaite déterminer la validité de la signature électronique.

13.6. Conclusion

Bien que de nombreuses études portent sur la protection des horodatages contre la falsification, la question fondamentale est de mettre en rapport le degré de sécurité requis et les vulnérabilités (juridiques, financières, etc.) pouvant résulter d'un horodatage prouvé invalide.

De plus, comme le souligne [GON 90], deux horloges synchrones n'indiquent pas nécessairement les mêmes valeurs de dates à un instant donné. L'autorité d'horodatage n'est de fait pas assurée de l'authenticité des horodatages signés et délivrés aux clients.

Ainsi, l'importance accordée à la mise en œuvre de la sécurisation d'un horodatage à l'aide de protocoles complexes (tels que ceux décrits précédemment) est relative : pourquoi vouloir sécuriser à tout prix les horodatages alors que la

plupart des informations temporelles signées manuscritement ne sont pas certifiées (par un notaire par exemple) mais font foi dès lors que la date est validée par les différentes signatures ?

14. Horodatage de contenus signés

Cette annexe suggère un format de données, reposant sur le format standard CMS [HOU 99]. Conformément aux préconisations de double horodatage (l'un apposé sur le contenu avant signature, l'autre joint à chaque signature), le format présenté définit une structure récursive qui pourra évoluer pour tenir compte des remarques de [PIN 01b] et [ETS 02b] concernant la validité des contenus signés sur le long terme. En l'état, seules les signature électroniques peuvent être protégées et validées sur le long terme, compte tenu de l'utilisation du format de signature électronique développé et critiqué dans ce document.

14.1. Présentation d'un premier format

L'intégration de la sur-signature et la contre-signature au sein du format de signature par récursivité via les champs « envelopSigns » et « counterSigns » de la structure de données « ElectronicSign » peut introduire des lourdeurs dès lors que plusieurs signatures doivent être sur-signées par un même signataire. En l'occurrence, le procédé de création de ces sur-signatures ou contre-signatures doit être réalisé pour chaque signature impliquée puisque chaque signature contient sa propre liste de sur-signatures et contre-signatures. Le format présenté dans cette annexe permet de pallier à cette limitation technique en n'autorisant les sur-signatures que par le biais de la signature d'un contenu signé (considéré ici comme un contenu non signé par le sur-signataire qui appose ainsi sa signature à la fois sur le contenu original et sur les signatures jointes).

> *La contre-signature peut être utilisée, même si elle s'applique également au contenu signé, bien que la contre-signature ainsi créée soit en contradiction avec la définition de celle-ci.*

Ce format repose néanmoins sur les définitions des données relatives à la signature électronique (« ElectronicSign » et formats sous-jacents).

Il apparaît au regard de la figure 49 que seul le contenu original non signé peut se prévaloir d'un horodatage non certifié (une date revendiquée par exemple). Dès lors que ce contenu daté est signé électroniquement, tout horodatage ultérieur doit être certifié.

Ce dernier apporte en outre plusieurs garanties :

- Il apporte une information temporelle permettant de borner les instants d'apposition des signatures conjointes datées par un majorant.

- Il protège les signatures contre toute fraude ultérieure. Les signataires peuvent ainsi employer des tailles de clé de signature réduites en comparaison de la clé de signature de chaque autorité d'horodatage.

- Il empêche l'ajout ou la suppression d'une signature en garantissant que la liste des signatures est authentique.

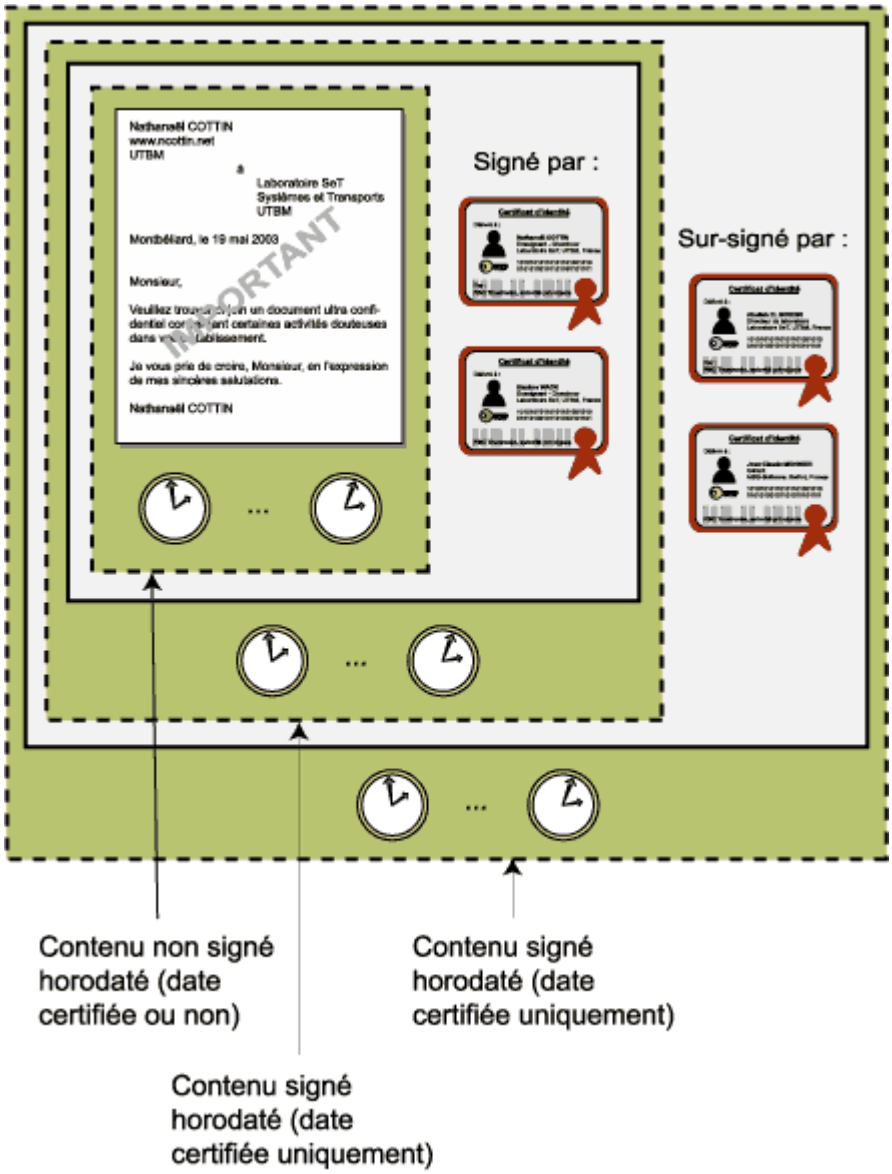

Figure 49 : premier format de contenu multi-signé et horodaté

Les horodatages, à l'exception du premier et du dernier apposé, sont facultatifs puisque les garanties mentionnées précédemment sont attribuées au dernier horodatage. Le premier horodatage (certifié ou non) apporte une information relative à un instant au cours duquel aucune signature n'a encore été apposée. Celui-ci est donc relativement important lors du processus de validation des signatures.

De plus, des informations relatives aux chemins de certification et éléments de vérification des signatures de CMS [HOU 99] peuvent être jointes aux signatures et horodatages.

Une possible mise en œuvre ASN.1 du format proposé, dérivée de [TRU 03], fait intervenir les structures de données de signature électronique et horodatage multiple :

```
MultiSignedContent ::= SEQUENCE {
  version             Version {v1(1)},
  tbsContent          SignedContent,
  envelopSigns    [0] ElectronicSigns       OPTIONAL,
  counterSigns    [1] ElectronicSigns       OPTIONAL,
  extns           [2] EXPLICIT Extensions   OPTIONAL
}

Version ::= INTEGER

SignedContent ::= SEQUENCE {
  data                Data,
  -- Can be a non-signed content or a MultiSignedContent
  ts                  Dates                 OPTIONAL,
  -- Must be a MultipleTimeStamp if data is of type
  -- MultiSignedContent
  extns               Extensions            OPTIONAL
}

Data ::= SEQUENCE OF ContentInfo

ContentInfo ::= SEQUENCE {
-- From RFC2315
  contentType         ContentType
  content         [0] EXPLICIT ANY
                  DEFINED BY contentType    OPTIONAL
}

ContentType ::= OBJECT IDENTIFIER
```

Le format du contenu original, noté « data », est exprimé de manière générique par une liste de contenus génériques « ContentInfo » [KAL 98b]. Il est cependant possible d'utiliser les travaux de [TRU 03] et [RIE 03a] pour définir un format davantage adapté aux différents types de contenus signés.

14.2. Second format

Une modification du format proposé précédemment permet de manipuler des signatures détachées. Les signatures détachées peuvent être extraites du contenu signé puis à nouveau insérées dans ce dernier sans que la cohérence de l'ensemble en soit affectée : les liens logiques entre le contenu et la hiérarchie des signatures est conservé.

Ce second format propose de gérer indépendamment le contenu et les signatures, comme l'indique la figure 50. L'apposition d'une sur-signature ou contre-signature s'effectue sur l'ensemble des signatures déjà apposées par emploi d'un procédé récursif.

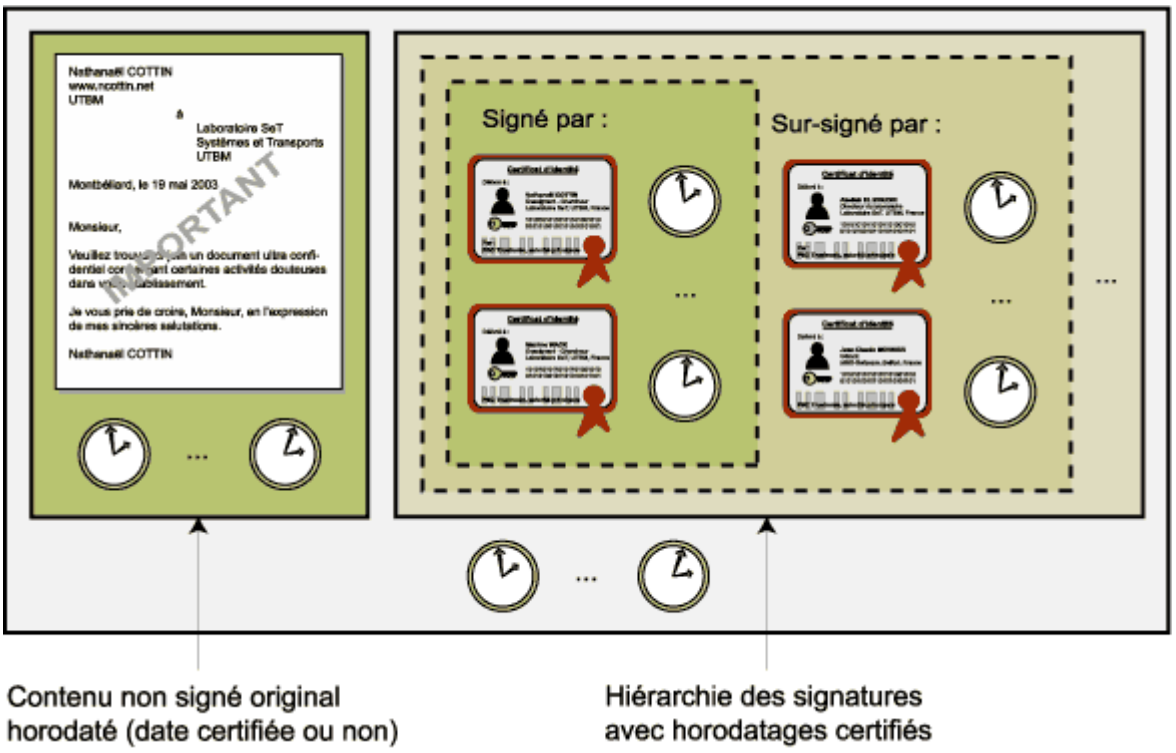

Figure 50 : second format de contenu multi-signé et horodaté

Les signatures conjointes apposées à un niveau donné peuvent être datées individuellement ou collectivement. Le second procédé apparaît plus simple à mettre en œuvre et permet de sceller l'ensemble des signatures du niveau concerné lorsqu'il est certifié par une autorité d'horodatage.

Le contenu signé est ici séparé de l'ensemble des signatures. Seul l'horodatage final associe définitivement le contenu et les signatures. Celui-ci n'empêche cependant pas le détachement des signatures. Ce dernier horodatage apporte en outre des garanties similaires à celles exposées précédemment.

Une description des structures de données mises en œuvre dans ce format aboutit aux définitions suivantes :

```
MultiSignedContent ::= SEQUENCE {
  version             Version {v1(1)},
  tbsContent          SignedContent,
  signatures      [0] DetachedSignatures    OPTIONAL
  extns           [1] EXPLICIT Extensions   OPTIONAL
}

Version ::= INTEGER

SignedContent ::= SEQUENCE {
  data                Data,
  -- Must be a non-signed content
  ts              [0] Dates                 OPTIONAL,
  -- Can be either certified or claimed
  extns           [1] EXPLICIT Extension    OPTIONAL
}

Data ::= SEQUENCE OF ContentInfo

DetachedSignatures ::= SEQUENCE {
  previousSigns   [0] DetachedSignatures    OPTIONAL,
  signs               Signs,
  ts              [1] Dates                 OPTIONAL,
  -- Should be certified and applied on "signs"
  extns           [2] EXPLICIT Extensions   OPTIONAL
}

Signs ::= SEQUENCE {
  envelopSigns    [0] ElectronicSigns       OPTIONAL,
  counterSigns    [1] ElectronicSigns       OPTIONAL,
}
```

Enfin, les informations relatives aux chemins de certification et éléments de vérification des signatures de CMS [HOU 99] peuvent être jointes aux signatures et horodatages (parmi les extensions de « DetachedSignatures » par exemple).

15. Quelques algorithmes de partage de secret

Le partage de secret [KUL 00] [ZIM 03] repose sur le concept de détention d'une portion d'une information secrète par plusieurs personnes, tel un coffre-fort bancaire dont l'ouverture est commandée par l'introduction simultanée de plusieurs clés.

Le partage de secret est traditionnellement utilisé en informatique pour scinder des clés de déchiffrement en plusieurs parties de sorte que chacune d'elles possède

une portion de la clé. Le partage de secret peut également s'appliquer à des fichiers plus volumineux afin de les disperser sur plusieurs sites.

15.1. Premiers algorithmes

Le cas le plus simple envisagé consiste à partager un secret M entre deux personnes A et B à l'aide d'une clé symétrique K. Le secret ainsi chiffré $C = crypt(M)_K$ est confié à A alors que K est donné à B.

Une seconde version offre la garantie supplémentaire que B n'est en mesure de créer un faux secret $C' = crypt(M')_K$ à partir d'une information M' qu'il a lui-même choisie. Au lieu d'utiliser une clé symétrique, une paire de clés $\{K_{priv}, K_{pub}\}$ est créée. Le secret M est alors chiffré à l'aide de K_{pub} pour donner $C = crypt(M)_{K_{pub}}$. Cette clé est ensuite détruite. Il revient à A de détenir le secret chiffré C et à B la clé de déchiffrement K_{priv}.

Une variante du premier algorithme consiste à générer un nombre aléatoire N et de remettre à A le secret $C = M \ xor \ N$ et de confier N à B. Le secret est obtenu en appliquant un second « OU exclusif » selon la relation $M = N \ xor \ (M \ xor \ N)$.

Le partage en un nombre quelconque de protagonistes est cependant difficile à gérer du fait du nombre important de clés ou de nombres aléatoires. Par exemple, le partage d'un secret entre n intervenants nécessite $n-1$ clés.

15.2. Algorithme de Shamir

Les algorithmes précédents nécessitent l'intervention de l'ensemble des protagonistes ayant reçu une partie du secret (ou de sa clé de déchiffrement). Chacun dispose alors d'un droit de veto ou ne peut être contacté pour apporter sa part de secret, ce qui peut être fâcheux (en cas de décès par exemple).

Le cryptologue Adi Shamir a proposé un algorithme reposant sur les propriétés de polynômes afin de partager un secret entre n intervenants de sorte que $0 < m \leq n$ parts permettent de reconstruire le secret : m est appelé « seuil ».

L'algorithme de Shamir [SHA 79] repose sur le postulat selon lequel un polynôme p de degré $d > 1$ représenté sous sa forme canonique par $p(x) = a_d \cdot x^d + a_{d-1} \cdot x^{d-1} + \ldots + a_0$ est parfaitement défini par $d+1$ éléments distincts. Par exemple, une droite $a \cdot x + a_0$ est connue dès lors que deux points lui appartenant sont donnés.

Les polynômes utilisés devront de plus passer par le point de coordonnées $(0, M)$ si M désigne la valeur du secret à partager.

Le seuil m présenté précédemment correspond alors à la valeur $d+1$, ce qui implique que le partage entre n intervenants avec un seuil de m nécessite l'utilisation d'un polynôme de degré $m-1$. Une fois ce polynôme déterminé, il reste à exprimer n points distincts (d'abscisse non nulle) et à les remettre aux différents protagonistes. Il suffit alors de rassembler m points pour recalculer le polynôme original est découvrir le secret M sachant que sa forme est $(0, M)$.

> *Il est également possible de hiérarchiser le partage du secret de telle sorte qu'une ou plusieurs parts soient nécessaires à l'obtention du secret. Cette hiérarchie des parts est réalisée en utilisant l'algorithme de partage de secret à plusieurs reprises.*

16. Proposition de format d'IOR

L'IOR ou « Interoperable Object Reference » permet à un objet serveur ou mixte d'informer les clients de sa localisation afin qu'ils puissent accéder à ses services (par le biais de son interface publique). Toute référence doit donc renfermer des informations structurées et notamment une adresse réseau et un numéro de port.

Nous suggérons ici un format d'IOR en langage ASN.1 :

```
IOR ::= SEQUENCE {
  version       INTEGER                 DEFAULT v1(1),
  hostName      VisibleString,
  hostIP        RELATIVE IDENTIFIER     OPTIONAL,
  port          INTEGER,
  key           OCTET STRING,
  extns         Extensions              OPTIONAL
}
```

Ce format inclue non seulement les informations mentionnées précédemment (champs « hostName » et « port ») mais également une clé sous la forme générique d'une chaîne d'octets. Cette clé permet par exemple au serveur de s'assurer que le client n'utilise pas une ancienne référence lors de ses requêtes.

De plus, ce format est évolutif du fait qu'il prend en charge des extensions dont le format correspond au format standard proposé par l'IETF au sein du certificat X.509.

Enfin, le nom utilisé lors de l'encodage en base 64 (format PEM pour la messagerie) de ce format peut être « INTEROPERABLE REFERENCE » ou « REFERENCE ».

17. Exemples de définitions S-IDL d'un PSCE

Les différents contrats IDL sécurisés conformes à la grammaire proposée sont présentés pour chaque entité intervenant dans l'architecture mise en œuvre par le PSCE.

Seuls les IDL des objets chargés des communications inter-autorités sont proposés. En particulier, les objets relatifs à la traçabilité et l'enregistrement des dossiers clients ne sont pas mentionnés.

Les interfaces IDL ci-après décrites utilisent une structure de données commune représentant les informations fournies par les demandeurs aux autorités locales d'enregistrement :

```
struct PrincipalInfo {
  in string commonName,
  in string organization,
  in string organizationUnit,
  in string postalAddress,
  in string emailAddress,
  in string ruleOID,
  in string state,
  in string locality,
  in string country
};

enum KeyType {
  RSA, DSA
};

typedef sequence<octet> Bytes;
```

Les erreurs manipulées par les objets répartis composant le PSCE sont regroupées dans le module suivant :

```
exception CertificateGenerationError(long errorCode);
exception PrincipalError(long errorCode);
exception UnavailableCertificateError(long certID);
exception ChainError(long certID) ;
exception UnavailableListError(long listID);
exception UnknownCertificateError(long certID);
exception UnavailableCertificateTrackError(
 long certID);
exception ListPublicationError(long errorCode);
```

17.1. Contrat S-IDL de l'autorité racine de certification

```
interface rootCA
  AUTHENTICATED {
    CLIENT  =>  ISSUER_ID IN [SN=24532657]
                             [O="CERT-SOFTWARE"]
  }
  SECURE {
    SSLv2, TLS
  }
{
  Bytes generateCertificate(
    in PrincipalInfo principal,
    in KeyType keyType,
    in long keySize,
    in string passPhrase)
  raises CertificateGenerationError;
}
```

17.2. Contrat S-IDL de l'autorité de certification principale

Au niveau de l'autorité de certification principale :

```
interface CA
  AUTHENTICATED {
    CLIENT  =>  ISSUER_ID IN [O="CERT-SOFTWARE",
                              R="RA_mainAgent"]
  }
  SECURE {
    SSLv2, TLS
  }
{
  int sendPrincipalInfo(
  // Returns the certificate id for retreival
    in PrincipalInfo principal,
    in KeyType keyType,
    in long keySize
    in string passPhrase)
  raises PrincipalError;

  Bytes getCertificate(in long certID)
  // Returns the certificate identified by certID
  raises CertificateGenerationError,
         UnavailableCertificateError,
         UnknownCertificateError;

  Bytes getRevocationList(in long listID)
  // Returns the CRL identified by listID
  raises UnavailableListError;
```

```
  Bytes getTrustedList(in long listID)
  // Returns the CTL identified by listID
  raises UnavailableListError;

  int getCertificateStatus(in long certID)
  // Returns the certificate status (for OCSP)
  raises UnavailableCertificateError,
         UnknownCertificateError;

  Bytes getCertificateChain(in long certID)
  // Returns the certificate chain of certID
  raises UnavailableCertificateError,
         ChainError,
         UnknownCertificateError;
}
```

17.3. Contrat S-IDL de l'AE principale

L'autorité d'enregistrement principale peut être décomposée en deux interfaces distinctes.

La première est utilisée pour réaliser les interactions avec l'AC et les ALE :

```
interface RA
  AUTHENTICATED {
    CLIENT  =>  ISSUER_ID IN [O="CERT-SOFTWARE",
                              R="LRA_mainAgent"]
  }
  SECURE {
    SSLv2, TLS
  }
{
  int sendPrincipalInfo(
  // Returns the certificate id for retreival
    in PrincipalInfo principal,
    in string passPhrase)
  raises PrincipalError;

  string getCertificateTrackInfo(in long certID)
  // Returns the certificate tracking info
  raises UnknownCertificateError,
         UnavailableCertificateTrackError;

  void publishRevocationList(in Bytes CRL)
    AUTHENTICATED {
       CLIENT  =>  ISSUER_ID IN [SN=1456734563]
                                [O="CERT-SOFTWARE"]
    }
```

```
  raises ListPublicationError;

  void publishTrustedList(in Bytes CTL)
    AUTHENTICATED {
       CLIENT  =>  ISSUER_ID IN [SN=1456734563]
                                [O="CERT-SOFTWARE"]
    }
  raises ListPublicationError;
}
```

La seconde interface, publique, permet à tout utilisateur de demander la liste des certificats révoqués (CRL) et la liste des certificats de confiance (CTL) :

```
interface PublicRA
 SECURE {
    SSLv2, TLS
  }
{
  Bytes getRevocationList()
  // Returns the current CRL
  raises UnavailableListError;

  Bytes getTrustedList()
  // Returns the current CTL
  raises UnavailableListError;
}
```

17.4. Contrat S-IDL des autorités locales d'enregistrement

Au niveau de chaque autorité locale d'enregistrement, le contrat S-IDL s'applique :

```
interface LRA
  AUTHENTICATED {
    CLIENT  =>  ISSUER_ID IN [O="CERT-SOFTWARE",
                              R="LRA_agent"]
  }
  SECURE {
    SSLv2, TLS
  }
{
  int sendPrincipalInfo(
  // Returns the certificate id for retreival
    in PrincipalInfo principal,
    in string passPhrase)
  raises PrincipalError;

  string getCertificateTrackInfo(in long certID)
```

```
  // Returns the certificate tracking info
  raises UnknownCertificateError,
         UnavailableCertificateTrackError;

  Bytes getCurrentRevocationList()
  raises ListPublicationError;
}
```

Index

Table des illustrations

Table des illustrations

Références bibliographiques

[ABE 97] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. Diffie, J. Gilmore, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, « The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption », rapport final, Mai 1997

[ACA 00] Projet ARCADE, « Architecture de Contrôle Adaptative des Environnements IP », consultable à l'adresse « www-rp.lip6.fr/arcade/ », 2000

[ADA 97] C. Adams, R. Zuccherato, « Notary Protocols », Internet Draft, disponible sur le site de l'IETF à « draft-adams-notary-01.txt », Février 1997

[ADA 99] C. Adams, D. Farrell, « RFC2510: Internet X.509 Public Key Infrastructure Certificate Management Protocols », Mars 1999

[ADA 01] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, « RFC3161: Internet X.509 Public Key Infrastructure: Time Stamp Protocol (TSP) », Août 2001

[ANS 01] A. Ansper, A. Buldas, M. Roos, J. Willemson, « Efficient long-term validation of digital signatures », Advances in Cryptology – PKC'01, Springer-Verlag, LNCS 1992, pp.402-415, Février 2001

[ARC 00] J.-L. Archimbaud, « Certificats (électroniques) : Pourquoi ? Comment ? », CNRS/UREC, 2000

[ARI 98] R. H. Arild, « Delegation of authority in a multi-domain distributed file repository », thèse de doctorat, université de Tromso, Norvège, Novembre 1998

[AUT 02] T. Autret, L. Bellefin, M.-L. Oble-Laffaire, « Sécuriser ses échanges électroniques avec une PKI – Solutions techniques et aspects juridiques », Eyrolles, 2002, ISBN 2-212-11045-6

[AZZ 03] S. Azzabi, « Signature électronique et droit de la preuve », thèse de droit privé, université Nancy II, à paraître

[BAL 00] H. Balen, « Distributed Object Architectures with CORBA », Cambridge University Press, 2000, ISBN 0-521-65418-1

[BOE 85] W. E. Boebert, R. Y. Kain, « A Practical Alternative to Hierachical Integrity Policies », proc. of the 8th National Computer Security Conference, Gaithersburg, Maryland, 1985

[BOE 94] B. den Boer, A. Bosselaers, « Collisions for the compression function of MD5 », Advances in Cryptology, LNCS 765, Springer Verlag, 1994

[BOE 99] S. Boeyen, T. Howes, P. Richard, « RFC2587: Internet X.509 Public Key Infrastructure LDAPv2 Schema », Juin 1999

[CAL 98] J. Callas, L. Donnerhacke, H. Finney, R. Thayer, « RFC2440: OpenPGP Message Format », Novembre 1998

[CAP 97] E. Caprioli, « Les tiers de confiance dans l'archivage électronique : une institution juridique en voie de formation », communication, Octobre 1997

[CHA 01] N. Chase, « XML et Java », CampusPress, 2001, ISBN 2-7440-1166-5

[CLA 01] D. E. Clarke, « SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI, thèse de doctorat, MIT, Septembre 2001

[CNU 01] Commission des Nations Unies, « Loi type de la CNUDCI sur les signatures électroniques », annexe II, 2001

[COT 00] N. Cottin, « DOMS : une architecture de suivi de la qualité de service dans les systèmes répartis objet », rapport de DEA, Juillet 2000

[COT 02a] N. Cottin, « Mise en œuvre de la signature électronique », 1er Colloque International sur les Multimédias, l'Archivage légal, la Dématérialisation de documents et la Signature électronique (CIMADS'02), Tunis, Mai 2002

[COT 02b] N. Cottin, B. Mignot, M. Wack, « Authentication and enterprise secured data storage », proc. of the 16th IEEE International Conference on Emerging Technologies on Factory Automation (ETFA'01), invited paper, Antibes Juan-les-Pins, France, Octobre 2002

[COT 03a] N. Cottin, M. Wack, A. Sehili, « Contribution à la validation des signatures électroniques dans le temps », proc. of the 2nd conférence francophone sur Sécurité et Architecture Réseaux (SAR'03), Nancy, France, Juin 2003

[COT 03b] N. Cottin, M. Wack, A. Sehili, « Time-stamping electronic documents and signatures », proc. of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03), Gammarth, Tunisie, Juillet 2003

[COT 03c] N. Cottin, « A long-term archival system: an open discussion », document interne, Mars 2003

[COT 03d] N. Cottin, « Etude technique de la gestion des attributs : problématiques et solutions », document interne, Mars 2003

[COT 03e] N. Cottin, « Quel format pour la signature électronique ? », document interne, Mai 2003

[COT 03f] N. Cottin, « La signature électronique dans les systèmes répartis : proposition de modèle », communication, 2nd Colloque International sur les Multimédias, l'Archivage légal, la Dématérialisation de documents et la Signature électronique (CIMADS'03), Yverdon-les-bains, Suisse, Juin 2003

[COT 03g] N. Cottin, « Proposal of certificate evolution for multiple public-keys support », à paraître

[DAN 00] J. Daniel, « Au cœur de CORBA avec Java », Vuilbert, Paris, 2000, ISBN 2-7117-8659-5

[DEL 00] J.-P. Delahaye, « Merveilleux nombres premiers, Voyage au cœur de l'arithmétique », Belin, Pour la Science, 2000, ISBN 2-84245-017-5

[DIE 99] T. Dierks, C. Allen, « RFC2246: The TLS Protocol Version 1.0 », Janvier 1999

[DIF 76] W. Diffie, M. E. Hellman, « New Directions in Cryptography », IEEE Transactions on Information Theory IT-22, pp. 644-654, 1976

[DOB 96] H. Dobbertin, A. Bosselaers, B. Preneel, « RIPEMD-160, a strengthened version of RIPEMD », Fast Software Encryption, LNCS vol. 1039, D. Gollmann Ed., pp. 71-82, 1996

[DUB 00] O. Dubuisson, « ASN.1: Communication between Heterogeneous Systems », Morgan Kaufmann Publishers, Juin 2000, ISBN 0-12-6333361-0

[EAS 94] D. Eastlake 3rd, S. Crocker, J. Schiller, « RFC 1750: Randomness Recommendations for Security », Décembre 1994

[EAS 02] D. Eastlake 3rd, J. Reagle, D. Solo, « RFC 3275: (Extensible Markup Language) XML-Signature Syntax and Processing », Mars 2002

[EES 99] European Electronic Signature Standardization Initiative (EESSI), « Final Report of the EESSI Expert Team », Juillet 1999

[ELL 99a] C. Ellison, « RFC2692: SPKI Requirements », Septembre 1999

[ELL 99b] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen, « RFC2693: SPKI Certificate Theory », Septembre 1999

[ELL 00] C. Ellison, B. Schneier, « Ten Risks of PKI: What You're not Being Told About Public Key Infrastructure », Computer Society Journal, vol.16, Novembre 2000

[ETS 01] European Telecommunications Standards Institute, ETSI TS 101 862 V1.2.1, « Qualified Certificate Profile », Juin 2001

[ETS 02a] European Telecommunications Standards Institute, ETSI TS 101 861 V1.2.1, « Time Stamping Profile », spécification technique, Mars 2002

[ETS 02b] European Telecommunications Standards Institute, ETSI TS 101 733 V1.4.0, « Electronic Signatures and Infrastructures (ESI); Electronic Signature Formats », spécification technique, Septembre 2002

[ETS 02c] European Telecommunications Standards Institute, ETSI TS 102 044 V1.1.1, « Electronic Signatures and Infrastructures (ESI); Requirements for role and attribute certificates », spécification technique, Décembre 2002

[FAR 02] S. Farrell, R. Housley, « RFC3281: An Internet Attribute Certificate Profile for Authorization », Avril 2002

[FLO 97] G. Florin, « Les techniques de cryptographie », présentation au CNAM, 1997

[FOU 99] P.-A. Fouque, G. Poupard, J. Stern, « Recovering Keys In Open Networks », proc. of the IEEE Information Theory Workshop (ITW'99), Kruger National Park, Afrique du Sud, Juin 1999

[FOW 01] M. Fowler, « UML », seconde édition, CampusPress, 2001, ISBN 2-7440-1090-1

[FOX 03] « Transférez vos archives familiales sur DVD », guide Fox, printemps/été 2003

[GAR 96] G. et O. Gardarin, « Le Client-Serveur », éditions Eyrolles, 1996, ISBN 2-212-08876-0

[GAS 89] M. Gasser, A. Goldstein, C. Kaufman, B. Lampson, « The Digital Distributed System Security Architecture », proc. of the 12th National Computer Security Conference, Octobre 1989

[GAS 90] M. Gasser, E. McDermott, « An Architecture for Practical Delegation in a Distributed System », proc. of the IEEE Symposium on Security and Privacy, Oakland, Mai 1990

[GEI 99] J. M. Geib, C. Gransart, P. Merle, « Corba, des concepts à la pratique, seconde édition », Dunod, Paris, Octobre 1999, ISBN 2-10-004806-6

[GON 90] L. Gong, « Cryptographic Protocols for Distributed Systems », thèse de doctorat, université de Cambridge, Avril 1990

[GRO 01] W. Grosso, « Java RMI », O'Reilly, Octobre 2001, ISBN 1-56592-452-5

[GTA 00] Groupe de Travail Commun Archivage, CSOEC, IALTA france, EDIFICAS, « Guide de l'archivage électronique sécurisé », Juillet 2000

[GTG 04] Groupe de Travail Commun Gestion des Attributs, CSOEC, IALTA france, EDIFICAS, « La qualité professionnelle dans la signature électronique », Février 2004

[GTH 03] Groupe de Travail Commun Horodatage, CSOEC, IALTA france, EDIFICAS, « Recommandations pour l'horodatage électronique », à paraître

[GUE 03] D. Guerin, « Service d'Horodatage », proc. of the 2nd conférence francophone sur Sécurité et Architecture Réseaux (SAR'03), Nancy, France, Juin 2003

[HOU 99] R. Housley, « RFC2630: Cryptographic Message Syntax – CMS », Juin 1999

[HOU 02] R. Housley, W. Polk, W. Ford, D. Solo, « RFC 3280: Internet X.509 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile », Avril 2002

[IAL 98] Comité IALTA, « Livre blanc : les nouveaux tiers de confiance impliqués dans les échanges électroniques », Novembre 1998

[ISO 88] International Standard Organization / Open System Interconnection, « Information processing system – basic reference model », ISO 7498 partie 2, 1988

[IMA 00] T. Imamura, H. Maruyama, « Mapping between ASN.1 and XML », RT0362, IBM Research Laboratory, Tokyo, Japan, 2000

[JAE 01] T. Jaeger, J. E. Tidswell, « Practical Safety in Flexible Access Control Models », communication de l'ACM Transactions on Information and System Security (TISSEC), vol. 4, Mai 2001, ISSN 1094-9224

[JAW 01] J. Jaworski, Paul J. Perrone, « Java Security », CampusPresss, 2001, ISBN 2-7440-1230-0

[JEN 96] R. J. Jenkins Jr, « ISAAC: a fast cryptographic random number generator », disponible à l'adresse « http://burtleburtle.net/bob/ »

[JUR 00] M. B. Juric, « CORBA, RMI and RMI-IIOP Performance Analysis and Optimization », proc. of the 4th World Multiconference on Systemics, Cybernetics and Informatics, Juillet 2000

[KAE 00] M. Kaeo, « Sécurité des réseaux », CampusPress, 2000, ISBN 2-7440-0850-8

[KAL 98a] B. S. Kaliski, « RFC2313: PKCS#1: RSA Encryption », Mars 1998

[KAL 98b] B. S. Kaliski, « RFC2315: PKCS#7: Cryptographic Message Syntax », version 1.5, Mars 1998

[KEL 03] S. Kelly, S. Ramamoorthi, « RFC3457: Requirements for IPSec Remote Access Scenarios », Janvier 2003

[KOH 78] L. M. Kohnfelder, « Towards a Practical Public-key Cryptosystem », thèse de doctorat, MIT, Mai 1978

[KRA 93] S. Krakowiak, « Systèmes Répartis : Evolution et Etat de l'Art », communication au laboratoire IMAG, Janvier 1993

[KUL 00] A. Kulkarni, H. Maheria, S. Pereira, « Secret Sharing Methods », présentation, 2000

[LAI 00] M. Lai, « Penser objet avec UML et Java », 2ème édition, Dunod, Paris, ISBN 2-10-005378-7

[LAM 92] B. Lampson, M. Abadi, M. Burrows, E. Wobber, « Authentication in distributed systems: Theory and Practice », communication de l'ACM Trans. Computer Systems 10, vol. 4, Novembre 1992

[LAN 01] S. Lanquetin, « Conception, modélisation et simulation d'un protocole de tiers certification », mémoire de DEA, Juin 2001

[LGI 02] Laboratoire LGI2P de l'EMA, « Projet I-CARE : Exemple du service de multisignature contrôlée », réf. I-CARE/EMA/LGI2P/DOC4/v0.2, Octobre 2002

[LIB 99] J. Liberty, « C++ – Ressources d'experts », CampusPress, 1999, ISBN 2-7440-0693-3

[LIP 99] H. Lipmaa, « Secure and efficient time-stmaping systems », thèse de doctorat, université de Tartu, Estonie, Avril 1999

[LOW 03] J. Löwy, « Programming .NET Components », O'Reilly, Avril 2003, ISBN 0-596-00347-1

[MAR 03] V. Marangozova, « Duplication et cohérence configurables dans les applications réparties à base de composants », thèse de doctorat, université Joseph Fourier, Grenoble I, Novembre 2003

[MBO 03] P. M. Mbow, « Modélisation et conception d'un système d'accès à un module de signature et d'archivage électroniques : application aux transports terrestres », Diplôme de Recherche Technologique, Université de Technologie de Belfort-Montbéliard, Octobre 2003

[MCC 96] B. W. McConnell, I. J. Appel et al., « Enabling Privacy, Commerce, Security and Public Safety in the Global Information Infrastructure », draft, Mai 1996

[MED 02] Medialsace, « Les systèmes de paiement sécurisé », article disponible à l'adresse « http://www.medialsace.fr/medialsace2/fr/commerce/securite.php »

[MEF 99] Ministère de l'Economie des Finances et de l'Industrie, « Politique de Certification-type », version 2.0, Décembre 1999

[MEL 01] H. X. Mel, D. Baker, « La cryptographie décryptée », CampusPress, 2001, ISBN 2-7440-1155-X

[MEN 01] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, « Handbook of Applied Cryptography », CRC Press, USA, 2001, ISBN 0-8493-8523-7

[MER 80] R. C. Merkle, « Protocols for public key cryptosystems », proc. of the IEEE Symposium on Security and Privacy (SSP'80), Oakland, Californie, 1980

[MIC 03] F. Michaut, « Adaptation des applications distribuées à la Qualité de Service fournie par le réseau de communication », thèse de doctorat, Université Henri Poincarré, Nancy I, Novembre 2003

[MIL 96] D. Mills, « RFC2030: Simple Network Time Protocol (SNTP) Version 4 for Ipv4, Ipv6 and OSI », Octobre 1996

[MIS 04] S. Aumont, « Les Infrastructures de Gestion de Clés: faut-il tempérer les enthousiasmes? », MISC 13, Mai-Juin 2004, ISSN 1631-9030

[MIT 98] J. C. Mitchell, V. Shmatikov, U. Stern, « Finite-State Analysis of SSL 3.0 », proc. of the 7th USENIX Security Symposium, pp. 201-215, 1998

[MOW 95] T.J. Mowbray, R. Zahavi, « The Essential CORBA: Systems Integration Using Distributed Objects », John Wiley & Sons, New York, NY, USA, 1995

[MUR 89] T. Murata, « Petri nets: Properties, analysis and applications », tutoriel invité, proc. of the IEEE, vol. 77, n°4, pp. 541-580, Avril 1989

[MYE 99] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, « RFC2560: X.509 Internet Public Key Infrastructure, Online Certificate Status Protocol – OCSP », Juin 1999

[MYE 01] M. Myers, R. Ankney, C. Adams, F. Farrell, « Online Certificate Status Protocol, version 2 », draft, Mars 2001

[NAG 97] N. Nagaratnam, « Secure Practical Delegation for Distributed Object Environments », thèse de doctorat, Université de Syracuse, 1997

[NAG 98] N. Nagaratnam, « Secure Delegation for Distributed Object Environments », 4ème USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Santa Fe, Nouveau Mexique, Avril 1998

[NAM 03] F. L. Nammour, N. Mansour, « Comparative Evaluation of Object Request Broker Technologies », proc. of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03), Gammarth, Tunisie, Juillet 2003

[NIS 95] National Institute of Standards and Technology, « Secure Hash Standard (SHS) », Federal Information Processing Standards Publication, FIPS PUB 180-1, Avril 1995

[NIS 96a] National Institute of Standards and Technology, « Key Recovery Demonstration Project », 1996, disponible à l'adresse « http://csrc.nist.gov/krdp/ »

[NIS 96b] National Institute of Standards and Technology, « Implementation Evaluation Criteria for the Key Recovery Demonstration Project », 1996, disponible à l'adresse « http://csrc.nist.gov/krdp/eadpic.html »

[NIS 00] National Institute of Standards and Technology, « Digital Signature Standard (DSS) », Federal Information Processing Standards Publication, FIPS PUB 186-2, Janvier 2000

[OEB 01a] M. Wack, B. Mignot, J.-C. Monnier, S. Gontard, N. Cottin, « Procédé d'apposition en ligne d'une pluralité de signatures numériques sur un document électronique », Office Européen des Brevets, Patentlaan 22280 HV Rijswiijk (ZH), 4 Otobre 2001, numéro de brevet : 01440334.9-2212

[OEB 01b] M. Wack, B. Mignot, J.-C. Monnier, S. Gontard, N. Cottin, « Procédé de dépôt d'un document numérique et de sa restitution », Office Européen des Brevets, Patentlaan 22280 HV Rijswiijk (ZH), 4 Octobre 2001, numéro de brevet : 01440333

[OEB 02a] J.-C. Monnier, N. Cottin, B. Hansz, B. Mignot, M. Wack, M. Souabni, « Support de données numériques », Office Européen des Brevets, Patentlaan 22280 HV Rijswiijk (ZH), 30 Janvier 2002, numéro de brevet : 02002306.5

[OEB 02b] J.-C. Monnier, N. Cottin, B. Hansz, B. Mignot, M. Wack, M. Souabni, « Etiquette électronique antivol », Office Européen des Brevets, Patentlaan 22280 HV Rijswiijk (ZH), 31 Mai 2002, numéro de brevet : 02012075.4

[OMG 97a] Object Management Group, « CORBAservices: Common Objet Services Specification », Juillet 1997

[OMG 97b] Object Management Group, « The Common Object Request Broker: Architecture and Specification », révision 2.1, Août 1997

[OMG 02] Object Management Group, « Security Service Specification », version 1.8, 2002

[ORF 95] R. Orfali, D. Harkey, J. Edwards, « Essential Distributed Objects Survival Guide », John Wiley and Sons Inc., New York, USA, 1995

[OSI 99] AFNOR, « Recommandations relatives à la conception et à l'exploitation de systèmes informatiques en vue d'assurer la conservation et l'intégrité des documents stockés dans ces systèmes », NF Z 42-013, Juillet 1999

[PAL 01] N. De Palma, « Services d'Administration D'applications Réparties », thèse de doctorat, université Joseph Fourier, Grenoble I, 2001

[PEC 99] Directive 1999/93/CE du Parlement Européen et du Conseil du 13 décembre 1999 sur un cadre communautaire pour les signatures électroniques

[PEL 00] M. C. Pellegrini, « Reconfiguration d'applications réparties : application au bus logiciel CORBA », thèse de doctorat, Institut National Polytechnique de Grenoble, Octobre 2000

[PIE 01] T. Piette-Coudol, « La signature électronique 2001 – Procédés techniques, valeur juridique, preuve », Juris Classeur Litec, Juin 2001, ISBN 2711133117

[PIE 02a] T. Piette-Coudol, « Conservation et archivage de l'écrit sous forme électronique », Juris Classeur, Communication & Commerce Electronique, Mai – Juin 2002

[PIE 02b] T. Piette-Coudol, « Classification des signatures électroniques et typologie des emplois », fascicule n°149, LAMY Droit de l'Informatique et des Réseaux, partie 1, Juillet 2002

[PIN 00] D. Pinkas, « Signature électronique équivalente à une signature signature manuscrite », communication, 2000

[PIN 01a] D. Pinkas, « PKI Disaster Planning and Recovery », informational RFC, disponible sur demande par courriel à « Denis.Pinkas@bull.net »

[PIN 01b] D. Pinkas, J. Ross, N. Pope, « RFC3126: Electronic Signature Formats for long term electronic signatures », Septembre 2001

[PIN 01c] D. Pinkas, « La signature électronique », communication, Septembre 2001

[PRE 97] B. Preneel, A. Bosselaers, H. Dobbertin, « The Cryptographic Hash Function RIPEMD-160 », The Technical Newsletter of RSA Laboratories, CryptoBytes, Automne 1997

[RAC 95] G. Rackl, « Load Distribution for CORBA Environments », thèse de doctorat, Technische Universität München, Janvier 1997

[RES 99] E. Rescorla, « RFC2631: Diffie-Hellman Key Agreement Method », Juin 1999

[RIE 03a] D. Rieupet, N. Cottin, « Scénarios d'apposition de multiples signatures », version 1.0, document technique, Janvier 2003

[RIE 03b] D. Rieupet, « Signatures électroniques multiples et hiérarchiques : proposition de formalisation et modélisation des processus associés », mémoire de Diplôme d'Etudes Approfondies, Septembre 2003

[RIV 92] R. L. Rivest, « RFC1321: The MD5 Message-Digest Algorithm », MIT Laboratory for Computer Science and RSA Data Security Inc., Avril 1992

[RIV 96] R. L. Rivest, B. Lampson, « SDSI - A Simple Distributed Security Infrastructure », Avril 1996

[RIV 98] R. L. Rivest, « Can we eliminate Certificate Revocation Lists? », proc. of Financial Cryptography (FC'98), volume 1465, p.178-183, LNCS, Springer-Verlag, Février 1998

[ROL 93] P. Rolin, « Réseaux locaux – normes et protocoles », 5ème édition revue et augmentée, Hermes, 1993, ISBN 2-86601-339-5

[ROO 99] M. Roos, « Integrating time-stamping and notarization », thèse de doctorat, université de Tartu, Estonie, 1999

[ROS 01] J. Ross, D. Pinkas, N. Pope, « RFC3125: Electronic Signature Policies », Septembre 2001

[RSA 93a] RSA Laboratories, « PKCS#3: Diffie-Hellman Key-Agreement Standard », Novembre 1993

[RSA 93b] RSA Laboratories, « PKCS#8: Private-Key Information Syntax Standard », version 1.2, Novembre 1993

[RSA 99] RSA Laboratories, « PKCS#5: Password-Based Cryptography Standard », version 2.0, Mars 1999

[RSA 00] RSA Laboratories, « PKCS#9: Selected Object Classes and Attribute Types », version 2.0, Février 2000

[SAN 88] R. S. Sandhu, « The schematic protection model: its definition and analysis for acyclic attenuating schemes », communication de l'ACM, vol. 35, Avril 1988, ISSN 0004-5411

[SAN 01] S. Santesson, W. Polk, P. Barzin, M. Nystrom, « RFC3039: Internet X.509 Public Key Infrastructure Qualified Certificates Profile », Janvier 2001

[SCH 01] B. Schneier, « Cryptographie appliquée », 2ème édition, Vuibert, 2001, ISBN 2-7117-8676-5

[SEN 00] Assemblée nationale, Sénat français, « Loi n°2000-230 du 13 mars 2000 portant adaptation du droit de la preuve aux technologies de l'information et relative à la signature électronique », Journal Officiel n°62 du 14 mars 2000, p.3968

[SEN 01a] Sénat français, « La signature électronique : note de synthèse », 2001

[SEN 01b] Sénat français, « Décret n°2001-272 du 30 mars 2001 pris pour l'application de l'article 1316-4 du code civil et relatif à la signature électronique », NOR : JUSC0120141D, Journal Officiel n°77 du 31 mars 2001, p.5070

[SEN 03] Sénat français, « Décret n°2003-659 du 18 juillet 2003 relatif aux obligations de facturation en matière de taxe sur la valeur ajoutée et modifiant l'annexe III au code général des impôts et la deuxième partie du livre des procédures fiscales », NOR : BUDF0300030D, J.O. n°166 du 20 juillet 2003, p.12272

[SHA 79] A. Shamir, « How to share a secret », communication de l'ACM, vol. 22, n°11, Novembre 1979

[SOL 88] K. R. Sollins, « Cascaded authentication », proc. of the IEEE Symposium on Research in Security and Privacy, Avril 1988

[STI 01] D. Stinson, « Cryptographie – Théorie et pratique », Vuibert, Paris, 2001, ISBN 2-7117-8675-7

[STU 91] S. Stuart, W. S. Stornetta, « How to Time-Stamp a Digital Document », Journal of Cryptology, vol. 3, 1991

[TOG 97] The Open Group, « DCE 1.2.2 Introduction to OSF DCE », The Open Group Publications Department, Novembre 1997, ISBN 1-85912-182-9

[TOG 99] The Open Group, « The COM/DCOM Reference », The Open Group Publications Department, 1999

[TRU 03] Trustronic, « TT TD S01-1.2e: EDCI: structures ASN.1 », version 1.2e, documentation technique, disponible à l'adresse www.trustronic.org, Janvier 2003

[VIL 02] J. Villemson, « Size-efficient interval time-stamps », thèse de doctorat, université de Tartu, Estonie, 2002

[WAC 02] M. Wack, B. Mignot, N. Cottin, A. Elmoudni, « Certification et Archivage Légal de Dossiers Numériques », revue Document Numérique, vol. 6, no 1-2/2002, Hermes Lavoisier, 2002, ISBN 2-7462-0532-7

[WOO 96] J. Woodcock, J. Davies, « Using Z – Specification, Refinement, and Proof », Prentice Hall, 1996, ISBN 0-13-948472-8

[ZIM 03] J. Zimmerman, « Le secret partagé », dossier cryptographie, Login: n°107, Juin 2003

Production scientifique personnelle

Les communications présentées dans les sections suivantes sont antérieures à la date de soutenance du doctorat. Pour connaître les dernières publications, prière de se référer à l'adresse « http://www.ncottin.net ».

Revues

« Certification et Archivage de Dossiers Numériques » (revue nationale) :

- M. Wack, B. Mignot, N. Cottin, A. Elmoudni
- Document Numérique, vol. 6, no 1-2/2002, Hermes Lavoisier
- 2002
- « La certification et l'archivage légal des données, alliée à la signature électronique des documents, ouvrent de nouvelles perspectives à la sécurisation des documents. Ainsi, ces technologies offrent des capacités : d'identification, d'authentification, de certification qui concourent à la capacité globale d'archivage sécurisé des dossiers numériques. Cependant, il apparaît que la certification et la signature électronique ne répondent pas complètement aux besoins des entreprises en ce qui concerne l'authentification et le stockage des données sécurisées. Dans la suite de cet article, nous proposons une solution à ces problèmes. »

« Cooperative Navigation in Multimedia Systems » (hors série LNCS) :

- M. Wack, N. Cottin, R. Bouyekhf
- Lecture Notes in Computer Science
- 2002
- « The emergence of the New Technologies of Information and Communication (NTIC), and the development of new tools open some perspectives for multimedia application design. In this paper we propose a graphical model of cooperative navigation of the multimedia applications. The model is based on the distinction between public and private areas. We use Petri nets to model several patterns which allow to build a complete navigation process. An example is worked out to illustrate the proposed approach. »

Conférences internationales avec comité de lecture

« Time-stamping electronic documents and signatures » :

- N. Cottin, M. Wack, A. Sehili
- Proc. of the ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03)
- Gammarth, Tunisia
- July 14-18, 2003
- « Time-stamping becomes a vital component of the emerging electronic business infrastructures. The main goal is to provide users with time management and - possibly signed- electronic content protection. We address in this article the purpose and usage of time-stamps on electronic documents and electronic signatures. A discussion is opened on the effective need of time-stamping electronic signatures through the study of two time-stamping technologies. »

« Contribution à la validation des signatures électroniques dans le temps » :

- N. Cottin, M. Wack, A. Sehili
- Proc. of the 2nd rencontre francophone sur Sécurité et Architecture Réseaux (SAR'03)
- Nancy, France
- June 30 - July 4, 2003
- « Les infrastructures à clé publique supportant la signature électronique proposent des services de gestion des certificats associés aux signataires. L'octroi du label « environnement sécurisé de signature » fait non seulement appel au système physique de stockage des informations de signature (clés, certificats) mais également aux applications faisant usage de ces informations ainsi qu'aux éventuelles connexions avec l'extérieur (LAN, WAN, Internet). Les applicatifs proposés aux clients doivent utiliser correctement les services de ces infrastructures afin d'authentifier et valider ces signatures. Nous nous intéressons à l'étude des principaux moyens de validation automatique des signatures électroniques et particulièrement la liste de révocation, le protocole OCSP, la signature à long terme ETSI et la notarisation électronique via le certificat de validité. »

« Authentication and Enterprise Secured Data Storage » :

- N. Cottin, B. Mignot, M. Wack
- Proc. of the 8th IEEE International Conference on Emerging Technologies on Factory Automation (ETFA'01)
- Antibes Juan-les-pins, France
- October 15-18, 2001
- « Data certification and digital signature are a new area of interest and many standards have emerged. Indeed, these technologies offer identification, authentication and non-repudiation capabilities during Internet transactions (emails and e-commerce). However, it appears that both certification and digital signature do not completely answer enterprise data authentication and secured data storage needs. We submit a proposition of an authorities-based architecture to answer these issues. This architecture relies on most of the available standards. »

« Distributed Object-Oriented Applications Supervision » :

- N. Cottin, J. Gaber, O. Baala, M. Wack
- Proc. of the 16th IEEE International Parallel and Distributed Processing Symposium (IPDPS'01)
- San Francisco, CA
- April 23-27, 2001
- « Distributed object-oriented computing allows efficient use of the Network Of Workstations (NOW) paradigm. However, the underlying middlewares used to develop and deploy such applications do not provide developers with any standard supervision mechanism so that they know exactly what happens during their applications execution. This paper analyzes distributed CORBA and JAVA-based applications to point out functional and management supervision information which has to be gathered from the objects. Developers will use this information to improve the Quality of Service (QoS) of their distributed object-oriented applications (DOA). Our framework is based on a CORBA service which uses our supervision API. This supervision service is composed of agents running on supervised computers. They interact with managed objects to perform local supervision measurements. These measurements may then be used to determine supervision indicators such as objects membership and computers exclusion degrees. »

« Management and QoS in Distributed Systems » :

- N. Cottin, O. Baala, J. Gaber, M. Wack
- Proc. of the IEEE International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'00), vol. III
- Las Vegas, NV
- June 26-29, 2000
- « Recent emerging of advanced networking technologies (IPv6, ATM, FastEthernet) together with the development of Internet technologies and distributed computing such as World Wide Web, Java and CORBA have the goal of offering heterogeneous services to users across various geographical and network boundaries. Consequently, many innovations in developing and adopting such emerged technologies are needed to guarantee the Quality of Service (QoS) in real-time distributed applications. An environment called Distributed Objects Management Protocol (DOMP) is defined to supervise and manage real-time distributed applications in order to improve their QoS. »

Mémoire de DEA

« DOMS : une architecture de suivi de la Qualité de Service dans les Systèmes Répartis Objet » :

- N. Cottin
- Février à Juillet 2000
- « L'objectif de ce travail est l'étude d'un moyen permettant d'améliorer la Qualité de Service (QoS), en termes de performance et fiabilité, des applications réparties basées sur l'objet et respectant la norme CORBA de l'OMG. Nous proposons ici un environnement de surveillance des applications réparties objet baptisé Distributed Objects Management System (DOMS). Basé sur le suivi et l'analyse des relations inter-objets, celui-ci fournit au développeur des outils de mesure de la QoS de ses applications. »