Problèmes de sécurité posés par les proxies d'adaptation multimédia : proposition de solutions pour une sécurisation de bout-en-bout

Ahmed Réda Kaced

Thèse

présentée pour obtenir le grade de docteur
de l'École Nationale Supérieure des
Télécommunications

Spécialité : **Informatique et Réseaux**

Ahmed Reda KACED

Problèmes de sécurité posés par les *proxies* d'adaptation multimédia : proposition de solutions pour une sécurisation de bout-en-bout

Soutenue le 09/06/2009 devant le jury composé de :

Samir Tohmé	Président
Julien Bourgeois	Rapporteurs
Pascal Lorenz	
Caroline Fontaine	Examinateurs
Pascal Urien	
Jean-Claude Moissinac	Directeur de thèse

Remerciements

Il était une fois une thèse, qui, m'avait-on dit, pouvait durer trois, voire quatre ans. La suite de l'histoire dépend un peu de moi, et beaucoup des autres, ou beaucoup de moi, et un peu des autres. Alors que les quelques 160 prochaines pages résument la partie qui vient de moi, je vais vous raconter ici la partie qui vient des autres.

Il y a tout d'abord Jean-Claude Moissinac, mon directeur de thèse, qui a accepté de me guider et de m'encourager dans ce parcours tout en me laissant une grande liberté, et que je veux remercier chaleureusement. La première fois qu'on a parlé de mes projets, c'était un 21 Mars 2004, lorsque je l'ai sollicité pour un stage de DEA, il a accepté de partager ses idées de recherche dans le domaine du multimédia et a su les adapter à mes projets en réseaux et sécurité ... Depuis, multimédia, réseaux et sécurité ont continué de faire bon ménage.

Je tiens à exprimer aussi mes remerciements aux membres du jury, qui ont accepté d'évaluer mon travail de thèse. Un grand merci donc au professeur Samir Tohmé pour avoir accepter de présider ce jury, à Julien Bourgeois et Pascal Lorenz qui m'ont fait l'honneur d'accepter de rapporter cette thèse. Mes chaleureux remerciements vont également à Caroline Fontaine et Pascal Urien pour avoir accepter de participer à ce jury.

Ensuite, l'histoire de ma thèse doit beaucoup aux membres de l'équipe MMA, et des laboratoire INFRES et TSI de l'ENST-Paris, Cyril, Jean, Maria, Mariam, etc. En effet, l'histoire n'aurait pas été la même sans leurs appréciations exigeantes (mais constructives !) et les conditions de travail exceptionnelles. De plus, les discussions fructueuses au sein d'un groupe de chercheurs, d'enseignants et de doctorants travaillant sur des thèmes liés au Multimédia ont largement contribué à me motiver dans mes travaux et à me guider dans mes recherches. J'ai donc beaucoup appris à leurs cotés, et je garderai de ces trois ans passés au 5éme étage de Dareau à l'ESNT le sentiment d'une ambiance de travail stimulante et sérieuse.

Une pensée particulière envers les membres du BDT et de la CJC, Loïs, Valentin, Nancy, Florent, etc. avec qui j'ai partagé des moments agréables et amicaux tout en découvrant le milieu associatif.

Mes prochains remerciements ont une portée de 1372 Km. Ils s'adressent à mes parents que j'aime tant, et qui m'ont soutenu durant tout ce parcours, ainsi qu'à tous mes huit frères er sœurs. sans eux, et sans le bonheur de notre famille nombreuse je ne serai jamais arrivé où j'en suis.

Je n'oublierai pas aussi mes amis qui ont fait le même parcours RedHa, Fahd, Rym, Nabil, Assia, Nesrine, Midou, Badro, Amine, et tous les autres de la même promo de l'USTHB.

Et pour garder le meilleur pour la fin, je souhaiterai adresser un grand merci à Florie-Anne la femme de ma vie qui n'a cessé de me soutenir et de me booster pour faire aboutir cette thèse. J'espère pourvoir t'apporter autant que tu m'apporte durant tout le restant de notre vie.

Résumé

Titre

Problèmes de sécurité posés par les *proxies* d'adaptation multimédia : proposition de solutions pour une sécurisation de bout-en-bout

Mots clés

Signatures numériques, adaptation de contenu, sécurité multimédia, proxy

Résumé

L'évolution des techniques d'adaptation et des contenus multimédias adaptables a montré la nécessité de définir des techniques et des pratiques concernant la sécurité des échanges sur les réseaux. Dans la mesure où l'adaptation des documents multimédia nécessite d'autoriser la modification de ces documents entre le serveur et le client, il est important d'étudier les conditions nécessaires pour assurer ces modifications de façon sécurisée.

Nous avons donc, dans ce cadre, à présenter un système de communication multimédia qui préserve l'authenticité et l'intégrité des contenus originaux de bout en bout tout en permettant l'adaptation de ces contenus par des intermédiaires. C'est l'objectif général de cette thèse.

Dans ce mémoire, nous présentons SEMAFOR, une plate-forme de communication multimédia offrant aux utilisateurs la possibilité d'adapter leur contenus (selon les besoins) par des nœuds intermédiaires. La particularité de cette plate-forme est sa capacité de sécuriser le contenu émis de bout-en-bout, cette sécurisation repose sur deux mécanismes proposés et décrits dans cette thèse : AMCA pour l'authentification des contenu et XSST pour le chiffrement et rechiffrement intermédiaire.

Les tests et les mesures de performances présentés à la fin de ce mémoire démontrent la validité des propositions décrites dans cette thèse et valident la pertinence des résultats obtenus.

Abstract

Title

Security issue in multimedia adaptation proxies : Solutions proposal for an end-to-end security

Keywords

Digital signatures, content adaptation, multimedia security, proxy

Summary

Due to the growing number of exotic networks, hardware and software and the increased demand for using multimedia services comes a tremendous need for providing adaptation techniques that end to deliver an understandable content to heterogeneous clients. These last are, generally, characterized by particular constraints and limitations which make the use of rich content difficult, in addition to the unsupported functionalities by the environment.

An adaptation method transforms the content from a state to another in order to meet the constraints of the client context. In some situations, several adaptations are applied on the original content to reach that goal, on the level system (ex : choice of flows), on the level of the organization of the data (ex : choice of level in a coding scalable), on the level of the modification of flows themselves (transcoding).

Offering an adaptable content for users showed the need for defining techniques and practices concerning the security of the exchanges on the networks. Insofar as the multimedia adaptation of the documents requires authorizing the modification of these documents between the Servers and the end-users, it is necessary to study the necessary conditions to ensure these modifications in a protected way. We have to present a multimedia content delivery system that preserves the end-to-end authenticity of original content while allowing content adaptation by intermediaries. It is the general aims of this thesis.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Ces dernières années ont vu l'apparition d'une multitude de nouveaux formats de document permettant à un utilisateur d'accéder et de consulter l'information qui l'intéresse. Avec l'avènement du *World Wide Web*, l'information n'est plus restreinte à un format textuel, mais peut être représentée selon différents formats numériques : audio, vidéo, voir une combinaison de ces technologies. Plus récemment, ce sont les supports physiques à partir desquels l'information est consultée qui ont fait leur révolution. Désormais on peut visionner des films sur son téléphone cellulaire, gérer son budget avec un assistant personnel (PDA), et bientôt envoyer des mails à partir de sa montre. Le défi des systèmes traitant l'information est de faire face à cette diversité croissante des modes d'accès, qui constituent le profil ou le contexte utilisateur, en proposant des solutions qui répondent aux besoins de chacun de ces contextes. Une des réponses pour ce défi est le déploiement de mécanismes d'adaptation de contenus pour les différents profils des récepteurs.

Dans le contexte des plates-formes de communication multimédia, il est inconcevable de prévoir une version d'un document multimédia pour chaque type de périphérique. La solution adoptée dans ce cas là est la mise en œuvre d'intermédiaires (*proxies*) d'adaptation qui, à partir d'un même flux original, peuvent fournir des versions adaptées aux récepteurs selon le profil de chacun. Cependant, ce type de solution se trouve aujourd'hui confronté à la conséquence même de son efficacité et qui est : «l'exposition de ces contenus adaptables aux risques d'attaques ou de piratage lors des opérations d'adaptation».

En effet, d'un coté, ces contenus doivent autoriser les opérateurs d'adaptation à effectuer les changements nécessaires sur le document avant la diffusion. Et d'un autre coté, les propriétaires et créateurs de contenu veulent protéger leurs produits de la dégradation résultant des opérations d'adaptation, de redistribution et du piratage aisés, alors que les clients finaux souhaitent protéger leurs informations personnelles et assurer que leurs transactions soient sécurisées.

Les méthodes de signature numérique et de chiffrement « classiques » ne permettent pas de telles opérations. En même temps, l'absence de ce type de sécurisation rend les documents originaux vulnérables et attaquables. Des solutions concernant des contenus à média unique (vidéo, image) ont été proposées [102, 4, 62], mais qu'en est-il lorsque ces contenus sont composites alliant textes, images, vidéos, etc. dans une même présentation ?

Motivations

L'histoire de cette thèse a commencé il y a un peu plus de quatre ans. Elle a fait suite à un stage de DEA qui avait porté sur la diffusion et l'adaptation de flux multimédias sur un réseau hétérogène, ce stage avait abouti à la réalisation d'une solution de diffusion de documents multimédias sur une plate-forme ad-hoc/filaire utilisant des nœuds intermédiaires (appelés plus communément dans la littérature par l'anglicisme *proxies*) pour le routage et l'adaptation.

Nonobstant l'intérêt de l'adaptation au niveau des *proxies*, nous avions relevé une problématique très pertinente relative à la sécurité. Nous avions remarqué la perte de toutes les protections sur les contenus au moment des opérations d'adaptation, le risque de piratage au niveau des *proxies* et la difficulté de garder secrètes des informations transitant par les intermédiaires d'adaptation.

Dans ce contexte et à partir des réflexions sur les dimensions sécurité des services d'adaptation, gestion des droits numériques et authentification des flux échangés, est née cette thèse. Nous nous sommes donc lancés dans la recherche de solutions pouvant préserver les dispositifs de sécurité mis en œuvre tout en autorisant les opérateurs d'adaptation à accomplir leur tâches.

Cette thèse a duré quatre années et a connu différentes orientations des travaux avant d'aboutir aux résultats escomptés ; ce sont ces résultats que nous allons tenter d'exposer dans les différents chapitres de ce mémoire. Nous garderons comme fil conducteur l'évolution dans le temps des travaux réalisés. Mais avant cela, nous continuons dans cette introduction par présenter brièvement les objectifs de cette thèse, énumérer les contributions majeures de ces quelques années de labeur, puis détailler le plan de la suite de ce document.

Objectifs

Le défi principal de cette thèse est de fournir une solution qui puisse garantir un niveau de sécurité assez élevé lors de l'adaptation de documents multimédias sur le réseau, tout en analysant les degrés de confiance nécessaires dans les intermédiaires d'adaptation. Par ailleurs, les objectifs généraux que nous nous sommes fixés au début des travaux de cette thèse étaient de :

(i) proposer de nouvelles solutions de protection de contenus multimédias «adaptables» ;

(ii) réaliser une plate-forme de diffusion multimédia, permettant une sécurisation de bout-en-bout des flux échangés tout en autorisant des opérations d'adaptation sur des nœuds intermédiaires.

Ces objectifs se situent clairement dans un contexte applicatif, par conséquent, toute proposition théorique pour y répondre doit être validée à travers la réalisation de prototype. Deux types de résultats sont donc visés par cette thèse :

- des résultats théoriques portant sur la spécification d'un modèle de documents multimédias génériques, la spécification d'une technique d'authentification de contenus multimédias adaptables et la proposition d'une architecture complète pour un système de diffusion de contenus adaptables ;
- des résultats pratiques constitués de l'implémentation d'une plate-forme sécurisée pour l'édition et la présentation de documents multimédias adaptables aux contextes des destinataires, ainsi que d'un *proxy* fournissant un service d'adaptation de contenu permettant la sécurisation de telles présentations.

Afin d'atteindre ces résultats nous avons abordé cette thèse selon le plan de travail suivant :

- durant les premières étapes des travaux nous avons effectué plusieurs états de l'art sur :
  - ▷ les différents formats multimédias,
  - ▷ leurs capacités d'adaptation et les techniques utilisées pour cette adaptation,
  - ▷ les concepts de base de la sécurité dans le domaine du multimédia,
  - ▷ les différents type d'attaque possibles sur les flux multimédias adaptables ;
- dans la deuxième phase de cette thèse, et suite aux conclusions tirées des différentes études effectuées, nous avons proposé deux mécanismes de sécurité (d'authentification et de chiffrement) autorisant l'adaptation de documents multimédias. Nous avons aussi implémenté des prototypes de ces mécanismes que nous

avons appelé respectivement AMCA et XSST ;
- nous avons ensuite procédé à la réalisation d'une plate-forme de diffusion complète que nous avons baptisé SEMAFOR, celle-ci utilise les services d'AMCA et d'XSST les raffinant avec des mécanismes de communication dans une plate-forme pair-à-pair ;
- la fin des travaux de cette thèse a été focalisée sur l'évaluation des performances des différentes approches proposées et la publication des résultats obtenus.

Contexte et Contributions

Les travaux de thèse ci-présents se sont déroulés au sein de l'équipe « MultiMédiA », dans le département « INFormatique et RESeaux » puis « Traitement du Signal et des Images » de «TELECOM ParisTech». Les domaines de compétences de cette équipe comprennent la représentation, la compression et plus généralement le cycle de vie des différents types de média (graphiques, image, vidéo) ainsi que l'étude des mécanismes de présentation de ces médias de manière interactive tout en préservant une bonne synchronisation. Plusieurs axes de recherche sont actuellement explorés :

- dessins animés : création, traduction, *streaming*, résistance aux erreurs, optimisation, compression, scalabilité ;
- évolutions des formats multimédia, normalisation, convergence de MPEG/W3C ;
- outils auteur pour simulateurs, pour services interactifs (mobiles, TV) ;
- compression vidéo scalable (MPEG-SVC), compression vidéo robuste aux erreurs, codage conjoint source-canal ;
- adaptation automatique et protection de contenus.

Les contributions apportées par ce travail de recherche s'insèrent notamment au sein de ce dernier thème, adaptation automatique et protection de contenus. Ils concernent plus précisément la protection de contenus lors des opérations d'adaptation par des nœuds intermédiaires sur le réseau. Les solutions proposées reposent sur les principes de signature/chiffrement des documents multimédia. En présence d'opérateurs d'adaptation, ces contenus gardent la signature de leur propriétaire tout en autorisant des modifications éventuelles.

Le travail que nous présentons dans cette thèse aborde les aspects sécurisation des documents multimédias adaptables ; pour les autres thématiques, le lecteur peut se référer à [50] pour la gestion du cache au niveau *proxy* et à [51] pour la signature au niveau *proxy*.

Nous tenons aussi à signaler que les contributions décrites dans cette thèse sont nos contributions strictement personnelles. Pour quelques autres contributions connexes de notre équipe le lecteur peut se référer à [95] pour des travaux sur les contenus scalables et à [6, 52] pour des travaux sur l'adaptation multimédia.

Plus concrètement, les contributions majeures de cette thèse sont les composants suivants :

AMCA une solution pour la signature de documents multimédias adaptables qui autorise les opérateurs d'adaptation à ajouter ou supprimer des éléments sans perdre l'intégrité et l'authenticité de ces documents. Cette solution utilise le principe des arbres de hachage de Merkle (AHM) ;

XSST une approche pour le chiffrement des contenus multimédias échangés entre les émetteurs et les récepteurs, facilitant l'intervention des *proxies* d'adaptation ;

SEMAFOR une plate-forme de diffusion de contenus multimédias adaptables, mettant en œuvre les solutions AMCA et XSST pour sécuriser les échanges de documents multimédias. SEMAFOR a permis la validation des différentes approches de sécurité proposées.

Par ailleurs, notre séjour à TELECOM ParisTech nous a permis de contribuer à deux projets importants, le projet RNRT SAFARI [1] et le projet IST TIRAMISU [2]. Une grande partie des travaux de cette thèse a été financée par ce dernier.

Organisation du manuscrit

Ce document résume ces années de travail sur la problématique précédemment mentionnée. Nous avons voulu le présenter selon l'évolution des travaux dans le temps, il est ainsi découpé en deux parties et six chapitres.

Dans la première partie nous présentons le contexte technique de nos travaux, la problématique de la sécurité rencontrée ainsi que certains états de l'arts sur des travaux connexes :

le chapitre 1 présente dans sa première partie les documents multimédias adaptables et la notion de l'adaptation multimédia. Cette description est dévelop-

1. https ://safari-rnrt.rd.francetelecom.com/
2. http ://www.tiramisu-project.org/

pée par l'étude de quelques techniques d'adaptation parmi les plus utilisées. La seconde partie de ce chapitre décrit les avantages de l'adaptation sur des nœuds intermédiaires en la comparant à d'autres approches possibles.

le chapitre 2 constitue une introduction aux concepts de base de la sécurité dans le multimédia, nous y présentons les différentes techniques de sécurité relatives aux documents multimédia et les différents critères de sécurité visés par nos recherches.

le chapitre 3 fournit un état de l'art sur les travaux existants relatifs à la sécurisation des opérations d'adaptation effectuées par des intermédiaires. Cette étude nous permet d'appréhender l'objet même de notre travail et de comprendre quels sont les critères qui peuvent caractériser une solution.

La deuxième partie de cette thèse présente nos contributions :

le chapitre 4 est consacré à la présentation de AMCA et XSST. AMCA étant notre approche pour la signature de documents multimédias adaptables autorisant des opérations d'adaptation sur le contenu original. Dans ce chapitre, nous décrivons les techniques et les algorithmes proposés pour la réalisation d'AMCA. Nous y décrivons aussi deux scénarios d'utilisation et déroulons nos algorithmes sur ces scénarios. XSST est une proposition de schéma de chiffrement robuste aux attaques dans une topologie à base de *proxies* d'adaptation.

le chapitre 5 expose notre chaîne complète de sécurisation lors de la diffusion de documents multimédias adaptables. Ce processus est présenté à travers la description de notre plate-forme SEMAFOR qui implémente les solutions proposées et valide les objectifs fixés pour notre travail. La fin de ce chapitre donne quelques détails d'implémentation de notre plate-forme ainsi que les outils utilisés.

le chapitre 6 montre des résultats expérimentaux permettant de valider les solutions proposées. Nous commençons par exhiber nos architectures de tests et les scénarios réalisés, puis nous étalerons les performances de l'utilisation de SEMAFOR en analysant chacun des résultats obtenus.

Nous finissons par une conclusion générale en rappelant les objectifs et les contributions de notre travail. Ainsi que quelques orientations possibles comme perspectives aux résultats obtenus.

Quelques publications relatives

Kaced, A R., Moissinac, J-C. *Sécurisation des flux multimédia adaptables - Proposition d'un schéma de signature sur des proxies*, in Proceedings of the 2nd French-speaking conference on Mobility and ubiquity computing UbiMob'05, Grenoble, France, Juin 2005.

Kaced, A R., Moissinac, J-C. *Signature de document multimédia adaptable par des proxies - application aux flux XML*, 2èmes rencontres des Sciences et Technologies de l'Information, Clérment-Ferrand, France, Octobre 2005.

Kaced, A R., Moissinac, J-C. *La sécurité, problème majeur pour les plates-formes de diffusion de flux multimédia adaptables*, 4ème édition du Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC), Rennes, France, Mai 2006.

Kaced, A R., Moissinac, J-C. *Protecting adaptive multimedia delivery and adaptation using proxy based approach.* In Proceedings of the International Conference on Security and Cryptography SECRYPT-2006, Setubal, Portugal, Août 2006.

Kaced, A R., Moissinac, J-C., *Multimedia content authentication for proxy-side Adaptation*, In Proceedings of the IEEE International Conference on Digital Telecommunications ICDT, Saint Raphael, France, Août 2006.

Kaced, A R., Moissinac, J-C., *SEMAFOR : a framework for authentication of adaptive multimedia content and delivery for heterogeneous networks*, In Proceedings of the IEEE International Conference on Internet Surveillance and Protection ICISP 2006, Saint Raphael, France, Septembre 2006.

Kaced, A R., Moissinac, J-C. *Sécurité dans les plates-formes de diffusion de flux multimédia adaptables*, in Proceedings of the 3rd French-speaking conference on Mobility and ubiquity computing UbiMob'06, Paris, France, Septembre 2006.

Kaced, A R., Moissinac, J-C. *Secure Intermediary Caching in Mobile Wireless Networks Using Asymmetric Cipher Sequences Based Encryption*, in Proceedings of the 3rd International Conference on Mobile Ad-hoc and Sensor Networks MSN 2007, Beijing, Chine, Décembre 2007

Première partie

État de l'art

Chapitre 1

Adaptation de contenu dans les systèmes multimédias

L'objectif de ce premier chapitre est d'introduire le contexte technique de la problématique décrite dans l'introduction générale. Il est découpé en quatre sections principales. Nous commencerons dans la première section par préciser les orientations de nos travaux et les types de contenu et de communication concernés en présentant un état de l'art sur les formats multimédias que nous traitons. La deuxième section est un état de l'art sur l'adaptabilité de ces flux multimédias, nous y définissons la notion de «document multimédia adaptable» avant de présenter quelques techniques d'adaptation et les caractéristiques liées aux opérations d'adaptation. La troisième section décrit les spécificités des opérateurs d'adaptation intermédiaires ainsi qu'un état de l'art sur des solutions fondées sur ce principe. Dans la dernière section, nous exposerons les risques liés à ce type d'adaptation en terme de sécurité afin de mieux analyser la problématique que nous traitons dans cette thèse.

1.1 Communications multimédias

Les communications multimédias sont le résultat du rapprochement inexorable entre l'informatique, les télécommunications (réseaux hauts-débits, téléphonie...) et l'audiovisuel (radio, télévision, cinéma...), les technologies les plus performantes et la conception/production de contenus les plus innovants. La notion même de multimédia s'est transformée, passant du sens restrictif d'objets numériques de type vidéo ou audio à celui plus global d'intégration sur un même flux d'éléments numériques de nature différente (texte, image, son, vidéo...), et de consultation interactive, voir

de production, stockage et diffusion de contenus avec des outils, sur du matériel, au travers de réseaux et sur des supports de plus en plus variés et complémentaires. Nous allons donc, avant d'aller plus loin, définir clairement quels sont les domaines du multimédia auxquels nous nous sommes intéressés dans nos travaux en donnant notre définition du document multimédia, et quels sont les types de communication que nous traitons dans cette thèse.

1.1.1 Document multimédia

La notion de document multimédia est bien connue de nos jours et les premières définitions datent du début des années 80 [33]. De façon informelle, nous définissons un document multimédia comme suit :

Définition 1 *Un document multimédia est caractérisé par l'intégration dans une même présentation d'informations exprimées dans divers médias (son, image, vidéo, texte, etc.). Cette présentation comporte une composante spatiale, hypertexte éventuellement, mais aussi temporelle.*

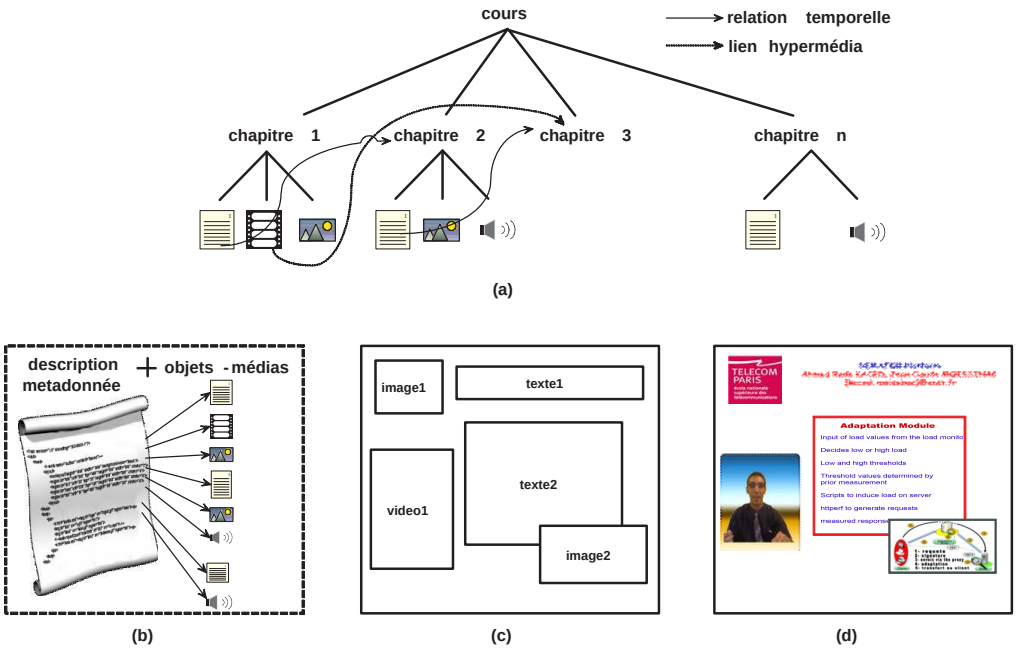


Figure 1.1 – Exemple de structure d'un document multimédia (a) Représentation arborescente (b) Décomposition en objets-médias (c) Organisation spatiale (d) Aperçu

Un document multimédia est ainsi composé de plusieurs médias distincts et d'une description, textuelle ou binaire, (métadonnées) décrivant l'interaction entre ces médias, où chacun des médias graphiques y est représenté par un rectangle (emplacement) défini à l'aide d'un point d'origine (le point supérieur gauche par exemple), d'une hauteur ainsi que d'une largeur. Un exemple de document multimédia est donné par la figure 1.1. Ce dernier, comme nous le voyons, est une présentation composite constituée de plusieurs images, vidéos, fichiers textes, etc. que nous appellerons **objets-médias**.

Définition 2 *Un objet-média est le plus petit composant d'un document multimédia. Il représente un élément média tel qu'une image, une vidéo, un texte, etc.*

Dans la suite de ce manuscrit, nous emploierons indifféremment les termes de **flux multimédia**, **contenu multimédia** ou **présentation multimédia** pour désigner un document multimédia. Par ailleurs, nous soulignons le vocabulaire utilisé pour désigner les composants d'un document multimédia structuré à travers l'exemple illustré par la figure 1.1, l'exemple proposé est une présentation d'un cours de plusieurs chapitres composés de différents objets-médias comme un logo (image), des zones de texte, des vidéos ainsi que des enregistrements audio. Le scénario du déroulement de la présentation de ces objets-médias est décrit dans une description métadonnée.

Dans cette thèse nous nous intéressons donc à ce format de documents multimédias composites. Plusieurs standards de documents multimédias existent dans la littérature ; le paragraphe suivant en décrit quelques uns sur lesquels ont porté nos travaux.

1.1.2 Standards multimédias ou apparentés

Plusieurs efforts de standardisation ont eu lieu dans le domaine du multimédia durant les années 90 sans réel succès. Cela dit, durant cette dernière décennie, la tendance semble s'inverser avec l'accroissement des performances des terminaux et l'apparition de plusieurs nouveaux standards. Un état de l'art sur les différents formats multimédias existants nous a permis de dégager deux grandes familles de standards, les formats utilisant des descriptions métadonnées textuelles et les formats binaires.

Dans ce qui suit, une présentation rapide de quelques uns de ces standards de flux multimédias. D'autres formats moins utilisés ne sont pas développés ici tels que X3D [19], MADEUS [49], ou encore la famille MHEG 1 à 7 [69].

1.1.2.1 SMIL

Le langage SMIL [1] [99] est un format standardisé proposé par le consortium W3C, il permet de décrire sous forme textuelle une présentation multimédia interactive. SMIL est un langage déclaratif, utilisant la syntaxe XML, qui permet aux auteurs de documents de spécifier ce qui doit être présenté, à quel endroit et à quel moment. Les auteurs peuvent ainsi définir des présentations multimédias où l'audio, la vidéo, les textes et les graphiques seront combinés en temps réel.

Il est à noter que SMIL ne définit pas de nouveaux codages de sources (audio, vidéo, image,...) mais se contente de proposer des mécanismes de synchronisation entre des sources déjà existantes.

1.1.2.2 SVG

SVG [2] [98] est aussi un format standardisé proposé par le consortium W3C décrivant des graphismes vectoriels bidimensionnels. SVG est un langage utilisant la syntaxe XML, ce qui lui confère une souplesse et une clarté d'utilisation très forte.

La description métadonnée SVG intègre des objets-médias tels que des formes géométriques de base (rectangles, ellipses, etc.), mais aussi des chemins (*paths*, qui utilisent les courbes de Bézier permettant d'obtenir presque n'importe quelle forme). Le format SVG permet aussi l'intégration d'animations, ou la manipulation des objets-médias grâce à des scripts qui peuvent être intégrés dans le SVG. Par ailleurs, comme SMIL, SVG gère aussi des objets-médias externes (ressources textuelles, dessins, images, son, etc.).

1.1.2.3 MPEG-4

MPEG-4 [1] a été standardisé par le consortium MPEG en 1998 mais le développement de la norme continue encore aujourd'hui. MPEG-4 contrairement aux standards MPEG 1 et 2 est orienté objet, cela signifie qu'il devient possible d'intégrer des objets-médias très divers pour constituer la scène que l'on souhaite transmettre. L'idée est de diviser une scène vidéo en objets et de décrire les relations entre ces objets. MPEG-4 définit un langage description de scène, BIFS [3], un format binaire

1. Synchronized Multimedia Integration Language
2. Scalable Vector Graphics
3. BInary Format for Scenes

fondé sur le standard de description de mondes 3D VRML 97, il permet de définir le comportement des objets-médias en fonction des commandes de l'utilisateur pour permettre une interactivité.

BIFS est un format binaire. Pour créer une scène MPEG-4, la scène peut d'abord être décrite sous forme textuelle. Une description textuelle peut se présenter sous plusieurs formes :

- Soit dans un des langages XML standardisés par MPEG, appelé XMT[4]. Il en existe plus exactement 2 versions : XMT-A ou XMT-Ω. XMT-A est l'exacte transcription du binaire BIFS en XML. XMT-Ω est une autre version du langage XMT de plus haut niveau très proche de SMIL.
- Soit dans une forme, non standard, XML ou non, comme le format BT (BIFS Text), qui utilise la syntaxe VRML.

Des encodeurs peuvent ensuite traduire ces descriptions de scènes textuelles en descriptions binaires.

1.1.2.4 FLASH

Flash [44] est un format fermé proposé par *Macromedia* en 1996 puis racheté par *Adobe* par la suite. Le contenu graphique des fichiers Flash est prévu pour être principalement vectoriel, à la façon du format SVG, et peut intégrer aussi des objets-médias tels que son, vidéo et image numérique.

Flash utilise *ActionScript* [44] un langage dont la syntaxe est similaire au *JavaScript*, ce dernier est utilisé pour créer la plupart des interactions présentes dans les animations Flash.

Les animations Flash sont aussi comparables aux animations SVG comme le montre Concolato et al dans [17]. Mais l'inconvénient de ce format est qu'il ne soit pas ouvert ce qui rend difficile son intégration avec d'autres formats tel que XML ; ce qui nous a donc empêché de tester nos travaux de thèse sur ce format.

4. eXtensible MPEG-4 Textual format

1.1.2.5 MPEG-LASeR

LASeR [5] [59, 27], porté par la société *Streamezzo* [89] et normalisé par le forum MPEG, est une nouvelle norme consacrée à l'amélioration et à la création d'applications et de services multimédias pour les terminaux mobiles. Ce format de codage de scènes graphiques est un format binaire dérivé de SVG, il comprend de la vidéo, de l'audio, des textes et des images pour une utilisation en mode interactif. Il s'appuie plus précisément sur le profil *Tiny* de SVG et notamment déjà adopté dans l'industrie du mobile. LASER complète SVG en définissant un ensemble de petites extensions clés paramétrées conformément aux besoins de mobilité. Ces extensions clés permettent entre autre : le *streaming* et une compression plus efficace des contenus SVG.

1.1.2.6 Récapitulatif sur les différents flux multimédias

Les formats de document multimédia discutés dans cette section ont été récapitulés dans le tableau 1.1, qui s'inspire de l'évaluation faite par Boll [13]. Le Tableau a été complété avec les évaluation de SVG, X3D, MPEG 4, XMT-W, et le Flash.

Standard	organisation	ouvert	binaire	actif	extensible
MPEG-4	MPEG	OUI	OUI	OUI	OUI
SMIL	W3C	OUI	NON	OUI	OUI
SVG	W3C	OUI	NON	OUI	OUI
FLASH	ADOBE	NON	OUI	OUI	OUI
LASeR	MPEG	OUI	OUI	OUI	OUI
X3D	WEB3DC	OUI	NON	OUI	OUI
OSG	OSGC	OUI	OUI	OUI	OUI

Table 1.1 – Tableau récapitulatif sur les différents flux multimédias

Boll présente aussi dans [13] une comparaison de ces flux multimédias en termes de niveau sémantique des langages utilisés et les fonctionnalités proposées par chacun. Nous avons raffiné cette comparaison en introduisant les langages SVG, Flash, X3D, JAVA, MPREG-4 comme décrit dans la figure 1.2. Un langage avec niveau sémantique élevé décrit la signification de la présentation (utilisation, détails sur le contenus, etc.), tandis qu'un niveau sémantique bas décrit la présentation (interaction des objets-médias). Concernant les fonctionnalités, un niveau élevé de fonctionnalités multimédias signifie un grand jeu de primitives liées aux caractéristiques

5. Lightweight Applications Scene Representation

du modèles, tandis qu'un niveau bas signifie que le format offre moins de souplesse expressive.

Comme le montre la figure 1.2, SVG est assez proche de SMIL 2.0. Cependant, il dispose de plus de primitives de dessin avec moins de fonctionnalités temporelles. Ainsi, il a plus de fonctionnalités, mais moins de sémantique que SMIL 2.0.

MPEG 4 utilise un modèle temporel à base de temps ou événement, mais dispose d'un riche ensemble de primitives de graphisme. Il a ainsi un niveau sémantique bas et un niveau de fonctionnalité élevé.

Le format textuel XMT-Ω est fondé sur SMIL 2.0 mais dispose d'un niveau de fonctionnalité plus élevé avec les primitives de graphisme associées.

X3D est compatible avec MPEG 4 et lui est très semblable. Cependant, MPEG 4 offre plus de fonctionnalités et décrit également les caractéristiques du flux. Flash utilise un langage déclaratif pour décrire les animations, il est similaire à SVG, mais a un niveau sémantique plus bas parce qu'il est optimisé pour un rendu rapide et non pour l'*authoring*.

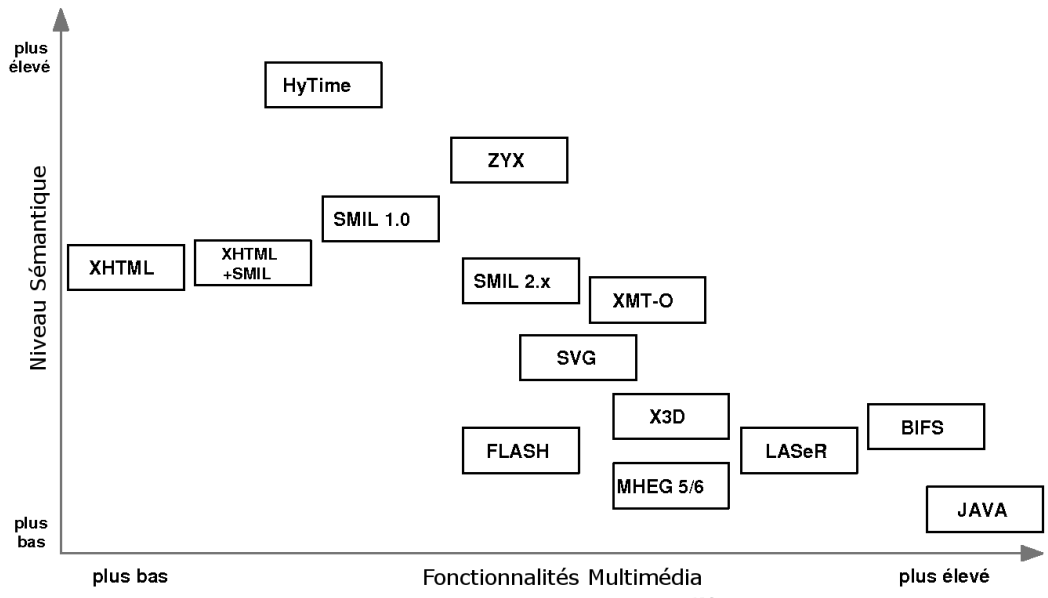


FIGURE 1.2 – Graphe de comparaison des flux

1.1.3 Réseaux multimédias

Les réseaux multimédias, notamment ceux diffusant des documents aux formats mentionnés dans le paragraphe précédent, sont de nos jours omniprésents. En plus du Web, nous les retrouvons dans le télé-enseignement, la télémedecine, la télévision numérique, etc., ainsi que dans plusieurs domaines de télécommunication. L'objectif d'un réseau multimédia est de permettre la diffusion sur le même canal d'un flux composé de plusieurs médias (audio, vidéo, image, ...) d'un point vers un autre.

Le processus de diffusion de contenu dans un réseau multimédia (que nous appellerons indifféremment système multimédia) implique d'un coté un ou plusieurs émetteurs de flux multimédias et un ou plusieurs récepteurs à l'autre extrémité. Selon les besoins de la plate-forme de diffusion en termes de fonctionnalités (routage, cache, adaptation, sécurité, facturation, ...), des nœuds intermédiaires (appelés communément *proxies*) peuvent être placés entre ces deux extrémités pour satisfaire ces besoins, on parle alors de «*proxy* de cache», «*proxy* de sécurité», «*proxy* d'adaptation», etc.

Les émetteurs et les récepteurs dans un système multimédia peuvent être répartis selon différentes topologies (client/serveur, pair-à-pair (*P2P*), etc.). La figure 1.3 présente deux exemples de topologies de systèmes multimédias et décrit les emplacements éventuels des *proxies* .

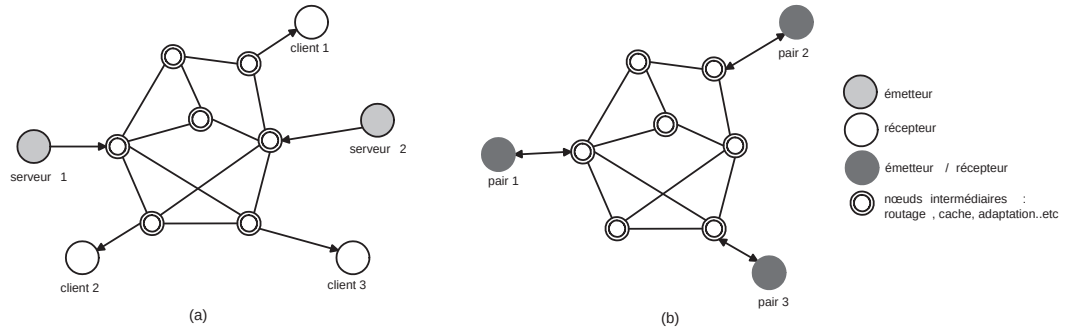


Figure 1.3 – Diffusion de contenu dans un réseau multimédia selon la topologie : (a) client/serveur (b) pair-à-pair

Dans les travaux présentés dans cette thèse, nous nous sommes particulièrement intéressés aux systèmes multimédias utilisant des *proxies* d'adaptation. Ces derniers permettent d'adapter un contenu émis par un émetteur avant de le transférer au récepteur destinataire. Plusieurs types d'adaptation peuvent être garantis par cette catégorie de *proxies*, il peut, par exemple, s'agir d'une conversion de format ou encore d'insertion de nouveaux objets-médias. Le paragraphe suivant définit cette notion d'adaptation de contenu ainsi que ses caractéristiques.

1.2 Adaptation multimédia

Nous définissons l'adaptation multimédia comme suit :

Définition 3 *Le processus d'adaptation multimédia représente le mécanisme qui s'applique sur une forme d'un contenu multimédia et produit en sortie une nouvelle forme alternative. L'adaptation peut être simple, comme par exemple une approche de sélection de version ou de couches de raffinement, ou complexe, en changeant les modalités du contenu comme, par exemple, l'adaptation d'une représentation audio vers une présentation textuelle.*

De nos jours, les besoins en communication multimédia se sont multipliés, des applications telles que la télé-diffusion (*streaming*) et la visioconférence, mais aussi la télémédecine, la vidéo-surveillance et bien d'autres encore nécessitent la transmission de telles données avec des contraintes de qualité, de fiabilité, de délai et surtout de scalabilité importantes. La technique d'adaptation des contenus multimédias répond à plusieurs de ces exigences.

En effet, une première difficulté dans la conception de schémas de communication qui respectent ces contraintes réside dans l'aspect hétérogène à la fois des récepteurs et des canaux de transmission. Il est concevable par exemple de participer à un cours en ligne aussi bien à partir d'un ordinateur personnel qu'à partir d'un PDA. Dans ce cas, il y a une différence énorme entre les capacités de ces récepteurs à traiter le contenu multimédia reçu. La mise en place de mécanismes pour l'adaptation du contenu émis aux profils des participants avant sa transmission à ces derniers est une solution qui assure une compatibilité entre les différents intervenants du système afin que les données soient exploitables par tous.

L'adaptation de contenu peut répondre à d'autres besoins dans les réseaux multimédias actuels tels que les besoins commerciaux. En effet, un mécanisme d'adaptation peut assurer, par exemple, l'ajout de messages publicitaires (ou autre) dans un flux émis avant sa transmission au destinataire.

La figure 1.4 décrit le processus de base d'une opération d'adaptation. Tout d'abord, une description du profil du récepteur et une description du contenu original sont transmis à l'application d'adaptation où un algorithme de décision (dit décisionnaire) décide quelles sont les opérations d'adaptation à effectuer. Ce dernier synchronise ensuite ces opérations d'adaptation au niveau du mécanisme d'adaptation qui reçoit le contenu original en entrée avant de produire en sortie un contenu adapté, selon les besoins, au profil du destinataire.

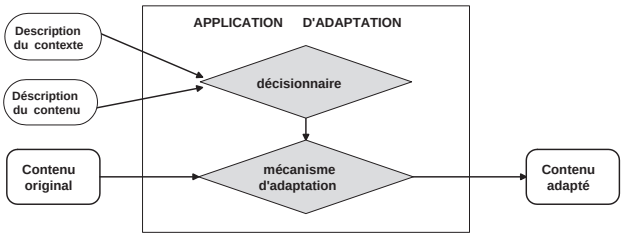


Figure 1.4 – Principaux éléments d'un processus d'adaptation

Dans une présentation multimédia, une opération d'adaptation peut concerner un ou plusieurs objets-médias séparément. Le décisionnaire, après analyse du contenu original, décide du sort de chaque objet-média constituant ce dernier. Pour chaque objet-média à adapter la procédure peut se décomposer alors en trois étapes :

1- **Suppression** de l'objet-média de son emplacement dans la présentation

2- **Transformation** de l'objet selon les besoins de l'opération

3- **Insertion** de l'objet adapté dans la présentation

Il est cependant utile de souligner que la procédure peut se limiter à l'étape 1 ou à l'étape 3 seulement. En effet, l'adaptation peut, par exemple, consister uniquement en une suppression de l'objet-média, les deux dernières étapes ne seront alors pas effectuées. L'adaptation peut aussi consister en une insertion d'un nouvel objet-média, dans ce cas là les deux premières étapes ne seront pas effectuées.

1.2.1 Stratégies d'adaptation

Nous avons étudié de près plusieurs mécanismes d'adaptation de documents multimédias. Cette étude nous a permis de faire ressortir plusieurs critères de classification qui permettent de mieux comprendre leur fonctionnement. Une classification possible est de distinguer l'emplacement où s'effectuent les opération d'adaptation. Nous verrons par la suite (dans la section 1.3.2) que cette classification est très utile pour situer le contexte des travaux de cette thèse. Un autre classification peut être considérée en utilisant comme critère la variation du contexte utilisateur. Trois stratégies d'adaptation peuvent être identifiées :

adaptation statique : consiste à préparer une collection de méthodes d'adaptation pour les différents contextes possibles. Chaque méthode considère un unique couple (contexte initial, contexte final) et ne peut pas être réappliquée si le

contexte change. Lorsqu'il y a une correspondance entre les caractéristiques du contexte courant et les caractéristiques du couple de contextes de la méthode d'adaptation, la méthode est appliquée et le résultat est utilisé pour transmettre finalement un contenu adapté ;

adaptation dynamique : applicable à un document ou à un objet média si la description du contexte de cette ressource correspond aux conditions d'entrée de la méthode d'adaptation et si la description de sortie de la méthode correspond aux contraintes du client cible. Elle peut engendrer en sortie une variété de ressources de caractéristiques différentes et qui peuvent être appliquées dans des contextes différents. La valeur d'un paramètre est amenée à changer durant la consultation d'un document. Ce changement intervient de façon peu fréquente (changement de la taille de la fenêtre, préférence entre une présentation interactive ou non, etc.) ;

adaptation temps réel : la décision du format de sortie change constamment durant la présentation du document. C'est le cas, par exemple, du débit réseau et de la charge CPU.

Bien que plusieurs travaux présentés dans ce mémoire soient fondés sur des mécanismes de transformation, nous n'effectuons pas un état de l'art sur ce sujet. En effet, un état de l'art assez complet de ces techniques a été accompli dans la thèse de Mariam Kimiaei [6]. Un des résultats de cet état de l'art est la classification des outils de transformation selon leur méthode de transformation et de génération. Trois types de transformation ont été identifiées :

Adaptation via *Transmodage* implique un changement de modalité du contenu, autrement dit une conversion de type. Par exemple pour produire une image représentant une vidéo ;

Adaptation via *Transcodage* implique une conversion sans changement de modalité. En exemple est la production d'une vidéo au format AVI pour remplacer une vidéo au format MPEG ;

Adaptation via *Transformation* implique un chargement sur les attributs d'un contenu sans le convertir complètement. Comme le redimenssionnement d'une image par exemple.

Bien entendu, les mécanismes d'adaptation peuvent combiner deux voire les trois types de transformation dans un même processus d'adaptation.

1.2.2 Adaptabilité d'un document multimédia

Définition 4 *Un document multimédia est dit* ***adaptable*** *si lors de sa diffusion il est envisageable de le restituer sous un format différent (adapté) sur un ou plusieurs périphériques destinataires de profils différents (PC, téléphones mobiles, ordinateurs, assistants personnels, etc.).*

Deux types de flux adaptables sont possibles, les flux adaptables *a priori* et les flux adaptables *a posteriori* :

- le premier type concerne les formats prédisposés à l'adaptation, autrement dit, les documents qui contiennent les informations de restitution pour plusieurs profils de périphériques, on peut parler alors de documents **scalables** ;
- le deuxième type concerne les formats qui ne font intervenir aucune information sur les appareils de restitution, l'adaptation se fait alors selon le profil du récepteur après sa requête de lecture, on peut parler dans ce cas de documents **transcodables**.

1.2.2.1 Contenus Scalables

Les formats scalables sont les formats de données prédisposés à l'adaptation ; c'est le cas du codage hiérarchique de vidéo, par exemple. En effet, le codage hiérarchique consiste à encoder un flux vidéo en plusieurs «couches», la couche de base est nécessaire pour le décodage de la vidéo. Cette couche correspond à la qualité minimale disponible, les autres couches dites «de raffinement» permettent l'amélioration de la qualité du flux de base. Les qualités sont ordonnées de manière hiérarchique. La qualité N par exemple ne sera disponible qu'après avoir traité la qualité $N-1$, et ce jusqu'à la couche de base. La figure 1.5 montre l'organisation d'une vidéo encodée grâce à un codage de type hiérarchique. D'autre types de hiérarchie existent, comme la hiérarchie spatiale (variation de la résolution) ou la hiérarchie temporelle (variation de la fréquence d'échantillonnage, ex : nombre d'images de la vidéo).

Cependant, l'encodage hiérarchique peut parfois être gourmand en ressources, ce qui empêche son utilisation sur des terminaux très légers. Dans ce cas une adaptation préalable (intermédiaire) du contenu scalable est préconisée avant de le transmettre à cette catégorie de terminaux.

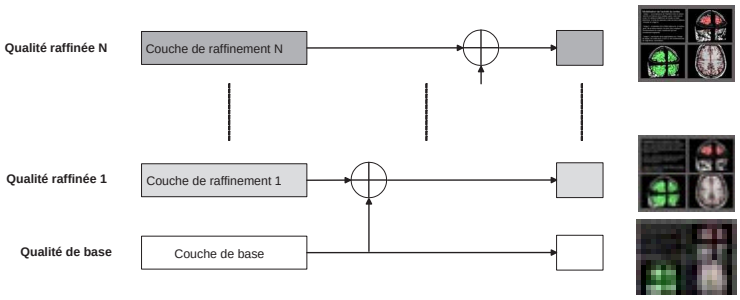


Figure 1.5 – Exemple de flux scalable : codage hiérarchique

1.2.2.2 Contenus Transcodables

La plupart des autres formats de documents multimédias sont dits transcodables, autrement dit, leur format initial ne leur permet pas d'être restitués sur des profils périphériques différents. La lecture de ce type de contenu sur chaque terminal de profil différent nécessite alors une adaptation spécifique à ce profil.

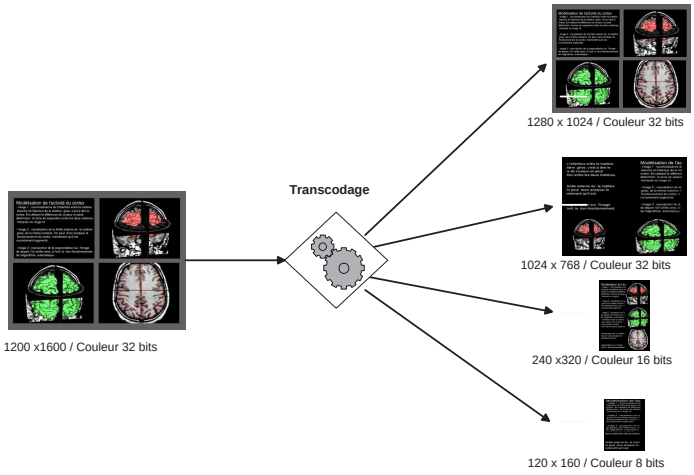


Figure 1.6 – Exemple d'un flux transcodable

1.3 Systèmes Multimédias Adaptatifs

Définition 5 *Les composants qui interviennent pour résoudre le problème de l'adaptation dans un système multimédia, constituent un système multimédia adaptatif (SMA). La fonction principale d'un tel système est de restituer du contenu multimédia regroupé au niveau de l'émetteur, tout en satisfaisant aux besoins et aux contraintes définis par le contexte du récepteur.*

1.3.1 Topologie d'un SMA

La figure 1.7 illustre les composants d'un SMA indépendamment de leur organisation physique (client, serveur, *proxy* , etc.).

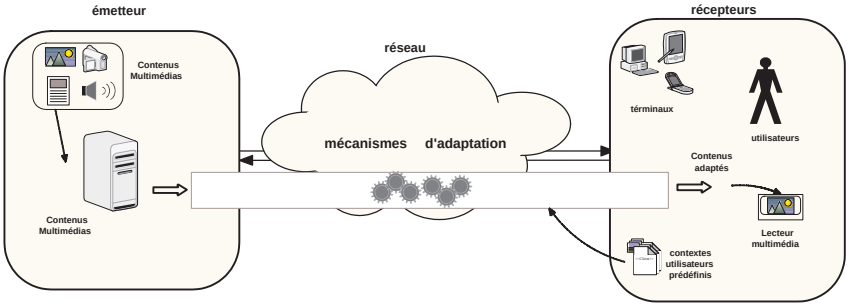


Figure 1.7 – Système multimédia adaptatif

La particularité d'un système multimédia adaptatif par rapport a un système multimédia classique est qu'il dispose de mécanismes d'adaptation placés dans la chaîne de diffusion entre l'émetteur et le récepteur, qui permettent, à partir de contenus originaux, la production de présentations qui correspondent aux besoins et aux contraintes des contextes des utilisateurs. Par exemple, pour un utilisateur malvoyant, une adaptation réalisée par le système adaptatif serait la production d'objets de nature audio de façon à faire entendre un texte plutôt que de l'afficher.

Ces systèmes adaptatifs varient selon la technique d'adaptation utilisée (outils, emplacement, etc.), la suite de cette section donne un panorama plus détaillé sur les différentes techniques d'adaptation et leurs caractéristiques.

1.3.2 Emplacement de l'adaptation

Un des critères de classification ressortis de notre état de l'art est de distinguer l'emplacement où s'effectuent les opérations d'adaptation. Dans la littérature, nous trouvons différents niveaux dans lesquels ce processus peut être appliqué. Nous distinguons trois niveaux possibles selon le chemin de transmission du contenu de l'émetteur (*serveur*) vers le destinataire cible (*client*) : niveau serveur, niveau client et niveau intermédiaire (*proxy*).

1.3.2.1 Adaptation au niveau serveur

La figure 1.8 montre un exemple où l'adaptation du contenu original est effectuée du côté du serveur avant de transmettre le résultat au client. Celle-ci peut aussi être appliquée afin de respecter le contexte de la transmission du contenu, par exemple pour optimiser l'utilisation des ressources du réseau si la bande passante est limitée ou pour réduire le temps de réponse et les délais de transmission du contenu. Plusieurs architectures adoptent l'adaptation côté serveur puisqu'elle est facile à mettre en œuvre et n'implique que l'entité serveur du réseau.

Figure 1.8 – adaptation sur le site émetteur

1.3.2.2 Adaptation au niveau client

Les mécanismes d'adaptation peuvent être mis en œuvre au niveau client (comme le montre la figure 1.9) si ce dernier est capable d'effectuer ces tâches sans avoir besoin de l'aide des autres entités du réseau. Certains terminaux peuvent être capables d'appliquer des transformations ou un transcodage du contenu par exemple en analysant des feuilles de style XSLT ou en exécutant certains scripts. Ces techniques peuvent inclure la conversion ou la traduction du contenu, le filtrage, la sélection, etc. Afin d'appliquer ce genre de techniques, le client doit être capable d'analyser les règles du langage de transformation ou d'appliquer les directives des programmes de transformation. Généralement, les techniques de transformation au niveau client sont réduites et se limitent à des domaines d'applications bien particuliers. De plus dans un contexte hétérogène impliquant des terminaux à très faible capacité cette solution devient vite inopérante vu la difficulté d'implémenter ces techniques d'adaptation sur ces terminaux.

1.3.2.3 Adaptation au niveau *proxy*

Une troisième approche utilisée pour l'adaptation est la mise en œuvre de nœuds intermédiaires que l'on mandate pour cette tâche. Les adaptations intermédiaires

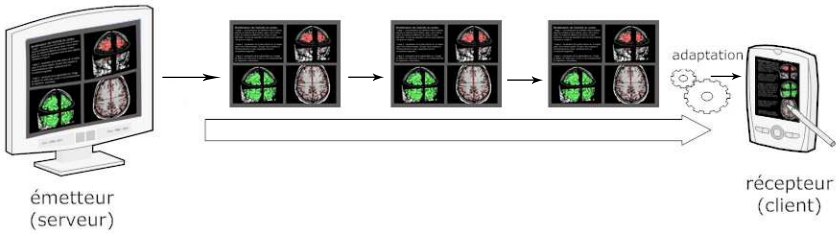


Figure 1.9 – adaptation sur le site récepteur

du contenu sont généralement appliquées dans les architectures à base de *proxies*. Ces architectures sont fondées sur l'ajout d'une troisième entité entre l'ensemble des serveurs et l'ensemble des clients. Un *proxy* d'adaptation peut être vu comme un processus qui reçoit le contenu, à la place du client, et qui applique un certain traitement sur le contenu avant de le transmettre au client cible. Ce type d'architecture représente une approche efficace pour traiter le problème de l'hétérogénéité des clients et des serveurs. En effet, dans une architecture à base de *proxies*, la plate-forme du réseau n'est pas modifiée et les caractéristiques de l'environnement déjà existantes sont préservées. Ce type d'architecture peut être très performant en incluant plusieurs *proxies* d'adaptation dédiés pour des tâches particulières telles que le transcodage de la vidéo et des images, la diffusion de la vidéo, etc.

L'adaptation au niveau *proxy* peut aussi concerner le protocole utilisé dans la transmission du contenu adapté. Après l'application d'une adaptation particulière, par exemple une transformation SMIL vers MPEG-2, le *proxy* peut contrôler la manière dont le contenu adapté est transmis au terminal cible. En effet, le *proxy* peut utiliser un nouveau protocole, différent du protocole d'origine utilisé entre le *proxy* et le serveur.

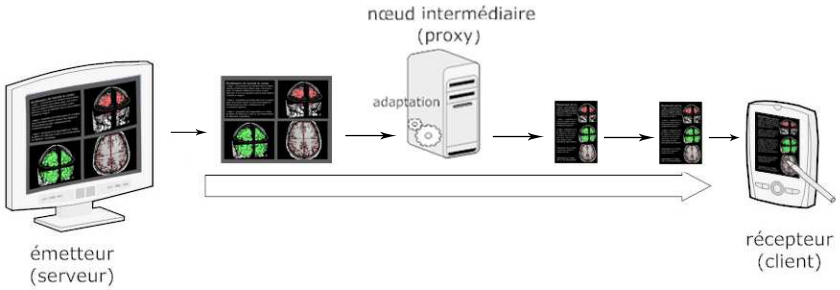


Figure 1.10 – adaptation sur un nœud intermédiaire

1.3.2.4 Discussion

Les différents niveaux d'adaptation utilisent un comportement adaptatif pour offrir aux applications la configuration la mieux appropriée avec la disponibilité des ressources. Ces schémas peuvent améliorer significativement les performances des applications dans un environnement *best-effort*. Le tableau 1.2 résume les principales caractéristiques de ces approches :

Propriétés \ Adaptation, coté	serveur	client	*proxies*
Modifications des applications existantes	OUI	OUI	NON
Modification de l'infrastructure réseau	OUI	NON	NON
Adaptation aux capacités du réseau	OUI	NON	OUI
Transformation des objets-médias	OUI	OUI	OUI
Insertion d'objets-médias	OUI	NON	OUI
Suppression d'objets-médias	OUI	OUI	OUI

Table 1.2 – Comparaison des approches adaptatives

Dans l'approche d'adaptation coté serveur, la responsabilité de l'adaptation est laissée à la charge de la source. Cette approche est souvent utilisée dans les systèmes à base de serveurs de fichiers. Elle nécessite une certaine capacité de traitement coté serveur.

Dans l'approche adaptation sur les clients, chaque client prend des décisions locales selon ses besoins et ses capacités sans affecter les autres clients. Mais certains clients ont des capacités de traitement, des ressources de mémoire et d'énergie très limitées (on cite le cas des PDA) ce qui rend cette technique difficilement applicable. Il est donc préférable qu'une adaptation destinée à ce type de récepteurs soit effectuée sur le serveur ou sur des nœuds intermédiaires.

Les techniques d'adaptation sur des nœuds intermédiaires donnent la possibilité de délivrer des niveaux de qualité de service appropriés pour chaque participant sans subir les surcharges associées à l'encodage et la transformation des flux. Il existe plusieurs avantages offerts par l'utilisation de *proxy* d'adaptation. Premièrement, le *proxy* peut être placé sur la position la plus appropriée pour effectuer l'adaptation, par exemple, sur le point de passage d'un réseau filaire à un réseau hertzien. D'autre part, les *proxies* permettent d'effectuer les adaptations de façon transparente aux serveurs et aux clients.

Cependant, cette solution présente aussi certains inconvénients. Le plus important concerne la sécurité des données transitant par le *proxy*. En effet, les émetteurs de

contenus doivent autoriser les *proxies* d'adaptation à effectuer les transformations sur leurs documents avant transmission. En même temps, ces émetteurs peuvent vouloir protéger leurs contenus des risques d'usurpation, de redistribution et du piratage aisés, et souhaitent s'assurer que leurs transactions soient sécurisées. C'est cet inconvénient que nous tentons d'analyser dans les travaux présentés dans cette thèse. Nous tacherons dans la section suivante de définir plus en détail la notion de *proxies* d'adaptation et de présenter un aperçu des risques liés à leur déploiement.

1.4 *Proxies* d'adaptation multimédia

Définition 6 *Un* proxy *d'adaptation (que nous appellerons indifféremment* ***PA*** *dans la suite du document) peut être vu comme un processus qui reçoit le contenu, à la place du client, et qui applique un certain traitement sur le contenu avant de le transmettre au client cible.*

L'idée d'utiliser des *proxies* n'est pas nouvelle. Elle a été introduite dans la spécification du protocole HTTP pour des raisons de sécurité. Depuis, elle a été utilisée dans plusieurs applications pour des besoins d'adaptation.

Les travaux sur les *proxies* d'adaptation multimédia sont eux plus récents [14, 10, 48, 60, 57, 91, 58]. La plupart de ces travaux s'intéressent principalement à l'adaptation des médias discrets tels que les pages Web et les images. D'autres se sont fondés sur des types adaptations simples comme le remplacement de la vidéo par un flux audio ou par une description textuelle. Certains d'entre eux utilisent cette technique pour implémenter des adaptations dynamiques sur les flux continus comme l'audio et la vidéo. En général, leurs objectifs consistent à adapter ces flux aux capacités réseau ou au profil du récepteur, soit par transcodage des médias, soit par l'élimination/remplacement de médias dans une présentations par exemple, soit en réduisant la qualité de ces médias, etc.

D'autres travaux se sont intéressés aux *proxies* d'adaptation de contenu composite [57, 91, 58, 52], dont plusieurs sont industrialisés [73, 43]. Le paragraphe 1.4.1 présentera quelques uns parmi les travaux étudiés.

1.4.1 Exemples de système utilisant des PA

Nous présentons dans ce paragraphe un rapide état de l'art sur des systèmes multimédias adaptatifs proposé par le monde de la recherche.

1.4.1.1 ViTooki

ViTooki [57], pour Video-ToolKit, supporte l'adaptation de flux vidéo scalables, le transcodage et le cache au niveau *proxy*. Il prend en charge des flux MPEG-1/2/4/7/21 et MP3/AAC, enregistrés dans différents conteneurs tels que MP4, AVI au dessus de RTSP/RTP/UDP avec retransmission.

L'outil complet ViTooKi comprend une bibliothèque principale, plusieurs petits exemples qui illustrent comment utiliser cette bibliothèque, et certaines applications multimédias plus importantes comme un serveur de médias adaptatifs, un *proxy* d'adaptation, un *player* multi-video, etc.

1.4.1.2 NAC

L'architecture NAC [91] est une architecture à base de *proxies* qui assure l'adaptation du contenu pour des appareils hétérogènes. L'adaptation est effectuée en utilisant une entité intermédiaire qui permet d'analyser les requêtes des clients, envoyer les requêtes aux serveurs d'origine, recevoir les réponses des serveurs, appliquer les méthodes d'adaptation et envoyer les réponses adaptées aux clients. L'adaptation du contenu passe par un protocole de négociation qui permet de bien adapter le contenu selon les caractéristiques de la plate-forme cible. La gestion et le maintien des descriptions des plates-formes et des différents clients sont assurés par une entité de l'architecture (*profiles repository*) exploitable directement par le *proxy* à travers des services Web.

1.4.1.3 APPAT

APPAT [80, 58] est une plate-forme d'adaptation qui consiste à utiliser un *proxy* qui adapte dynamiquement les données transmises. Le *proxy* bénéficie de mécanismes d'adaptation comme des filtres (transformation vidéo, modification automatique d'images : redimensionnement/recadrage, etc.), des mécanismes de transco-

dage (changement de format vidéo/audio/images appelé Mixer RTP) et des mécanismes de changement de protocole (HTTP, RTP). Les données sont adaptées en fonction des caractéristiques du terminal et des préférences de l'utilisateur.

Le principe novateur de cette plate-forme est basé sur la mise en œuvre de l'adaptation globale. APPAT fonctionne comme un bus d'échange de données : les utilisateurs d'applications utilisant la plate-forme font abstraction de tout le fonctionnement de celle-ci. Ils sont connectés à celle-ci et lui soumettent leurs requêtes d'échange de données. L'utilisation de *proxies*, à l'image de *proxies* cache, permet de rendre l'adaptation transparente aux applications existantes.

1.4.1.4 PAAM

PAAM [52, 53], une architecture qui adapte les documents multimédias au contexte des participants en s'inspirant des architectures P2P. Les contenus multimédias ainsi que les adaptateurs sont mis à disposition par des systèmes librement répartis sur le réseau. Sur la base d'une requête de document et d'un contexte d'utilisation, l'architecture d'adaptation recherche les adaptateurs utiles, les compose pour réaliser des opérations d'adaptation complexe et pilote l'adaptation. Pour chaque document, le système d'adaptation permet de redéfinir l'ordonnancement des tâches entre des adaptateurs choisis sur le réseau ; cela facilite la prise en compte des changements dans l'environnement des usagers et dans celui des adaptateurs.

1.4.1.5 IBM Transcoding Proxy

Devenu par la suite «WebSphere Transcoding Publisher»[43]. Développé au début par le Mobile Networking Group d'IBM en 1999 en tant que prototype de recherche pour l'adaptation de contenu pour PDA. Il est devenu aujourd'hui un produit commercial orienté entreprise permettant de réaliser une multitudes de transformations de formats de texte et d'image standard, notamment HTML, WML et XML, GIF et JPEG, etc.

1.4.2 Risques liés à l'adaptation au niveau *proxy*

Comme nous l'avons signalé dans l'introduction générale de ce mémoire, le problème majeur dans une architecture d'adaptation à base de *proxy* est la vulnérabilité du contenu lors de son passage par le PA où le contenu doit être adapté (voir la figure 1.11). En effet, dans ce type d'architecture, même si la sécurité point à point (où lien par lien) reste applicable entre l'émetteur et le *proxy* puis entre le *proxy* et le destinataire, la sécurisation de «bout-en-bout» des flux ne peut être garantie car les flux doivent être accessibles au *proxy* pour les opérations d'adaptation. Ces flux sont alors facilement piratables à cet endroit de la chaîne, rien n'empêche un utilisateur de ces machines intermédiaires d'intercepter le trafic qui transite par elles.

Plusieurs techniques d'attaques réseaux sont envisageables pour récupérer les flux transitant, *Sniffing*, *Spoofing*, *Deny Of Service*, etc. (lire [92] pour plus de détails sur ces attaques).

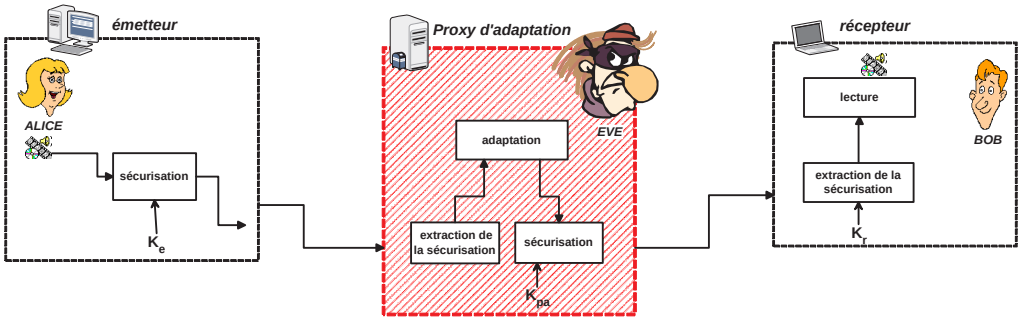


FIGURE 1.11 – Vulnérabilité des flux au niveau du PA

Le chapitre 3 reviendra plus en détail sur ces failles en termes de sécurité en analysant les différentes attaques possibles lors de l'acheminement d'un contenu multimédia adaptable à partir d'un émetteur jusqu'au récepteur.

1.5 Synthèse

Ce chapitre a décrit les concepts de base dans le domaine du multimédia et de l'adaptation de contenu multimédia plus précisément. Nous avons commencé par faire un tour d'horizon sur les différents formats multimédias existants où nous avons retenu que la forme générique d'un document multimédia est un ensemble d'«objets-médias» additionné à un document (souvent textuel) dit «description métadonnée» décrivant les interactions entre ces «objets-médias».

Nous avons ensuite définit l'adaptation dans le domaine du multimédia où nous retenons la décomposition de l'opération d'adaptation en trois étapes : 1) suppression des objets-médias à adapter, 2) transformation puis 3) insertion des objets-médias adaptés (ou de nouveaux objets-médias).

L'état de l'art sur les différentes techniques d'adaptation nous a permis de montrer l'intérêt de la mise en œuvre des opérateurs d'adaptation dans le réseau sur des nœuds intermédiaires (appelés *proxies* d'adaptation).

Dans la dernière section de ce chapitre nous avons soulevé la problématique de la sécurité liée à l'utilisation de l'adaptation au niveau *proxy*, cette problématique sera analysée plus en détail dans le chapitre 3. Cela dit, l'objectif de cette thèse étant de proposer des solutions pour cette problématique, nous allons exhiber dans le chapitre suivant les outils de base utilisés pour l'élaboration de ces solutions.

Chapitre 2

Systèmes multimédias et sécurité

Le développement rapide des réseaux multimédias a créé les conditions d'une mutation profonde dans ce domaine, qui, au-delà de la transition progressive vers une distribution dématérialisée des flux, se traduit par de nouvelles contraintes de sécurité. Cette mutation s'accompagne d'un développement important des pratiques de copie et d'échanges d'œuvres multimédias protégées sur les réseaux. Si en l'état du droit positif international comme national, l'essentiel de ces pratiques relèverait du régime juridique de la contrefaçon, il apparaît important de définir des techniques qui permettent aux propriétaires de contenus de protéger leurs documents contre toute usurpation et de contrôler la sécurité de leurs contenus jusqu'aux destinataires finaux. Ce deuxième chapitre a pour objectif de présenter les outils de bases qui permettent de définir de telles techniques.

Nous commençons dans la première section par rappeler les impératifs en termes de sécurité dans le domaine du multimédia du point de vue des contenus, des communications et des infrastructures ; nous exposerons ensuite dans la seconde section les techniques de sécurité utilisées dans ce contexte, qui s'appuient essentiellement sur un catalogue de fonctions de sécurité que nous tâcherons d'exhiber dans l'annexe A de ce document. Ces techniques de sécurité mettent en œuvre des solutions pour protéger les contenus multimédias ou pour sécuriser les interfaces entre les émetteurs et les récepteurs.

La section 2.3 décrit les systèmes de gestion des droits dans le domaine du multimédia, avant de clôturer ce chapitre dans la section 2.4 par une synthèse où nous identifions les limitations des solutions présentées dans le contexte des documents multimédias adaptables.

2.1 Besoins de sécurité dans les communications multimédias

Tout composant du système multimédia dans une plate-forme de communication est potentiellement vulnérable à une attaque visant à récupérer le flux multimédia échangé, le modifier, se l'approprier, etc., ou perturber le fonctionnement du système. Ces attaques peuvent être lancées automatiquement à partir de machines du système infectées (virus, chevaux de Troie, vers, etc.), à l'insu de leur propriétaire, ou encore activées par des pirates informatiques.

Si l'on considère une plate-forme de diffusion utilisant des communications via l'Internet, l'acheminement des flux sans itinéraire préconçu fait qu'il est impossible de savoir par où passent les données, et donc d'avoir la garantie que le chemin emprunté est sans danger. En effet, les informations envoyées d'un ordinateur à l'autre peuvent passer par un certain nombre de machines avant d'atteindre leur destination. Il devient ainsi aisé pour un pirate d'intercepter le trafic qui transite par ces machines intermédiaires en utilisant des techniques d'attaque telle que le *Sniffing* par exemple.

Afin de contrer les risques d'attaque dans les systèmes d'informations en général, plusieurs catalogues de critères de sécurité ont été publiés [72, 78, 36, 45]. L'un des plus importants est le *Common Criteria* (CC) de l'ITSEC [45], qui définit des critères avec plusieurs classifications selon les trois types de menaces :

- menaces de confidentialité (révélation de données secrètes)
- menaces d'intégrité (modification non-autorisée des données)
- menaces de disponibilité (refus anormal d'informations ou de ressources)

En se reposant sur ce *CC*, nous avons effectué une analyse de sécurité de notre problématique. Analyse qui a montré que les services de sécurité les plus pertinents et qu'il faut assurer pour sécuriser un réseau de communication multimédia sont :

1. l'authentification (de l'origine) des documents ;
2. l'intégrité des données ;
3. et la confidentialité des contenus transmis.

2.1.1 Authentification

L'authentification dans notre contexte peut être définie par « l'assurance de l'origine et de l'identité de la source ». En effet, la source doit pouvoir fournir un mécanisme prouvant l'origine des documents émis contre les possibles risques d'usurpation d'identité.

Ce problème peut se découper en un double problème : (i) tout d'abord seul la source doit pouvoir générer l'empreinte et un grand nombre de récepteurs doivent pouvoir la vérifier ; (ii) par ailleurs, une modification (altération, adaptation, etc.) du contenu original ne doit pas provoquer la perte de l'authentification une fois ce dernier reçu.

De plus, en cas d'adaptation du contenu multimédia original, la technique d'authentification doit permettre de différencier les parties provenant de l'émetteur initial des parties provenant des opérateurs d'adaptation.

Les techniques utilisées pour l'authentification des contenus sont généralement la signature numérique (*cf.* section 2.2.1.2) ou le tatouage numérique (*cf.* section 2.2.2).

2.1.2 Intégrité des documents échangés

De manière générale, l'intégrité des données désigne l'état des données qui, lors de leur traitement, de leur transmission ou de leur conservation, ne subissent aucune altération ou destruction non autorisée, et conservent un format permettant leur utilisation.

Dans le domaine du multimédia, l'expression «intégrité des données» correspond à deux notions légèrement différentes, selon que le contexte soit celui des télécommunications ou celui de la cryptographie.

Si le principe général est le même, les données ne doivent pas avoir été modifiées depuis leur création, à comprendre au sens large (écriture sur un support de stockage, transmission ...), la cryptographie veut pouvoir affirmer que les données ont ou n'ont pas été modifiées, ce qui se fait souvent via une fonction de hachage (*cf.* section A.1.2) ou, un MAC (*Message Authentication Code*), tandis qu'en télécommunication, on souhaite simplement pouvoir détecter et souvent corriger ces modifications.

Dans le cadre de cette thèse, nous emploierons le sens cryptographique de l'intégrité. Notre but étant de garantir que les documents multimédias transmis ne soient pas modifiés de manière illicite.

2.1.3 Confidentialité

Le concept de confidentialité constitue une motivation fondamentale en matière de sécurité. Par confidentialité, on entend généralement la protection contre tout accès non autorisé aux contenus échangés.

Par extension, la confidentialité d'un document multimédia est en outre associée à certains moyens techniques (par exemple la cryptographie (*cf.* section 2.2.1)) visant à garantir que l'ensemble (ou une partie) des objets-médias le constituant ne soit divulgué qu'aux personnes voulues et ce, afin que seules les parties explicitement autorisées puissent interpréter le contenu échangé entre elles.

Le chiffrement, les listes de contrôle d'accès (ACL) et les permissions d'accès aux fichiers sont des méthodes souvent utilisées pour assurer la confidentialité des données.

2.2 Les techniques de sécurité

La protection des données médias, que ce soit des images, de l'audio ou de la vidéo, utilise de manière essentielle des méthodes cryptographiques ou des méthodes de tatouage numérique, même si d'autres techniques de sécurité telles que la stéganographie [64], le *fingerprinting* [94], etc. peuvent les compléter. Ces techniques de protections procurent des éléments de réponse aux impératifs de sécurité décrits dans la section 2.1. Nous décrivons dans ce qui suit ces deux approches, qui ne sont d'ailleurs pas exclusives l'une de l'autre, en précisant le degré de standardisation et les limites de sécurité technologiques connues de chacune. Une troisième technique accompagne souvent l'une ou l'autre de ces techniques pour garantir un accès hiérarchique aux documents, cette technique est le «contrôle d'accès» que nous présenterons dans le paragraphe 2.2.3.

Une discussion à propos de l'apport et les limites de chacune de ces techniques pour notre problématique est ensuite présentée.

2.2.1 Approche cryptographique

Dans le domaine du multimédia, comme dans la majorité des domaines des télécommunications, la cryptographie est la solution la plus utilisée pour assurer l'inintelligibilité des informations échangées.

Par définition la cryptographie, est un ensemble de procédés et de techniques mathématiques permettant de transformer un message dit «clair», en un autre message, dit «chiffré». Les deux applications majeures de la cryptographie sont le «Chiffrement» et la «Signature électronique». Nous allons dans la suite de ce paragraphe décrire le principe général de chacune de ces deux applications dans les systèmes multimédias.

2.2.1.1 Chiffrement et multimédia

Les systèmes multimédias utilisent les techniques de chiffrement symétriques et asymétriques (*cf.* section A.1.1). Les flux sont protégés par l'émetteur en les chiffrant à l'aide d'un algorithme de chiffrement et d'une clé. Le récepteur doit alors disposer de la clé pour déchiffrer un flux reçu.

D'une manière générale, comme le montre la figure 2.1, Alice envoie un document multimédia M à Bob en évitant que Eve puisse voir son contenu ; pour cela elle chiffre M en utilisant la fonction $E()$ et une clé k avant de transmettre le résultat $\mathcal{C}$ à Bob. Ce dernier, à la réception de $\mathcal{C}$, le déchiffre en utilisant la fonction de déchiffrement $D()$ et une clé k' ($k = k'$ dans le cas d'un algorithme symétrique). Le résultat de cette opération est le document M en clair.

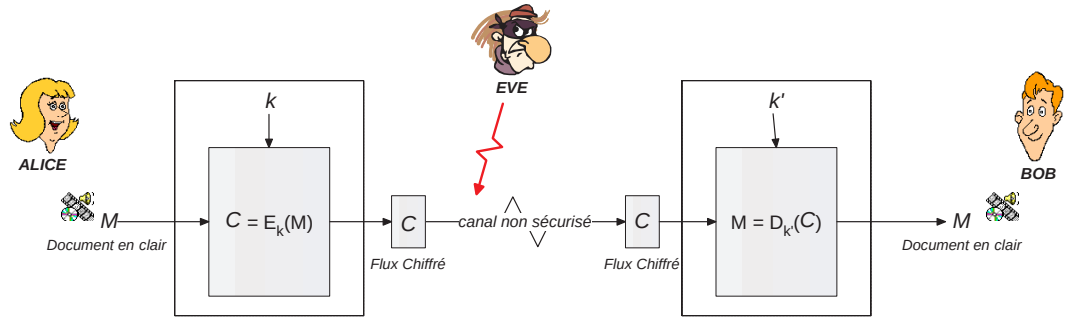


Figure 2.1 – Principe général du chiffrement

Cela dit, on peut distinguer deux différents procédés pour chiffrer un document multimédia composite :

(a) la première serait de chiffrer l'ensemble de ses objets-médias après les avoir rassemblés dans une même archive ;

(b) la seconde serait de chiffrer chaque objet-média séparément, puis d'envoyer les différents résultats chiffrés.

Pour la suite de cette thèse, nous appellerons ces manières de procéder «les méthodes classiques» pour chiffrer un document multimédia.

Par ailleurs, pour des besoins divers (performances, scalabilité, adaptabilité, etc.), plusieurs autres méthodes ont été proposées pour chiffrer des documents multimédias en se basant sur leur codage, leur utilisation, etc. Nous citons par exemple les systèmes de chiffrement scalables qui s'appuient sur le codage scalable de certains flux multimédias comme la technique proposée par Zhu dans [104] pour les flux MPEG-4 FGS, ou encore [102, 41] pour des flux d'images scalables. Des standards plus génériques tels que XML Encryption [96], constituent aussi des solutions pour chiffrer des documents multimédias composites.

La partie II de l'ouvrage de Furht [31] présente un état de l'art sur les différentes technique de chiffrement spécifiques au domaine du multimédia.

2.2.1.2 Signature numérique et multimédia

La signature numérique est utilisée à des fins de vérification d'identité et de non répudiation. L'émetteur d'un document le signe avec sa clé privée, ensuite, l'application qui permet de lire le document chez le récepteur peut vérifier l'identité de l'émetteur et peut garder la signature comme preuve de l'obtention licite du document.

Concrètement, comme le montre la figure 2.2, Alice signe le document M avant de le transmettre à Bob. M est alors passé à $hash()$ une fonction de hachage à sens unique, le résultat $\mathcal{H}$ est ensuite chiffré par la clé privée d'Alice et donne le résultat $\mathcal{S}$. Le couple $(M, \mathcal{S})$ est ensuite transmis à Bob. Ce dernier à la réception :
– déchiffre la signature à l'aide de la clé publique d'Alice et obtient ainsi $\mathcal{H}'$;
– applique la fonction de hachage $hash$ sur le document M reçu et obtient $\mathcal{H}$;
– compare $\mathcal{H}$ et $\mathcal{H}'$ et en déduit la validité du document reçu.
Comme pour le chiffrement, nous distinguons deux méthodes «classiques» pour signer un document multimédia composite :

(a) la première serait de signer l'archive rassemblant l'ensemble des objets-médias

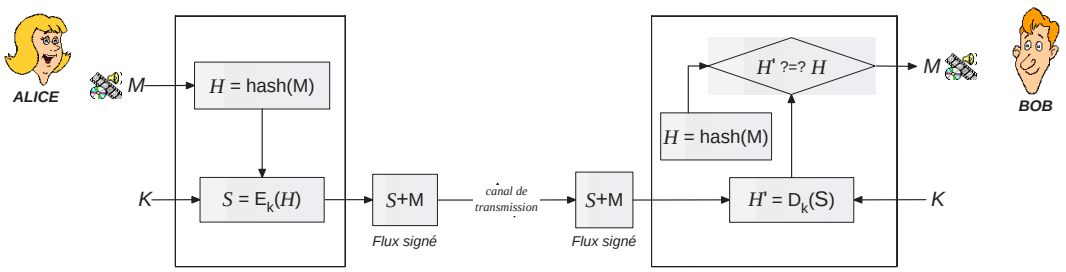


Figure 2.2 – Principe général de la signature numérique

le constituant ;

(b) et la seconde serait de signer chaque objet-média séparément, puis d'envoyer les différents résultats des signatures.

La seconde méthode peut s'avérer assez coûteuse en termes de performances et manque surtout de flexibilité. En effet, signer chaque objet séparément génère par exemple un surcoût de consommation mémoire considérable.

Nous présenterons dans le chapitre suivant un état de l'art sur des solutions de signatures numériques de documents multimédias proposées dans la littérature dans le cadre de différents besoins fonctionnels.

2.2.2 Tatouage de donnnées

Le tatouage numérique, également connu sous le nom de *watermarking* [25], est une discipline plus récente que la cryptographie, il trouve son origine dans le manque de techniques fiables de protections de données médias. En effet, associé à d'autres techniques, cet axe de recherche a pour but de résoudre des problèmes aussi variés que la protection du *copyright* et des droits d'auteurs, la réglementation des copies, la prévention de la redistribution non autorisée, le suivi de documents et l'intégrité du contenu d'une donnée. Nous développerons dans la suite de cette section les principes de base du fonctionnement du tatouage ayant trait à la protection du *copyright* et des droits d'auteurs de contenus médias, typiquement image ou vidéo. Pour plus de détails le lecteur pourra consulter [25], et [32].

L'idée derrière le tatouage est d'insérer une marque imperceptible dans les valeurs de la donnée, la marque insérée, appelée *watermarque* ou *filigrane*, correspond au code du *copyright* qui permettra alors d'identifier le propriétaire. Ce type de tatouage doit répondre à des contraintes fortes en termes de robustesse. Idéalement, quelles que

soient les transformations (licites ou illicites) que la donnée tatouée subit, la marque doit rester présente tant que la donnée reste exploitable. De plus, la présence de la marque ne doit être détectée que par des personnes autorisées (possédant une clé de détection privée).

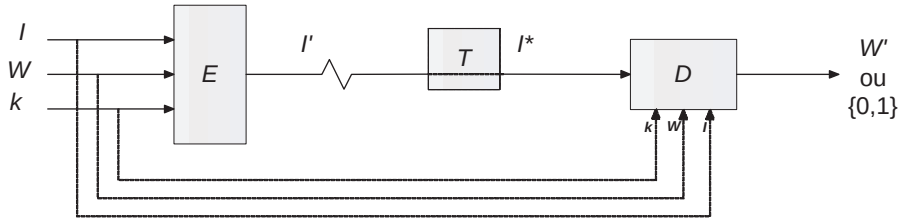


Figure 2.3 – Schéma général d'un processus de tatouage

La figure 2.3 décrit le schéma d'implémentation et de détection de la marque, issu du formalisme donné par Petitcolas et al. dans [75] et généralement utilisé par la communauté des tatoueurs. Le signal d'entrée I, appelé signal hôte sera modifié par une application $E()$. Cette étape est l'incrustation de la marque W dans I, avec l'intervention de la clé privée k. Le signal sortant I', est diffusé. Il est alors soumis à des transformations $T()$ licites ou illicites de nature inconnue, cette version attaquée est notée I^*. A la détection $D()$, la marque est extraite (on obtient alors une marque W') ou sa présence est contrôlée (une variable booléenne indique si la marque est présente dans l'image testée).

Le tatouage numérique est, dans la plupart des cas, assimilé aux contenus de type vidéo, image, voir audio. Néanmoins, nous retrouvons dans la littérature quelques travaux portant sur le tatouage de documents structurés. Sakr propose dans [81] une solution pour l'authentification de documents SMIL en se basant sur des techniques de tatouage standards pour sécuriser chaque objet-média du document. [82] propose aussi une technique similaire pour l'authentification de documents structurés au formats MPEG-4 XMT en utilisant des techniques de tatouage combinées à la signature numérique.

2.2.3 Le contrôle d'accès

Une troisième approche pour la préservation de la sécurité des données dans les systèmes multimédias peut être la mise en application d'une politiques de contrôle d'accès.

Le contrôle d'accès par définition consiste à vérifier si un sujet (un utilisateur, une machine, ...) demandant d'accéder à une ressource a les droits nécessaires pour le

faire. Il permet aussi d'accorder des privilèges à des sujets afin qu'ils puissent réaliser des actions sur des ressources.

Les sujets peuvent être des utilisateurs, c'est-à-dire des personnes physiques, mais aussi les processus que ces personnes utilisent. Les ressources, aussi appelées objets, sont les fichiers du système multimédia. Enfin, les actions possibles dans un système multimédia sont par exemple «lire», «écrire», «exécuter», etc.

Une politique de contrôle d'accès est alors un ensemble d'autorisations qui peut prendre différentes formes selon le modèle de données sous-jacent. Par exemple, une autorisation sur un document XML est généralement exprimée par une combinaison de règles positives (respectivement négatives) permettant d'accéder dans le document aux sous arbres autorisés (respectivement interdits) grâce à des expressions XPath [9].

Une autre dimension des modèles de contrôle d'accès est le modèle d'administration de ces autorisations. Il existe deux familles traditionnelles de modèles. Les modèles dits discrétionnaires (DAC) [34] et les modèles dits obligatoires (MAC) [74]. Dans les modèles discrétionnaires, des permissions sont accordées aux sujets en fonction de leur identité uniquement. Les modèles obligatoires s'appuient sur le niveau de confiance accordé aux sujets. Ainsi, si un sujet est habilité à un certain niveau de confiance, alors il pourra accéder aux ressources ayant un besoin de sécurité équivalent ou inférieur.

Il existe aussi d'autres modèles plus récents. Nous trouvons par exemple un modèle de contrôle d'accès fondé sur les rôles (RBAC) [30, 84]. Dans ce modèle, on considère qu'un sujet se voit attribuer des permissions en fonction des rôles qu'il joue. Ainsi, dans une organisation, des rôles sont définis et des permissions sont accordées à ces rôles. Ensuite, les sujets sont affectés aux différents rôles et obtiennent les permissions correspondantes.

Contrairement aux modèles DAC, les modèles MAC et RBAC tentent de structurer les sujets, soit en fonction de la confiance qu'on leur accorde (habilitation) soit en fonction de leurs rôles. En effet, la gestion et l'administration d'une politique de contrôle d'accès deviennent vite ardues si le système comporte un grand nombre de sujets, d'actions et d'objets. On voit ainsi apparaître l'idée qu'il peut exister d'un côté une forme de structuration des sujets, et de l'autre la définition de l'ensemble des règles de contrôle d'accès. Ainsi, dans un modèle MAC, on s'attachera dans un premier temps à définir les niveaux de confiance, puis à attribuer des permissions en fonction de ces niveaux. Dans un modèle RBAC, on commencera par définir l'ensemble des rôles puis on leur accorder des privilèges.

Dans le domaine du multimédia, le contrôle d'accès est très utilisé notamment dans les plates-formes de DRM que nous présenterons dans la section suivante (*cf.* section 2.3), mais on trouve quelques solutions de contrôle d'accès au niveau flux. Kodali présente dans [55] et [56] DAML, un système de contrôle d'accès pour des documents SMIL en utilisant les principes du modèle RBAC. [29] propose une solution similaire pour des documents SVG. Tandis que [22] présente une solution plus générique pour plusieurs formats multimédias utilisant des metadonnées XML.

2.2.4 Discussion

Nous avons vu dans cette section un panorama sur les différentes outils de bases utilisables pour sécuriser un document multimédia, en l'occurrence, les techniques cryptographiques, le tatouage numérique et le contrôle d'accès. Néanmoins, les besoins en termes de sécurité pour les documents multimédias diffèrent d'un domaine d'utilisation à un autre. Dans cette thèse nous posons la problématique de la sécurisation de documents multimédias adaptables. Les critères de sécurité recherchés sont comme décrit dans la section 2.1 : l'authentification, l'intégrité et la confidentialité, dans le cadre des systèmes multimédias adaptatifs.

Dans un premier temps, posons nous la question à savoir si les techniques décrites dans cette section peuvent garantir ces besoins de sécurité pour des systèmes multimédias **classiques** (non adaptatifs), le tableau 2.2.4 répond à cette question. Si les approches cryptographiques peuvent souvent garantir les trois critères de sécurité il n'en est pas de même pour le tatouage qui est souvent utilisé pour des fin d'authentification seulement. Idem pour le contrôle d'accès qui est utilisé pour assurer la confidentialité des documents ou de parties d'un document.

Critères de sécurité	Authentification	Intégrité	Confidentialité
- Cryptographie	Oui	Oui	Oui
- Tatouage	Oui	Non	Non
- Contrôle d'accès	Non	Non	Oui

Table 2.1 – Comparaison des approches de sécurisation de contenus multimédias

La seconde question que nous nous sommes posée est de savoir si chacune de ces techniques suffit pour garantir nos besoins de sécurité pour des documents multimédias **adaptables**.

– D'abord les techniques cryptographiques, ici la réponse est clairement non, de par leurs principes de fonctionnement. Il est, par exemple, par définition interdit

de modifier un document signé sous peine d'invalider sa signature, idem pour le chiffrement.
- Le tatouage numérique est a priori plus adapté pour des modifications intermédiaires du contenu et peut donc préserver l'authentification ainsi que l'intégrité des données, malheureusement il ne peut garantir leur confidentialité.
- Le contrôle d'accès quant à lui, à l'inverse du tatouage, ne peut garantir leur authentification ou intégrité et sert souvent à garantir la confidentialité des données. Mais dans notre contexte, un contrôle d'accès au niveau flux donnerai à un *proxy* d'adaptation un accès complet aux documents.

Par ailleurs, l'étude que nous avons mené sur ce sujet nous a conduit à voir différentes solutions de sécurisation de contenus multimédias pour différents besoins fonctionnels. Parmi les conclusions que nous avons tiré de cette étude, le fait que la majorité de ces solutions soient des combinaisons de plusieurs techniques parmi celles cités dans cette section, le contrôle d'accès est ainsi souvent accompagné de chiffrement ou de tatouage des contenu. Le tatouage est aussi souvent utilisé avec du chiffrement, etc.

A partir de ce constat nous avons orienté nos recherches vers une solution composite en combinant signature, chiffrement et contrôle d'accès. Nos différentes résultats seront décrits dans la deuxième partie de cette thèse.

2.3 Gestion des droits numériques dans l'univers du multimédia

Nous avons vu dans la section précédente les outils de base pour sécuriser les contenus multimédias. L'utilisation de ces techniques de sécurité pour protéger les droits de propriété intellectuelle dans le contexte multimédia a amené les industriels et les producteurs de contenus (textes, images, vidéo, musique, logiciels, ...) à concevoir des architectures complètes permettant le contrôle et la gestion des droits intellectuels tout au long de la chaîne de distribution et d'utilisation de ces contenus. On dénomme usuellement ces systèmes sous le terme générique de DRM pour « *Digital Rights Management* », même si d'autres terminologies ont été également (ou sont encore parfois) employées, comme par exemple *Electronic Copyright Management Systems* (ECMS), ou encore *Intellectual Property Management and Protection* - IPMP, ...).

Nous allons dans cette section fournir une vue d'ensemble sur les nouvelles normes

de DRM et les technologies qui déploient un cadre global pour traiter la protection des droits numériques et la propriété intellectuelle. Mais avant cela définissons ce qu'est une DRM.

Définition 7 *La gestion des droits numériques (DRM) est une technique proposant un ensemble de technologies permettant de protéger les droits d'auteurs en chiffrant les contenus et en n'autorisant qu'un accès limité et contrôlé en fonction des droits associés à ceux-ci.*

Ces mesures techniques de protection visent à limiter et à interdire plusieurs actions comme :

- Interdire la lecture d'un contenu hors de la zone géographique définie ;
- Limiter voire rendre impossible le transfert des contenus entre les appareils de lecture ;
- Interdire l'extraction numérique des contenus ;
- etc.

Toutes les formes de contenus médias peuvent être soumise à la gestion des droits numériques. Les fichiers musicaux représentent la plus grosse part des œuvres numériques étant protégés par ce procédé technologique. Ce sont avant tout les plates-formes de téléchargement payantes et légales sous la pression de l'industrie du disque qui utilisent en masse les DRM sur tous les titres vendus via l'Internet. Plus récemment, les fichiers vidéo font également l'objet de mesures de protection afin d'en limiter la copie et de lutter contre le piratage.

2.3.1 Principe de fonctionnement d'un système de DRM

Le procédé technologique des DRM peut se décomposer en quatre éléments, à savoir :

- L'encodeur : c'est lui qui va chiffrer les fichiers sources en leur ajoutant une protection ou une limitation ;
- Le serveur : c'est par lui que le fichier protégé va être diffusé ;
- Le lecteur : une fois téléchargé, l'utilisateur va avoir recours à un lecteur logiciel afin de pouvoir lire le fichier protégé ;
- Le contrôleur de la diffusion : qui se charge de la gestion des droits d'utilisation (nombre de lectures du fichier, le nombre de transferts sur d'autres appareils et la durée dans le temps de l'utilisation de ce fichier, etc.).

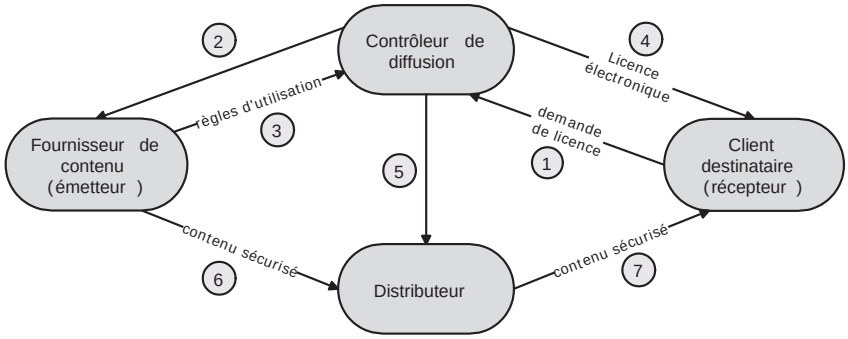


Figure 2.4 – Schéma du principe de fonctionnement d'un système de DRM

Les systèmes de DRM reposent sur l'utilisation de licences électroniques. L'idée est de séparer la possession du contenu des droits d'utilisation de ce contenu. Pour ce faire les contenus sont fournis par l'encodeur sous une forme altérée (chiffrée en général), inutilisable en tant que telle. Pour pouvoir lire un contenu il est nécessaire de disposer d'une «licence» fournie par le serveur pour les utilisateurs ayant le droit de lecture sur ce contenu.

Une licence est un fichier écrit dans un langage déclaratif décrivant les différents droits dont le titulaire dispose sur le contenu. Par exemple, elle peut autoriser la lecture, accorder la permission de le copier, définir une date après laquelle le fichier n'est plus utilisable ..., la licence contient également les informations nécessaires à l'utilisation du contenu telles que les clés de chiffrement utilisées.

Un langage permettant d'exprimer de telles licences se doit d'être simple et expressif afin de couvrir les besoins exprimés dans le langage naturel. David Parott en 2001 a mené une étude pour le groupe de travail sur la gestion des droits numérique MPEG-21 [24]. Dans cette étude, il analyse les différents cas d'utilisation des licences et spécifie les exigences que doit satisfaire tout langage d'expression des droits REL (Rights Expression Language). Des langages comme XrML (*eXtensible Rights Markup Language*) [20], ODRL (*Open Digital Rights Language*) [42] ou PDRL (*Portable Document Rights Language*) [2] ont été développés pour pouvoir exprimer de telles licences. Ces langages ont été proposés dans une perspective de protéger des données telles que du son ou de la vidéo, de spécifier les conditions d'utilisation d'un logiciel ou encore de contrôler l'accès à des web-services. La section suivante propose un panorama plus complet sur les différents REL utilisés de nos jours.

2.3.2 Langages d'expression des droits - REL

Le caractère stratégique du langage de description des droits dans la chaîne numérique d'un système de DRM, notamment parce qu'il est à la croisée des univers industriels et des producteurs de contenus, peut rendre compte de la très forte concurrence entre les standards appelés à normaliser le langage et donc les groupes d'acteurs industriels sur ce segment. Cette section a pour objectif d'exposer quelques éléments de base sur ces Langages d'Expression de Droits et faire un bref état des lieux sur les différents standards utilisés.

Définition 8 *Un REL est un type de langage informatique de haut niveau qui peut exprimer des instructions humaines pour interprétation, sans ambiguïté et dans une façon sécurisée, par un dispositif de traitement. Les instructions concernent ce qu'un propriétaire de droits permet à un utilisateur de faire avec un contenu. L'objectif principal d'un REL est de permettre à un système de DRM de contrôler, autant que voulu, l'utilisation de matériel ou de médias protégés délivrés sur un réseau ou sur un support physique quand ils sont accédés par des utilisateurs.*

Les REL sont souvent fondés sur le métalangage XML ce qui facilite leur interprétation par le lecteur humain ainsi que par les machines et qui rend plus aisé leur interopérabilité avec les autres technologies.

Historiquement, la technologie des langages d'expression de droits a été développée au début des années 1990 par le *Xerox Parc Research Centre*, à Palo Alto, par Mark Stefik, comme une branche de la recherche en intelligence artificielle [87, 88, 21]. Depuis, cette technologie est devenue de plus en plus élaborée. Aujourd'hui, même s'il n'existe pour l'instant aucun langage universellement commun pour l'expression des droits, deux langages majeurs disponibles dans une forme raisonnablement mature, l'un et l'autre fondés sur XML, il s'agit des langages ISO REL (basé sur XrML) et de ODRL. Cependant il existe plusieurs autres REL qui méritent d'être mentionnés sans être des concurrents majeurs. Le tableau 2.2 dresse une liste de plusieurs groupes de travail qui se sont formés autour de différents standards dans le domaine des REL.

2.3.2.1 ISO REL

Appelé aussi MPEG-21 REL ou MPEG REL, L'ISO REL est fondé sur XrML v2.0, il a été créé par le scientifique américain, Mark Stefik, qui est à l'origine du langage

Comité	Membres	Travaux
MPEG-21	IBM, Universal Music Group, NTT, Reuters, Microsoft, Intel, Philips, Panasonic ainsi que d'autres propriétaires de contenuS tels que MPAA et le RIAA	Standard ISO REL (MPEG 21 REL)
OASIS	Hewlett Packard, Verisign, Microsoft, IBM, Reuters, CommerceOne	Standard XrML
CC [a]	Matsushita, Philips, Samsung, Sony NBC, Twentieth Century Fox, Universal Music...	NEMO [b].
CMLA [c]	Intel, Nokia, Panasonic (Matsushita), Real-Networks, Warner Brothers Studios ...	Modèle de confiance pour les téléphones mobiles qui utilisent un système OMA DRM v2

a. Coral Consortium

b. Networked Environment for Media Orchestration, structure expérimentale développée par Intertrust (racheté en 2002 par Philips et Sony). Leur position tend vers l'établissement de relations de confiance sans toutefois adopter un REL commun

c. Content Management Licensing Administrator

Table 2.2 – Standards dans le monde des REL

DPRL (Digital Rights Permission Language) au début des années 1990. Depuis, le langage a été transcodé en XML et fournit la brique de base pour la norme internationale (ISO/IEC 21000/5), développé par le groupe MPEG

L'ISO REL a un modèle de données simple et extensible pour beaucoup de ses concepts clés (voir la figure 2.5-a). Le modèle de données inclut quatre entités de base. Ces entités sont intégrées dans une ***grant*** (permission). Structurellement, une *grant* consiste en les éléments suivants :

- L'élément ***principal*** (sujet) pour qui la permission est publiée ;
- L'élément ***right*** (action) que la permission spécifie ;
- L'élément ***resource*** (ressource) à laquelle s'applique l'action dans la permission ;
- L'élément ***condition*** qui doit être rencontrée avant que l'action ne puisse être exercée.

La *grant* (permission) n'est pas en soit une expression de droits qui peut être transférée sans ambiguïté d'une partie à une autre. Une expression de droits complète est décrite dans une ***License***.

Pour satisfaire les exigences de granularité fonctionnelle, une *license* type consiste

en une ou plusieurs *grants* et un *issuer*, qui identifie la partie qui a publié la licence. Une licence simple est illustrée par la figure 2.5-b.

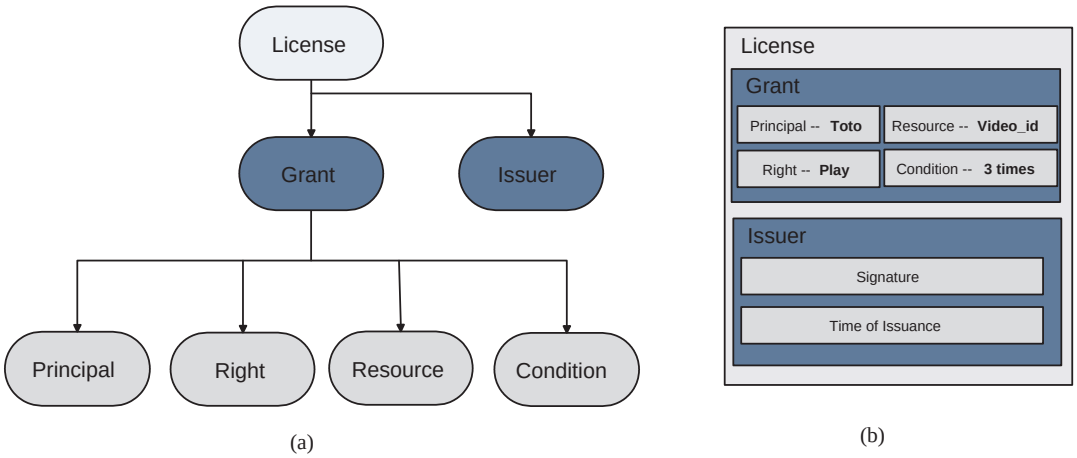


Figure 2.5 – (a) Modèle de données ISO REL - (b) Exemple de licence ISO REL

Dans cette figure, le *issuer* identifie la partie qui accorde la licence. Le *issuer* peut être authentifié par une signature, qui est utile pour traiter la question de la confiance et assurer l'intégrité de la licence. L'élément *issuer* peut aussi contenir des détails relatifs à l'émission de la licence (exemple, date de publication ou le mécanisme de révocation).

Le *issuer* de la licence, accorde les privilèges contenus dans la licence. L'élément *grant* est la partie de la licence ISO REL qui accorde à l'entité identifiée par l'élément *principal* le droit d'utiliser la *resource* sous certaines *conditions*. Par exemple, considérons un fichier vidéo (video_id) distribué à un utilisateur (Toto). Le document ISO REL possède une entrée qui exprime le fait que Toto a le droit de visionner 3 fois ce fichier vidéo. La figure 2.5-b montre l'élément *grant* qui spécifie cette règle.

2.3.2.2 ODRL

ODRL est une une proposition de standard pour l'expression des droits sur les contenus numériques. Un sous ensemble de ODRL est utilisé par le forum des industries leaders du mobile OMA (Open Mobile Alliance) dans le cadre du système OMA DRM.

ODRL permet de fournir des mécanismes flexibles et interopérables pour la publication, la distribution et l'usage transparent des ressources numériques telles que la musique, la vidéo, les logiciels et autres créations numériques. Ce langage est une recommandation du W3C, il ne nécessite pas de licence et est disponible librement.

ODRL est fondé sur un modèle extensible d'expression de droits qui comprend trois principales entités :

Party comprend les utilisateurs finaux ou les détenteurs de droits. Les détenteurs de droits sont les entités qui participent à la création, à la production, ou à la distribution des ressources numériques. Cet élément est analogue aux éléments *principal* et *issuer* du standard ISO REL ;

Permission représente les autorisations, qui peuvent contenir des contraintes, des obligations et des conditions. Les contraintes sont les limites imposées à l'usage de la ressource (par exemple regarder une vidéo au maximum 5 fois). Les obligations sont des prérequis pour l'obtention des permissions (par exemple payer 2 euros chaque fois afin de regarder la vidéo). Les conditions spécifient des exceptions, qui si elles sont satisfaites, révoquent les permissions ou entraînent la renégociation de celles-ci (par exemple si la carte de crédit expire alors toutes les permissions sur la vidéo sont révoquées) ;

Asset définit la ressource à protéger. Elle doit être identifiée de façon unique. C'est l'équivalent de l'élément *ressource* du standard ISO REL.

Considérons l'exemple précédent exprimé avec ISO REL à la figure 2.5-b. La figure 2.6-b donne sa représentation en ODRL en utilisant les entités *party*, *permission* et *asset*.

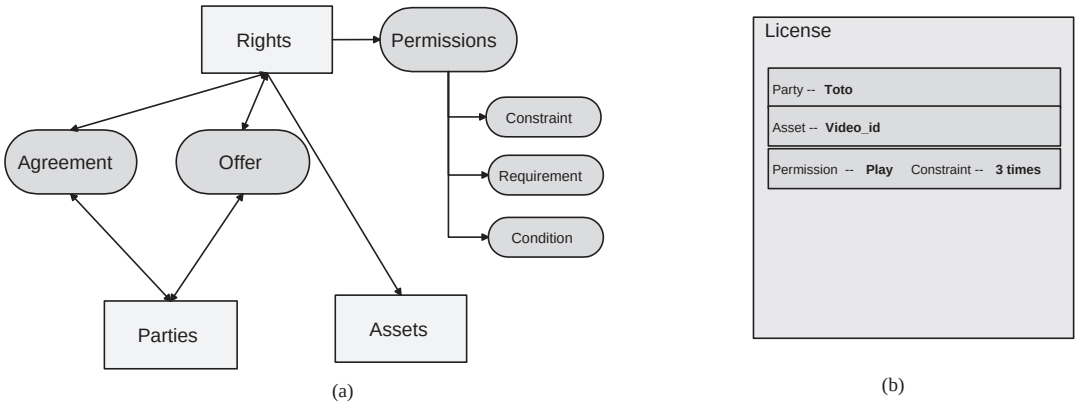


Figure 2.6 – (a) Modèle de données ODRL - (b) Autorisation en ODRL

2.3.2.3 XrML

Comme ODRL, XrML dans sa version 2.0 est un « schéma » au sens du W3C, c'est-à-dire un modèle générique permettant d'instancier des objets spécifiques conformes

à un standard. Une licence est construite sur une phrase, XrML repose sur la phrase suivante : « une licence est un ensemble de concessions qui procurent à certaines personnes certains droits sur certaines ressources sous certaines conditions ». En sachant qu'une concession et un droit sont des ressources, on peut produire des documents très complexes et une très large et flexible capacité de description.

XrML assure une intégrité totale des droits tout au long des chaînes de communication grâce à l'intégration d'un système de confiance. XrML implémente la possibilité de définir des organismes de certification, dont le rôle est d'assurer que les échanges sont conformes, par exemple qu'il n'y a pas d'abus d'identité ou de promesse non tenue. Cette fonction de superviseur peut se déléguer.

Comme cité précédemment, le standard ISO REL a pris XrML v2.0 comme base pour ses travaux. Cependant il existe une version antérieure de XrML encore active ; en l'occurrence la version 1.2. Celle ci étant adoptée par Microsoft comme REL intégré aux composants du services RMS (Rights Management Services).

Le fait que Microsoft ait utilisé la version antérieure de XrML (qui n'est pas un schéma mais plutôt une DTD au sens W3C), est une question politique de Microsoft. Cette version précédente de XrML emploie une sémantique qui diffère de l'ISO REL et les deux ne sont pas vraiment interopérables. Les travaux sur XrML 1.2 continuent au sein de Microsoft et ne seront plus conduit par ContentGuard, qui se concentre sur le développement de l'ISO REL.

2.3.3 Discussion

Nous avons vu dans cette section le principe général des systèmes de DRM ainsi qu'un panorama des standards dans ce domaine. Vu d'un certain angle, la problématique traitée dans cette thèse relève aussi de la gestion des droits, même si le but n'est pas de gérer les accès au contenu du destinataire final, l'objectif peut être assimilé à la gestion des droits d'accès aux contenus au niveau des *proxies* d'adaptation. L'idée serait alors de gérer les accès aux différents objets-médias avec le principe des DRM.

L'étude que nous avons effectué ici nous a donc permis de dégager, parmi les différentes techniques utilisées dans les systèmes de DRM, celle qui correspond le mieux à nos attentes en termes de gestions des profils, contrôle d'accès, gestions des droits, etc., rajouté à cela, la disponibilité des implémentations et surtout la complémentarité avec nos approches pour la sécurisation des contenus.

La seconde partie de ce mémoire est consacrée à la description de nos différentes approches ; nous y présentons les détails concernant notre utilisation des techniques de DRM pour la gestion des accès aux contenus lors des opérations d'adaptation.

2.4 Synthèse

Ce chapitre a été axé sur le concept de la sécurité dans le multimédia, nous y avons décrit les différentes briques de base pour la construction d'un système de sécurisation de contenus multimédias.

Après avoir exhibé le catalogue des critères de sécurité exigés par nos objectifs, nous avons présenté les techniques permettant de garantir ces critères, en l'occurrence les techniques cryptographiques, le tatouage numérique et le contrôle d'accès. La transposition de ces solutions à notre problématique nous a permis de dessiner les grandes lignes de notre approche pour la sécurisation des contenus multimédias adaptables par des *proxies* d'adaptation. Ainsi nous avons vu que la meilleure approche serait de combiner les techniques de signature numérique, chiffrement et contrôle d'accès.

Dans un deuxième temps, nous avons présenté les systèmes de Gestion des Droits Numériques (DRM) dans le domaine des médias. Après avoir présenté le principe général de ces système, nous avons constaté une similitude entre l'objectif de ces systèmes et nos besoins en termes de gestion d'accès aux contenus à la différence que nous cherchons à contrôler ces accès au niveau des *proxies* d'adaptation et non au niveau des destinataires. Nous avons donc présenté un état de l'art sur les langages d'expression des droits (REL) afin de dégager celui qui conviendrait le mieux à notre approche pour le mécanisme de gestion de droits.

Toutes les décisions qui ont résultées des études présentées dans ce chapitre seront concrétisées par les approches proposées dans la partie deux de ce mémoire. Le chapitre 4 présentera nos techniques de signature et de chiffrement des contenus multimédia, tandis que le chapitre 5 présente notre architecture de diffusion complète englobant notre mécanisme de gestion des droits.

Chapitre 3

Intermédiaires d'adaptation et sécurité, état de l'art

«... By breaking the end-to-end nature of the communication, proxies render the task of providing end-to-end security much harder or even impossible in some cases ...».

M. Portmann dans [77]

Après avoir introduit dans le chapitre précédent la notion de sécurité dans les flux multimédias et les techniques utilisées pour garantir cette sécurité, nous allons dans ce chapitre aborder les spécificités, toujours sous l'angle de la sécurité, des systèmes multimédias utilisant des intermédiaires d'adaptation.

Nous commençons dans la section 3.2 par rappeler les failles en termes de sécurité caractérisant ces réseaux ainsi que quelques techniques d'attaque possibles sur ce type de plate-forme.

Nous tâcherons ensuite de présenter un rapide état de l'art sur les solutions traitant de la sécurité en présence de *proxies* d'adaptation ; la section 3.3, définit une classification des solutions proposées à ce jour dans la littérature, puis fait un tour d'horizon des approches relevant de la problématique de l'adaptation de contenus par des opérateurs intermédiaires.

Enfin, nous synthétisons dans la section 3.4 les différents aspects abordés dans ce chapitre en présentant nos conclusions de ces études.

3.1 Introduction

L'article de Portmann [77] décrit d'une manière très pertinente la problématique de sécurité liée à l'utilisation des *proxies* d'adaptation et la résume parfaitement par la phrase suivante :

«... By breaking the end-to-end nature of the communication, proxies render the task of providing end-to-end security much harder or even impossible in some cases...»

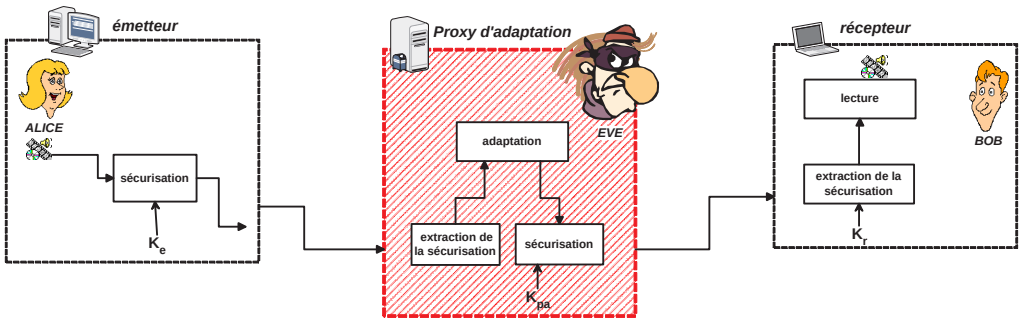


Figure 3.1 – Vulnérabilité des flux au niveau du *proxies* d'adaptation

La sécurité bout-en-bout dans les systèmes multimédias (dans notre contexte) implique la garantie de la confidentialité, de l'intégrité et de l'authentification des documents échangés, de l'émetteur jusqu'au récepteur final sans interruption. Cela dit, l'utilisation de *proxies* d'adaptation rend cet objectif difficile ; en effet, l'adaptation de contenu par des *proxies* nécessite souvent l'accès de ces derniers au contenu à adapter afin de fournir ce service. Cette contrainte pose des restrictions importantes en termes de sécurité ; l'intégrité des données, par exemple, dont le but est de détecter n'importe quels changements apportés aux données lors de l'acheminement, se voit fortement compromise du fait des adaptations apportées par les *proxies* d'adaptation.

Le même problème se pose pour le chiffrement des contenus. Pour exécuter une adaptation, un *proxy* a besoin de l'accès aux données en clair, car il est souvent impossible de transformer des données chiffrées d'un format vers un autre sans avoir à révéler leur contenu. Bien qu'il existe quelques travaux intéressants qui proposent des techniques d'adaptation de contenus chiffrés, aucune solution générique n'existe pour tout type d'adaptation.

Par ailleurs, même si le *proxy* dispose des droits pour accéder au contenu (chiffré) à adapter (*proxy* de confiance), il reste vulnérable à une attaque visant à récupérer le flux multimédia une fois déchiffré. Les *proxies* d'adaptation ouvrent ainsi une faille de sécurité importante dans le système multimédia.

3.2 Les menaces et les vulnérabilités

La manière «standard» d'adapter un contenu chiffré pour un *proxy* d'adaptation est de le déchiffrer à la réception, l'adapter puis de le rechiffrer. En partant de ce postulat, l'étude que nous avons mené concernant les failles de sécurité nous a conduit à distinguer deux types de techniques d'attaque possibles ; des attaques dites **à point simple** et des attaques dites **multipoint** ou par collusion (voir la figure 3.2). Nous allons dresser, dans la suite de cette section, une liste non exhaustive de ces différents points d'attaque possibles dans notre contexte.

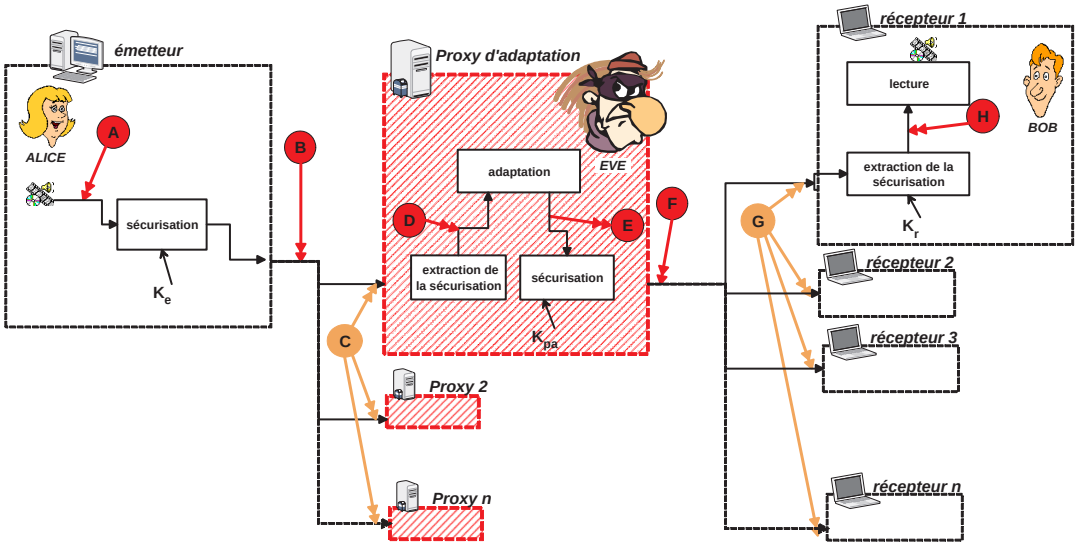


FIGURE 3.2 – Attaques possibles sur un système multimédia adaptatif

3.2.1 Attaques à point simple

Un point d'attaque simple signifie une menace de sécurité de base qui est capable de mettre en danger la sécurité bout-en-bout d'une communication. Plusieurs points d'attaques simples peuvent être identifiés dans notre système de communication à base de *proxies* d'adaptation. Nous les avons identifiés dans la figure 3.2 par les points A, B, D, E, F et H.

A- Attaque de la copie originale : l'attaque de la copie originale suppose que le document multimédia original soit piraté ou illégalement redistribué avant que n'importe quelle opération de protection ne lui soit imposée, comme il est montré dans la figure 3.2. En général, la version originale appartient exclusivement à l'émetteur du contenu multimédia, à qui on doit garantir que ce

contenu ne sera pas exposé au risque d'être piraté.

B- Attaque par Snooping : l'attaquant ici peut faire une écoute clandestine sur la liaison réseau entre l'émetteur et le *proxy* d'adaptation comme le montre l'attaque B de la figure 3.2 pour récupérer le flux émis. Si le document multimédia est transmis en clair ou bien si la clé de chiffrement utilisée n'est pas assez sécurisée, les données écoutées peuvent être retrouvées et illégalement redistribuées sans aucun effort. De ce fait, pour une transmission sécurisée, le flux émis devrait être correctement chiffré avant toute transmission.

D- Attaque de la copie déchiffrée : appelée dans d'autres contextes attaque en entrée de la Set-top Box. Ici l'attaquant au niveau du *proxy* récupère le flux après son déchiffrement et avant les opérations d'adaptation. Cela est possible en local via un utilisateur sur un *proxy* de non-confiance ou bien à distance via des méthodes d'écoute sur la machine du *proxy*.

E- Attaque du flux adapté : appelée dans d'autres contextes attaque à la sortie de la Set-top Box. Comme pour l'attaque précédente, les flux déchiffrés peuvent être piratés à la sortie des opérations d'adaptation.

F- Attaque de la copie adaptée chiffrée : à l'image de l'attaque B, le flux chiffré sortant du *proxy* vers le destinataire pourrait être écouté par un attaquant au niveau du lien réseau. Là aussi, si la communication n'est pas assez sécurisée de sorte que les clés utilisées puissent être calculées, la sécurité du système entier est dégradée et le flux peut être déchiffré.

H- Attaque de la copie reçue : quand le document est déchiffré à la réception, il devient vulnérable à un piratage éventuel. Il est donc impératif de s'assurer d'un niveau de protection élevé sur la machine du récepteur de contenu.

3.2.2 Attaque multipoint par collusion

Une attaque multipoint par collusion est une combinaison de plus d'une attaque à point simple. Dans cette attaque plusieurs exemplaires de documents chiffrés sont rassemblés ; ils sont ensuite combinés pour calculer (via des techniques de cryptanalyse par exemple) la clé utilisée pour le chiffrement des documents afin de déchiffrer ces derniers. La figure 3.2 montre deux possibilités d'attaques multipoint :

C- Attaque multipoint des *proxies* : dans le cas d'un envoi de l'émetteur vers plusieurs *proxies* , les différentes versions reçues par ces derniers sont utilisables pour ce type d'attaque.

G- Attaque multipoint des récepteurs : dans le cas d'un envoi du *proxy* vers plusieurs récepteurs, les différentes versions reçues par ces derniers sont aussi utilisables pour ce type d'attaque.

3.2.3 Discussion

L'étude présentée dans cette section a porté sur l'analyse des failles de sécurité lors d'une communication, de bout-en-bout, dans un système multimédia utilisant des *proxies* d'adaptation. Nous avons donc recensé tous les points d'attaque possibles sur une telle communication. Néanmoins, parmi ces différentes failles, quelques unes seulement sont directement liées à l'utilisation des *proxies* d'adaptation. Il s'agit typiquement des attaques C, D, E et G décrits par la figure 3.2. Les autres attaques étant clairement des attaques «génériques», que l'on peut retrouver dans tous type de plates-formes de communication, de nombreuses solutions de sécurité sont présentées dans la littérature permettent de se protéger contre ces failles [85, 92, 54, 66, 64, 94, 25, 32, 28].

Ainsi, dans la suite de ce chapitre (et du document) nous allons nous restreindre aux risques liés aux *proxies* d'adaptation et aux solutions permettant de sécuriser les failles provoqués par ces derniers.

La section suivante présente donc un état de l'art sur les solutions traitant les risques provoqués par la présence de *proxies* d'adaptation dans les architectures et dans différents contextes.

3.3 Travaux existants

La problématique de la sécurité liée à l'utilisation des *proxies* d'adaptation dans les systèmes de communication multimédia est assez récente comparée aux premiers travaux sur la sécurité dans les réseaux. Elle a été souvent posée notamment dans l'article [77] de Portmann cité précédemment ; ainsi que par le groupe de travail de l'IETF OPES WG qui présente dans [8] les menaces et les risques en termes de sécurité pour les services déployés au niveau des *proxies* d'adaptation dans l'architecture OPES [1][40, 8] ; ou encore, plus récemment, dans l'article de Zeng et al [103]. Ces travaux décrivent parfaitement les risques et les failles liés à l'utilisation de *proxies* d'adaptation, mais ne proposent pas de solutions pour palier à cela.

1. L'architecture OPES (*Open Pluggable Edge Services*) est une architecture proposée par l'IETF afin de permettre le déploiement des services appliqués sur les données d'une application par des entités intermédiaires du réseau. Son objectif est de définir un protocole et un ensemble d'APIs pour l'application d'un ensemble de services assurant une transmission efficace d'un contenu complexe et l'utilisation de services relatifs au contenu. Les entités intermédiaires peuvent être des *proxies* ou des serveurs auxiliaires au système.

Par ailleurs, ce qui a retenu notre attention, dans la littérature, lors de la réalisation de cet état de l'art est que plusieurs travaux proposant (ou utilisant) des *proxies* d'adaptation multimédia soulèvent la question des risques en termes de sécurité ; mais souvent, cette question est laissée en perspective de ces travaux, sans y apporter de solution.

Néanmoins, nous avons pu trouvé dans la littérature plusieurs réflexions qui tentent de résoudre cette problématique pour des flux médias. La plupart de ces études traitent de l'adaptation de contenus vidéo. Nous avons étudié quelques travaux parmi ceux-la, que nous allons présenter dans le paragraphe 3.3.2.

Parallèlement à cela, nous avons trouvé d'autres travaux relevant de la sécurité lors de l'utilisation d'autres types de *proxies* tels que les *proxies* de cache (par exemple). Le paragraphe 3.3.3 présente un panorama de quelques approches utilisées.

Toutefois nous avons pu trouver quelques travaux sur des contenus composites ou des formats multimédias spécifiques, que nous présenterons dans le paragraphe 3.3.4.

3.3.1 Critères d'étude

La plupart des classifications de solutions de sécurisation de l'adaptation au niveau *proxy* existantes se focalisent sur des domaines d'application particuliers comme la télé-diffusion (*streaming*), les images scalables, ou encore les flux audio. De plus, les classifications existantes sont rares et se concentrent sur un sous-ensemble de critères [103]. Dans cette état de l'art, nous proposons une étude des différentes solutions de gestion de la sécurité de bout-en-bout, dans des architecture à base de *proxies*, en se fondant sur les critères suivants :

- type de média : afin de distinguer les solutions traitant des flux (single) média des flux traitant des flux multimédia (au sens défini dans le chapitre 1) ;
- type d'adaptation : ce critère définit le type de transformation applicable au flux (*cf.* section 1.2.1) : Transmodage, Transcodage et Transformation ;
- stratégie d'adaptation : ce critère définit la stratégie d'adaptation utilisée. Comme présentée dans la section 1.2.1, la stratégie d'adaptation peut être statique, dynamique ou temps réel ;
- niveau de sécurité : ce critère permet de distinguer les différents objectifs, en termes de sécurité, garantis par une solution. Dans notre étude les niveaux de sécurité considérés sont l'authentification, la confidentialité et l'intégrité des documents (*cf.* section 2.1).

Dans la suite de cette section, nous classons les solutions qui traitent le problème de la sécurité de l'adaptation multimédia suivant la granularité manipulée par les applications. Nous donnons pour chaque solution notre position vis-à-vis des autres critères cités ci-dessus. Le tableau 3.1 page 64 donne un récapitulatif des différentes solutions étudiées.

3.3.2 Solutions traitant l'adaptation vidéo

La gestion de la sécurité des opérations d'adaptation sur des *proxies* a été souvent abordée dans le cadre de l'adaptation de flux vidéo. Les solutions proposées peuvent être réparties en deux approches distinctes selon la granularité du contenu : (i) approches par blocs, (ii) approches par couches.

Nous décrivons plus en détails dans cette section trois propositions présentes dans la littérature utilisant l'une des deux approches sus-citées : les travaux de Wee et d'Apostolopoulos dans [3, 102, 4], les travaux de Gentry et al dans [37], et les travaux de Tieyan Li dans [63, 61].

3.3.2.1 SSS

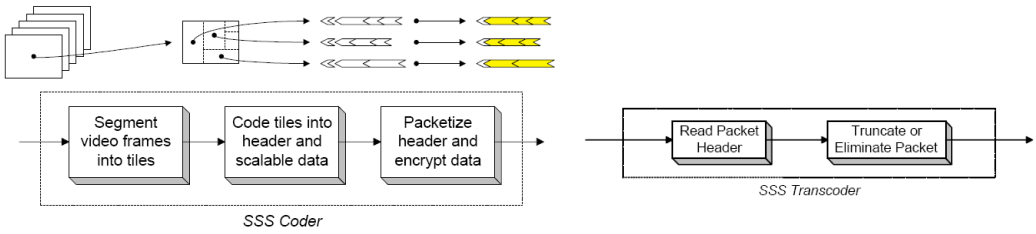


Figure 3.3 – Solution SSS (Secure Scalable Streaming)

SSS (*Secure Scalable Streaming*) est une une technique de chiffrement de contenu proposée par Wee et Apostolopoulos pour des contenus vidéo scalables MPEG-FGS [101] et pour des images scalables JPEG 2000 [102], puis portée sur les formats vidéo non scalables [4].

L'idée est, comme le montre la figure 3.3, de diviser le flux en paquets, chaque paquet contenant plusieurs blocs ; toutes les données dans chaque paquet (sauf les champs des entêtes sont chiffrés par bloc (avec un algorithme tel que DES) en utilisant le

mode CBC *«Cipher Block Chaining»* [2]. Parallèlement à cela, des informations sont insérées dans l'entête non-chiffrée de chaque bloc pour permettre des opérations de suppression éventuelles de manière optimale.

Cette technique exploite le fait que le déchiffrement d'un *cipher block* dans le mode CBC est causal, autrement dit, le déchiffrement d'un bloc ne dépend pas des blocs précédents, de sorte que la suppression des données de lien dans un paquet n'affecte pas le déchiffrement des blocs précédents dans le paquet.

Les opérations d'adaptation permises dans SSS sont donc la suppression de paquets dynamiquement pour un transcodage du média émis.

3.3.2.2 LISSA et TRESSA

Gentry et al proposent dans [37] deux techniques d'authentification de contenus streamés adaptables par des *proxies*. Les deux techniques concernent des flux codés en couche et sont fondées sur des méthodes de hachage particulières ; LISSA utilise un schéma linéaire tandis que TRESSA utilise des arbres de hachage appelés arbres de hachage de Merkle.

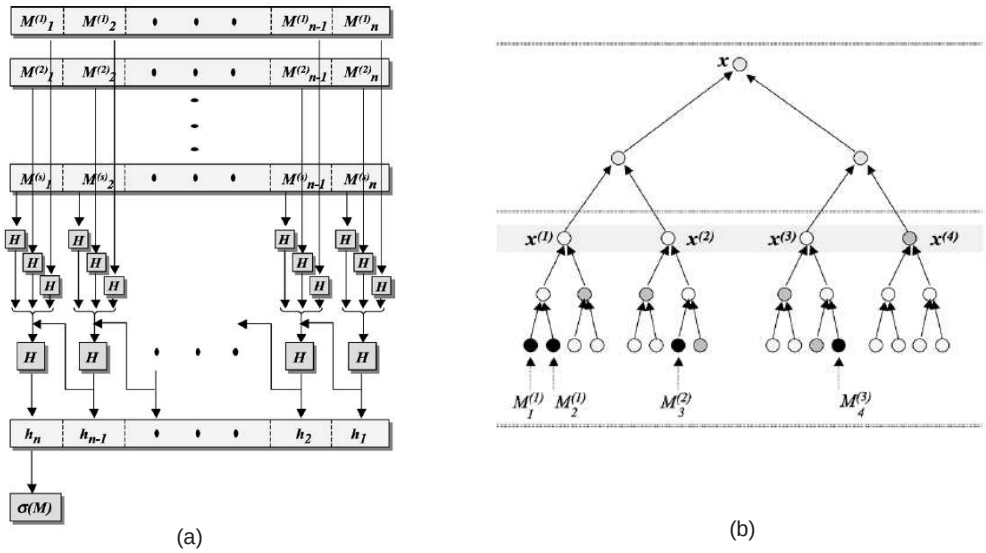


Figure 3.4 – (a) Solution LISSA (b) Solution TRESSA

2. Le mode CBC (Cipher Block Chaining) rajoute au chiffrement par bloc un mécanisme de retour. Chaque bloc chiffré Y_i est additionné par un ou exclusif avec le bloc clair rentrant X_{i+1} avant d'être chiffré avec la clé k, d'où la notion de «chaîne». Un vecteur d'initialisation (IV) est utilisé pour la première itération

Les schémas proposés permettent au *proxy* d'adaptation d'enlever sélectivement des parties du flux média et, ainsi, de permettre aux applications d'effectuer des opérations de transcodage telles que la compression scalable. De plus, un fournisseur de contenu n'encode son flux qu'une seule fois pour des envois multiples, par opposition au codage et signature non dynamiques de plusieurs versions différentes pour chaque combinaison prévue de *device*, configuration de réseau et les conditions du canal de transmission.

3.3.2.3 SSMS

Secure Scalable Multimedia Streaming [15] est un projet qui vise aussi à résoudre le conflit inhérent entre l'adaptabilité de contenus vidéo scalable et la sécurité en proposant le framework SSMS. Ce dernier implique la structuration des données chiffrées utilisant XML et ajoutant des metadonnées qui fournissent les informations nécessaires pour le transcodeur pour adapter le contenu sans avoir à le déchiffrer. Cela permet une sécurité de bout-en-bout maintenant une pleine adaptabilité du contenu.

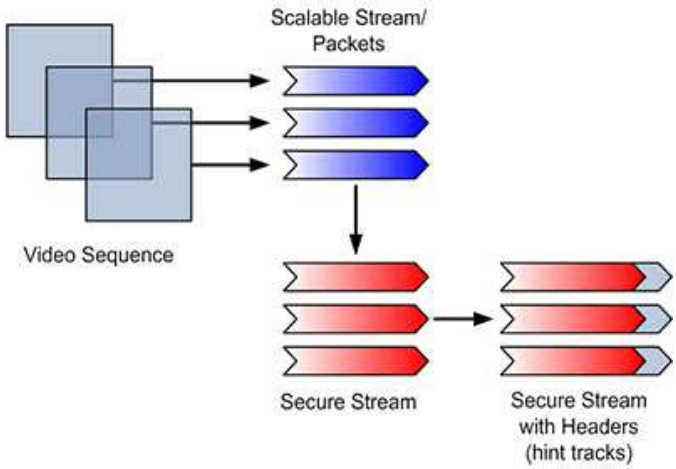


Figure 3.5 – SSMS : *Secure Scalable Multimedia Streaming*

Les opérations d'adaptation permises dans SSMS sont, à l'image de SSS, la suppression de paquets chiffrés dynamiquement pour un transcodage du flux vidéo émis.

3.3.3 Solutions traitant divers *proxies*

En dehors de l'adaptation multimédia, les *proxies* sont utilisés dans beaucoup d'autres domaines. Les risques en termes de sécurité, cela dit, sont un peu différents mais parfois très proches. Nous présentons dans cette section deux approches de sécurisation de contenu lors de l'utilisation de *proxies* de communication et de cache.

3.3.3.1 MC-SSL

MC-SSL (*Multiple-Channel SSL*) est une solution proposée par Song et al. dans [86] dans le cadre de la sécurisation de communication réseau utilisant des *proxies*. MC-SSL est un protocole de sécurité qui étend le protocole SSL (fonctionnant en mode client-serveur) pour obtenir un fonctionnement Client-*Proxy*-Serveur. L'objectif de MC-SSL est de fournir une solution de sécurisation de communication dans tout type d'applications et de réseaux utilisant des *proxies* , et plus particulièrement les applications mobiles.

Le protocole MC-SSL a été conçu à l'origine comme une solution pour les problèmes de sécurité pour le WAP 2.0, qui implique une confiance inconditionnelle des applications *proxies*. Cependant, le protocole implémenté est assez générique pour être utilisé dans d'autres domaines comme la sécurité scalable. Une mise en oeuvre utilisant l'API Java JSSE[3] [46] a été proposée, celle ci permet d'évaluer les capacités d'optimisation de performances lors de l'utilisation de MC-SSL.

3.3.3.2 CoDeeN

CoDeeN présenté par Wang et al. dans [100] est un système de *proxies* de cache pour le web, créé à l'université de Princeton et déployé pour un usage général sur le réseau PlanetLab [76]. Le principe général de CoDeen est de permettre aux utilisateurs de fixer (déléguer) leurs caches internet chez un *proxy* à proximité participant au réseau PlanetLab. Les *proxies* coopèrent, ensuite, entre eux, afin de fournir collectivement un accès plus robuste et plus rapide aux contenus mis en cache pour les autres utilisateurs du réseau.

Pour la question de sécurité des contenus mis en cache dans CoDeen, les auteurs proposent un mécanismes fondé sur un modèle de contrôle d'accès, fixant une clas-

3. Java Secure Socket Extension

sification des intervenants dans la session et des privilèges pour chaque classe de nœuds.

Pour réaliser leur politiques d'accès, ils classent les adresses IP des différents nœuds en trois groupes :

- les adresses locales du nœud dans CoDeeN
- les adresses de tout site abritant un nœud PlanetLab
- et enfin les adresses externes au réseau PlanetLab

Les contenus sont alors sécurisés, et une licence d'accès (distribuée selon la classe/privilège des nœuds) est nécessaire pour l'accès à ces contenus.

3.3.4 Solutions traitant l'adaptation de flux Multimédia

L'article de Zeng et al [103] décrit la problématique de sécurité liée à l'utilisation des PAs dans les systèmes multimédias, et propose un panel de direction de recherche pour des architectures et des solutions possibles. Même si nos travaux ont commencé bien avant la publication de cet article, nous trouvons que cette manière de décrire la problématique est assez pertinente, nous en donnons donc les grandes lignes dans cette section.

3.3.4.1 SESSC

«Selective Encryption and Scalable Speech Coding» est une autre solution utilisant un chiffrement sélectif. Elle fut proposée par Gibson et all dans [38] dans le cadre de la sécurisation de contenu audio (voix) sur des réseaux sans fil.

La solution consiste à diviser le contenu en plusieurs couches en fonction de l'importance lors de la création de contenu, et de sécuriser les différentes couches séparément avec des annotations de metadonnées. Lorsqu'une adaptation de contenu est nécessaire, il suffit lors d'élaguer les couches les moins importantes en se fondant sur les metadonnées. Ceci est particulièrement approprié lorsque le contenu multimédia est composé de divers médias, par exemple, texte, audio et vidéo qui peuvent être sécurisés séparément et de façon sélective en fonction de la variation de bande passante/ressources contrainte.

3.3.4.2 mSSA

L'un des projets les plus intéressants pour notre état de l'art est le projet *mSSA* proposé par Li et al dans [63, 61], un système d'authentification de *Streaming* MPEG-4 scalables (codés en couche) sur des nœuds intermédiaires. L'objectif de *mSSA* est, entre autre, de supporter des altérations causées par les pertes de paquets sur le réseau. Il traite aussi le problème d'un flux multi-source Il combine des techniques de correction d'erreur (*ECC*) et de signature (*Arbres de Merkle*). L'idée proposée par les auteurs est de signer chaque couche séparément utilisant les propriétés des arbres de Merkle pour permettre aux PAs la suppression des couches de raffinement tout en gardant la possibilité de vérification de signature par le récepteur.

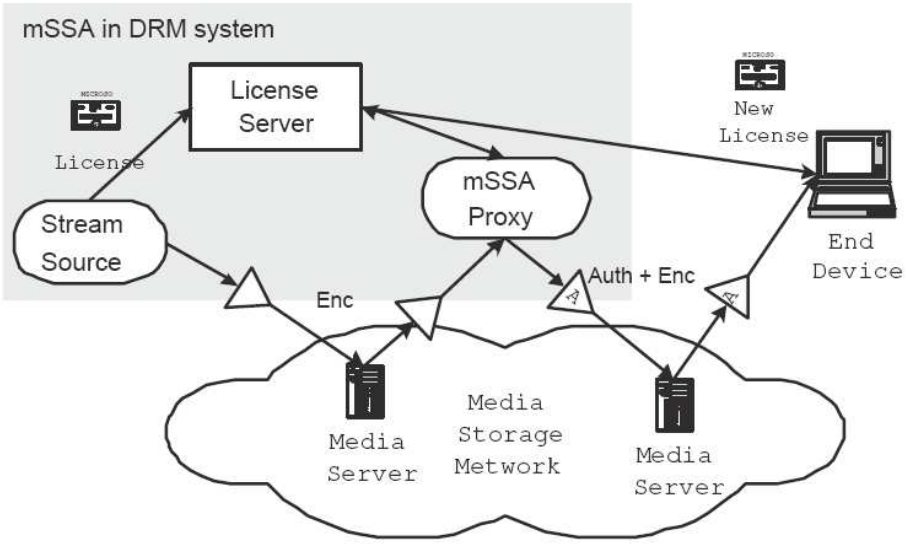


Figure 3.6 – Architecture de la solution mSSA

3.3.5 Discussion

Après avoir passé en revue ces quelques solutions relevant de la sécurité lors de l'utilisation de *proxies*, le tableau 3.1 ci-dessous récapitule l'essentiel des points remontés de cette étude.

Critère \ Solution	SSS	LISSA	TRESSA	SSMS	MC-SSL	CODEEN	SESSC	MSSA
format	video	video	video	video	/	/	multi	multi
adaptation statique					/	/		
adaptation dynamique	X	X	X	X	/	/	X	X
adaptation temps réel			X	X	/	/	X	X
transmodage								/
transcodage	X	X	X	X	X	X	X	/
transformation	X	X	X	X			X	/
Authentification		X	X					X
Confidentialité	X			X	X	X	X	
Intégrité		X	X				X	X

TABLE 3.1 – Tableau récaputulatif des solutions étudiées

3.4 Synthèse

Dans la première partie de ce chapitre, nous avons étudié les risques, en termes de sécurité, liés à l'utilisation de *proxies* d'adaptation dans une plate-forme de communication multimédia ; cette étude fait ressortir deux types d'attaques possibles sur les contenus échangés.

Tout d'abord, les attaques dites à points simple ; nous en avons énuméré six. Parmi cela nous retrouvons des attaques «classiques» qui concernent tous les type de réseaux de communication, ainsi que des attaques liées directement à l'utilisation de *proxies* d'adaptation.

Deuxièmement, nous avons présenté les attaques dites à points multiples ; celles-ci impliquent la fédération de plusieurs pirates ou plusieurs flux attaqués pour le calcul des clés de chiffrement utilisées par l'émetteur.

Ensuite, nous avons présenté dans ce chapitre un état de l'art sur les solutions traitant de la sécurité liée à l'utilisation des *proxies* dans les réseaux de communication. Pour cela, nous avons classé les solutions étudiées selon des critères d'études bien définis, en dégageant les différentes approches possibles dans chacun des domaines d'utilisation des *proxies*.

Nous avons ainsi décrit des solutions proposées pour des *proxies* d'adaptation de contenu vidéo, des *proxies* de cache, différentes autres architectures à base de *proxies* et quelques solutions relatives à l'utilisation de *proxies* d'adaptation de contenus

multimédias.

Cette étude nous a permis d'analyser les différentes techniques utilisées pour assurer chacun des niveaux de sécurité visés par nos travaux, et ce, dans les différents domaines abordés. C'est sur cet état de l'art que nous nous sommes appuyé pour proposer nos approches de sécurisation de contenu pour des flux multimédias. Notamment pour l'utilisation des Arbres de Hachage de Merkle (*cf.* section 4.2.2.2) comme structure de hachage pour l'authentification des documents.

La seconde partie de cette thèse, qui commence dès le chapitre suivant, présente nos différentes solutions pour la sécurisation des opérations d'adaptation de contenus multimédias sur des nœuds intermédiaires.

Deuxième partie

Contributions

Chapitre 4

Sécurisation des documents multimédias adaptables

La première partie de ce document a permis d'abord de se familiariser avec les plates-formes d'adaptation de contenus multimédias (*cf.* chapitre 1) et le monde de la sécurité dans le multimédia (*cf.* chapitre 2). Ensuite, nous avons parcouru, à travers le chapitre 3, une synthèse de la littérature concernant les solutions proposées dans le cadre de la sécurisation de flux adaptables. Cette deuxième partie présente les différentes contributions de cette thèse.

Dans ce chapitre, nous présentons nos approches pour la sécurisation de contenus multimédias pouvant être transmis en présence d'opérateurs intermédiaires d'adaptation dans la chaîne de diffusion entre un émetteur et un récepteur. Les deux contributions présentées sont AMCA [1] pour l'authentification des contenus et XSST [2] pour le chiffrement de ces derniers.

Dans un premier temps, nous allons commencer par rappeler les objectifs visés par les solutions proposées et décrire les motivations qui ont guidé nos choix pour ces approches. Ensuite nous donnerons une présentation détaillée de chacune de ces deux solutions, la section 4.2 présentera l'approche AMCA, les concepts théoriques sur lesquels elle repose, son fonctionnement en l'illustrant par des exemples d'utilisation, tandis que la section 4.3 présente les détails de l'approche XSST, en commençant par les concepts théoriques qui la guident, son fonctionnement et des exemples d'utilisation.

1. AMCA est l'acronyme de *Adaptive Multimedia Content Authentication*.
2. XSST est l'acronyme de *eXtended Secure System Transaction*.

4.1 Motivations et objectifs

Le travail que nous présentons dans cette thèse s'intéresse à la sécurité dans les plates-formes d'adaptation de contenus multimédias utilisant des *proxies*. L'architecture référence considérée dans nos recherches et pour la mise en œuvre de nos solutions est donc composée d'émetteurs/récepteurs et de *proxies* d'adaptation. Pour simplifier la présentation de nos solutions nous allons considérer une architecture référence présentée dans la figure 4.1. Cette architecture, que nous désignerons aussi dans la suite de document par «**Architecture** $\mathcal{SPC}$», se compose d'un émetteur de contenus multimédias que nous appellerons «$\mathcal{S}$erveur» et d'un récepteur que nous appellerons «$\mathcal{C}$lient», les communications passent via un nœud intermédiaire que nous appellerons indifféremment «$\mathcal{P}roxy$ d'adaptation» ou bien $\mathcal{PA}$, ce dernier adapte le contenu émis par le serveur avant transmission au client destinataire.

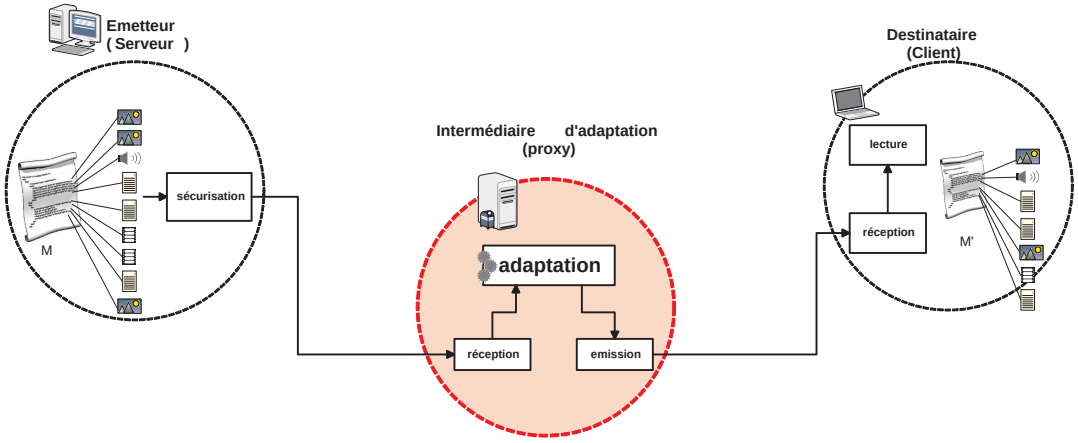


Figure 4.1 – Architecture $\mathcal{SPC}$ d'un système multimédia à base de *proxy* d'adaptation

L'étude que nous avons présentée dans le paragraphe 2.2.4 du chapitre 2 nous a conduit à opter pour une approche cryptographique afin de garantir tous les besoins de sécurité nécessaires dans notre contexte. Plus précisément, nous avons choisi la signature numérique pour assurer l'authentification des contenus et le chiffrement pour assurer l'intégrité et la confidentialité des données.

Néanmoins, comme nous l'avons souligné précédemment, les définitions classiques de la signature et du chiffrement numériques interdisent toute modification intermédiaire (et par conséquent adaptation) des documents sécurisés. La solution que nous recherchons pour notre problématique a pour objectif de palier à cette «déficience» en définissant une nouvelle technique générique permettant de contrôler la sécurité d'un document multimédia ayant éventuellement subi des opérations d'adaptation.

Par conséquent, cette solution doit permettre à des entités bien déterminées l' ajout de nouveaux objets-médias ou la suppression/modification d'objets-médias initiaux dans une présentation originale. Elle doit ainsi :

- garantir le maintien de l'identité de l'émetteur, tout en identifiant tous les opérateurs d'adaptation intermédiaires éventuels ;
- offrir aux émetteurs de contenus multimédias adaptables la possibilité de garder confidentiel les (ou une partie des) données émises et ce jusqu'aux destinataires finaux.
- éviter tout risque de piratage ou d'attaque au niveau des opérateurs d'adaptation ou au niveau des clients récepteurs.

Comme nous l'avons présenté dans le chapitre 3, plusieurs approches de sécurisation de contenus adaptables [90, 63, 12] ont tenté de trouver des solutions pour signer/chiffrer des flux adaptables en combinant signature/chiffrement numérique avec d'autres solutions de sécurisation (*watermarking*, hachage, etc.). Cependant, la plupart de ces approches sont des réponses «ad hoc» à des contextes précis ou à des formats de flux précis.

C'est en se reposant sur cet état de l'art et avec les objectifs présentés ci-dessus, que nous avons proposé AMCA et XSST comme solutions de sécurisation de contenus multimédias adaptables.

Les deux sections suivantes présenteront chacune de ces solutions et détailleront les techniques utilisées.

Par ailleurs, dans le but d'illustrer les techniques proposées dans les sections suivantes, nous allons dérouler nos approches sur un cas d'application réel nécessitant une sécurisation de contenus multimédias et utilisant des intermédiaires d'adaptation. L'application exemple utilisée et dont les intervenants sont représentés par La figure 4.2 est une application de télémedecine dont le but est de permettre à un médecin de transmettre le diagnostic d'un patient, sous forme d'un document multimédia, à plusieurs collègues distants disposant de terminaux de différentes capacités. Nous déroulerons par la suite chacune de nos solutions sur ce cas.

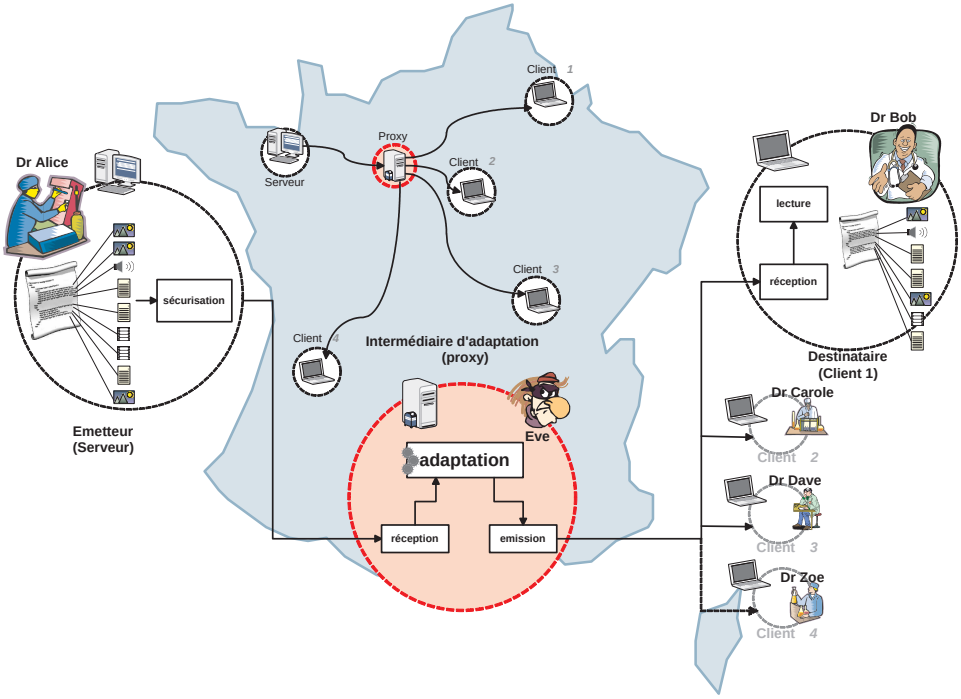


Figure 4.2 – Application exemple d'une adaptation multimédia sur un *proxy*

4.2 AMCA pour l'authentification des documents Multimédias

Afin d'authentifier les documents multimédias nous utilisons le principe général de la signature électronique, autrement dit, l'émetteur signe son contenu et le récepteur contrôle la validité de cette signature. Mais l'utilisation d'une technique de signature «classique» étant inadaptée, les recherches menées nous ont conduit à proposer une nouvelle approche combinant des techniques de hachage, de signature numérique et de contrôle d'accès pour assurer une authentification «bout-en-bout» des flux multimédias adaptables. Cette solution a été baptisée AMCA.

En termes de fonctionnalités, notre solution est capable de permettre aux récepteurs de vérifier l'authenticité et l'intégrité d'un document adapté par des *proxies* intermédiaires et d'identifier (authentifier) chacun des objets-médias insérés par ces derniers.

L'idée principale est d'exploiter les propriétés des arbres de hachage de Merkle (*cf.* paragraphe 4.2.2.2) pour signer les documents multimédias. Ce type d'arbre de hachage est utilisé en général pour le contrôle d'intégrité des données, mais nous en proposons ici une utilisation pour la signature de documents multimédias compo-

sites.

Par ailleurs, afin de permettre une signature sélective et de restreindre les possibilités d'adaptation à un sous-ensemble d'objets-médias, nous avons mis en en place une technique de contrôle d'accès aux documents avec une plus fine granularité. L'idée ici est de contrôler l'accès aux objets-médias plutôt qu'au fichier entier.

Le paragraphe suivant décrit plus en détail le fonctionnement de notre approche avant de présenter les différents concepts théoriques utilisés.

4.2.1 Fonctionnement

Nous allons dans ce paragraphe décrire le principe général du fonctionnement de l'approche AMCA et introduire sommairement les idées sur lesquelles elle repose. Le suite du chapitre présentera les détails des étapes les plus importantes de notre processus de sécurisation.

Le procédé proposé par AMCA ne se limite pas uniquement à la sécurisation du contenu multimédia adaptable, mais s'étend à la sécurisation de la session de communication de bout-en-bout, incluant le serveur, le(s) *proxy(ies)* et le client. Le processus de sécurisation du contenu avant son émission passe ainsi par trois étapes distinctes, 1) l'identification des intervenants dans la session, 2) la définition de la politique d'adaptation par l'émetteur, et 3) la signature du flux adaptable.

1. La première étape est d'identifier les intervenants dans une session, pour cela nous utilisons un système d'authentification à la session par certificat, ce dernier contient les détails concernant les clés publiques/privées ou partagées de son propriétaire. Toutefois, en plus d'identifier chaque intervenant, il est important de connaître son profil dans la session de communication (serveur, *proxy* ou client) afin de pouvoir définir les autorisations d'accès au contenu pour chacun des intervenants. Pour cela nous utilisons un modèle de contrôle d'accès à base de rôles (RBAC) où chaque catégorie de profil se verra attribuer des droits d'accès particuliers.
2. La seconde étape de notre processus de sécurisation s'appuie sur ce contrôle d'accès et l'aspect composite des documents multimédias pour générer les politiques d'accès et d'adaptation qui permettront à l'émetteur de décider quelles parties d'un document ne peuvent être adaptées et éventuellement quels *proxies* peuvent procéder à l'adaptation de son contenu.

3. La troisième étape est la signature du document à émettre, la solution que nous proposons pour cela utilise la technique des arbres de hachage de Merkle. Où nous associons à chaque document multimédia un arbre de hachage de Merkle de sorte que les feuilles soient les valeurs de hachage des objets-médias du document à émettre et que la signature soit la signature de la racine de l'arbre de hachage de Merkle.

L'utilisation des arbres de hachage de Merkle dans notre approche nous permet de maintenir la validité de la signature d'un document initial signé, même après «élagage» d'une partie des objets-médias le composant. Plus encore, afin de permettre les opérations d'insertion de nouveaux objets-médias (ou d'objets adaptés) par le *proxy* d'adaptation, nous avons raffiné cette solution en créant des «*freeleaves*», qui sont des feuilles spéciales destinées à accueillir ces objets adaptés (ou insérés). Cette technique sera décrite plus en détail dans le paragraphe 4.2.3.1.

Après les opérations de signature, le document est transmis au destinataire via le *proxy* intermédiaire qui procède aux opérations d'adaptation. Dans AMCA nous considérons deux types d'opération d'adaptation possibles : la suppression d'objets-médias et l'insertion, l'adaptation d'un objet-média pouvant être décomposée en la suppression de ce dernier suivie de l'insertion de sa version adaptée. Le *proxy* procède ensuite à l'adaptation de l'arbre de hachage à transmettre en élaguant les valeurs de hachage correspondant aux objets-médias supprimés tout en s'assurant d'envoyer un minimum d'informations nécessaires au client pour vérifier la signature transmise. Parallèlement il signe aussi séparément les objets insérés dans les *freeleaves*. Puis il transmet ces résultats au client final.

Le client à la réception du flux, reconstruit l'arbre de hachage puis vérifie la validité de la signature reçue par rapport à la racine calculée, l'authentification du document dépendra donc de la validité de cette signature. De plus, le client a aussi la possibilité de vérifier la validité des objets insérés par les *proxies* intermédiaires grâce à la technique des *freeleaves*.

La section 4.2.3 décrit plus en détail la signature des documents dans l'approche AMCA, en l'illustrant par un exemple d'utilisation. Toutefois, les deux premières étapes, concernant l'identification des utilisateurs et le contrôle d'accès, sont des implémentations de solutions classiques et seront donc décrites dans le chapitre 5 qui est consacré aux aspects implémentation des approches proposées.

4.2.2 Concepts de base

Dans cette section nous introduisons les concepts théoriques utilisés dans notre schéma d'authentification des contenus, nous commençons par proposer un modèle formel pour représenter les documents multimédias avant de présenter les arbres de hachage de Merkle que nous utilisons dans nos algorithmes de signature.

4.2.2.1 Modèle de document multimédia

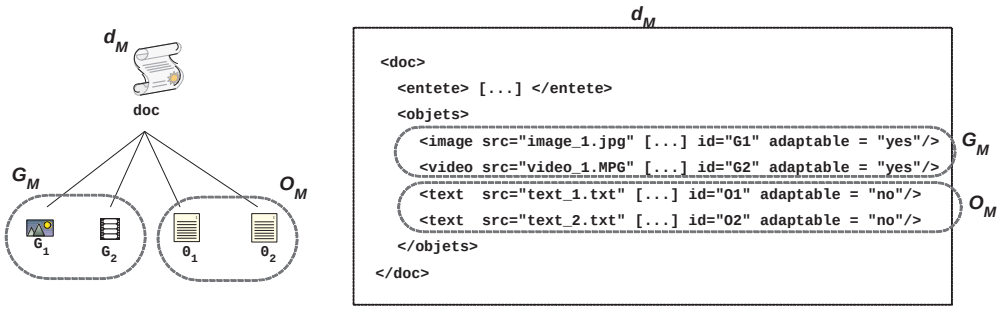
Comme nous l'avons décrit dans la section 1.1.1, un document multimédia peut être découpé, selon sa granularité, en plusieurs objets-médias. Dans le cas de documents structurés, les objets-médias peuvent être des fichiers (audio, vidéo, image, texte), ou des objets synthétiques (comme pour MPEG 4). Dans le cas de documents scalables, les objets-médias varient selon la nature de la scalabilité des documents, ils peuvent par exemple représenter des couches de raffinement. Dans les deux cas, ces objets-médias composant un document initial peuvent être classés en deux groupes : objets-médias adaptables et objets-médias statiques.

Les objets-médias adaptables sont les objets susceptibles d'être modifiés lors des opérations d'adaptation. Les objets-médias statiques sont ceux qui doivent rester inchangés tout au long du processus d'adaptation, comme par exemple la couche de base dans un contenu scalable, ou les objets dont l'émetteur a interdit la modification.

A partir de ces postulats, et pour une approche plus formelle de notre problématique, nous proposons le modèle suivant pour représenter un document multimédia adaptable.

Définition 9 *Un document multimédia est un tuple* $M = (d_M, O_M, G_M)$*, où :*
- d_M *est la description métadonnée de la présentation multimédia ;*
- $O_M = \{o_1, o_2, \ldots, o_l\}$ *est l'ensemble des objets-médias statiques (non adaptables) constituant la présentation multimédia ;*
- $G_M = \{g_1, g_2, \ldots, g_k\}$ *est l'ensemble des objets-médias adaptables constituant la présentation multimédia.*

Cette présentation sera utilisée dans la suite de ce manuscrit. La figure 4.3 montre un exemple de document multimédia en décrivant les ensembles O_M et G_M dans les formats en graphe et structuré.



(a) représentation en graphe (b) représentation structurée

Figure 4.3 – Exemple de document multimédia $M = (d_M, O_M, G_M)$

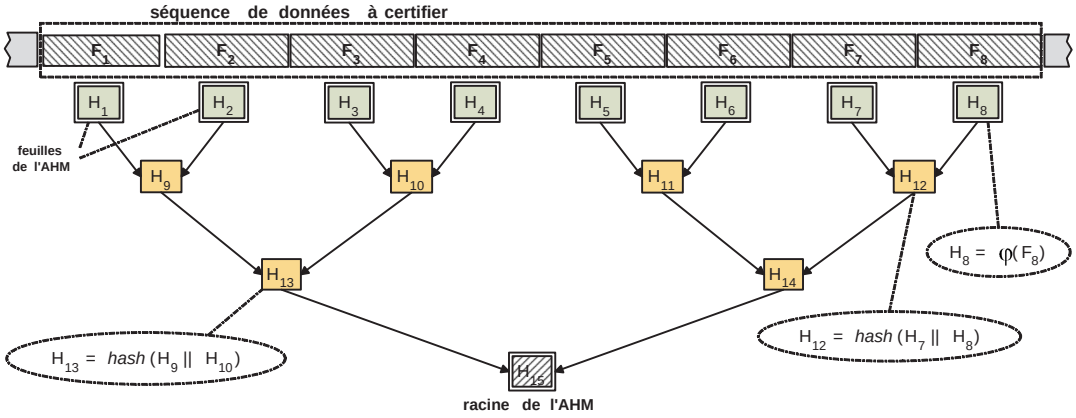
4.2.2.2 Arbres de hachage de Merkle

Définition 10 *Un arbre de hachage de Merkle [67] est une structure de données hiérarchique récursive capable de certifier et de révoquer des données. Il permet de garantir l'intégrité de n'importe quel sous-ensemble (ou groupe de sous-ensembles) de données représentant les feuilles de l'arbre, en fonction d'une seule valeur de hachage correspondant à la racine de l'arbre.*

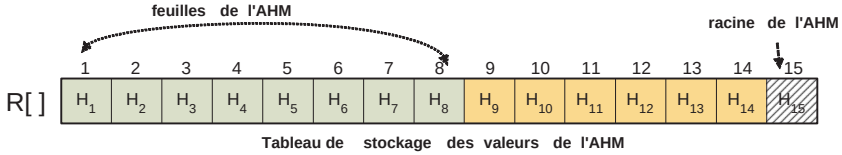
Cette technique d'arbres de hachage a été présentée la première fois par Ralph Merkle dans sa thèse en 1979 [68], par la suite plusieurs chercheurs ont repris le principe général pour des utilisations diverses. Actuellement, les arbres de hachage de Merkle sont très utilisés dans les systèmes de fichiers distribués sécurisés ou dans les réseaux pair-à-pair pour vérifier l'intégrité des données ou de zones mémoire sécurisées [16, 35, 71]. Dans la suite du manuscrit, nous désignerons indifféremment un arbre de hachage de Merkle par **arbre de Merkle** ou **AHM**. A présent nous allons introduire les techniques de construction et de stockage des AHM utilisées dans notre approche AMCA.

Construction d'un AHM Dans un AHM, les feuilles correspondent à des valeurs de hachage de portions de données, et la racine à la valeur de hachage du document entier. Le principe général est de calculer la valeur de hachage au niveau de chaque feuille en hachant le sous-ensemble de données correspondant, puis de construire un arbre de hachage au dessus des résultats, chaque valeur de hachage intermédiaire est calculée en hachant la concaténation des valeurs de hachage de ses fils.

Plus précisément, Comme le montre la figure 4.4-a, un arbre de Merkle est un arbre binaire complet équipé d'une fonction de hachage $hash()$ et d'une application φ,



(a) Construction d'un AHM



(b) Stockage d'un AHM dans un Vecteur R[ ]

FIGURE 4.4 – Exemple d'un AHM à 8 feuilles

qui relie l'ensemble des feuilles de l'arbre à l'ensemble des chaînes de longueur k : $x \rightarrow \varphi(x) \in \{0,1\}^k$.

Pour les feuilles de l'arbre, nous associons pour chaque feuille F_i la valeur H_i calculée comme suit :

$$H_i = \varphi(F_i) \qquad 0 \leq i \leq n \tag{4.1}$$

Pour les nœuds intérieurs de l'arbre, chaque nœud H_i doit satisfaire l'égalité :

$$H_i = hash(H_{i_{gauche}} \parallel H_{i_{droit}}) \qquad n < i \leq 2n-1 \tag{4.2}$$

Où $\parallel$ est l'opérateur de concaténation, $H_{i_{gauche}}, H_{i_{droit}}$ sont respectivement les fils gauche et droit de H_i et la fonction $hash()$ est une fonction de hachage à sens unique comme, par exemple *MD5, SHA-1 ou SHA-256.*

Notons que l'application φ peut être une fonction de hachage, de chiffrement ou

toute autre fonction de transformation. Dans notre solution de sécurisation finale, comme nous le verrons dans le chapitre 5, nous allons remplacer la fonction φ par une fonction de chiffrement. Mais, pour l'instant, nous supposons q ue l'application φ est remplacée par la fonction de hachage $hash()$, dans quel cas $H_i = hash(F_i)$ comme le montre l'exemple décrit dans la figure 4.4-a.

Stockage d'un AHM Comme pour tout arbre binaire, le stockage des valeurs peut se faire dans différentes structures de données (tableau, liste, pile, ...). Pour la description de notre approche nous avons opté pour le stockage dans un tableau de taille $2n-1$ (où n est le nombre de feuilles de l'arbre. La figure 4.4-b représente le tableau qui stocke les valeurs de l'AHM de l'exemple présent. La méthode générale est décrite dans la figure 4.5, le stockage se fait palier par palier et de gauche à droite.

Soit R le tableau des valeurs de l'AHM, R contient alors $2n-1$ cases (numérotées de 1 à $2n-1$), les feuilles de l'arbres occupent les cases de 1 à n, le premier niveau de parents est dans les $n/2$ cases suivantes, le deuxième niveau de parents dans les $n/4$ cases suivantes, ... De cette manière, la racine de l'arbre se trouve de cette manière dans la case $2n-1$ du tableau.

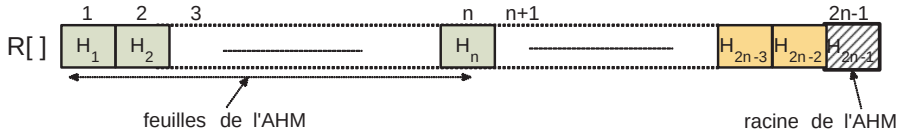


FIGURE 4.5 – Tableau de stockage des valeurs d'un AHM

Chemin d'authentification d'une feuille Un arbre de Merkle sert à certifier ou à révoquer des données. Pour valider une donnée F_i reçue, il faut vérifier qu'en recalculant la valeur de la racine $R[2n-1] = H_{2n-1}$ on retrouve celle reçue ou stockée initialement. Si on ne retrouve pas la même valeur c'est que la donnée reçue est corrompue. Pour faire ce calcul on utilise le chemin d'authentification de F_i.

Définition 11 *Dans un AHM, le chemin d'authentification d'une feuille F_i est représenté par «l'ensemble des nœuds frères des nœuds reliant la feuille à la racine».*

Si l'on prend l'exemple de l'arbre présenté dans la figure 4.6, nous pouvons prouver l'intégrité de la donnée F_7 (à partir de H_7) en utilisant son chemin d'authentification. Les nœuds reliant F_7 à la racine étant $\{H_7, H_{12}, H_{14}\}$, son chemin d'authentification

est constitué des frères de chacun de ces nœuds et est donc $\{H_8, H_{11}, H_{13}\}$. La certification de la données F_7 se fait alors en calculant la valeur de la racine et la comparant avec la racine reçue $R' = hash(hash(hash(H_7 \| H_8) \| H_{11}) \| H_{13})$.

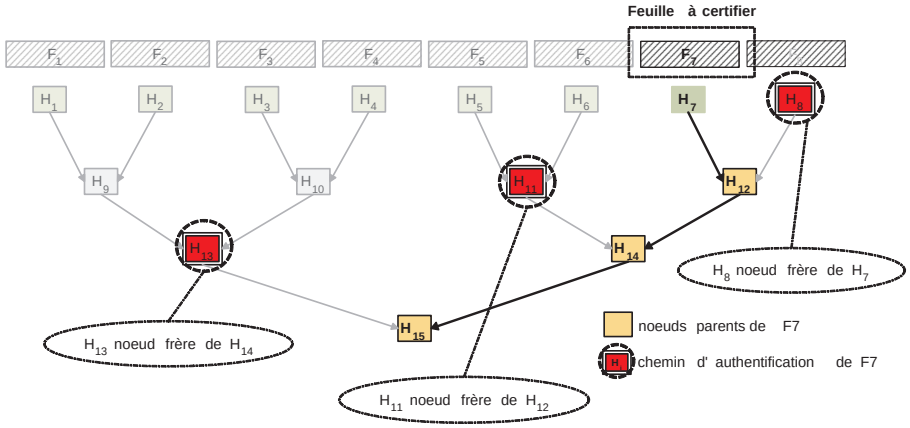


Figure 4.6 – Chemin d'authentification d'une feuille dans un AHM

4.2.3 Algorithmes de signature proposés

Après avoir exposé dans la section précédente les concepts théoriques sur lesquels se fonde l'approche AMCA. Nous allons dans cette section voir comment nous procédons, en utilisant ces concepts, pour authentifier les contenus multimédias adaptables.

Reprenons maintenant la topologie $\mathcal{SPC}$ (figure 4.1). Soit $\mathcal{S}$ l'émetteur qui crée et signe le document original, $\mathcal{C}$ le récepteur qui vérifie la signature et $\mathcal{P}$ le *proxy* d'adaptation. Le schéma peut s'étendre à plusieurs *proxies* mais pour simplifier nous nous limiterons ici au cas d'un seul $\mathcal{P}$.

$\mathcal{S}$ transmet un document multimédia $M = (d_M, O_M, G_M)$ à $\mathcal{C}$ en passant par $\mathcal{P}$. M ne peut être exploité par $\mathcal{C}$ dans l'état, il doit être adapté à son profil. $\mathcal{P}$ reçoit M procède à des opérations d'adaptation selon la politique précisée par $\mathcal{S}$ et transmet le résultat $M' = (d'_{M'}, O'_{M'}, G'_{M'})$ à $\mathcal{C}$.

De manière générale, $\mathcal{P}$ peut supprimer des objets-médias de M (ex. retirer une vidéo qui illustre un texte), insérer de nouveaux objets-médias à des endroits précis (ex. rajout de messages publicitaires), ou adapter des objets-médias, dans ce dernier cas, l'opération d'adaptation sur un objet-média est considérée comme une suppression de la version originale du média suivie de l'insertion de la version adaptée. Une vidéo peut être ainsi remplacée par une image, une bande audio par un texte, un texte en

français par sa traduction en anglais, etc. Nous tâcherons cela dit de nous assurer que la politique d'adaptation précisée par $\mathcal{S}$ soit bien respectée.

Supposons que chacun des intervenants dispose d'une paire de clés privée/publique $(K, \overline{K})$ comme suit : $\mathcal{S}(K_S, \overline{K}_S)$, $\mathcal{P}(K_P, \overline{K}_P)$ et $\mathcal{C}(K_C, \overline{K}_C)$.

$\mathcal{S}$ dispose d'une fonction de signature $sign(d, K)$ qui désigne un algorithme qui produit une signature θ sur une donnée d en utilisant la clé K.

$\mathcal{C}$ dispose d'une fonction $verif(d, K, \theta)$ qui désigne un algorithme qui permet de vérifier la validité de la signature θ de la donnée reçue d en utilisant la clé K.

Soit $hash()$ une fonction de hachage qui prend en entrée une donnée d, et produit un condensé de k bits en sortie. Nous supposons que cette fonction cryptographique de hachage est résistante aux collisions ; i.e., trouver deux entrées $d_1 \neq d_2$ tels que $hash(d_1) = hash(d_2)$ est difficile. Un exemple pratique d'une telle fonction cryptographique de hachage est *SHA-1* qui a une sortie de 160 bits.

Le processus d'authentification se décompose alors en trois étapes, *(i)* l'émetteur signe le contenu, *(ii)* le PA adapte le contenu et signe les contenus ajoutés, *(iii)* le récepteur vérifie la validité des signatures reçues. Nous allons dans la suite de cette section détailler chacune de ces étapes en présentant leurs algorithmes formels et en les illustrant par un exemple réel d'utilisation.

4.2.3.1 L'émetteur signe

L'approche classique pour signer M serait de signer chacun de ses composants ; à la réception toutes les signatures devraient être validées une à une. L'idée que nous proposons dans notre schéma est d'associer à M un AHM puis de ne signer que la racine de ce dernier. La vérification de l'authenticité de chaque composant se fera en utilisant son *chemin d'authentification*.

Pour signer M, l'émetteur $\mathcal{S}$ procède en deux temps : (a) il commence par créer l'AHM associé au document M, puis (b) il signe la racine de l'AHM résultat. Regardons cela en détail :

(a) Génération de l'AHM d'un document multimédia L'arbre de Merkle associé à un document multimédia M, composé de n objets-médias, est un arbre binaire équilibré à n feuilles, de manière que pour chaque feuille F_i, $H_i = \varphi(F_i)$, $1 \leq i \leq n$

et que pour un sommet intérieur H_i, $\varphi(H_i) = hash(\varphi(H_{i_{gauche}}) \parallel \varphi(H_{i_{droit}})$, où $H_{i_{gauche}}$ et $H_{i_{droit}}$ sont les enfants de H_i respectivement gauche et droit, et $\parallel$ dénote l'opérateur de concaténation.

$\mathcal{S}$ calcule la valeur de la racine ρ de l'AHM associé à M après avoir calculé les valeurs de hachage de chaque feuille.

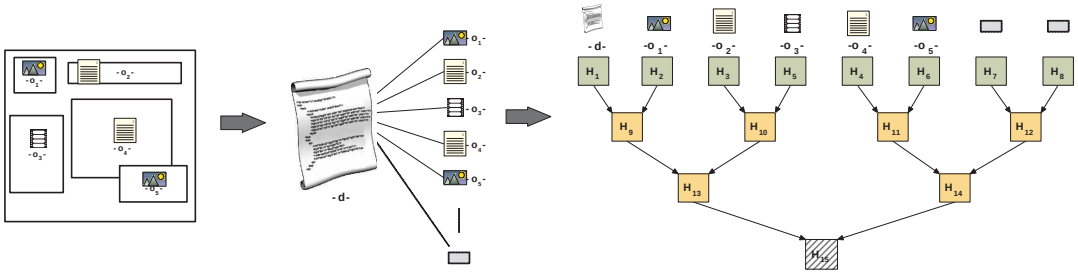


Figure 4.7 – Génération de l'AHM d'un document multimédia structuré

La figure 4.7 décrit la manière dont les feuilles de l'AHM sont générées dans le cas d'un document multimédia structuré.

(b) Signature L'algorithme 1 décrit la procédure suivie par $\mathcal{S}$ pour la signature du document M. Après avoir généré l'AHM de M, $\mathcal{S}$ signe la racine de ce dernier en utilisant la fonction $sign()$ La signature générée est $\theta = sign(K_S, H_{2n-1})$, cette valeur sera ensuite transmise par l'intermédiaire de $\mathcal{P}$ au client $\mathcal{C}$. L'algorithme décrit aussi les techniques utilisées pour la construction d'un AHM et l'insertion des *freeleaves*. Ces dernières sont utilisées par $\mathcal{P}$ lors des opérations d'adaptation de M au profil de $\mathcal{C}$.

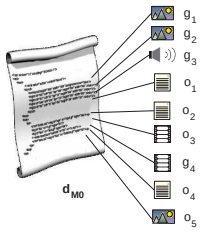
Algorithme 1 : Signature

```
   /* L'algorithme reçoit en entrée le document multimédia M et la clé privée de l'émetteur K_s,
      il donne en sortie le résultat de la signature θ.                                        */
   Données : M = (d_M, O_M, G_M), Key = K_S
   Résultat : θ
 1 begin
   |  /* commencer par calculer le nombre de Freeleaves à insérer                             */
 2 |  int l ← |O_M|     /* l est le nombre d'objets médias NON adaptables                     */
 3 |  int k ← |G_M|     /* k est le nombre d'objets médias adaptables                         */
   |  /* ⇒ le nombre de freeleaves est de k                                                   */
   |  /* créer ces k freeleaves (la fonction CreateFreeleaf() sera décrite plus loin)        */
 4 |  pour int i = 1 à k faire
 5 |  |  CreateFreeleaf(G_M[i], fl_M[i])
   |  /* calculer le nombre de feuilles de l'AHM (puissance de 2)                           */
 6 |  int s ← 2
   |  /* s est la première puissance de 2 supérieure à l + k + k                           */
 7 |  tant que s <= (l + k + k) faire s ← s * 2
   |  /* compléter avec k freeleaves vides pour atteindre les s objets                     */
 8 |  j ← k + 1
 9 |  pour int i = 1 à s − (l + k + k) faire
10 |  |  CreateFreeleaf(NULL, fl_M[j])
11 |  |  j ← j + 1
   |  /* déclarer un vecteur pour stocker l'AHM                                            */
12 |  String A[2s − 1]   /* tableau de taille 2s − 1                                       */
   |  /* commencer par y inclure les valeurs de hachage de chaque objet NON adaptable       */
13 |  pour int i = 1 à l faire
14 |  |  A[i] ← hash(O_M[i])
15 |  int j ← i
   |  /* puis inclure les valeurs de hachage des objets adaptables                         */
16 |  pour int i = 1 à k faire
17 |  |  A[j] ← hash(G_M[i])
18 |  |  j ← j + 1
   |  /* ensuite inclure les valeurs de hachage des freeleaves                             */
19 |  pour int i = 1 à s faire
20 |  |  A[j] ← hash(fl_M[i])
21 |  |  j ← j + 1
   |  /* calculer les valeurs des nœuds intermédiaires, la racine sera dans A[2 * s − 1]  */
22 |  i ← 1
23 |  j ← s + 1
24 |  tant que i < 2 * s − 3 faire
25 |  |  A[j] ← hash(A[i]||A[i + 1])
26 |  |  i ← i + 2
27 |  |  j ← j + 1
   |  /* signer la racine de l'AHM avec la fonction de signature sign()                    */
28 |  θ = sign(key, A[2 * s − 1])
29 |  return(M, A[], θ)          /* renvoyer tous les résultats                           */
30 end
```

exemple de déroulement Reprenons l'application exemple décrite dans la figure 4.2. Un médecin utilisant cette application décide d'envoyer un diagnostic sous la forme d'un document multimédia que nous appellerons M_0 à un collègue.

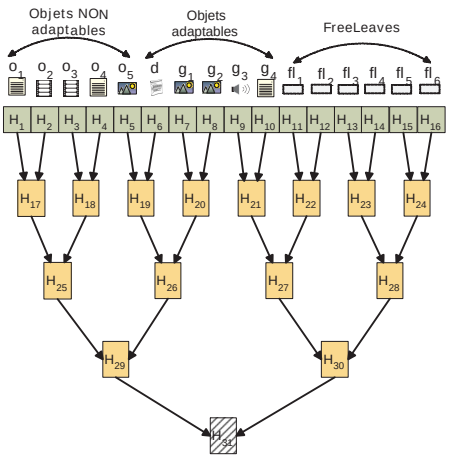
Soit $M_0 = (d_{M_0}, O_{M_0}, G_{M_0})$ un document multimédia structuré composé d'une description métadonnée et de neuf objets-médias dont quatre sont adaptables, sans oublier la metadonnée qui est aussi adaptable.

La première étape du processus d'authentification est la génération de l'AHM. M_0 contient cinq objets adaptables et nécessite donc cinq «*freeleaves* ». La figure 4.8 déroule l'algorithme 1 sur M_0. Ce déroulement permet de générer le document M_1 (la version signée de M_0), un tableau $A[\,]$ stockant les valeurs des nœuds de l'AHM associé ainsi que θ_{M_1} la signature du document M_1. Ces trois résultats seront transmis ensuite au *proxy* d'adaptation.



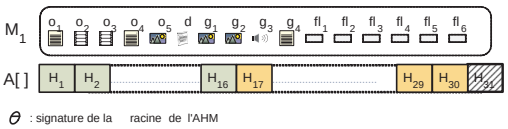
Données : $M_0 = (d_{M_0}, O_{M_0}, G_{M_0})$, $Key = K_S$
avec $O_{M_0} = \{o_1, o_2, o_3, o_4, o_5\}$, $G_{M_0} = \{d_{M_0}, g_1, g_2, g_3, g_4\}$
Résultat : M_1, A[ ], θ

$l = |O_{M_0}| = 5$ /* l = le nombre d'objets-médias non adaptables */
$k = |G_{M_0}| = 5$ /* k = le nombre d'objets adaptables + d */
$\Rightarrow s = 16$ /* la première puissance de 2 supérieure à $2 * k + l$ */
/* le nombre de feuilles est donc de 16 : 5 objets non adaptables */
/* + 5 objets adaptables + leur 5 *freeleaves* + 1 feuille vide */
$\Rightarrow |A| = 2 * 16 - 1 = 31$ /* la tailles de l'AHM est donc de 31 valeurs */
$CreateFreeLeaf(d_{M_0}, fl_1)$ /* */
$CreateFreeLeaf(g_1, fl_2)$ /* */
$CreateFreeLeaf(g_2, fl_3)$ /* */
$CreateFreeLeaf(g_3, g_4, fl_4)$ /* */
$CreateFreeLeaf(g_4, fl_5)$ /* */
$CreateFreeLeaf(NULL, fl_6)$ /* */



Construction de l'AHM
/* nous calculons les valeurs de hachage des feuilles */
/* puis les valeurs de hachage des nœuds intermédiaires */

$A[1] = hash(d_{M_0})$ — $A[17] = hash(A[1] \parallel A[2])$
$A[2] = hash(o_1)$ — $A[18] = hash(A[3] \parallel A[4])$
$A[3] = hash(o_2)$ — $A[19] = hash(A[5] \parallel A[6])$
$A[4] = hash(o_3)$ — $A[20] = hash(A[7] \parallel A[8])$
$A[5] = hash(o_4)$ $\Rightarrow$ $A[21] = hash(A[9] \parallel A[10])$
$A[6] = hash(g_1)$ — $A[22] = hash(A[11] \parallel A[12])$
$A[7] = hash(g_2)$ — $A[23] = hash(A[13] \parallel A[14])$
$A[8] = hash(g_3)$ — $A[24] = hash(A[15] \parallel A[16])$
$A[9] = hash(g_4)$ — $A[25] = hash(A[17] \parallel A[18])$
$A[10] = hash(g_5)$ — $A[26] = hash(A[19] \parallel A[20])$
$A[11] = hash(fl_1)$ $\Rightarrow$ $A[27] = hash(A[21] \parallel A[22])$
$A[12] = hash(fl_2)$ — $A[28] = hash(A[23] \parallel A[24])$
$A[13] = hash(fl_3)$ — $A[29] = hash(A[25] \parallel A[26])$
$A[14] = hash(fl_4)$ — $A[30] = hash(A[27] \parallel A[28])$
$A[15] = hash(fl_5)$ — $A[31] = hash(A[29] \parallel A[30])$
$A[16] = hash(fl_6)$



Signature de la racine de l'AHM
$\theta = sign(key, A[31])$

et préparation des données à transmettre
/* nous retournons au *proxy* : le document M_1, le tableau */
/* A stockant l'AHM et θ la signature du document */

return(M_1, A, θ)

Figure 4.8 – Déroulement de l'algorithme de signature d'AMCA sur l'exemple 1

4.2.3.2 Le PA adapte

L'utilisation de la technique des AHM permet de recalculer la valeur d'une signature même après suppression d'une partie des données initiales (constituant les feuilles de l'arbre de Merkle) sous réserve de disposer du reste de l'arbre, autrement dit, les valeurs des nœuds parents des feuilles supprimées.

Une opération d'adaptation d'un objet-média pouvant être considérée comme une suppression de l'objet-média initial puis une insertion de l'objet adapté. L'algorithme 2 décrit d'une manière plus formelle chacune des deux opérations que nous détaillons dans ce qui suit.

Suppression En utilisant les propriétés des AHM, il devient possible de vérifier l'intégrité (et par extension la signature) d'un sous ensemble des données de départ sous condition de disposer des chemins d'authentification de toutes les parties reçues. Nous exploitons cette propriétés pour mettre en œuvre un algorithme de suppression des objets-médias au niveau de $\mathcal{P}$. Ce dernier doit s'assurer qu'à la réception $\mathcal{C}$ puisse certifier toutes les données reçues.

L'idée pour la procédure de suppression est de fournir à $\mathcal{C}$ le minimum d'informations nécessaires pour calculer tous les chemins d'authentification des objets reçus. Pour cela $\mathcal{P}$ retire tous les objets à supprimer de O_M et remplace (en itérant sur les valeurs restant) chaque deux paires correspondant au même nœud père par ce dernier. L'ensemble des valeurs de hachage reçues à la fin permettra de certifier chaque objet séparément. L'algorithme 2 décrit d'une manière plus formelle cette technique de suppression.

Insertion Pour les opérations d'adaptation nécessitant l'ajout d'éléments dans le flux multimédia, nous proposons un schéma pour la réalisation des *freeleaves* utilisant des techniques de signatures à clé publique standards (ex, RSA). Nous définissons les *freeleaves* comme suit :

Définition 12 *Une freeleaf est un objet-média «fictif» qui sert comme support à l'insertion éventuelle d'un nouvel objet-média (ou un objet-média adapté). A la création, une freeleaf contient la clé publique du client signée par la clé publique du PA.*

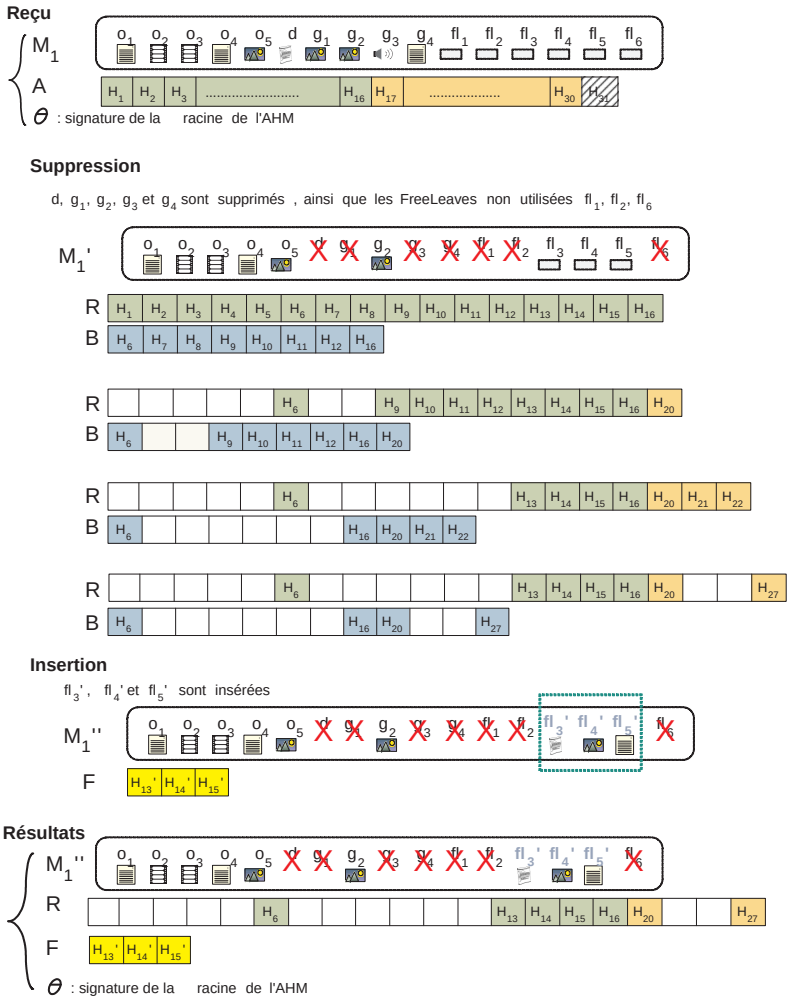
Pour cela $\mathcal{S}$ place la clé publique de $\mathcal{P}$ dans le bloc d'une *freeLeaf*. $\mathcal{S}$ crée alors un condensé de l'arbre de Merkle puis signe comme décrit ci-dessus. $\mathcal{P}$ à son tour insère

son contenu et le signe séparément. $\mathcal{C}$ vérifie la validité des deux signatures. Ainsi, il pourra authentifier le flux comme provenant de l'émetteur $\mathcal{S}$ ayant été modifié par le *proxy* $\mathcal{P}$ tout en repérant les parties qui sont le résultat d'une insertion/adaptation.

Algorithme 2 : Adaptation de la signature

```
   /* L'algorithme reçoit en entrée A[ ] le tableau contenant les valeurs de l'AHM et B la liste
      des éléments à supprimer. Il donne en sortie le tableau R[ ] stockant les nouvelles valeurs
      de l'AHM                                                                              */
   Données : A[ ], B
   Résultat : R[ ], F[ ]
 1 begin
 2   StringR[ ]            /* tableau qui contiendra les résultats                          */
 3   pour int i = 1 à k faire
 4     R[i] ← A[i]          /* R est initialisé avec les valeurs de A                       */
 5   pour int i = 1 à m faire
 6     B[i] ← A[i]          /* B est initialisé avec les valeurs des objets à supprimer     */
     /* SUPPRESSION                                                                        */
 7   pour int i = 1 à l faire
 8     R[i] ← NULL        /* On supprime de R les valeurs concernant le objets non adaptables */
 9   tant que i < |B[]| faire
10     si frere(B[i], B[i + 1]) alors
11       R[MAX + 1] ← pere(B[i], B[i + 1])
12       R[i] ← NULL
13       B[MAX + 1] ← pere(B[i], B[i + 1])
14       B[i] ← NULL
15       i ← i + 2
16     sinon
17       i ← i + 1
     /* INSERTION                                                                          */
18   pour int i = 1 à k faire
19     F[i] ← sign(hash(fl_{H_j})‖R[H_j])          /* signature des objets insérés dans les
          freeleaves                                                                    */
20   return (M'', R, F, θ)
21 end
```

exemple de déroulement Supposons que l'adaptation qui doit être effectuée sur le document M_1 par le *proxy* soit le remplacement de la vidéo o_2 par une image et la suppression de l'image o_1 et un texte o_3. Déroulons à présent l'algorithme 2 sur notre exemple afin de voir son fonctionnement. La figure 4.9 décrit ce déroulement :



Données : $M'_1 = (d_{M'_1}, O_{M'_1}, G_{M'_1})$, A[ ], θ
Résultat : $M''_1, R[\,], F, \theta$

/* Nous initialisons $R[\,]$ avec les valeurs de hachage des feuilles */
$R = \{H_1, H_2, H_3, H_4, H_5, H_6, H_7, H_8, H_9, H_{10}, H_{11}, H_{12}, H_{13}, H_{14}, H_{15}, H_{16}\}$

Suppression

/* L'adaptation nécessite la suppression de $d, g_1, g_2, g_3, g_4, g_5, fl_1, fl_2, etfl_6$. $B[\,]$ est initialisé avec les valeurs de hachage des feuilles supprimées */
$B = \{H_6, H_7, H_8, H_9, H_{10}, H_{11}, H_{12}, H_{16}\}$

/* Les objets o_1, o_2, o_3, o_4, o_5 seront transmis tel quel, alors nous supprimons leur valeurs de hachage de $R[\,]$ */
- $R = R - \{H_1, H_2, H_3, H_4, H_5\}$

/* Puis, tant qu'il y'a des nœuds frères dans $B[\,]$ les remplacer par leur père dans R et dans B */
$R = R - \{H_7, H_8\} + H_{20}$ $\quad B = B - \{H_7, H_8\} + H_{20}$
$R = R - \{H_9, H_{10}\} + H_{21}$ $\quad B = B - \{H_9, H_{10}\} + H_{21}$
...
$R = R - \{H_{21}, H_{22}\} + H_{27}$ $\quad B = B - \{H_{21}, H_{22}\} + H_{27}$

$\Rightarrow \quad R = \{H_6, H_{13}, H_{14}, H_{15}, H_{16}, H_{20}, H_{27}\}$

Insertion

/* Pour chaque objet inséré, nous calculons la signature */
$F[1] = sign(hash(fl'_3) \| H_{13})$
$F[2] = sign(hash(fl'_4) \| H_{14})$
$F[3] = sign(hash(fl'_5) \| H_{15})$

return(M''_1, R, F, θ)

FIGURE 4.9 – Déroulement de l'algorithme d'adaptation d'AMCA sur l'exemple 1

4.2.3.3 Le récepteur vérifie

La fonction $Verif()$ citée précédemment permet de calculer la validité d'un ensemble de nœuds donnés en entrée, en comparant les résultats des chemins d'authentification de chaque feuille avec les valeurs de la signature initiale reçue. Pour cela, $\mathcal{C}$:

- génère un nouvel AHM à partir des objets-médias reçus et des valeurs de hachage reçues, soit N_0 la racine de cet AHM ;
- déchiffre la signature θ avec la clé publique de $\mathcal{S}$, soit N_0 le résultat ;
- compare N_0' et N_0 et déduit la validité des objets-médias reçus.

L'algorithme 3 décrit la procédure de vérification.

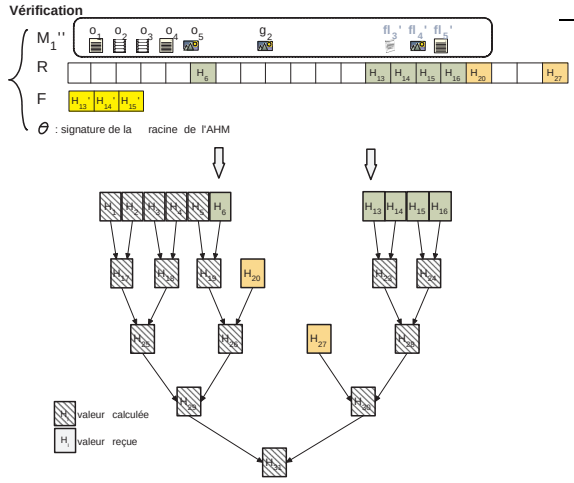
Algorithme 3 : Vérification

```
   /* L'algorithme reçoit en entrée le flux adapté M'', le R[] tableau contenant les nouvelles
      valeurs de l'AHM, le tableau F[] contenant les signatures des objets insérés, et θ la
      signature originale du document.                                                    */
   /* Il retourne une valeur booléenne déterminant la validité de θ                       */
   Données : M'', R, F, θ
   Résultat : estValide
 1 begin
     /* déclarer un vecteur pour stocker le nouvel AHM                                    */
 2   String V[2s − 1]   /* tableau de taille 2s − 1                                       */
     /* on initialise V[] avec les valeurs de R[]                                         */
 3   V[ ] ← R[ ]
     /* puis on calcule les valeurs de hachage des objets-médias reçus afin de compléter l'AHM
        dans V[]                                                                          */
 4   pour int i = 1 à s faire
 5     si O_{M''}[i] reçu alors
 6       V[i] ← hash(O_{M''}[i])
     /* calculer les valeurs des nœuds intermédiaires                                     */
 7   j ← s + 1
 8   pour i = 1 à 2 ∗ s − 3 faire
 9     V[j] ← hash(V[i]‖V[i + 1])
10     j ← j + 2
     /* on calcul la validité de la signature reçue                                       */
11   estValide ← verif(V[2s − 1], K̄_S, θ)
12   afficher(estValide)
     /* puis on calcule la validité des signatures des freeleaves reçues                  */
13   int q ← |F|
14   pour int 1 à q faire
15     estValide ← verif(hash(fl''_M[i]), K̄_P, F[i])
16     afficher(estValide)
17 end
```

Nous pouvons montrer que $N_0' = N_0$, à partir de quoi il est facile de voir pourquoi l'algorithme précédent fonctionne. Si on a toutes les feuilles initiales, alors le procédé ci-dessus est l'algorithme standard pour le calcul de la racine de l'arbre de Merkle. Maintenant, observons qu'à chaque fois que $\mathcal{C}$ ne reçoit qu'une partie des valeurs de hachage $H_1, \ldots, H_s$, ceux-ci proviennent de $\mathcal{P}$ utilisant le même algorithme sur le

sous-ensemble de parties manquantes. Par conséquent, $\mathcal{P}$ et $\mathcal{C}$ ont, ensemble, lancé l'algorithme sur toutes les s feuilles, ce qui ramène à la valeur de la racine de l'AHM.

exemple de déroulement Le client final reçoit le document adapté M_1'', le tableau R contenant les informations nécessaires pour reconstituer l'AHM, le tableau F contenant les signatures des objets insérés dans les *freeleaves* et θ la signature du document. En déroulant l'algorithme 3, le client peut vérifier l'authenticité du document reçu. La figure 4.10 déroule l'algorithme de vérification sur l'exemple décrit précédemment.



Données : $M_1'' = (d_{M_1''}, O_{M_1''})$, R[ ], F[ ], $\theta, \overline{K}_S, \overline{K}_P$
Résultat : estValide = VRAI ou FAUX

/*Nous construisons un AHM à partir des données reçues : Nous initialisons un tableau V[ ] avec les valeurs de R */
$V[\,] = R[\,]$
/*Puis nous calculons les valeurs de hachage des objets reçus afin de compléter l'AHM du document reçu.*/
$V[1] = hash(o_1)$
. . .
$V[5] = hash(o_5)$

/* En suivant l'instruction «si deux nœuds sont non null est leur père est null calculer le père» Nous remplissons V en le parcourant jusqu'à l'obtention de la racine */
$V[17] = hash(V[1] \parallel V[2])$
. . .
$V[31] = hash(V[29] \parallel V[30])$

$estValide = verif(V[31], \overline{K}_S, \theta)$
afficher(estValide)
$estValide = verif(fl_3', \overline{K}_P, F[1])$
afficher(estValide)
$estValide = verif(fl_4', \overline{K}_P, F[2])$
afficher(estValide)
$estValide = verif(fl_5', \overline{K}_P, F[3])$
afficher(estValide)

FIGURE 4.10 – Déroulement de l'algorithme de vérification d'AMCA sur l'exemple 1

4.2.4 Discussion

Le but de l'approche AMCA n'est pas de proposer une nouvelle fonction de signature spécialement pour des documents multimédias adaptables, mais plutôt de proposer une solution combinant des fonctions de signature et des techniques existantes de sorte que l'on puisse assurer une authentification de bout-en-bout du flux adaptable. De ce fait, nous avons adopté une approche orientée blocs de hachage centrée sur le niveau de granularité de nos flux adaptables et pilotée par des cas d'utilisation de notre technique. Les concepts que nous avons proposés dans AMCA sont assez généraux pour être utilisés avec d'autres types de document proposant un niveau de scalabilité ou de granularité minimal, dans la mesure où ils séparent les préoccupation fonctionnelles et extrafonctionnelles, et utilisent des fonctions de hachage et de signature génériques.

La section suivante décrit XSST notre approche de chiffrement de contenus adaptables.

4.3 XSST pour le chiffrement des objets-médias

Toujours dans le cadre de la sécurisation de bout-en-bout des flux multimédias adaptables par des *proxies*, nous avons proposé une nouvelle technique de chiffrement sur le réseau que nous avons baptisée XSST. XSST est un schéma de chiffrement, ou plutôt de rechiffrement, unidirectionnel mis en place au niveau des nœuds intermédiaires. L'objectif principal de XSST est de garantir la confidentialité de bout-en-bout de flux chiffrés d'un émetteur vers plusieurs récepteurs en passant par des *proxies* d'adaptation. XSST permettra ainsi de compléter la technique d'authentification AMCA pour fournir le niveau de confidentialité requis par nos critères de sécurité en gardant les flux (ou des parties du flux) chiffrés de bout-en-bout.

En effet, le troisième critère de sécurité cité dans le paragraphe 2.1 est la confidentialité des données, autrement dit, comment offrir aux émetteurs de contenus multimédias adaptables la possibilité de garder confidentiel les (ou une partie des) données émises et ce jusqu'aux destinataires finaux ? La solution la plus appropriée est, comme nous l'avons montré dans le paragraphe 2.2.1, de chiffrer ces contenus avant de les transmettre ; mais, comme pour l'authentification, des questions se posent : comment un *proxy* peut adapter un contenu chiffré sans avoir à tout déchiffrer ? Et comment éviter les attaques au niveau des *proxies* ? Nous devons donc trouver une technique permettant aux *proxies* d'adapter les contenus sans avoir à les déchiffrer complètement ou du moins offrir la possibilité de garder chiffrées les parties que l'émetteur juge confidentielles.

Une autre difficulté pour la conception d'un tel système réside dans son aspect multipartite ; c'est à dire qu'un même flux peut être envoyé à plusieurs destinataires éventuels. La gestion des clés dans le système devient alors délicate car elle doit rester résistante aux attaques par collusion. Comme l'explique Doërr dans sa thèse [26], un système de chiffrement doit avant tout résister aux attaques par collusion. Autrement dit, en prenant la figure 4.2 de notre application exemple, si un même flux chiffré est envoyé par le Dr Alice vers Dr Bob et Dr Carole (et à plusieurs autres éventuellement), ces derniers ne doivent pas pouvoir, à partir de leurs contenus reçus, calculer la clé utilisée pour le chiffrement.

C'est avec ces objectifs, et en se reposant sur l'état de l'art présenté dans le chapitre 3, que nous avons proposé XSST comme solution de chiffrement sélectif, unidirectionnel et résistant aux attaques par collusion.

L'avantage majeur de notre approche est qu'elle soit unidirectionnelle, autrement dit, Dr Alice peut déléguer des droits à Dr Bob sans qu'il ait à lui déléguer les siens.

Nous utilisons aussi des algorithmes de chiffrement asymétriques ce qui fait que Dr Alice n'a pas à divulguer sa clé secrète à Dr Bob. Un autre apport de notre approche est la résistance aux collusions, autrement dit, plusieurs récepteurs d'un même flux original ne peuvent faire de collusion pour retrouver la clé initiale de l'émetteur, et ce grâce aux concepts utilisés et que nous détaillons dans la suite de ce chapitre.

4.3.1 Concepts théoriques

Dans ce paragraphe nous allons présenter les concepts et théorèmes sur lesquels repose notre approche pour le rechiffrement au niveau des *proxies*.

Vu les contraintes du contenu transmis en termes d'adaptabilité, notre solution doit permettre un chiffrement scalable, autrement dit la possibilité de chiffrer séparément les objets médias afin de pourvoir restreindre l'accès des *proxies* qu'aux objets adaptables.

Le principe général de notre solution est fondé sur l'utilisation des **Séquences de Chiffrement Asymétriques** (*Asymetric Cipher Sequences*) que nous appellerons dans la suite du document **SCA**. La notion de *Cipher Sequences* a été introduite par Molva et Pannetrat dans [70] dans le cadre de la sécurité des échanges dans les réseaux multicast. Dans nos travaux, nous avons exploré l'utilisation de cette technique dans le cadre des systèmes multimédias à base de *proxies* d'adaptation en étendant ses concepts de base afin de proposer une approche de rechiffrement unidirectionnel pour des transmissions chiffrées de bout-en-bout à travers les *proxies*. Nous allons donc commencer par définir ce que nous entendons par «**Rechiffrement Unidirectionnel**» et présenter notre définition des SCA.

4.3.1.1 Schéma de rechiffrement intermédiaire

Dans un schéma de chiffrement « classique », un message m chiffré avec la clé publique d'Alice ne peut être déchiffré que par la clé privé d'Alice. Dans un schéma «de rechiffrement intermédiaire» un nouveau rôle est introduit, celui du «*proxy* de rechiffrement», ce dernier, en utilisant une clé $K_{Alice \mapsto Bob}$ a la capacité de transformer un flux chiffré avec la clé publique d'Alice en un flux déchiffrable par la clé privée de Bob. Plusieurs techniques de rechiffrement intermédiaire peuvent être trouvées dans la littérature [65], [11], [7]. D'une manière plus formelle un schéma de rechiffrement peut se définir comme suit :

Définition 13 *(Rechiffrement Intermédiaire)*

Un schéma de Rechiffrement Intermédiaire est un tuple d'algorithmes à temps polynomial $(KeyGen, RekGen, \vec{Enc}, Renc, \vec{Dec})$*, où les composants sont définis comme suit :*

- **KeyGen** est un algorithme de génération de clés
- **RekGen** est un algorithme de génération de clés de rechiffrement
- $\vec{\textbf{Enc}}$ est un algorithme de chiffrement
- **Renc** est la fonction de rechiffrement
- $\vec{\textbf{Dec}}$ est l'algorithme de déchiffrement

– Pour un utilisateur λ donné en entrée, l'algorithme de génération de clés $KeyGen$ génère en sortie une paire de clés $(kp_\lambda, \bar{ks}_\lambda)$
– Pour une entrée $(kp_A, \bar{ks}_A, kp_B, \bar{ks}_B)$, l'algorithme de génération de clés de rechiffrement $RekGen$ génère en sortie une clé $kr_{A \mapsto B}$ pour le *proxy*
– Pour un message m en entrée et une clé kp_A, $\vec{Enc}$ donne en sortie un texte chiffré $\bar{m}_A$
– Pour une entrée ks_A et un texte chiffré $\bar{m}_A$, $\vec{Dec}$ donne en sortie $m \in M$;
– Pour une entrée $kr_{A \mapsto B}$ et un texte chiffré $\bar{m}_A$ la fonction de rechiffrement $Renc$ génère en sortie un texte rechiffré $\bar{m}_B$.

La méthodologie générale pour la délégation des droits de déchiffrement fut introduite par Mambo et Okamoto dans [65], typiquement en tant qu'amélioration de l'approche traditionnelle «déchiffrer-et-puis-rechiffrer». Une année plus tard Blaze et al proposent dans [11] la notion de «*atomic proxy cryptography*» où un *proxy* de confiance calcule une fonction qui convertit un texte chiffré pour Alice en un texte chiffré pour Bob sans voir le contenu initial en clair. Dans leur schéma (avec EL-Gamal) avec un modulo de protection $p = 2q + 1$, le *proxy* est doté de la clé de délégation $b/a \ mod \ q$ dans le but de détourner un texte chiffré d'Alice vers Bob en calculant $(mg^k \ mod \ p, (g^{ak})^{b/a} mod \ p)$. Les auteurs reconnaissent cela dit que leur schéma contient une certaine restriction qui est à notre sens importante : c'est le fait que leur schéma soit *bidirectionnel*. Autrement dit la valeur b/a peut être utilisée pour détourner du contenu chiffré d'Alice vers Bob et vise versa. Ce schéma n'est donc utilisable que lorsque la confiance entre Alice et Bob est mutuelle. Plusieurs autres problèmes peuvent être trouvé dans ce schéma, sa transitivité par exemple, ce qui signifie qu'un *proxy* peut créer, à lui seul, une délégation de droits entre a/b et b/c et ainsi rechiffrer un texte d'Alice pour Carole. Un autre problème majeur est que le *proxy* et Bob peuvent faire une collusion pour retrouver la clé d'Alice avec $(a/b) * b = a$!

4.3.1.2 Séquence de Chiffrement

Soit $g : \mathcal{N}^2 \longmapsto \mathcal{N}$ une fonction de transformation de données (chiffrement), et k une clé de chiffrement. $g()$ a la propriété suivante : si $X = g(Y, k)$, il est mathématiquement impossible de calculer k connaissant X et Y.

Soit $(k_{1 \leq i \leq n})$ une séquence finie de n clés de chiffrement. $\mathcal{M} = (M_{0 \leq i \leq n})$ est une séquence finie de $n+1$ éléments définie comme suit :

$$\begin{cases} M_0 & \text{valeur initiale de la séquence} \\ M_i = g(M_{i-1}, k_i) & \text{pour } 1 \leq i \leq n \end{cases}$$

Définition 14 *$\mathcal{M} = (M_0, M_1, \ldots, M_n)$ est dite **Séquence de Chiffrement** de la fonction g.*

Définition 15 *$\mathcal{M}$ est dite **Réversible** (Séquence de Chiffrement Réversible de la fonction g), désignée par SCR_g, si pour tout $(M_i, M_j) \in \mathcal{N}^2$ et $0 \leq i < j \leq n$, il existe une fonction φ de sorte que $M_i = \varphi_{i,j}(M_j)$, pour $0 \leq i < j \leq N$.*

Définition 16 *Une SCR_g est dite **Symétrique** désignée par $SCRS_g$ si la fonction $\varphi_{i,j}$ peut être calculée à partir du sous ensemble de clés $\{k_{i+1}, \ldots, k_j\}$.*

Définition 17 *Une SCR_g est dite **Asymétrique** désignée par $SCRA_g$ si la fonction $\varphi_{i,j}$ **ne** peut être calculée à partir du sous ensemble $\{k_{i+1}, \ldots, k_j\}$.*

Pour illustrer ces définitions nous utilisons la figure 4.11. Supposons que M_0 soit le document original à envoyer à plusieurs destinataires en passant par différents *proxies*. A chaque *proxy* intermédiaire P_i est affecté une clé secrète k_{P_i}. P_i reçoit les données à partir de son nœud parent P_j, calcule $M_i = g(M_j, k_{P_i})$, et transmet le résultat M_i à ses fils. Le destinataire final reçoit ainsi M_n. Chaque destinataire C_i dispose d'une fonction inverse g_i^{-1} qui lui permet de retrouver les données originales en calculant : $M_0 = g_i^{-1}(M_n)$.

L'exemple de la figure 4.11 montre un réseau simple avec trois Séquences de Chiffrement :

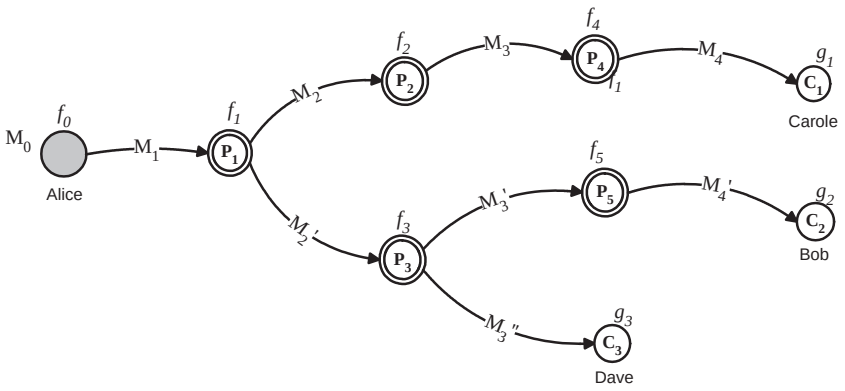


Figure 4.11 – Trois Séquences de chiffrement dans un système adaptable

$\mathcal{M} = (M_0, M_1, M_2, M_3, M_4)$
$\mathcal{M}' = (M_0, M_1, M'_2, M'_3, M'_4)$
$\mathcal{M}'' = (M_0, M_1, M'_2, M''_3)$

Suivons la Séquence de Chiffrement $\mathcal{M}'$ d'Alice vers Bob (C_2) : $\mathcal{M}' = (M_0, M_1, M'_2, M'_3, M'_4)$.

a) Premièrement, Alice calcule $g(M_0, k_0)$ et envoie le résultat au *proxy* P_1 auquel elle est reliée ;

b) P_1 reçoit $M_1 = g(M_0, k_{P_1})$, calcule et envoie $g(M_1, k_1)$ à P_3 ;

c) Puis P_3 reçoit $M'_2 = g(M_1, k_3)$ et envoie $g(M'_2, k_3)$ à P_5 ;

d) P_5 reçoit alors $M'_3 = g(M'_2, k_3)$ et envoie $g(M'_3, k_5)$ au client C_2.

e) Enfin, C_2 reçoit $M'_4 = g(M'_3, k_5)$ et retrouve la donnée originale en calculant $M_0 = g^{-1}(M'_4, k_{C_2})$.

4.3.2 Schéma de rechiffrement XSST

Nous allons utiliser les propriétés des séquences de chiffrement décrite précédemment pour implémenter notre système de rechiffrement unidirectionnel XSST. Comme nous l'avons défini dans le paragraphe 4.3.1.1, XSST est donc un tuple d'algorithmes polynomiaux $(KeyGen, RekGen, \vec{Enc}, Renc, \vec{Dec})$ tel que :

- $KeyGen$ est un algorithme de génération de clés
- $RekGen$ est un algorithme de génération de clés de rechiffrement
- $\vec{Enc}$ est un algorithme de chiffrement de type SCR_g
- $Renc$ est la fonction de rechiffrement
- $\vec{Dec}$ est l'algorithme de déchiffrement

Pour un utilisateur λ donné en entrée, l'algorithme de génération de clés $KeyGen$ génère en sortie une paire de clés $(kp_\lambda, \bar{ks}_\lambda)$.

Pour une entrée $(pk_A, sk_A^+, pk_B, sk_B^*)$, l'algorithme de génération de clés de rechiffrement $RekGen$ génère en sortie une clé $rk_{A \leftarrow B}$ pour le *proxy* ;

Pour une entrée $rk_{A \leftarrow B}$ et un texte chiffré C_A la fonction de rechiffrement $Renc$ génère en sortie un texte rechiffré C_B.

Pour cela, considérons la représentation graphique d'un système d'adaptation avec un *proxy* d'adaptation desservant un contenu original M_0 à N clients.

En général, notre approche utilisant les $SCRA_g$ assure les propriétés suivantes :

- Confidentialité des données durant le transfert
- Confidentialité de bout-en-bout des données
- Protection des données contre des *proxies* malveillants
- Protection contre des attaques par collusion

Développons un peu cette dernière propriété. Supposons que nous utilisons une technique de chiffrement $SCRS_g$, une attaque par collusion est réalisable si le client j dispose de :

1. k_i et k_j (avec $i, j > 0$)
2. le flux reçu par le client i, M_i
3. et sa fonction de déchiffrement g_j^{-1}

Dans ces conditions, le client j (l'attaquant) peut aisément accéder au document original M_0. Par exemple :

1. A partir de k_i, j peut obtenir $g^{-1}(M_i)$ et obtient ainsi $M_0 = g^{-1}(M_i)$
2. A partir de k_j et M_0, l'attaquant peut obtenir M_j par $M_j = g(M_0, k_i)$
3. Et comme l'attaquant dispose de la fonction de déchiffrement $g_{0,j}$, la donnée originale est calculable par $M_0 = g_{-1,j}(M_j)$

Cependant, si le procédé de chiffrement est une $SCRS_g$, l'attaquant ne peut en aucun cas retrouver les données originales car, par définition, la fonction $g_{0,i}$ est mathématiquement non-calculable.

4.3.3 Réalisation

Pour réaliser la fonction $SCRS_g$ nous avons besoin d'un algorithme de chiffrement initial. Molva et Pannetrat dans [70] utilisent l'algorithme d'El Gamal. Dans notre contribution, nous avons choisi d'étendre l'algorithme RSA [85] afin d'obtenir une $SCRS_g$. Nous présentons dans ce qui suit les principes de notre réalisation :

D'abord, l'émetteur génère deux entiers premiers p et q assez grand qui satisfassent les égalités suivantes :

$$n = p \,.\, q \tag{4.3}$$

$$\phi = (p-1)(q-1) \tag{4.4}$$

La clé de chiffrement est calculée selon l'algorithme RSA, et est donc générée comme suit :

$$1 < k < \phi \quad et \tag{4.5}$$

$$pgcd(k, \phi) = 1 \tag{4.6}$$

Le chiffrement du document original M_0 est effectué en générant le document chiffré M_1 comme suit :

$$M_1 = (M_0)^k \ mod \ n \tag{4.7}$$

Ce dernier peut être déchiffré en utilisant une clé de déchiffrement d calculée comme suit :

$$1 < d < \phi \quad et \tag{4.8}$$

$$k.d = 1 \ mod \ \phi \tag{4.9}$$

En utilisant cette clé, le déchiffrement de M_1 se fait selon l'équation suivante :

$$M_0 = (M_1)^d \ mod \ n \tag{4.10}$$

Cela dit pour satisfaire les besoins en clés de toute la séquence de chiffrement, l'émetteur doit générer non pas une mais un ensemble des clés de chiffrement $\{k_{0\leq i\leq N}\}$. Nous étendons donc l'algorithme RSA du fonctionnement mono-clé à un fonctionnement **multi-clés**. Le calcul de ces clés devient comme suit :

$$1 < k_i < \phi \quad et \tag{4.11}$$

$$pgcd(k_i, \phi) = 1 \; pour \; 0 \leq i \leq N \tag{4.12}$$

De plus, il doit générer l'ensemble des clés de déchiffrement correspondantes (Le calcul de ces clés de déchiffrement peut être effectué en utilisant l'Algorithme Euclidien Étendu [23]). Chaque clé de déchiffrement doit satisfaire les deux équations suivantes :

$$1 < d_i < \phi \quad et \tag{4.13}$$

$$(k_0.k_i).d_i = 1 \; mod \; \phi \tag{4.14}$$

L'émetteur envoie n et les clés de chiffrement k_i au *proxy*, et il chiffre le contenu original M_0 avec k_0 et génère le format chiffré M_1 avec l'équation (4.7) :

$$M_1 = (M_0)^{k_0} \; mod \; n$$

Le *proxy* reçoit la donnée chiffrée M_1 puis calcul pour chaque destinataire C_i une nouvelle version chiffrée utilisant les clés de chiffrement k_i de la manière suivante :

$$M_{c_i} = (M_1)^{k_i} \; mod \; n \tag{4.15}$$

Le *proxy*, après avoir envoyé la clé de déchiffrement d_i et n au client C_i, envoie la donnée chiffrée M_{c_i}. Le client C_i à la réception déchiffre le contenu reçu utilisant d_i avec :

$$M_0 = g_{-1,i}(M_{c_i}) = (M_{c_i})^{d_i} \; mod \; n \tag{4.16}$$

Exemple Afin d'illustrer cette implémentation prenons l'exemple suivant :

Supposant que les nombres premiers sont $p = 5$ et $q = 7$ (même si dans la réalité p et q doivent être très grands). Selon l'équation (4.3) n = 5 x 7 = 35 et ϕ = (5-1) x (7-1) = 24.

Soit les trois clé de déchiffrement $k_0 = 5$, $k_1 = 11$, $k_2 = 13$.

Si la donnée originale $M_0 = 10$, la données transmise au *proxy* est selon l'équation (4.7) $\boxed{M_1 = (10)^5 \ mod \ 35 = 5}$

$\Rightarrow$ Pour transmettre la données M_1 au client C_1, le *proxy* génère la clé de déchiffrement d_1 tel que (5 x 11) x $d_1 = 1 \ mod \ 24$. En utilisant l'algorithme Euclidien Étendu nous obtenons $d_1 = 7$. Alors, la version chiffrée pour le client C_1 est : $\boxed{M_{c_1} \ = \ (M_1)^{k_1} \ mod \ n \ = \ (5)^{11} \ mod \ 35 = 10}$

Le Client C_1 peut alors déchiffrer la version M_{c_1} pour retrouver la donnée originale M_0 comme suit : $\boxed{M_0 \ = \ (M_{c_1})^{k_1} \ mod \ n = (10)^7 \ mod \ 35 = 10}$

$\Rightarrow$ Pour transmettre la données M_1 au le client C_2, le *proxy* génère la clé de déchiffrement d_2 tel que (5 x 13) x $d_2 = 1 \ mod \ 24$. En utilisant l'algorithme Euclidien Étendu nous obtenons $d_2 = 17$. Alors, la version chiffrée pour le client C_2 est : $\boxed{M_{c_2} \ = \ (M_1)^{k_2} \ mod \ n = (5)^{13} \ mod \ 35 = 5}$

Le Client C_2 peut alors déchiffrer la version M_{c_2} pour retrouver la donnée originale M_0 comme suit : $\boxed{M_0 = (M_{c_2})^{k_2} \ mod \ n = (5)^{17} \ mod \ 35 = 10}$

Théorème 1 *Le chiffrement décrit précédemment est une Séquence de Chiffrement* ***Inversible***

preuve :

Pour montrer que le mécanisme précédent est une Séquence de Chiffrement **Inversible**, nous devons montrer qu'à partir de M_i pour $i \geq 1$ nous pouvons calculer M_0 et M_1. Sans perte de généralisation, considérons M_2 comme la donnée en entrée. La génération de M_2 est faite avec :

$$\begin{aligned} M_2 &= (M_1)^{k_2} \ mod \ n \\ &= [(M_0)^{k_1} mod(n)]^{k_2} \ mod \ n \\ &= (M_0)^{k_1.k_2} \ mod \ n \end{aligned}$$

Soit R le résultat du déchiffrement de M_2, R est alors égal à :

$$\begin{aligned} R &= (M_2)^{d_2} \ mod \ n \\ &= [(M_0)^{k_1.k_2} \ mod \ n]^{d_2} \ mod \ n \\ &= (M_0)^{k_1.k_2.d_2} \ mod \ n \end{aligned}$$

A partir du moment où la clé de chiffrement k_1 et la clé de déchiffrement d_2 sont générées de sorte que $k_1.k_2.d_2 = j.(p-1).(q-1)+1$, où j est un entier, nous pouvons

réecrire le résultat obtenu R par :

$$R = [(M_0)(M_0)^{j.(p-1).(q-1)}] \; mod \; n \tag{4.17}$$

En s'appuyant sur le théorème d'Euler [85] qui définit $X^{p-1} = 1 \; mod \, n$ nous obtenons l'expression suivante :

$$\begin{aligned} R &= (M_0)(M_0)^{j.(p-1).(q-1)} \\ &= (M_0).1^{j.(q-1)} \\ &= (M_0) mod(p) \end{aligned}$$

$$\begin{aligned} R &= (M_0)(M_0)^{j.(p-1).(q-1)} \\ &= (M_0).1^{j.(p-1)} \\ &= (M_0) mod(q) \end{aligned}$$

Et comme p et q sont premiers, en utilisant le théorème de Chinese Remainder [85] nous avons :

$$\begin{aligned} R &= (M_0)(M_0)^{j.(p-1).(q-1)} \\ &= (M_0) \; mod \; n \\ &= M_0 \end{aligned}$$

Par conséquent, à partir de M_2 nous avons pu extraire M_0 en disposant de la clé d_2. Nous pouvons appliquer une procédure similaire afin d'obtenir M_2 à partir de M_1 et une autre clé de déchiffrement correspondante. En résumé, à partir de M_i nous pouvons extraire M_1 et la donnée originale M_0 si les clé de déchiffrement correspondantes sont connues.

■

Théorème 2 *Le chiffrement décrit précédemment est une Séquence de Chiffrement Inversible* ***Asymétrique***

preuve :

Pour montrer que le mécanisme précédent est une Séquence de Chiffrement Inversible **Asymétrique**, nous devons montrer qu'à partir de M_i pour $i \geq 1$ il est impossible

de de calculer la donnée originale M_0 même si nous disposons de k_1, k_2, k_N et n. Comme nous utilisons le chiffrement RSA, trouver les clés de déchiffrement dans les équations (4.13) et (4.14) revient à trouver les deux facteurs premiers p et q, qui est, par convention, mathématiquement impossible à calculer si p et q sont assez grands et correctement choisis. Par conséquent il est mathématiquement impossible de trouver $g_{i,j}$.

■

4.3.4 Fonctionnement

Dans XSST, le chiffrement des documents multimédias repose sur une granularité au niveau objet-média. Autrement dit, les mécanismes de chiffrement/rechiffrement/ déchiffrement décrits précédemment s'appliquent sur un objet-média du document original. Tous les objets-médias qu'ils soient adaptables ou non sont chiffrés de la même manière. Une session de communication entre un client $\mathcal{C}_i$, un *proxy* $\mathcal{P}$ et un serveur $\mathcal{S}$ utilisant XSST fonctionne alors la manière suivante :

A **Initialisation de la connexion**

Le client $\mathcal{C}_i$ envoie une requête et un certificat de son identité au *proxy* $\mathcal{P}$. Le *proxy* transfère cette requête au serveur $\mathcal{S}$.

Soit Cer_i le certificat de $\mathcal{C}_i$ tel que ($Cer_i = (A_i, \{A_i\}_{B_i})$, où A_i et B_i sont respectivement les clés publique et privée de $\mathcal{C}_i$, et M_k désigne le message M chiffré avec la clé k.

Le serveur garde une liste des clés publiques de tous ses clients authentifiés. Pour vérifier l'identité du client $\mathcal{C}_i$ ayant envoyé une requête, $\mathcal{S}$ déchiffre $\{A_i\}_{B_i}$ utilisant B_i comme clé de déchiffrement, la requête est satisfaite seulement si le résultat du déchiffrement est égale à A_i et correspond à la clé publique sauvegardée dans la liste du serveur. Ceci est fondé sur le fait que seulement le vrai $\mathcal{C}_i$ dispose de la clé B_i. Par conséquent, seul $\mathcal{C}_i$ peut générer $\{A_i\}_{B_i}$ qui, déchiffrée utilisant B_i produit le même A_i que la copie sauvegardée dans la liste de $\mathcal{S}$.

Une étape supplémentaire nous sert à éliminer les faux clients éventuels, et ce comme suit : Le serveur génère un message aléatoire, l'envoie au *proxy* , qui le transfert à $\mathcal{C}_i$, ce dernier chiffre le message reçu avec sa propre clé privée et le renvoie au *proxy* . $\mathcal{P}$ à son tour chiffre le message avec sa propre clé privée et inclus ce message chiffré dans sa réponse. Enfin, le $\mathcal{S}$ déchiffre le message utilisant les clés publiques de $\mathcal{P}$ et de $\mathcal{C}_i$. La requête ne pourra être satisfaite uniquement si le message déchiffré correspond au message aléatoire initial, si l'authentification réussit le serveur passe à l'étape de la génération des clés.

B **Génération de clé de rechiffrement**
Le serveur génère deux grands nombres premiers aléatoires p et q, puis calcule le produit $n = p * q$, $\phi = (p-1) * (q-1)$, la clé de chiffrement k_0, la clé de rechiffrement k_i et la clé de déchiffrement correspondante d_i. Le serveur sauvegarde les paramètres k_0 et n avec un identifiant unique ID. Il répond à $\mathcal{P}$ avec (a) la clé de déchiffrement K_i, qui est chiffrée avec la clé publique de $\mathcal{C}_i$ et (b) la clé de déchiffrement correspondant (d_i). Et comme la clé de rechiffrement K_i est chiffrée avec la clé publique du *proxy* , personne ne peut la retrouver par une écoute clandestine sur le canal entre le serveur et $\mathcal{P}$ par exemple. Les attaques par collusion peuvent ainsi être évitées. Considérons deux attaquants qui connaissent la clé de rechiffrement k_i et k_j (par écoute clandestine) et les clés de déchiffrement d_i et d_j, il est démontable qu'ils sont capables de calculer le paramètre secret. A partir du moment que le paramètre secret est révélé, les attaquants peuvent dériver la clé de déchiffrement de $\mathcal{P}$ ou n'importe quelle clé de déchiffrement. C'est pour cela que la clé de rechiffrement k_i doit être chiffrée de sorte qu'elle ne soit accessible que par le *proxy* .

C **Calcul de la clé de déchiffrement** $\mathcal{P}$ répond avec un acquittement au client $\mathcal{C}_i$ en incluant la clé d_i chiffrée. Le client la déchiffre utilisant sa propre clé privée pour retrouver d_i.

D **Chiffrement des données et transmission au *proxy***
$\mathcal{S}$ utilise la clé de chiffrement k_0 et n pour chiffrer les données. Il transmet ensuite les paquets chiffrés au *proxy* via son canal de communication classique. A la réception le *proxy* $\mathcal{P}$ adapte les données sans avoir à les déchiffrer.

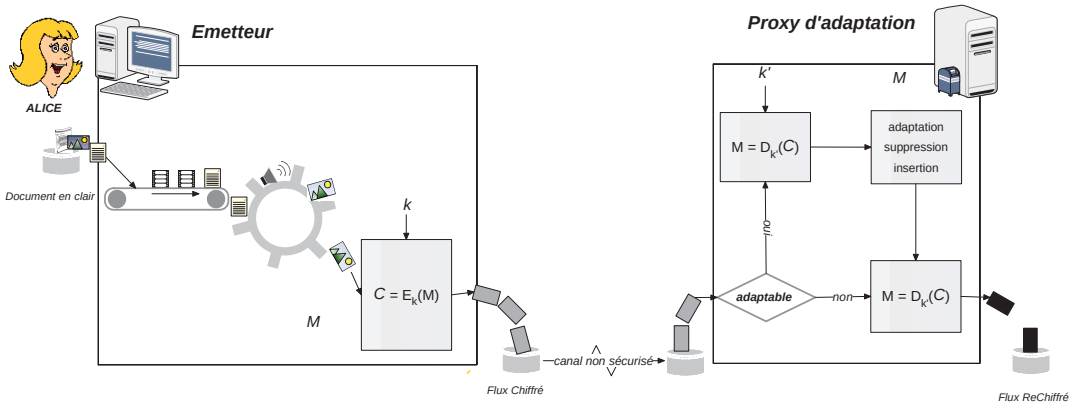


FIGURE 4.12 – Chiffrement XSST entre Serveur et*proxy* d'adaptation

E **Rechiffrement des données et envois vers le client**
Le *proxy* $\mathcal{P}$ utilise la clé de rechiffrement k_i et n pour rechiffrer les données déjà chiffrées reçues. Il transmet ensuite le résultat au client i par le canal de communication classique. Après réception le client $\mathcal{C}_i$ utilise la clé de déchiffrement d_i pour déchiffrer les paquets reçus.

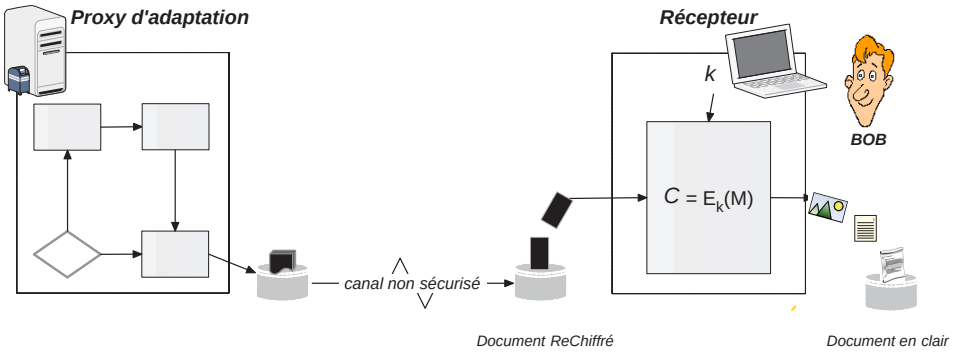


Figure 4.13 – Chiffrement XSST entre *proxy* et client final

4.3.5 Discussion

L'objectif premier de la solution XSST est de garantir le troisième critère de sécurité visé par nos travaux en l'occurrence la confidentialité de bout-en-bout des données (*cf.* section 2.1). Comme nous l'avons démontré dans les paragraphes précédents, cet objectif est largement satisfait pour tous les objets-médias dont l'émetteur (ou le mécanisme d'émission) décide la confidentialité à condition qu'ils ne soient pas obligatoirement adaptables. De plus, la solution XSST garantit la sécurité contre les différents types d'attaques par collusion.

Cependant, s'agissant des objets-médias adaptables, XSST ne peut garantir la confidentialité de bout-en-bout, et ce à cause de contraintes liées aux techniques d'adaptation de contenus, car à ce jour, à notre connaissance, aucune solution «générique» d'adaptation de contenu ne peut être appliquée à tout type de contenu chiffré. Certes, il existe quelques solutions d'adaptation de contenus chiffrés destinées à des formats médias spécifiques tels que la vidéo scalable, mais aucune de ces solutions ne peut être généralisée à tout type de contenu.

XSST garantit ainsi le niveau le plus élevé possible en termes de confidentialité des données pour les documents multimédias adaptables, et permet de ne déchiffrer que les objets-médias adaptables lors des opérations d'adaptation.

4.4 Synthèse

Nous avons présenté dans ce chapitre deux de nos contributions majeures, la technique d'authentification de contenu AMCA, et la technique XSST, notre approche de chiffrement de contenus adaptables.

La première contribution, AMCA, est une technique de signature de contenus multimédias adaptables garantissant l'authentification et l'intégrité de ces derniers. L'approche utilisée dans cette solution est celle des arbres de hachage de Merkle afin de permettre les opérations d'adaptations sur des contenus signés.

L'apport de l'approche AMCA peut se résumer en trois éléments : une technique de signature, une technique d'adaptation (modification) de contenu signé et une technique de contrôle de flux multi-signés.

Nous avons vu dans ce chapitre les détails du fonctionnement de l'approche AMCA, les algorithmes utilisés et des exemples de déroulement de chacun des algorithmes proposés.

La seconde contribution, XSST, a comme objectif de garantir la confidentialité de bout-en-bout de flux chiffrés d'un émetteur vers plusieurs récepteurs en passant par des *proxies* d'adaptation. L'approche proposée est fondée sur l'utilisation de séquences de chiffrement asymétrique dans un schéma de rechiffrement unidirectionnel. Appliquée aux documents multimédias adaptables, cette solution garantit à l'émetteur du contenu la confidentialité de tous les objets-médias non adaptables et protège les contenus échangés de tout type d'attaque par collusion.

Dans ce chapitre, nous avons volontairement limité la présentation de AMCA et de XSST aux détails de fonctionnement et aux algorithmes utilisés, les détails d'implémentation seront présentés dans le chapitre 5 consacré à la description de SEMAFOR, une solution de sécurisation globale pour les documents multimédias adaptables intégrant les deux solutions AMCA et XSST. Des mesures de performances validant les apports de ces approches seront eux présentés dans le chapitre 6 dédié à cet effet.

Chapitre 5

Présentation de la plate-forme SEMAFOR

Après avoir présenté dans le chapitre précédent AMCA et XSST nos approches pour l'authentification et le chiffrement de documents multimédias adaptables, nous présentons dans ce chapitre l'architecture complète de notre chaîne de diffusion sécurisée de bout-en-bout. L'implémentation de cette architecture s'appelle SEMAFOR [1], une plate-forme de diffusion de contenus multimédias adaptables totalement sécurisée, où l'adaptation des contenus est à la charge de nœuds intermédiaires répartis sur le réseau dans une topologie $\mathcal{SPC}$.

Ce chapitre est composé de quatre sections, la première a comme objet de présenter la plate-forme SEMAFOR, ses objectifs, son fonctionnement et ses apports, tandis que la deuxième décrit l'architecture logicielle de la plate-forme et les mécanismes qui viennent raffiner AMCA et XSST pour le contrôle d'accès aux documents, l'adaptation et la gestion des droits. La troisième section quant à elle donne quelques détails sur les approches de développement utilisées dans SEMAFOR avant que la section quatre illustre l'utilisation de la plate-forme SEMAFOR à travers un exemple d'utilisation.

1. SEMAFOR est l'acronyme de *SEcure Multimedia Adaptation platFORm*

5.1 Introduction

Nous avons initié le développement de la plate-forme SEMAFOR en tant que projet *open source* au sein de la communauté JXTA [2][39]. JXTA définissant un ensemble de protocoles ouverts pour les liaisons pair-à-pair, permettant de connecter et faire collaborer n'importe quel matériel connecté au réseau (téléphone mobile, PC...), nous nous sommes appuyés sur ces solutions pour implémenter nos propositions pour la diffusion et la sécurisation de contenus multimédias adaptables. Mais très vite, les choix d'implémentation et les critères de performance nous ont conduit à faire de SEMAFOR un projet autonome en développant nos propres solutions de communication. Nous avons donc fait de SEMAFOR un projet interne qui nous a servi à valider les différents choix proposés.

Plusieurs objectifs et motivations ont guidé les travaux réalisés dans SEMAFOR, il était par exemple important pour nous de :

- réaliser un système multimédia adaptatif, capable d'interconnecter différents types de terminaux (PC, PDA, *smartphone*...) ;
- mettre en œuvre les propositions théoriques AMCA et XSST dans une architecture de sécurisation de bout-en-bout ;
- fournir une plate-forme de base pour des utilisations multiples (personalisable) ;
- gérer le maximum de formats adaptables possible (scalables ou non) ;
- réduire les délais de transmission d'un document adaptable sécurisé.

L'architecture de SEMAFOR est le fruit de notre collaboration et participation à plusieurs projets (JXME [5], MyJXTA [47], JxtaSound [3] ...), certains ont servi pour la mise en œuvre de quelques composants logiciels dans la plate-forme SEMAFOR, plus de détails seront donnés plus loin dans la section 5.3.

5.1.1 Topologie réseau

SEMAFOR est une plate-forme de diffusion de contenus multimédias adaptables. Son architecture réseau, telle que nous la décrivons dans la figure 5.1, se com-

2. `http://www.jxta.org`, JXTA étant une plate-forme de développement d'application initiée par *Sun* en 2001 dont le but est de fournir les outils de base pour réaliser des réseaux P2P
3. Projet maintenant obsolète car non retenu par la communauté JXTA

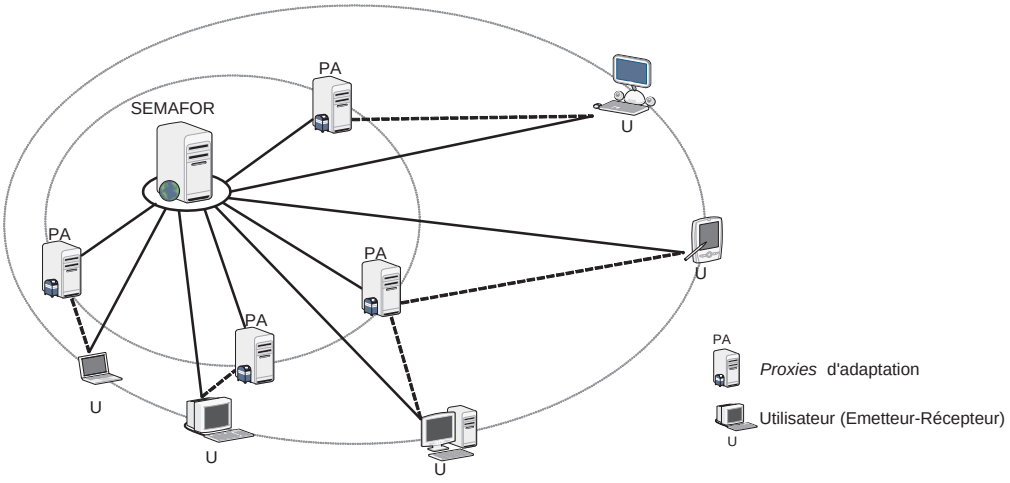


FIGURE 5.1 – Topologie pair-à-pair utilisée dans SEMAFOR

pose de deux types d'intervenants : des utilisateurs (émetteurs/récepteurs) et des *proxies* d'adaptation, tous connectés à un serveur hébergeant l'application SEMAFOR.

Dans sa mise en œuvre actuelle (et que nous présentons dans ce chapitre), la plate-forme SEMAFOR se déploie sous la forme d'une application Web sur un réseau pair-à-pair, autrement dit tous les utilisateurs connectés à l'application peuvent émettre et recevoir des contenus multimédias. Mais elle peut aussi être déployée sur une topologie client/serveur, il suffit pour cela de séparer les modules d'émissions et les modules de réception, les premiers seront chargés uniquement par les serveurs et les seconds par les clients.

5.1.2 Fonctionnement

Le fonctionnement de SEMAFOR est celui d'une plate-forme de communication multimédia classique. Autrement dit, les utilisateurs disposent d'une boîte de réception, d'une messagerie instantanée et une gestion automatique des messages multimédias. Mais la particularité, c'est la gestion du format de réception des messages multimédias. Un message multimédia transmis est restitué au destinataire dans un format adapté à son profil (capacité du terminal, profil utilisateur, etc.).

Les utilisateurs doivent s'enregistrer auprès de la plate-forme SEMAFOR qui centralise une base de données des profils et des contextes de ces utilisateurs. Chaque utilisateur enregistré peut émettre des messages multimédias à un ou plusieurs utilisateurs. En cas de nécessité d'adapter un message émis, l'application redirige les flux

vers un nœud intermédiaire (*proxy* d'adaptation) qui génère une version adaptée du contenu émis avant de la transmettre au destinataire final.

La sécurisation des documents et des échanges dans SEMAFOR s'inspire du principe général des systèmes de DRM pour la gestion de la chaîne de sécurité de l'émetteur jusqu'au récepteur. Elle reprend l'idée des licences d'utilisation et de description des droits d'accès aux documents échangés.

Ainsi, pour l'émission d'un flux adaptable M, l'émetteur génère deux licences :

– une **licence d'adaptation (LA)** à destination des *proxies*
– et une **licence de lecture (LL)** à destination du (ou des) client(s) final(aux)

Ces licences contiennent les informations nécessaires pour l'adaptation et la lecture du contenu émis telles que les signatures, les politiques d'adaptation. La figure 5.2 décrit un exemple de transfert de ces licences lors d'une émission d'un document adaptable d'un émetteur vers un destinataire.

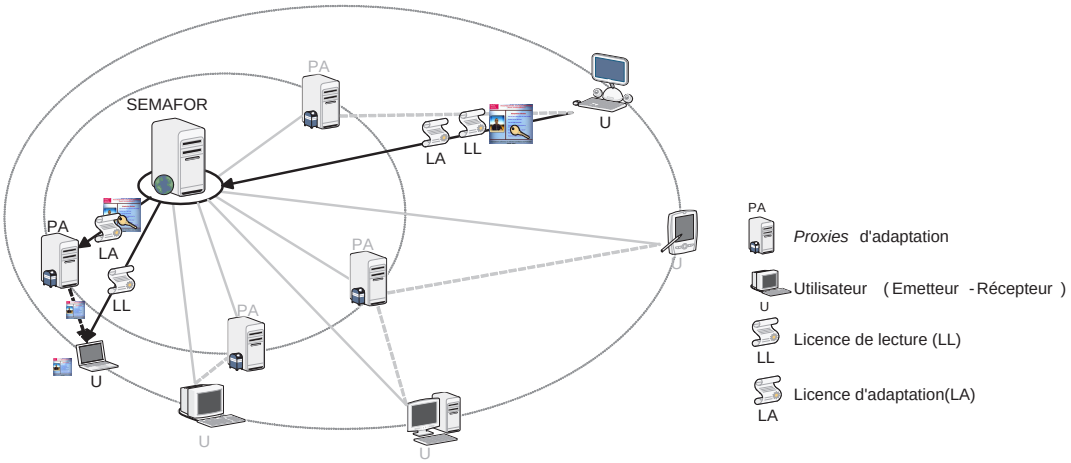


Figure 5.2 – Diffusion des licences dans SEMAFOR

Par ailleurs, avant de générer ces licences l'émetteur doit signer et chiffrer le contenu. Pour cela SEMAFOR implémente les deux mécanismes présentés dans le chapitre précédent AMCA et XSST. Ainsi, l'authentification/vérification des documents suit les principes d'AMCA et le chiffrement/rechiffrement des objets-médias suit les principes d'XSST.

Le processus de sécurisation d'un document multimédia peut alors être découpé en quatre étapes :

1. génération de la politique d'adaptation

2. chiffrement des objets médias
3. génération de l'AHM à partir des objets chiffrés et signature du document
4. génération des licences

L'adaptation quant a elle se décompose en quatre étapes :

5. déchiffrement des objets-médias à adapter
6. adaptation du contenu
7. adaptation de l'AHM et signature des *freeleaves*
8. rechiffrement des objets médias

La lecture du contenu au niveau du récepteur peut être faite après les deux étapes suivantes :

9. déchiffrement des objets-médias
10. vérification de l'authenticité du contenu reçu

La section suivante décrit l'architecture logicielle de la plate-forme SEMAFOR et donne plus de détails sur les mécanismes qui assurent ces opérations de sécurisation, mis en œuvre pour compléter AMCA et XSST.

5.2 Canevas logiciel SEMAFOR

L'architecture logicielle de l'application SEMAFOR est répartie sur trois types de nœud selon la topologie $\mathcal{SPC}$. Cette architecture est donc découpée verticalement en trois parties :

Les composants Serveur comprenant tous les composants nécessaires à l'émission des documents multimédias ;

Les composants *Proxy* qui interviennent pour adapter les documents émis ;

Les composants Client qui sont composés des outils nécessaires à la lecture/vérification des contenus reçus.

A noter que l'utilisateur qui se connecte à l'application SEMAFOR peut (selon son profil matériel) être émetteur/récepteur et dans ce cas il disposera des composants Serveur et des composants Client, ou tout simplement récepteur (ex. terminal léger) et dans ce cas il ne disposera que des modules Client.

L'application SEMAFOR est aussi découpée horizontalement selon trois niveaux de sécurité :

le contrôle d'accès avec les gestionnaires de profils et de politiques d'accès ;

la signature qui regroupe les composants de AMCA ;

le chiffrement qui regroupe les composants d'XSST.

La figure 5.3 présente une vue d'ensemble des composants et des services de SEMAFOR et leur répartition sur les trois types de nœud. On y retrouve aussi la description des 10 étapes de l'acheminement des flux multimédias présentées dans la section précédente (*cf.* section 5.1.2).

Dans la suite de cette section nous allons décrire chacun des composants de l'application SEMAFOR en utilisant le découpage horizontal de l'application, autrement dit, en couches de sécurité.

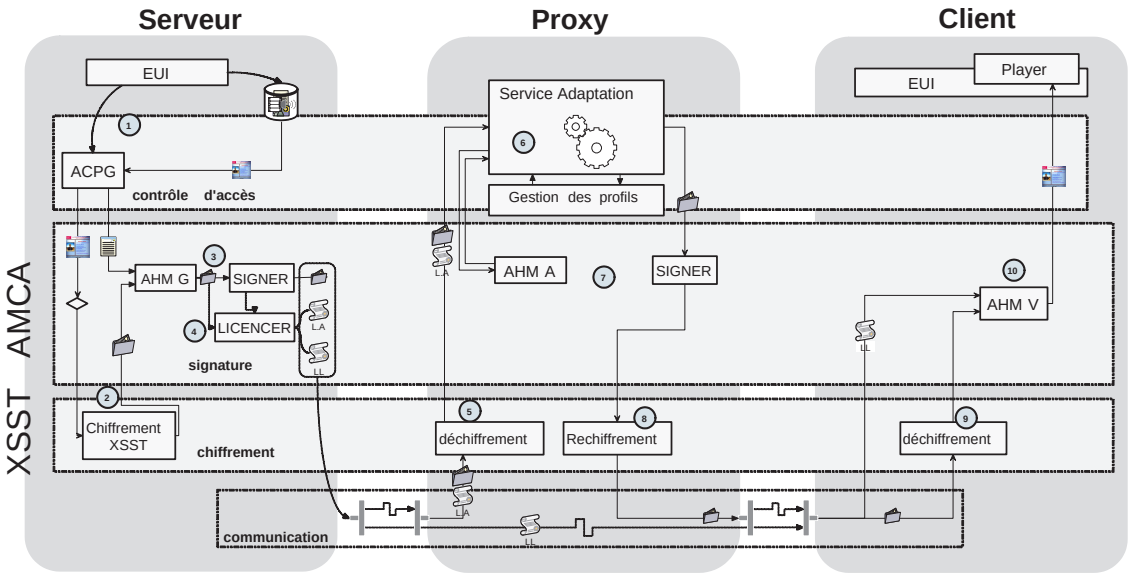


Figure 5.3 – Architecture logicielle de SEMAFOR

5.2.1 Le gestionnaire du contrôle d'accès (ACM)

L'objectif de notre gestionnaire de contrôle d'accès, que nous appellerons aussi ACM [4], est double, premièrement, il doit gérer les différents types d'intervenants dans une communication réseau (serveur, client ou *proxy*), deuxièmement, il doit permettre à un émetteur (serveur) de spécifier des autorisations d'accès aux différentes parties d'un même document, autrement dit pour chaque partie d'un document à émettre, définir les parties non modifiables par les *proxies* , celles où ces dernières peuvent insérer du nouveau contenu.

Pour mettre en œuvre ce contrôle d'accès, nous avons trouvé dans la littérature une multitude de techniques (ACL, DAC, MAC, RBAC, TMAC, etc.) dont un panorama assez complet est présenté dans [83]. Dans une précédente version de SEMAFOR nous avions opté pour une ACL (*Access Control List*) qui centralisait les noms des utilisateurs de la plate-forme et leurs mots de passe ; cette solution fut très vite obsolète vu la difficulté d'administration et la nécessité d'avoir plus d'informations sur chaque utilisateur. Après quelques essais avec des schémas DAC, MAC, et RBAC, la topologie de notre plate-forme $\mathcal{SPC}$ nous a emmenés au choix d'un schéma de type RBAC «simplifié» définissant trois rôles principaux, *owner* pour l'émetteur du contenu, *adapter* pour le PA, et *consumer* pour le client destinataire. Ce service intervient donc dans deux composants de l'architecture SEMAFOR, le générateur

4. Access Control Manager

de politiques d'accès ACPG [5] et le Gestionnaire de profils.

5.2.1.1 ACPG

Concevoir un générateur de politiques d'accès nécessite de définir les sujets, les objets avec et les règles d'autorisations spécifiant si tel ou tel sujet a la permission ou l'interdiction d'accéder à tel ou tel objet.

Dans SEMAFOR l'ACPG est le composant responsable de la génération des politiques d'accès aux documents, il s'appuie sur l'ACM pour définir les trois sujets (*owner*, *adapter*, *consumer*). Les objets sont les objets-médias composant le document multimédia (selon sa granularité), et un exemple d'autorisation serait de permettre aux *proxies* d'adapter une vidéo ou de leur interdire de modifier un texte, etc. selon les désirs de l'émetteur.

Concrètement, l'ACPG est composé d'un analyseur de contenus multimédias, qui, en recevant la description métadonnées d'un document multimédia, va extraire tous les objets-médias de ce contenu et les présenter dans l'interface graphique décrite par la figure 5.4. L'émetteur pourra ensuite sélectionner les objets qu'il veut garder inchangés et inaccessibles aux *proxies*.

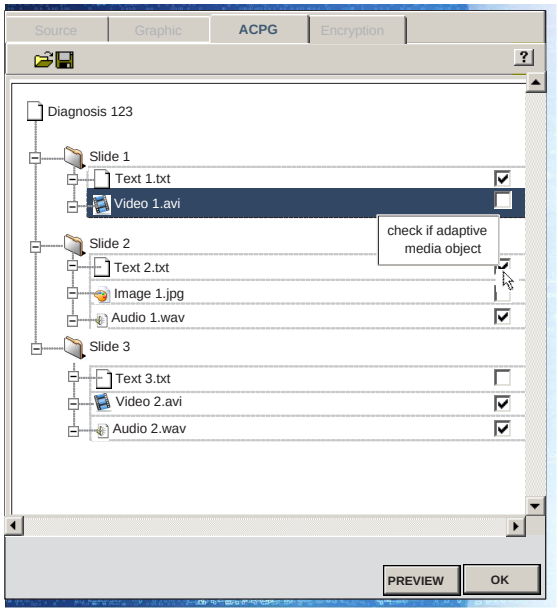


Figure 5.4 – Interface graphique du composant ACPG dans SEMAFOR

5. Access Control Policy Generator

Le mécanisme proposé permet une expression simple et précise de la politique de sécurité. Un document multimédia est représenté selon le modèle décrit précédemment dans le paragraphe 4.2.2.1 où l'objet-média est la plus petite entité à laquelle une autorisation peut être associée. Cela signifie qu'une règle d'autorisation peut permettre ou interdire l'accès à un simple objet (par exemple, une vidéo, un texte) ou à un ensemble de nœuds (par exemple à toute une séquence contenu dans un même groupe d'éléments donné).

Cette étape fournit à l'analyseur les éléments nécessaires pour générer les *freeleaves* dans l'arbre de Merkle du document initial.

5.2.1.2 Gestionnaire de profils

Ce module est une base de données stockant différentes informations concernant le profil des utilisateurs. Parmi ces informations nous retrouvons le contexte de l'utilisateur, c'est-à-dire les caractéristiques et préférences de l'utilisateur, les capacités et propriétés de son terminal et éventuellement les propriétés de son réseau d'accès.

Les utilisateurs peuvent mettre à jour et modifier leur profil à tout moment. Les informations contextuelles dynamiques telles que la localisation d'un utilisateur ou les paramètres dynamiques du réseau sont déterminées lors de l'exécution des requêtes.

5.2.1.3 Gestionnaire des droits

Dans SEMAFOR, la gestion des droits s'inspire du principe général des plates-formes de DRM, autrement dit, nous générons des licences d'utilisation pour permettre l'exploitation des contenus émis. L'idée n'étant pas de proposer une nouvelle plate-forme de DRM mais plutôt de fournir aux émetteurs de contenus un moyen pour le contrôle d'accès à leurs documents. De cette sorte, afin de s'adapter à nos besoins de sécurité et à la topologie $\mathcal{SPC}$ nous avons modifié certaines contraintes spécifiques aux plates-formes de DRM telles qu'elles étaient précédemment décrites dans le paragraphe 2.3 page 43.

Modèle de données Le modèle de données utilisé dans notre architecture est fondé sur le celui de l'ISO REL, la figure 5.5 décrit ce modèle de données.

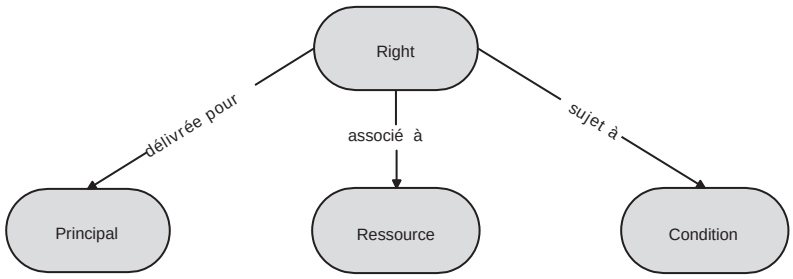


Figure 5.5 – Model de Données dans SEMAFOR

5.2.2 Le service d'authentification

Comme nous l'avons vu dans le chapitre 4, le service d'authentification des contenus est fondé sur l'utilisation des Arbres de Hachage de Merkle. Le composant responsable des traitement relatifs à ces structures dans SEMAFOR est désigné par gestionnaire d'arbres de Merkle, il fournit trois modules AHMG, AHMA et AHMV :

AHMG [6] ou le générateur d'AHM, qui à partir d'un document multimédia et de sa politique d'adaptation génère l'AHM correspondant et crée les deux licences (adaptation et utilisation) nécessaires à son émission ;

AHMA [7] ou l'adaptateur d'AHM, qui à partir d'un document multimédia adapté, sa licence d'adaptation et ses objets-médias adaptés adapte l'AHM du document en complétant les *freeleaves* et génère une nouvelle signature englobant les objets-médias adaptés ;

AHMV [8] ou le vérificateur d'AHM, qui vérifie la validité d'un contenu adapté en utilisant la licence d'utilisation correspondante.

5.2.3 Le chiffrement/rechiffrement/déchiffrement

Le service de chiffrement/rechiffrement/déchiffrement des contenus utilise le principe des algorithmes présentés dans XSST. Il collabore directement avec les composants du contrôle d'accès pour la gestion des clés utilisées pour le chiffrement et pour la politique d'adaptation. Les modules le constituant dans SEMAFOR sont désigné par *XSSTEncrypter, XSSTReencrypter* et *XSSTDecrypter* :

XSSTEncrypter utilisé par l'émetteur, ce module permet de chiffrer les objets-

6. AHM generator
7. AHM adapter
8. AHM validator

médias d'un document multimédia en se fondant sur l'adaptabilité de chaque objet, telle qu'elle est décrite dans la politique d'adaptation du document reçue de la part des composants de contrôle d'accès ;

XSSTReencrypter situé au niveau du *proxy* d'adaptation, ce module gère le déchiffrement des objets-médias adaptables et le rechiffrement des objets pour la transmission aux client finaux ;

XSSTDecrypter ce module permet aux clients destinataires de déchiffrer un contenu chiffré par l'émetteur et rechiffré par le *proxy* d'adaptation selon les principe de la solution XSST.

5.2.4 Le service d'adaptation

Notre solution, telle qu'elle est présentée dans ce manuscrit, consiste à utiliser des *proxies* qui adaptent dynamiquement les données transmises, de façon à ce que le terminal récepteur puisse les exploiter efficacement. Le service d'adaptation est donc un composant présent sur tous les *proxies* de la plate-forme SEMAFOR. Cela dit, l'adaptation n'étant pas le *focus* principal de nos travaux ci-présents nous allons nous limiter dans ce paragraphe à la description du fonctionnement de ce service.

Comme le montre la figure 5.6, le composant «Service d'Adapataion» est divisé en deux modules : un module de décision et un module de traitement fournissant les mécanismes d'adaptation.

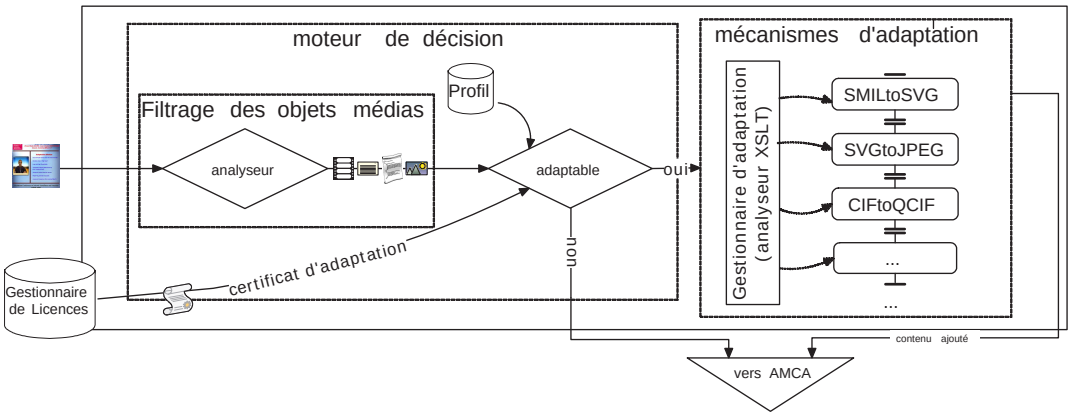


Figure 5.6 – Service d'adaptation dans SEMAFOR

5.2.4.1 Décision

La décision est importante dans la mise en place d'une transmission nécessitant une adaptation : le service doit être en mesure de déterminer quelle configuration optimale le système doit adopter. Pour cela il bénéficie des éléments suivants :

- **gestionnaire de session** Ce composant est en charge d'interagir avec les intervenants avec la même session SEMAFOR, et de récupérer les informations sur les profils clients ;
- **moteur de décision** c'est le composant principal de la partie décisionnelle. Il utilise une procédure de décision multicritères qui prend en paramètres la licence d'adaptation les informations relatives au profil du client, et au type de données à adapter. Il oriente ensuite les objets-médias dont l'adaptation est faisable et nécessaire vers les mécanismes d'adaptation et les autres vers les modules AMCA.

5.2.4.2 Mécanismes d'adaptation

Ce module implique aussi deux sous-modules : le gestionnaire d'adaptation et un ensemble de filtres d'adaptation.

- **gestionnaire d'adaptation** Ce sous-module est chargé de la gestion de l'adaptation au niveau du PA, autrement dit, la mise en place des mécanismes d'adaptation, le démarrage et arrêt des processus d'adaptation, la modification des paramètres ...
- **filtres d'adaptation** Il s'agit de la liste des mécanismes d'adaptation, ces différents sous-modules mis en œuvre au niveau de chaque *proxy* agissent comme des filtres (transformation vidéo, modification automatique d'image, redimensionnement, recadrage, etc.), des mécanismes de transcodage, etc. Les données sont adaptées en fonction des directives données par le moteur de décision.

5.3 Détails d'implémentation

Le prototype de l'architecture logicielle de SEMAFOR (*cf.* section 5.2 figure 5.3) a été développé afin de montrer la faisabilité et la validité de nos propositions. La figure 5.7 montre l'interface graphique de l'application cliente d'un utilisateur de la plate-forme SEMAFOR.

Ce prototype a été implémenté en JAVA 1.5 sous Eclipse 3.2 [9] et Liquid XML Studio [10] pour le traitement des documents au format XML. Nous avons utilisé un serveur Tomcat/Cocoon [11] pour le stockage des données et pour la mise en œuvre du *proxy* d'adaptation, et avons utilisé MySQL-5.0.51 pour les différentes bases de données mises en œuvre dans le prototype. Les techniques de signatures et de chiffrement sont des extensions aux solutions XML-SIGNATURE [97] et XML-ENCRYPTION [96] du W3C, et enfin Les licences utilisées sont décrites sous un format XML.

Le paquetage logiciel de SEMAFOR comprend à ce jour plus de 15 000 lignes de code. Nous allons dans cette section donner quelques informations sur les choix d'implémentations pour chacun des composants décrits précédemment.

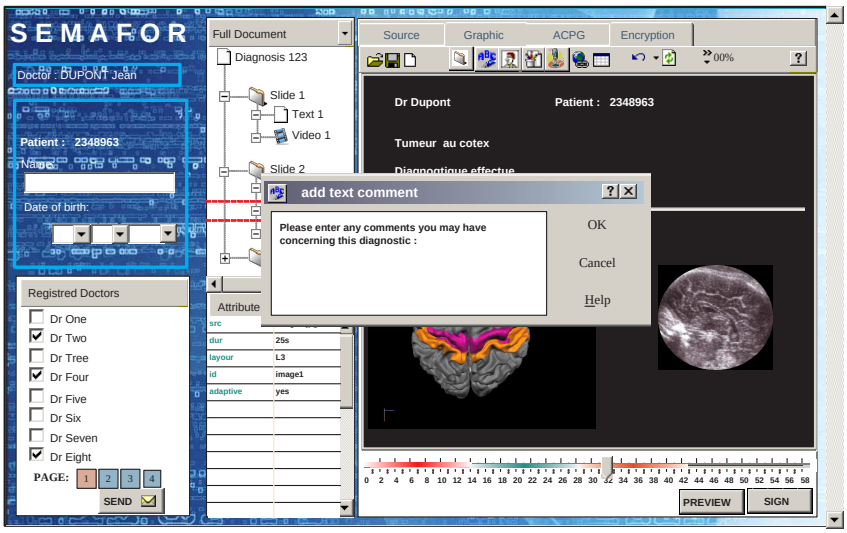


FIGURE 5.7 – Interface graphique de l'application *SEMAFOR*

9. http://www.eclipse.org
10. http://www.liquid-technologies.com/Product_XmlStudio_XmlEditor.aspx
11. http://cocoon.apache.org

5.3.1 ACPG

Les règles qui définissent les autorisations relatives à un document multimédia particulier sont définies dans un document indépendant : une feuille d'autorisation, appelée feuille XAS (XML Authorization Sheet). Lors d'une étape préliminaire, la feuille XAS est traduite en une feuille de style XSLT. Par la suite, à chaque fois qu'un utilisateur demande à accéder au document multimédia, un processeur XSLT applique la feuille de style XSLT ainsi obtenue et fournit à l'utilisateur une vue du document original compatible avec ses droits.

La hiérarchie des sujets est décrite dans un document XML indépendant appelé feuille XSS (XML Subject Sheet). La figure 5.8 présente un exemple de feuille XSS.

```
<subjects>
        <owner>
                        <member id="dupont">
                                <name>Docteur Dupont</name>
                        </member>
        </owner>
        <adapter>
                        <proxy video="yes", image="yes">
                                <member id="p1"/>
                        </proxy>
                        <proxy video="non", image="yes">
                                <member id="p1"/>
                        </proxy>
                        ...
        </adapter>
        <consumer>
                        <client>
                                <member id="Dr 1"/>
                                <profile name=".."/>
                        </client>
                        <client>
                                <member id="Dr 2"/>
                                <profile name=".."/>
                        </client>
                        ...
        </consumer>
</subjects>
```

Figure 5.8 – Description des sujets dans une feuille XSS

5.3.2 Gestionnaire de profils

La Gestionnaire de profils dans SEMAFOR est fondé sur les mécanismes fournis par JXTA et utilise la norme CC/PP [12] [18] pour la description de ces profils. Comme le montre la figure 5.9, CC/PP permet à l'utilisateur de préciser différents types de contraintes particulières d'utilisation. Par exemple, l'utilisateur peut déclarer une infirmité comme la surdité, ce qui aura pour effet de provoquer la réception d'informations uniquement visuelles, etc.

12. Composite Capability/Preference Profiles

Pour la mise en œuvre du Gestionnaire de profils, nous avons utilisé le toolkit *open source* DELI [13] proposé par HP Labs. DELI fournit une API pour permettre à des *servlets* Java de déterminer le contexte de réception d'un dispositif client en utilisant CC/PP ou UAProf.

DELI a également été intégré dans l'environnement d'adaptation XSLT du serveur Apache Cocoon [14].

Ces standards permettent une transmission efficace du contexte du client même dans des environnement à faible bande passante. En effet, au lieu d'envoyer le profil entier avec chaque requête, le client envoie uniquement une référence vers un profil stocké au niveau du serveur dans un *repository* de profils, avec une liste des différences spécifiques à ce client particulier.

```
[ex:LeProfil]
        |
        +--ccpp:component--> [ex:MaterielFinal]
        |                    |
        |                    +--rdf:type-------> [ex:PlateformeMateriel]
        |                    +--ccpp:defaults--> [ex:MaterielDefaut]
        |
        +--ccpp:component--> [ex:LogicielFinal]
        |                    |
        |                    +--rdf:type-------> [ex:PlateformeLogiciel]
        |                    +--ccpp:defaults--> [ex:LogicielDefaut]
        |
        +--ccpp:component--> [ex:NavigateurFinal]
                             |
                             +--rdf:type-------> [ex:AgentUtilNavigateur]
                             +--ccpp:defaults--> [ex:AgentUtilDefaut]
[ex:MaterielDefaut]
        |
        +--rdf:type----> [ex:PlateformeMateriel]
        +--ex:displayWidth--> "240"
        +--ex:displayHeight--> "320"
[ex:LogicielDefaut]
        |
        +--rdf:type----> [ex:PlateformeLogiciel]
        +--ex:name-----> "PocketSMIL"
        +--ex:version--> "3.0"
[ex:AgentUtilDefaut]
        |
        +--rdf:type----> [ex:AgentUtilNavigateur]
        +--ex:name-----> "Pocket Internet Explorer"
        +--ex:version--> "5.0"
        +--ex:vendor---> "Microsoft"
        +--ex:htmlVersionsSupported--> [ ]
                                        |
                                        +--rdf:type---> [rdf:Bag]
                                        +--rdf:_1-----> "2.0"
                                        +--rdf:_2-----> "1.1"

```

Figure 5.9 – Exemple d'un profil CC/PP d'un PDA

13. http://delicon.sourceforge.net
14. http://xml.apache.org/cocoon/developing/deli.html

5.3.3 Composants d'authentification

Les composants du service d'authentification ont été, dans un premier temps, implémentés pour le langage SMIL, en utilisant et en étendant la technologie XML Signature [97] du W3C ; puis ont été porté à d'autres langages comme SVG, MPEG4, etc.

Les figures 5.10 et 5.11 montrent un exemple de document SMIL avant et après signature par le service d'authentification au niveau du serveur.

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="http://www.w3.org/2005/SMIL21/Language">
 <head>
  <meta name="Semafor DOC" content="diagnostic"/>
  <layout type="text/smil-basic-layout">
    <root-layout  id="rootLayout"  width="1020"  height="594"  background-color="black"/>
    <region id="region_1"  left="60"  top="48" width="906"  height="120"/>
    <region id="region_2"  left="204" top="42"  width="552" height="174"/>
    <region id="region_3"  left="60"  top="216" width="366" height="337"/>
    <region id="region_4"  left="486" top="216" width="480" height="335"/>
  </layout>
  </head>

  <body  id="body" begin="0ms">
  <seq>
     <par>
       <img  dur="28" src="image2.jpg"  region="region_3"  id="img_2"/>
       <img  dur="28"  src="image1.jpg"  region="region_2" id="img_1"/>
       <audio dur="5"  src="chimes.wav"   region="region_4"  id="audio_f"/>
     </par>
     <par>
       <text   dur="18"  src="txt1.txt"   region="region_1"  id="text_1" begin="7"/>
       <text  dur="21" src="txt2.txt"  region="region_3"   id="text_2"/>
       <video  dur="27" src="video1.MPG" region="region_4" id="video_1"/>
     </par>
     <par>
       <video dur="37" src="video2.MPG"   region="region_2"  id="video_2"/>
       <text  dur="37"  src="txt3.txt"  region="region_3"  id="text_3"/>
       <img  dur="37"  src="image3.jpg" region="region_4" id="img_3"/>
     </par>
  </seq>
 </body>
</smil>
```

Figure 5.10 – Document multimédia SMIL -avant signature-

5.3.4 Interfaces de Communication

Les protocoles de communication utilisés dans la plate-forme se basent sur les standard : HTTP, SOAP, RTP et SNMP (figure 5.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<smil xmlns="http://www.w3.org/2005/SMIL21/Language">
[...]
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315" />
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />

    <Reference URI="">
       <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
       </Transforms>
       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
       <DigestValue>3F2IwAyZDSZ7Q1s7PNyQIUG/jS8=</DigestValue>
    </Reference>

    <Reference URI=/Licence/smil/body>
             <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

       <Transforms>
          <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
       </Transforms>
       <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
       <DigestValue>2DBFC870E59a09b3a3fe972A28bb</DigestValue>
    </Reference>


  </SignedInfo>

    <SignatureValue>jYqSwvRGlMI5DCno[...]</SignatureValue>
  <KeyInfo>
    <X509Data>
       <X509Certificate>MIIC0zCCAjy[...]</X509Certificate>
    </X509Data>
  </KeyInfo>
</Signature>
</smil>
```

Figure 5.11 – Document multimédia SMIL -après signature-

interface client		interface serveur		interface proxies	
HTTP (cc/pp)	RTP	SNMP	RTP	HTTP	SOAP
TCP	UDP			TCP	

Table 5.1 – Les différents protocoles de communication disponible sur un PA

5.3.5 Bases de données utilisées dans le prototype

Trois principales bases de données ont été développées dans le prototype : le répertoire de profils client, l'annuaire des services d'adaptation, une base de contenus et de métadonnées. Le répertoire des profils stocke les informations relatives aux utilisateurs telles que le nom, l'identifiant, le mot de passe et un lien vers le fichier du profil contextuel (CC/PP). L'annuaire des services d'adaptation stocke les informations relatives aux services d'adaptation telles que les identifiants et le lien vers le fichier de description de service. Enfin, la base de contenus héberge les données d'application et les contenus téléchargeables par l'utilisateur. Les bases de données sont déployées sous MySQL-5.0.51.

L'architecture SEMAFOR de sécurisation de contenus multimédias est générique

et indépendante du domaine applicatif et des configurations matérielle et réseau. Cependant, dans le but de valider concrètement nos concepts et propositions, nous avons développé une implémentation typée de SEMAFOR pour une application médicale. Notre base de données stocke ainsi des modèles de dossiers médicaux.

5.4 Synthèse

Nous avons décrit dans ce chapitre les différents aspects de développement du prototype de la plate-forme SEMAFOR. L'objectifs de cette implémentation est d'intégrer les solutions AMCA et XSST dans une chaîne complète d'outils de sécurisation de contenu adaptable.

Dans la première partie de ce chapitre nous avons décrit l'architecture fonctionnelle de SEMAFOR ainsi que son déploiement logiciel sur les trois sites (Serveur, *Proxy* et Client) avant de détailler le rôle de chacun des composants constituant la plate-forme.

La seconde partie de ce chapitre a présenté les choix d'implémentation du prototype de SEMAFOR.

Le chapitre suivant est consacré aux tests et aux évaluations de performance concernant les opérations de sécurisation de données avec SEMAFOR. Ces tests nous permettrons d'une part de démontrer la faisabilité de notre approche pour réaliser la sécurisation de bout-en-bout, et d'autre part d'évaluer l'apport de l'utilisation de notre approche comparé à une sécurisation objet par objet.

Chapitre 6

Étude de performances

Nous avons présenté dans les deux chapitres précédents l'ensemble de nos propositions pour sécuriser la diffusion de contenus multimédias adaptables par des *proxies*, en insistant sur les apports en sécurité de ces propositions, mais sans donner de mesures concrètes de gain ou de perte en termes de performances. Nous avons ainsi préféré consacrer ce chapitre pour l'évaluation des performances des services d'authentification, de chiffrement et de transmission au sein de la plate-forme SEMAFOR.

Certes, nous aurions pu analyser les solutions proposées séparément, mais la pertinence des résultats nous semble plus importante une fois mise dans une chaîne complète de sécurisation. Cela dit nous tâcherons de séparer les résultats de chaque étape de sécurisation afin de localiser les points forts des points faibles de nos propositions.

Nous avons divisé ce chapitre en deux parties. Dans la première partie et après une description de l'environnement utilisé pour réaliser les mesures, nous donnons les coûts des mécanismes élémentaires, puis le coût global de la sécurisation d'une transmission d'un contenu adaptable, et ce afin d'identifier les surcoûts éventuels liés aux langages de développement utilisés et aux latences de communication. La section 6.2 présente ensuite les différents scénarios de test de performances réalisés.

La seconde partie décrit les résultats obtenus sur les différentes étapes de sécurisation durant les tests. La section 6.3 mesure ainsi les impacts des services de sécurisation de SEMAFOR sur les ressources (utilisation du processeur, de la mémoire, etc.) des différents intervenants dans une session de communication.

Enfin, nous synthétisons dans la section 6.4 les différentes interprétations des résultats des différents tests de performances que nous avons réalisé.

6.1 Environnement de mesure

Étant donné que les objectifs de notre thèse visent principalement la sécurisation des opérations d'adaptation dans un environnement ouvert, nous avons effectué nos expérimentations dans des conditions réelles entre des machines éloignées interconnectées par l'Internet, mais à des fins de comparaison, nous avons opéré les mêmes mesures sur un réseau local (sans passer par l'Internet). La figure 6.1 montre ces deux réseaux de test que nous appellerons «réseau local» et «réseau distant». Voici les deux configurations que nous avons ainsi utilisées :

Réseau local Il est composé de quatre stations de travail hébergeant un Serveur, un PA, et deux Clients, avec les configurations décrites par le tableau 6.1, et reliées par un réseau Ethernet à 10 Mb.

Machine	Serveur	*Proxy* d'adaptation	Client 1	Client 2
Type	Laptop	PC	PC	PDA
Système d'exploitation	WinXP	Linux FC4	WinXP	WM6
Puissance CPU	1,7 Gh	3,5 Gh	2Gh	433 Mh
Mémoire	512 Mo	1Go	512Mo	64 Mo

TABLE 6.1 – Configuration matérielle du réseau local utilisé

Réseau distant Il est composé des mêmes quatre stations de travail reliées cette fois ci par l'Internet. Le Serveur et le *proxy* se situent à Paris 14^{eme} distant de 2 km, et les clients se situent à *Colombes* à environ 25 km du *proxy*.

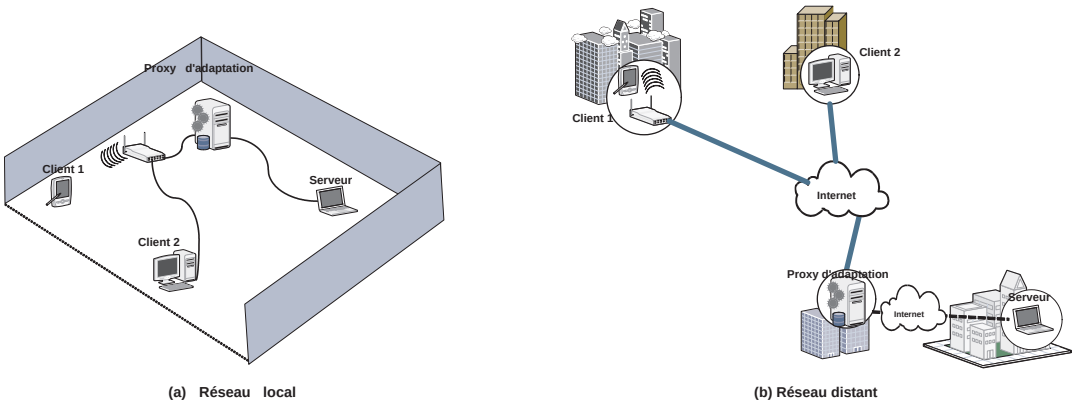


Figure 6.1 – Architectures de test utilisées : répartition des nœuds (a) sur le même site (b) dans un réseau distant

6.1.1 Composants élémentaires de l'analyse de performance

Afin de mesurer les coûts élémentaires, nous avons décomposé l'opération d'envoi d'un document adaptable en plusieurs parties qui correspondent aux principales étapes de cette opération. Pour ces mesures, nous avons programmé un *proxy* minimal qui peut réaliser des adaptations par remplacements (vidéo par image, image par texte, etc.), des suppressions d'objets-médias et des insertions de nouveaux objets à des endroits réservés.

L'envoi d'un document adaptable d'un émetteur vers un récepteur se décompose ainsi en 7 phases décrites dans le tableau 6.2 :

Étape 1	Préparation du document, qui se résume à la récupération des clés de sécurisation et la préparation de la politique d'adaptation
Étape 2	Authentification du document selon l'approche AMCA
Étape 3	Chiffrement des objets signés selon l'approche XSST
Étape 4	Transfert vers le *proxy*
Étape 5	Adaptation du document par le *proxy*
Étape 6	Transfert vers le client
Étape 7	Vérification de la signature au niveau du client avant lecture

Table 6.2 – Étapes réalisées lors de l'émission d'un document multimédia adaptable

6.1.2 Coûts de base de la plate-forme

Nous avons réalisé nos expérimentations en utilisant le JDK 1.4.2. Afin de fournir des éléments plus précis concernant cet environnement, nous avons mesuré les performances de base du réseau et de la JVM sur ces stations de travail (table 6.3). Les mesures de latence (aller simple) ont été effectuées en utilisant UDP et les mesures de débit en utilisant TCP.

Latence sur le réseau local	0.6 ms
Latence sur Internet	20 ms
Débit effectif sur le réseau local	1842 Ko/s
Débit effectif sur l'Internet	129 Ko/s
Appel local de méthode Java	0,1 ms
Appel à distance de méthode Java sur le réseau local	3,1 ms
Appel à distance de méthode Java sur Internet	44 ms

TABLE 6.3 – Coûts de base de l'environnement expérimental

Cette table montre les différences importantes de performances entre le réseau Internet et le réseau local. Elle montre également comment se comportent les mécanismes de base de Java (appel local, appel à distance) sur l'environnement que nous avons utilisé.

6.2 Scénarios de test

Les objectifs de notre étude de performances sont multiples et peuvent être regroupés dans les point suivants :

- montrer la faisabilité de l'adaptation sécurisée de bout en bout ;
- évaluer l'apport de la sécurisation *proxy* ou pas ..
- évaluer les coûts de l'intégration de SEMAFOR sur les délais de transmission.

L'évaluation des transmissions traitées par la plate-forme et le comportement des *proxies* d'adaptation, même s'ils peuvent être déterminés, impliquent un nombre trop élevé de paramètres. En effet, afin d'évaluer les performances du système, il est nécessaire de prendre en compte la puissance et la charge de toutes les machines jouant un rôle dans le système (émetteurs, récepteurs, PAs), les caractéristiques des transmissions (format, paramètres...), le type et la charge des réseaux sous-jacents.

Pour cette raison, nous avons choisi d'évaluer les performances de la plate-forme en ne nous basant pas sur un modèle déterministe. C'est pourquoi nous avons préféré étudier notre plate-forme selon des scénarios d'utilisation réalistes et concrets.

Pour cela, nous avons retenu deux types de scénarios d'utilisation de la plate-forme :

scénario 1 émission d'un flux composite impliquant un nombre élevé d'objets-médias d'un émetteur vers un récepteurs

scénario 2 émission au sein d'une application multipartite, autrement dit, un même flux est émis à plusieurs destinataires simultanément

Ces deux scénarios ont pour but de démontrer :

- le coût induit par chacune des étapes de l'émission en termes de délais
- le rapport surcoût/sécurité induit par l'utilisation de la plate-forme SEMAFOR
- l'influence de l'adaptation sur les opérations de sécurisation
- l'influence du type de récepteur sur les performances de SEMAFOR
- la possibilité de l'intégration de la plate-forme avec des applications non spécifiquement développées pour la sécurisation de contenu
- la sécurisation globale des données transmises d'une source unique vers plusieurs utilisateurs, en tenant compte des différences des contextes

6.2.1 Fichiers utilisés

Pour les besoins des tests nous avons choisi plusieurs documents multimédias de sortes qu'ils soient les plus représentatifs possible des différents cas éventuels dans la réalité. Le tableau 6.4 résume les propriétés des différents documents utilisés dans le cadre de nos tests :

Document	Format	Taille (Mo)	Objets-Médias	
			nombre total	adaptables
M1	SMIL	10,3	5	3
M2	SMIL	22,8	12	8
M3	SVG	3,6	35	23

Table 6.4 – Propriétés des fichiers de test utilisés

Chacun des scénarios décrit précédemment a été déroulé sur les trois fichiers M1,M2 et M3 afin de comparer les comportements de la plate-forme SEMAFOR dans différentes situations.

6.2.2 Phases de tests

Pour chacun des tests effectués nous avons réalisé une série de mesures de performance. Nous avons ensuite analysé ces mesures selon le découpage en étapes proposé précédemment par le tableau 6.2. Par ailleurs, afin que les résultats obtenus soit plus pertinents nous avons réalisé, pour chaque test, deux autres sessions témoins que nous utilisons comme éléments de comparaison. Ainsi, pour chaque fichier testé dans un scenario donné, trois sessions sont effectuées comme suit :

Session SEMAFOR : le test ici est réalisé en utilisant la plate-forme SEMAFOR de bout-en-bout ;

Session témoin 1 : cette session est réalisée sans SEMAFOR et consiste à émettre le document sans aucune sécurisation ;

Session témoin 2 : ici la sécurisation se fait en utilisant des méthodes de signature/chiffrement classiques avec un déchiffrement complet du flux avant adaptation

Nous avons ensuite analysé les résultats obtenus en étudiant d'abord les résultats de la session SEMAFOR et en effectuant ensuite une comparaison de ceux-ci avec ceux des deux sessions témoins de manière à évaluer l'apport de nos approches.

6.2.3 Éléments évalués

Les critères de mesures pris en considération lors de nos tests ont été les suivants :

durée de la transmission ou bien le temps nécessaire pour l'acheminement du document transmis à partir de l'émetteur jusqu'au récepteur final, dans un format adapté

pourcentage de charge processeur c'est le pourcentage du temps écoulé que tous les threads des applications de la plate-forme SEMAFOR passent pour exécuter des instructions. Le code exécuté pour gérer des interruptions dues au matériel et gérer des conditions de déroutement est inclus dans ce compte.

consommation mémoire représente le nombre de méga-octets dédiés aux composants de la plate-forme de communication, correspondent à la taille de la mémoire physique dédiée + la mémoire virtuelle pour laquelle de l'espace a été réservé dans le fichier d'échange du disque.

6.3 Évaluation des performances

Nous allons dans cette section présenter en détail les résultats obtenus lors des différents tests effectués et qui ont été décrits dans la section précédente. Nous décrivons pour chacun des deux scénarios les différents tests réalisés et les résultats obtenus pour chaque critère d'évaluation lors de chaque test. Une synthèse des résultats est ensuite présentée pour chaque test.

6.3.1 Scénario 1

Le test réalisé dans ce premier scénario consiste à envoyer les documents multimédias M1, M2 et M3 à partir d'un émetteur $\mathcal{S}$ vers un client $\mathcal{C}$ équipé d'un terminal léger. L'environnement utilisé pour ce test est le réseau local décrit dans le paragraphe 6.1. Les résultats obtenus sont décrit dans les paragraphes suivants classés par nom du document transmis et selon les critères évalués pour chacun.

6.3.1.1 Évaluation de la durée de la transmission

Le premier critère d'évaluation auquel nous nous sommes intéressés, et qui, sans doute, est l'une des motivations principales de nos travaux, est le surcoût en temps provoqué par l'utilisation de la plate-forme SEMAFOR. Nous avons donc mesuré le délai consommé pour chacune des étapes du transfert (en se fondant sur le découpage présenté dans la section 6.1.1).

La technique utilisée pour cette prise de mesure est l'intégration d'une fonctionnalité de log de début et de fin de chaque étape. Cela dit les mesures prises avec et sans l'activation de cette fonctionnalité ont montré que le temps consommé par l'activation de cette dernière était négligeable si on le compare au temps global du transfert des documents.

La figure 6.2 montre les résultats obtenus sur une moyenne de 10 tests effectués pour le transmission de chacun des trois documents M1, M2 et M3 en utilisant à chaque fois la plate-forme SEMAFOR ainsi que les deux plates-formes témoins décrites précédemment.

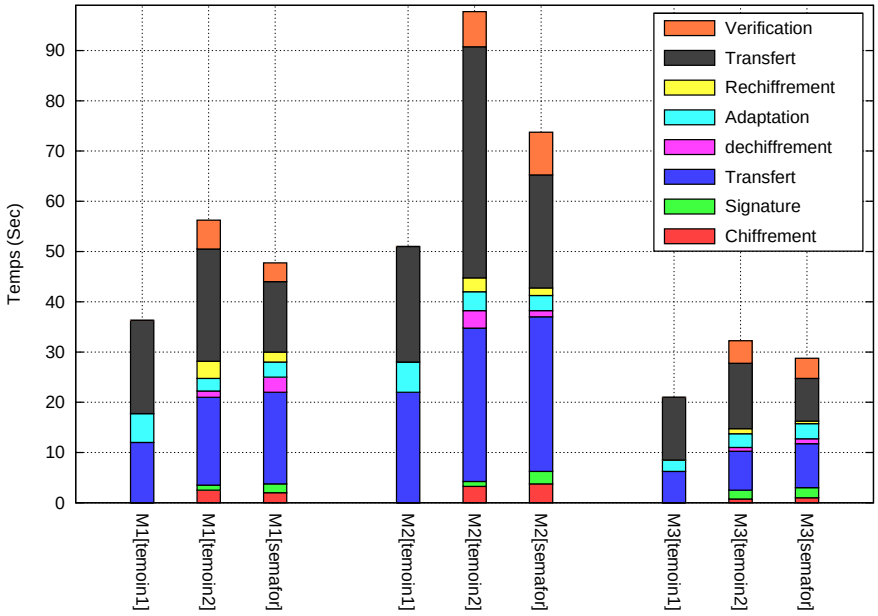


Figure 6.2 – Histogrammes des temps de transfert de bout-en-bout des documents multimédias

Analyse Les résultats exposés par la figure 6.2 montrent à la fois un surcoût induit par l'utilisation de SEMAFOR comparé à la première session témoin (sans sécurisation) mais surtout un gain important comparé à la seconde session témoin.

Les tests sur le fichier M1, par exemple, montrent que l'utilisation de SEMAFOR permet un gain de 43,4% du surcôut induit par une sécurisation classique. Ceux de M2 montrent un gain de 51,8% et ceux de M3 un gain de 32,1%.

En regardant de plus près, nous pouvons comparer les délais consommés pour chacune des étapes des transfert. Nous nous apercevons, par exemple, que le gain sur la phase de transfert entre le *proxy* et le client terminal est de 52% pour le fichier M1, de 25% pour le fichier M3 et atteint 91% de gain pour le fichier M2. Idem pour les étapes de déchiffrement/rechiffrement qui se voit a chaque fois réduites avec SEMFAFOR, en termes de temps consommé, par rapport à la sécurisation classique.

En revanche, d'autres phases se voient prolongées à cause de l'utilisation de SEMAFOR, telles que les étapes de chiffrement, de signature et de vérification. Ceci est dû aux techniques utilisées dans SEMAFOR afin de garantir l'adaptabilité des documents sécurisés. Néanmoins ces pertes sont aussi la raison des gains dans les autres phases. Nous pouvons ainsi affirmer que l'utilisation de SEMAFOR permet un gain considérable en termes de temps comparé à une sécurisation classique.

6.3.1.2 Évaluation de la charge CPU

Le second critère d'évaluation auquel nous nous sommes intéressés est le surcoût en consommation de charge CPU induit par les mécanismes de sécurisation que nous avons proposés. Pour cela nous avons mesuré la charge CPU dans les différentes étapes du processus d'émission au niveau de chaque intervenant, puis avons comparé avec les mêmes mesures prises en utilisant nos deux plates-formes témoins. Ces mesures ont été réalisées avec des outils de monitoring lancés sur chaque site.

Les figures 6.3, 6.4 et 6.5 montrent les résultats obtenus lors des transferts de (respectivement) M1, M2 et M3, a raison d'une test par fichier. Chacune de ces trois figures est composée de trois ensembles de courbes comme suit :

- la sous-figure (a) représente les mesures prises au niveau de l'émetteur du contenu
- la sous-figure (b) représente les mesures prises au niveau du *proxy* d'adaptation
- la sous-figure (c) représente les mesures prises au niveau du récepteur du contenu

Les courbes représentées dans ces figures sont les suivantes :

- en vert : la session témoin 1 (sans sécurisation)
- en rouge : la session témoin 2 (sécurisation avec méthodes classiques)
- en bleu : la session SEMAFOR

Nous avons aussi illustré sur ces figures le début et la fin de chaque étape du transfert (voir lignes en pointillé), et ce afin de localiser les phases les plus consommatrices en charge CPU.

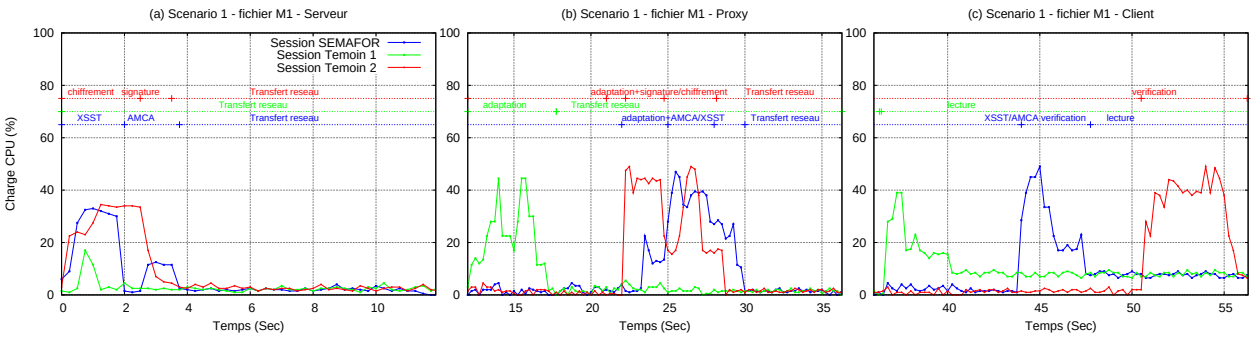


FIGURE 6.3 – Charge CPU utilisée lors de la transmission du document M1

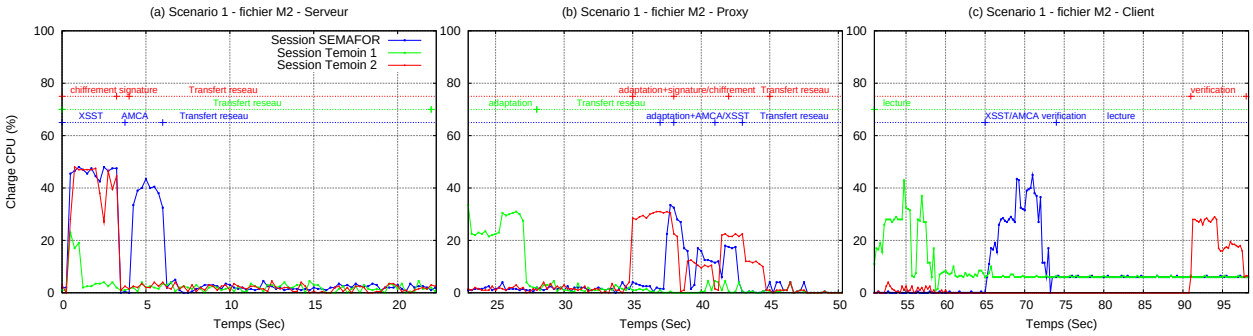


FIGURE 6.4 – Charge CPU utilisée lors de la transmission du document M2

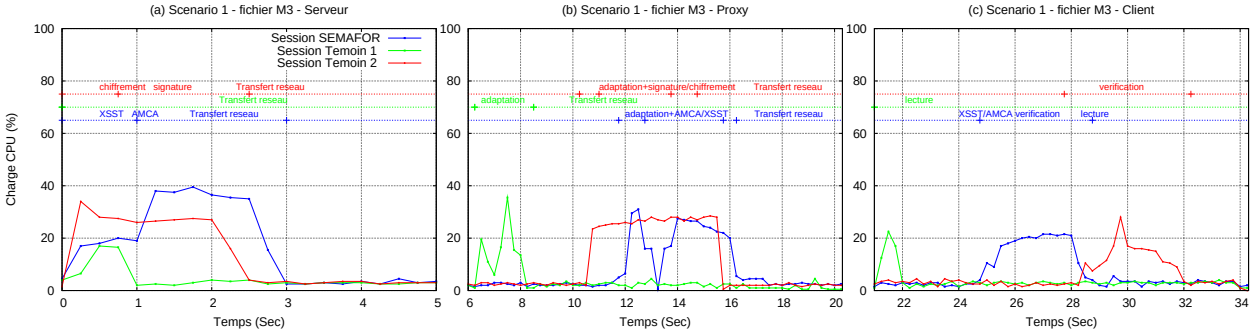


FIGURE 6.5 – Charge CPU utilisée lors de la transmission du document M3

Analyse Une première analyse rapide des résultats exposés par les figures 6.3, 6.4 et 6.5 serait de constater que, sur toutes les phases d'une transmission, et sur chacun des sites, SEMAFOR n'induit aucun pic de consommation en charge CPU considérablement supérieur à une sécurisation classique. En revanche la consommation est nettement supérieure comparée à une session sans sécurisation.

Au niveau du serveur, le point qui ressort clairement de ces figures est le découpage (assez net sur les 6.3-(a), 6.4-(a)) de la phase de signature en utilisant SEMAFOR en deux paraboles : la première correspondant à la phase de signature, la seconde correspondant à la phase de chiffrement.

Lors de ces étapes, la consommation en charge CPU reste relativement équivalente à une sécurisation classique. Néanmoins, la durée de ces étapes variant entre chaque session, ceci nous ramène à une moyenne de consommation de -33% pour le fichier M1, +39% pour le fichier M2 et +22% pour le fichier M3 comparé à une sécurisation classique dans la session Témoin 2. Ces variations sont corrélées avec le nombre d'objets adaptables dans le document initial et s'expliquent par la technique utilisée

pour la génration des arbred de hachage de Merkle utilisée.

Comparé à une session sans sécurité, pour ce test, SEMAFOR induit un surcoût moyen de +32% pour le transfert du fichier M1, +41% pour M2 et +38% pour M3.

Au niveau du *proxy* d'adaptation, les résultats montrent un gain moyen de 14% en consommation de charge CPU pour le transfert du document M1, de 66% pour le M2 et de 34% pour M3 comparé à une session avec sécurisation classsique. Ce gain important est obtenu grâce à la solution XSST qui permet de garder chiffré la partie de contenu non adaptable, là où la sécurisation classique nécessite un déchiffrement total du contenu avant toute adaptation, et de ce fait une consommation supérieure de charge CPU.

Cela dit, comparé à une session sans sécurité, SEMAFOR induit un surcoût de +34% pour le transfert du fichier M1, +23% pour M2 et +23% pour M3.

Au niveau du client, l'analyse des figures 6.3-(c), 6.4-(c) et 6.5-(c), montre que la phase de vérification de la signature reste assez proche de la charge CPU nécessaire lors de la session Témoin 2. Néanmoins, les durée de traitement étant différentes, nous mesurons un gain moyen de 29% pour la vérification de M1, un surcout de 9% pour M2 et un gain de 11% pour M3.

6.3.1.3 Évaluation de la consommation mémoire

Après la charge CPU, nous avons mesuré le surcoût en consommation mémoire induite par notre plate-forme. Pour cela nous avons effectué un monitoring relevant la charge mémoire dédiée à chaque processus des applications utilisées dans SEMAFOR, puis avons additionné les résultats relevés pour obtenir la charge totale dédié à SEMAFOR au niveau de chaque intervenant.

Comme pour les tests précédents, les mêmes mesures ont également été effectuées en utilisant les deux plates-formes témoins.

Les figures 6.6, 6.7 et 6.8 montrent les résultats obtenus lors des transferts de (respectivement) M1, M2 et M3, a raison d'un test par fichier.

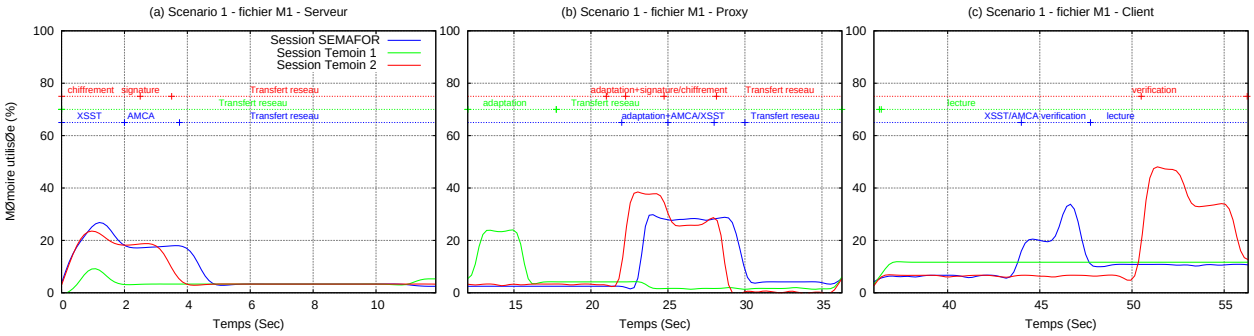


FIGURE 6.6 – Consommation mémoire lors de la transmission du document M1

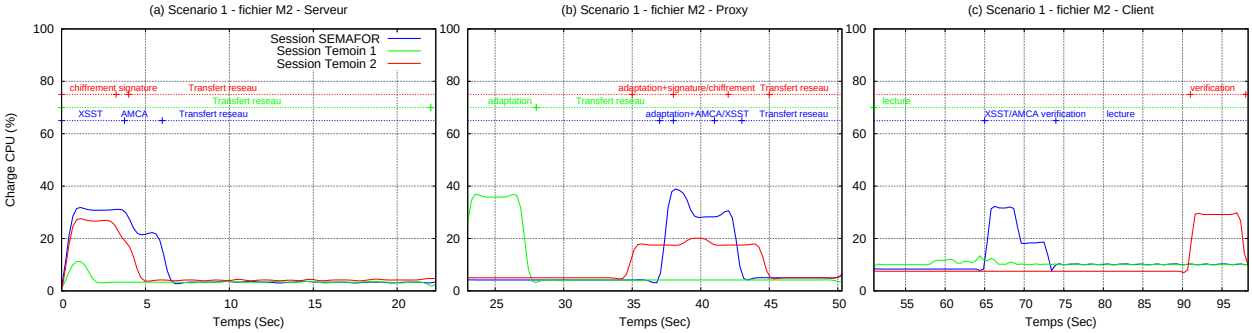


FIGURE 6.7 – Consommation mémoire lors de la transmission du document M2

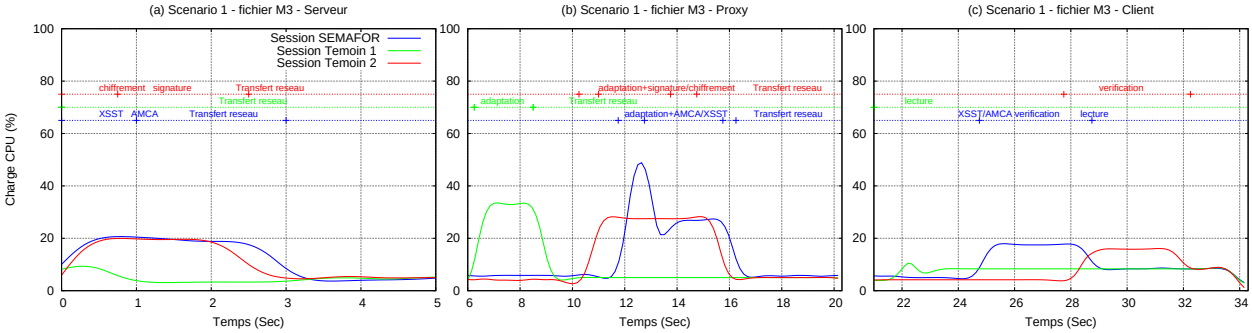


FIGURE 6.8 – Consommation mémoire lors de la transmission du document M3

Analyse En essayant d'analyser le comportement de la partie serveur dans SEMAFOR, autrement dit, en regardant les figures 6.6-(a), 6.7-(a) et 6.8-(a) nous apercevons que les besoins en mémoire se confondent avec la session Témoin 2 dans la première et la troisième figure mais sont légèrement supérieurs dans la seconde. La cause de cette différence est sans doute le nombre d'objets-médias adaptables dans le document M2, qui est relativement important par rapport au nombre total

d'objets-médias. La moyenne donne donc un surcoût de 3% pour M1, un surcoût de 8% pour M2 et de 3% pour M3.

Pour le *proxy* d'adaptation, le gain en termes de charge CPU consommée est très clair sur les figures 6.6-(b), 6.7-(b) et 6.8-(b), en effet, comparé à une sécurisation classique, SEMAFOR permet un gain moyen de 19% lors de l'envoi de M1, un gain de 27% pour M2 et de 12% pour M3.

L'étape de vérification au niveau du Client dans SEMAFOR est aussi performante que les étapes précédentes, et permet un gain de 39% pour le docuement M1, un gain de 12% pour le document M2 et un gain de 8% pour le docuement M3.

6.3.2 Scénario 2

Le test réalisé dans le second scénario consiste à envoyer le document multimédia **M1** à partir de l'émetteur $\mathcal{S}$ vers les deux clients $\mathcal{C}$ 1 et $\mathcal{C}$ 2 dont les caractéristiques sont décrites dans le tableau 6.1.

L'environnement utilisé pour ce test est le réseau Internet décrit dans le paragraphe 6.1. Les résultats obtenus sont décrits dans les paragraphes suivants classés selon les critères évalués.

6.3.2.1 Évaluation de la durée de la transmission

De la même manière que lors de scénario 1, nous avons calculé la durée consommée pour chacune des étapes de la transmission du fichier M1 de $\mathcal{S}$ vers $\mathcal{C}$ 1 et $\mathcal{C}$ 2. La figure 6.9 présente les résultats obtenus.

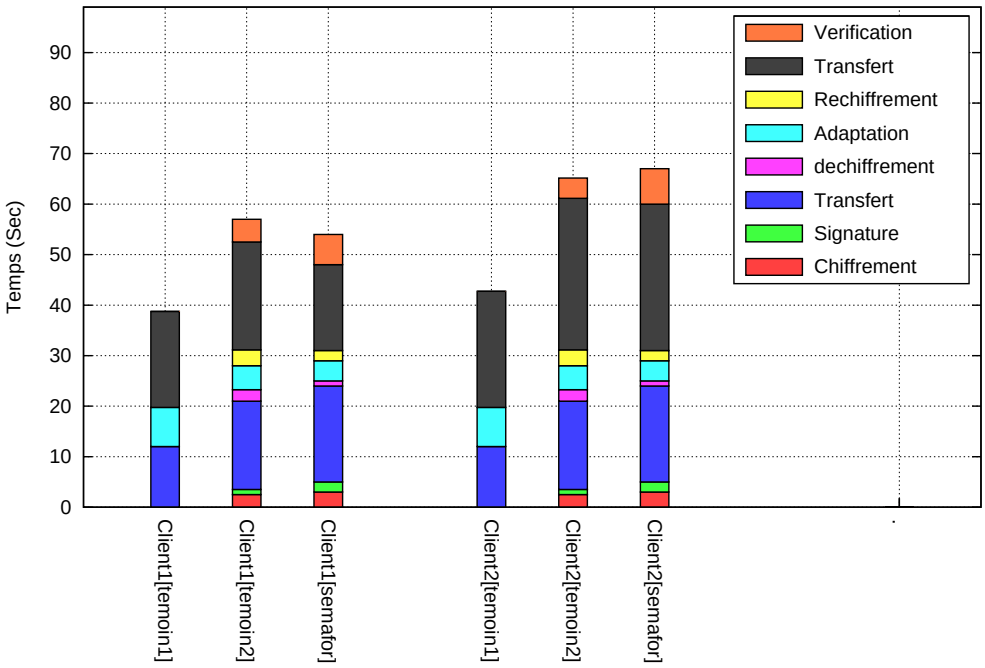


FIGURE 6.9 – Histogrammes des temps de transfert avec deux clients $\mathcal{C}$ 1 et $\mathcal{C}$ 2

Analyse Comme lors du scénario 1, les résultats obtenus pour l'envois du document M1 vers le Client $\mathcal{C}$ 1 et exposés par la figure 6.9 montrent que l'utilisation de SEMAFOR induit un gain en termes de durée de transmission comparé à la

seconde session témoin et un surcoût comparé à la première session témoin (sans sécurisation).

En effet, les tests sur le fichier M1 dans cette session multipartite montrent que l'utilisation de SEMAFOR, même si elle induit un surcôut de 12,8% de temps de transmission comparé à la session témoin 1, permet un gain de 24,6% du surcôut induit par une sécurisation classique dans la session témoin 2.

Cela dit, la figure 6.9 montre aussi que lors de l'envoi de M1 vers le client $\mathcal{C}$ 2, l'utilisation de SEMAFOR induit à chaque fois un surcoût comparé avec les deux sessions témoins. Ainsi, si la session témoin 2 induit un surcoût de 13,5% par rapport à une session sans sécurisation, l'utilisation de SEMAFOR induit 14,8% de surcoût.

Par ailleurs, en analysant les étapes de transmission séparément, nous pouvons distinguer clairement que l'utilisation de SEMAFOR reste plus performante pour chacune des étapes sauf celle de la vérification de la signature, où le surcoût est plus important que lors de la session Témoin 2.

Cette latence dans l'étape de vérification sur le terminal léger s'explique par les solutions d'implémentation que nous avons utilisé sur ce type de terminaux pour la mise en œuvre du générateur d'arbre de hachage Merkle. En effet, suite à l'absence de librairie dédiée pour le calcul des AHM nous avons fait le portage de notre implémentation d'un environnement J2SE [1] vers l'environnement J2ME [2] [93] ce qui n'est pas très optimisé en soit. Les librairies utilisées pour la sécurisation classique étant dédiées pour ce type de terminaux, la vérification de signature était donc plus optimisée lors de cette session.

1. Java 2 Standard Edition, le framework Java destiné aux applications pour poste de travail
2. Java 2 Micro Edition, le framework Java spécialisé dans les applications mobiles

6.3.2.2 Évaluation de la charge CPU

Comme pour le scénario 1, le second critère d'évaluation auquel nous nous sommes intéressés est le surcoût en consommation de charge CPU induit par les mécanismes de sécurisation que nous avons proposés. Pour cela nous avons mesuré la charge CPU dans les différentes étapes du processus d'émission au niveau de chaque intervenant, puis avons comparé avec les mêmes mesures prises en utilisant nos deux plates-formes témoins. Ces mesures ont été réalisées avec les mêmes outils de monitoring utilisés précédemment.

Les figures 6.10-(a),...,(d) montrent les résultats obtenus sur chaque site lors du transfert du document M1 vers les deux clients $\mathcal{C}$ 1 et $\mathcal{C}$ 2.

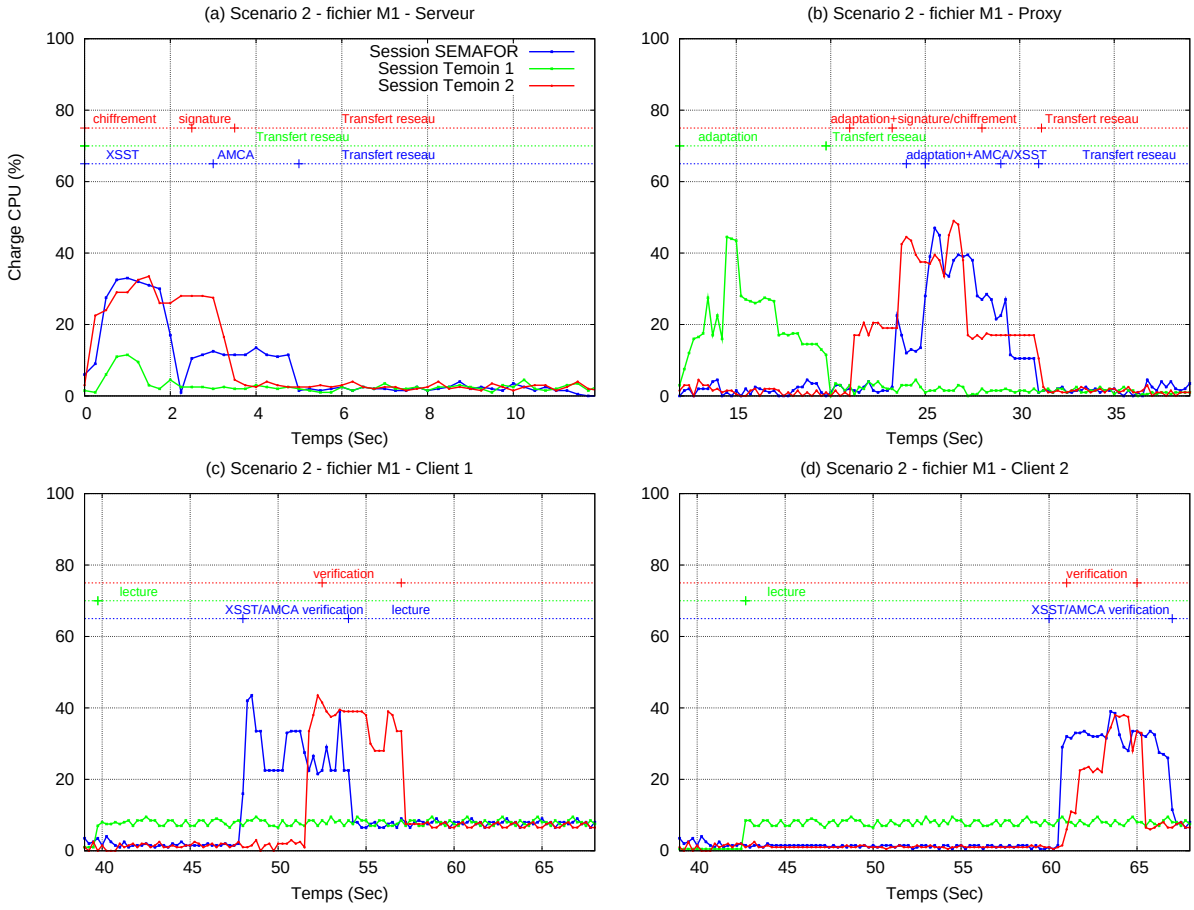


FIGURE 6.10 – Consommation en charge CPU lors de la transmission du document M1

Analyse La figure 6.10-(a) montre clairement que, malgré une durée de traitement supérieure dans SEMAFOR, pour la phase de sécurisation de M1, la charge CPU utilisée reste inférieure en comparaison avec la session Témoin 1. La moyenne calculée donne un gain de 26% de charge CPU consommée. Le surcoût de la sécurisation dans SEMAFOR étant de 43% et celle de la session Témoin 2 de 54% (en comparant avec la session Témoin 1).

La figure 6.10-(b) montre qu'au niveau du *proxy* d'adaptation, la charge CPU nécessaire dans SEMAFOR reste relativement égale à celle de la session Témoin 2. Mais la durée de traitement est cette fois ci inférieure dans SEMAFOR ce qui donne en moyenne un gain de 47%. Le surcoût de l'adaptation/sécurisation dans SEMAFOR étant de 19% et celle de la session Témoin 2 de 27%.

La figure 6.10-(c) montre que la vérification de l'authenticité du document M1 par le client $\mathcal{C}$ 1 induit un gain de 16% de charge CPU consommée. Le surcoût de cette phase dans SEMAFOR étant de 31% et celle de la session Témoin 2 de 36%.

Concernant le client $\mathcal{C}$ 2, la vérification dans SEMAFOR, comme le montre la figure 6.10-(d), induit un surcoût de 21% comparé à la session Témoin 2. Le surcoût de cette phase dans SEMAFOR étant de 37% et celle de la session Témoin 2 de 30%.

6.3.2.3 Évaluation de la consommation mémoire

Après la charge CPU, nous avons aussi mesuré le surcoût en consommation mémoire induite par la plate-forme SEMAFOR pour la session multidestinataire. Pour cela nous avons effectué un monitoring relevant la charge mémoire dédiée à chaque processus des applications utilisées dans SEMAFOR, puis avons additionné les résultats relevés pour obtenir la charge totale dédiée à SEMAFOR au niveau de chaque intervenant.

Comme pour les tests précédents, les mêmes mesures ont également été effectuées en utilisant les deux plates-formes témoins.

Les figures 6.11-(a),...,(d) montrent les résultats obtenus lors du transfert de M1 vers les deux clients $\mathcal{C}$ 1 et $\mathcal{C}$ 2.

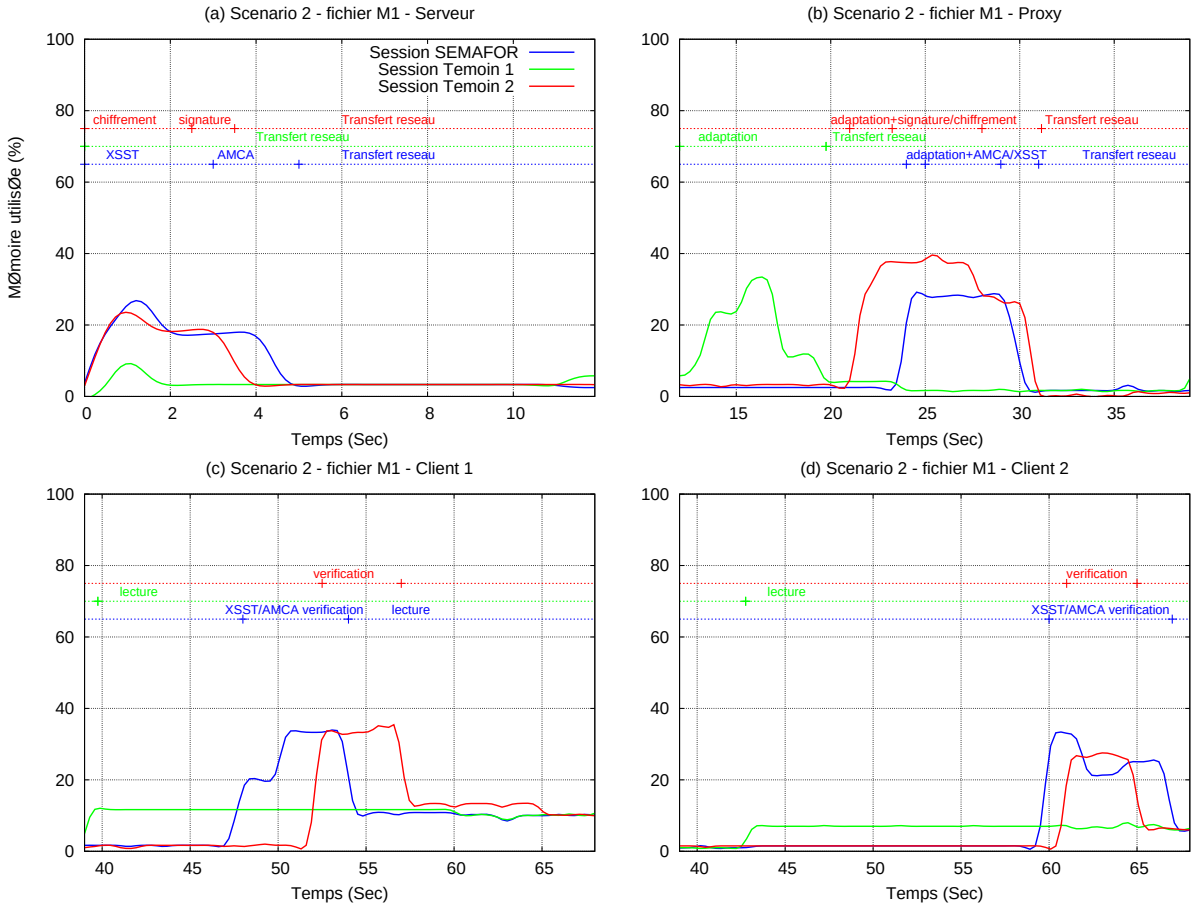


FIGURE 6.11 – Consommation mémoire lors de la transmission du document M1

Analyse L'analyse du comportement des intervenants dans ce scénario révèle plusieurs résultats intéressants :

Au niveau du Serveur, d'abord, la comparaison entre SEMAFOR et une session avec sécurisation classique ne révèle aucun surcoût de consommation mémoire. Ceci démontre que les mécanismes utilisés pour la a gestion des session multi-clés (génération et transmission) lors d'une session multidestinataire n'induisent pas de surcoût significatif.

Au niveau du *proxy* , même si le surcoût par rapport à la session Témoin 1 est de 29% en moyenne, nous mesurons un gain de 18% de consommation mémoire en comparaison avec la session Témoin 2. Ceci démontre l'intérêt de l'utilisation de SEMAFOR pour la sécurisation au niveau du *proxy* , et s'explique par le déchiffrement partiel de M1 lors de l'operation d'adaptation via SEMAFOR alors que ce fichier est entiérement déchiffré, adapté puis réchiffré lors de la session Témoin 2.

Au niveau du client $\mathcal{C}$ 1, après un premier palier (correspondant à la phase déchiffrement de M1) où la consommation mémoire est plus faible dans SEMAFOR, celle ci reste très comparable entre la session SEMAFOR et la session témoin 2. Néanmoins, la moyenne montre un gain de 12% dans SEMAFOR.

Au niveau du client $\mathcal{C}$ 2, le premier palier dans SEMAFOR (correspondant à la phase déchiffrement de M1) est plus consommateur en mémoire que la session Témoin 2. Mais la suite du traitement situe les deux session au même niveau de consommation. La différence dans la première phase s'explique par l'implémentation du composant de vérification faite dans SEMAFOR pour les terminaux légers, qui est comme nous le constatons, n'est pas très optimisée à case de l'absence de librairies standards pour le déchiffrement sur le terminal utilisé. Cette partie reste donc à améliorer dans un futur proche.

6.4 Synthèse

Ce dernier chapitre, avant de conclure ce mémoire, avait pour but de démontrer l'intérêt du compromis entre «sécurité de bout-en-bout» et «surcoût induit par cette sécurisation». Nous avons présenté deux scénarios de test sur la plate-forme SEMAFOR dans des conditions réelles d'utilisation.

Le premier scénario de test nous a permit de valider le fonctionnement de SEMAFOR dans une session de communication basique. Nous avons ainsi pu montrer que le (sur)coût de la sécurisation SEMAFOR était en moyenne de +25% en termes de durées de transmission, et de +30% en moyenne de charge CPU supplémentaire, et de presque autant de mémoire consommée. Ces chiffres étant fortement corrélés au nombre d'objets-médias à adapter par le *proxy* d'adaptation. Mais ce qui sorti également de ce test, c'est que, en plus de satisfaire le cahier des charges en termes de critères de sécurité, SEMAFOR permet un gain moyen de 40% de durée de transmission comparé à une session utilisant des techniques de sécurisation classiques, idem pour la charge CPU où le gain et de 33% en moyenne, et pour la consomation mémoire avec un gain moyen de 23%.

Le second scénario de test nous a permit de valider le fonctionnement de SEMAFOR dans une session de communication multipartite avec un émetteur, plusieurs récepteurs et un *proxy* intermédiaire dans un environnement ouvert (L'internet). Nous avons ainsi pu montrer que le (sur)coût lié à l'aspect multipartite de la session SEMAFOR était en moyenne de +5,1% en termes de durées de transmission, et de +3,7% en moyenne de charge CPU supplémentaire, et de 6,3% de mémoire dédiée suplémentaire.

Les analyses de ce test montrent aussi que SEMAFOR permet un gain moyen de 16% de durée de transmission comparé à une session utilisant des techniques de sécurisation classiques, idem pour la charge CPU où le gain et de 24% en moyenne, et pour la consomation mémoire avec un gain moyen de 12%.

Toutefois, d'autres tests plus approfondis ont été réalisés sur la plate-forme SEMAFOR, mais nous avons préféré insister dans ce chapitres sur les critères les plus importants qui sont la durée de transmission, la charge CPU et la consommation mémoire.

Troisième partie

Conclusions et perspectives

Conclusions et perspectives

Les travaux que nous avons réalisés au cours de cette thèse se placent dans le domaine de la sécurisation du partage de documents multimédias adaptables, plus précisément la sécurisation des documents lors des opérations d'adaptation effectuées par des prestataires intermédiaires.

L'aspect recherche de nos travaux s'est axé autour de l'étude des problématiques d'authentification et du chiffrement de contenus multimédias adaptables. Nous nous sommes intéressés à faciliter la signature de documents composites, la résistance de la signature numérique aux opérations d'adaptation et le chiffrement des flux lors des échanges multipartites. Nous avons notamment été plongés au cœur d'un projet de recherche de chiffrement de flux vidéo avec un grand groupe de distribution de biens culturels. Nous avons pu présenter un prototype en comité de direction, mais ce projet n'a hélas pas pu continuer pour des questions économiques.

Toutes les études bibliographiques que nous avons effectuées, toutes les réflexions et les échanges que nous avons pu avoir, tant avec les membres de la communauté JXTA qu'avec les membre du W3C, lors des réunions de projets ou au cours des conférences auxquelles nous avons pu participer ont été très enrichissantes à la fois sur le plan théorique que sur le plan applicatif.

L'aspect applicatif quant à lui s'est axé autour de notre collaboration avec la communauté JXTA et la réalisation de la plate-forme SEMAFOR. Nous nous sommes intéressés à la conception et l'implémentation d'une solution pour la fourniture de services d'adaptation totalement sécurisée.

Pendant notre thèse, nous avons également participé au démarrage de plusieurs projets et groupes de travail, ce qui nous a permis d'élargir le champ d'application de nos travaux. Tout d'abord, le projet TIRAMISU dont l'objectif est d'étudier ce que pourrait être la télévision du futur, mêlant à la fois la diffusion de contenus multimédias, sécurité, partage de documents, etc. Ensuite, le groupe de travail CESAME qui

s'intéresse aux fondements des «systèmes diffus», fondements aptes à soutenir d'un point de vue conceptuel et implémentationnel, des éclairages particuliers à finalité, par exemple, d'adaptation ou d'exploitation des mondes physique et numérique et qui réuni compétences (IHM, ergonomie , génie logiciel , système , etc.) s'y trouvent réunies, stimulées par des cas d'usage émanant ou inspirés de partenaires industriels.

Notre démarche

Études préliminaires

Afin de comprendre comment se déroulaient les services d'adaptation multimédia, nous avons été amenés à étudier en détail plusieurs formats de documents multimédias adaptables ainsi que les différentes techniques d'adaptation. Plusieurs états de l'art et comparatifs ont été dégagés dans cette phase de travail. Ces études nous ont permis de délimiter notre champs d'intervention dans le cadre de cette thèse. Nous avons ainsi orienter nos travaux de recherche vers des type de flux multimédias composites adaptables par des nœuds intermédiaires présents dans le réseau.

Des intermédiaires pour une communication adaptative ... mais

Dans un second temps, nous avons étudié les différentes techniques d'adaptation de contenus multimédias. Nous avons ainsi pu décrire les services proposés par des intermédiaires d'adaptation ; les comparer avec les autres solutions d'adaptation ; analyser les avantages de cette topologie, etc. ... Mais, malheureusement, la chose qui a le plus retenu notre attention était avant tout l'impact en termes de sécurité de l'utilisation de ces *proxies* d'adaptation. En effet, les solution classiques d'adaptation, ne permettent aucune sécurité des données échangées (perte d'authentification, pas de confidentialité des données, etc.).

Faire confiance mais se protéger avant

Deux solutions s'offraient à nous pour l'utilisation de *proxies* d'adaptation, faire confiance aux *proxies* d'adaptation où trouver de nouvelles solutions garantissant sécurité et adaptabilité des documents. Nous avons bien évidement choisi la seconde,

même si un minimum de confiance reste obligatoire pour l'utilisation de *proxies* d'adaptation, ne serait ce que pour l'intervention dans la session de communication.

Ainsi, d'abord inspirés par des systèmes de sécurisation de vidéo adaptable, permettant un découpage des documents en paquets ou en couches, nous avons émis l'hypothèse qu'un document multimédia peut se traduire en un ensemble d'objets-médias. Au travers d'une études des formats multimédias, nous avons constaté des similitude entre plusieurs de ces formats et des structures de données arborescentes. En effet, nous avons pu modéliser différents formats multimédias sous forme de modèle en arbre. Nous nous sommes ainsi intéressés à des applications de sécurité ciblant ce type de structures. C'est là que nous avons identifié les arbres de Merkle et la technique de hachage qu'ils proposent.

Ensuite, inspirés par les systèmes de DRM et de gestion des propriétés intellectuelles dans les réseaux de partage de musique et de vidéo, nous avons commencer par émettre l'hypothèse qu'un contenu multimédia adaptable ne doit être accédé que par les titulaires d'une licence décrivant leur droits d'accès. Et vu que dans notre architecture deux type de récepteurs sont possibles (*proxies* et client finaux) nous avons proposé un modèle de données utilisant deux types de licences, une licence de lecture et surtout une licence d'adaptation.

Rester toujours maître de ses choix

Un objectif majeur en termes de sécurité dans les réseaux multimédias est le critère de confidentialité des données échangées. Or, l'utilisation de *proxies* d'adaptation altérait les techniques utilisées pour satisfaire cet objectif. Et imposait aux émetteurs de divulguer leurs contenus aux *proxies* pour les opérations d'adaptation. De plus l'aspects multipartite des sessions de communications multimédias faisait apparaître de nouvelles failles de sécurité telle que les attaques par collusion entre plusieurs destinataires recevant le même contenu où même entre plusieurs *proxies* par lesquels transite un même contenu.

Nous nous sommes alors inspirés des plates-formes utilisant des groupes de communication, afin de proposer notre schéma de chiffrement/rechiffrement unidirectionnel, garantissant une confidentialité de bout-en-bout des parties d'un document émis choisies par l'émetteur d'un contenu adaptable.

Une solution clé en main

Nous avons, pour chacune des différentes solutions proposées, démontré séparément l'intérêt et les avantages pour les critères de sécurité que nous nous sommes fixés. Néanmoins, rien de mieux que de les utiliser dans des cas réels et surtout les combiner pour une sécurisation maximale.

Nous nous sommes donc penché sur la réalisation de solution complète implémentant toutes nos propositions théoriques, et permettant de valider la faisabilité et l'intérêt de ces propositions.

Nos apports

Dans notre thèse, nous espérons tout d'abord avoir apporté au lecteur une vue d'ensemble de la problématique complexe à laquelle nous nous sommes attaqués. Ce que nous estimons être l'un des principaux apports est d'avoir mis la lumière sur un problème très pointu, celui du partage d'informations sécurisées, et puis d'avoir proposé des solutions qui démontrent la possibilité d'atteindre ce niveau de sécurité.

Cette solution, nous l'avons modélisée en l'articulant autour de la notion de rôle. Nous avons en particulier défini les trois rôles génériques : Client, *Proxy* et Serveur.

Ces rôles permettent à leur tour de construire des classes de niveau de sécurité et de droit d'accès aux ressources. Nous avons de plus montré que notre modélisation pouvait se généraliser et prendre en compte les autres applications que nous avons rencontrées, comme l'authentification et le chiffrement des contenus.

Nous avons également pu valider notre approche en implémentant un prototype d'application pour la télémédecine avec le projet SEMAFOR. D'un point de vue applicatif, nous nous sommes plongés au cœur de plusieurs plates-formes d'adaptation afin d'en comprendre les rouages et notamment comment elles unifiaient l'accès aux ressources multimédias. Nous avons pu en dégager une architecture abstraite, dans laquelle les *proxies* ont pleinement leur place pour interagir dans le cadre des services d'adaptation et de sécurité.

Nous avons, de ce fait, exposé nos réflexions pour concevoir un système de sécurisation de bout-en-bout. Pour cela, nous avons cherché à montrer la palette des possibilités qui s'était offerte à nous, tant en ce qui concerne l'authentification des

données que pour leur chiffrement. Nous n'avons pas proposé des solutions applicables à tout type de flux multimédia ni à tout type d'adaptation, mais nous avons envisagé plusieurs façons de rendre conforme nos propositions à ces derniers.

Dans le cas particulier du problème de l'authentification, les résultats de nos expérimentations ont montré que même si notre modélisation par des Arbres de Merkle dans AMCA souffre d'une complexité relativement importante, elle était utilisable conjointement avec des méthodes de hachage pour obtenir une signature résistante aux opérations d'adaptation.

S'agissant de la confidentialité des données émises, nous avons proposé XSST, une technique garantissant un chiffrement de bout-en-bout des parties jugées confidentielles par l'émetteur. Cependant, s'agissant des parties adaptables, XSST ne peut garantir un chiffrement de bout-en-bout, et ce à cause de contraintes liées aux techniques d'adaptation de contenus. XSST garantit ainsi un niveau élevé de confidentialité des données pour les documents multimédias adaptables, et permet de ne déchiffrer que les objets-médias adaptables lors des opérations d'adaptation. De plus, la solution XSST fournit des solutions et des parades contre les différents types d'attaques par collusion.

Perspectives

Qu'il s'agisse de travaux théoriques ou d'applications pratiques, de nombreuses perspectives sont possibles aux travaux décrits dans ce mémoire.

Un premier challenge est la réduction de la complexité liée aux étapes de vérification et d'améliorer les temps de vérification sur les clients légers. En effet bien que la modélisation que nous avons effectuée reposant sur le recalcul complet de l'AHM d'un document multimédia nous a permis de valider nos propositions, nous pensons qu'il est possible d'utiliser d'autres structures de données pour la transmission permettant d'alléger ce calcul.

Par ailleurs, nous pensons qu'il pourrait être aussi intéressant de se pencher sur les méthodes qui permettent de générer les politiques d'accès aux documents adaptables. De plus, notre modèle de contrôle d'accès, même s'il s'inspire des modèles de DRM, n'en est pas un à part entière. Cette piste de recherche est aussi explorable afin de proposer des architecture de DRM proposant des licences pour l'adaptation de contenu.

Pour la partie chiffrement de bout-en-bout des données, il nous semble important que des travaux soient menés pour permettre une plus grande convergence entre les formats multimédias afin d'aboutir à une solution générique de confidentialité bout-en-bout des données.

Fonctionnellement parlant, nous comptons intégrer dans notre solution de contrôle d'accès, dans SEMAFOR, le nouveau protocole OpenID [3][79] un système d'authentification décentralisé qui permet l'authentification unique, ainsi que le partage d'attributs, afin de pouvoir s'interfacer avec d'autres plates-formes de diffusion de contenus.

À plus long terme, nous souhaitons réutiliser les différentes contributions de cette thèse dans le cadre d'une plate-forme industrialisée. Plusieurs contacts ont déjà été pris afin de donner suite au projet SEMAFOR.

3. Permet à un utilisateur de s'authentifier auprès de plusieurs sites en utilisant un unique identifiant OpenID. `http://openid.net/`

Quatrième partie

Annexes

Annexe A

Introduction à la cryptographie

Nous présentons ici quelques notions de cryptographie. Nous détaillons les mécanismes régissant le chiffrement, l'authentification et la signature numérique, et présentons les quelques vulnérabilités et attaques possibles sur les algorithmes de cryptographie. Cette introduction ne se prétend pas exhaustive, mais vise simplement à fournir quelques précisions utiles dans le cadre de cette thèse.

Plus de détails sont disponibles dans de nombreux ouvrages de référence comme [85], [54] ou encore [66].

A.1 Les outils cryptographiques fondamentaux

Dans le multimédia comme dans les autres technologies numériques, la cryptographie demeure la technique indispensable pour, d'une part, protéger la confidentialité des documents transmis sur les réseaux ou stockées dans les serveurs de données et pour, d'autre part, assurer l'intégrité d'un document ou pour prouver l'authenticité d'une opération ou d'une transaction. Elle applique des concepts mathématiques et met en place des paradigmes informatiques afin de résister aux attaques potentielles de pirates ou de prouver, de manière quasi sûre, qu'une procédure est incorruptible. Les deux concepts les plus importants sur lesquels repose la cryptographie sont : les algorithmes de chiffrement et les fonctions de hachage.

A.1.1 Algorithmes de chiffrement

Un algorithme de chiffrement est un formule mathématique dont les opérations, paramétrées par une clé de chiffrement, conduisent au chiffrement et au déchiffrement des données. En principe, l'algorithme est normalisé et connu de tous, le secret ne réside que dans la clé utilisée. L'algorithme de chiffrement peut être utilisé de deux façons : en mode signature (*cf.* section A.1.3) et en mode chiffrement (*cf.* section A.1.3). Dans le premier cas, il permet de garantir l'intégrité des données et, dans le second cas, la confidentialité des éléments transmis. Il existe deux familles d'algorithmes cryptographiques :

La cryptographie symétrique (ou *à clé secrète*) avec des algorithmes comme DES, 3DES, AES, n'emploie qu'une unique clé pour chiffrer et déchiffrer un document. Il est donc nécessaire de distribuer cette même clé aux deux intervenants de la communication. Si un serveur s'adresse séparément à plusieurs clients distincts, il aura besoin d'autant de clés distinctes. Cette famille d'algorithmes sert à chiffrer en temps réel ou en différé, des documents, car ces algorithmes sont puissants et nécessitent assez peu de ressources. La taille des clés est faible, par exemple 128 bits.

La cryptographie asymétrique (ou *à clé publique*) emploie deux clés différentes : si l'on chiffre avec une clé, il faut déchiffrer avec l'autre. Il existe ainsi deux possibilités d'applications qui servent des objectifs distincts. Dans le cas du système RSA, le système asymétrique le plus utilisé, la taille des clés les plus courantes est 1024 bits. Notons que les clés sont beaucoup plus longues que dans le cas du chiffrement symétrique. Ceci est dû à la nature arithmétique du système RSA dont la sécurité repose sur la difficulté de factoriser des grands entiers. Or les progrès mathématiques et informatiques dans le domaine de la factorisation imposent maintenant cette taille de clé. Ces algorithmes sont beaucoup plus lents (dans un ordre de 10 à 100) que les algorithmes symétriques. On les réserve donc au chiffrement et déchiffrement des messages courts.

Un message court peut justement être une clé d'un algorithme symétrique. Cryptographie asymétrique et cryptographie symétrique sont donc exploitées de manière complémentaire et successive dans les protocoles cryptographiques pour authentifier l'émetteur, le récepteur, énoncer la non-répudiation des interlocuteurs, et déployer les systèmes de secrets qui vont permettre de communiquer de manière sécurisée.

A.1.2 Les fonctions de hachage

Une fonction de hachage est utilisé entre autres pour la signature électronique, et rend également possible des mécanismes d'authentification par mot de passe sans stockage de ce dernier. Elle permet de calculer un condensé (*digest*) de taille réduite et fixe (160 bits, par exemple) d'un document volumineux, écrit au niveau atomique à l'aide de 0 et de 1. Ces fonctions ne sont absolument pas continues, si bien que si le document est un tant soit peu modifié, le condensé sera radicalement transformé. C'est en particulier cette propriété que l'on utilise pour valider l'intégrité d'un transfert de documents.

Pour présenter des garantie de sécurité suffisantes, une fonction de hachage doit être résistante aux collisions, c'est-à-dire que deux messages distincts doivent avoir très peu de chances de produire la même condensé. De par sa nature, tout algorithme de hachage possède des collisions mais on considère le hachage comme cryptographique si les conditions suivantes sont remplies :

- il est très difficile de trouver le contenu du message à partir du condensé (attaque sur la première préimage)
- à partir d'un message donné et de son condensé, il est très difficile de générer un autre message qui donne le même condensé (attaque sur la seconde préimage)
- il est très difficile de trouver deux messages aléatoires qui donnent le même condensé (résistance aux collisions)

Par très difficile, on entend «techniquement impossible» que ce soit au niveau algorithmique ou matériel. Des exemples de fonctions de hachage nous citons le MD5, le SHA-1, le RIPEMD.

A.1.3 Applications classique des outils cryptographiques

Le chiffrement des communications et des fichiers Le chiffrement est un procédé grâce auquel on peut rendre la compréhension d'une information intelligible (document, message, etc.) impossible à toute personne qui n'a pas la clé de chiffrement. Pour que deux personnes échangent des documents de manière sécurisée, il suffit que l'émetteur chiffre le message intelligible grâce à un algorithme de chiffrement. Le récepteur va ensuite déchiffrer le message avec ce même algorithme réversible et la clé lui correspondant.

La signature numérique La cryptographie à clé publique rend possible l'utilisation des signatures numériques. Pour signer un document, on utilise une fonction de hachage (*cf.* section A.1.2) qui produit un condensé (*digest*) du message. Le condensé obtenu est chiffré à l'aide de la clé privée de l'expéditeur. Le résultat, qui constitue la signature numérique, est annexé au document. Le destinataire du document peut ensuite s'assurer l'origine du document et l'intégrité de l'information.

A.1.4 Les infrastructures de confiance

La difficulté de la cryptographie asymétrique provient de l'authenticité non établie de la clé publique d'un interlocuteur. N'importe qui peut, à présent, engendrer un couple de clés privée-publique en récupérant un algorithme sur Internet. Aussi, lorsqu'une personne publie son nom et sa clé publique associée, un pirate peut se glisser sous cet affichage et usurper la clé, en proposant sa propre clé publique. De cette façon, le pirate pourra décoder avec sa propre clé privée les messages à destination de l'interlocuteur véridique, quitte à restituer les messages chiffrés à l'interlocuteur de départ avec la clé publique originelle. C'est ce qu'on appelle une attaque par l'homme au milieu.

Les autorités de confiance Pour éviter cette méprise, la solution consiste à s'en remettre à un tiers, en faisant signer la clé publique par une autorité digne de confiance, qui va ainsi garantir que la clé publique appartient bien au bon interlocuteur. On va donc signer numériquement le couple composé du nom du propriétaire de la clé publique et de la clé publique. La clé privée de l'autorité va donc signer ce couple de manière que la clé publique de cette autorité puisse permettre de vérifier cette signature.

Les certificats numériques Un certificat numérique est un message signé par la clé privée d'une autorité de confiance. Cette autorité de confiance est un tiers qui est reconnu digne de confiance par les deux parties d'une transaction.

Un certificat X.509 version 3 est un standard qui contient notamment les renseignements suivants :

- l'identité du porteur du certificat ;
- l'identité de l'autorité de certification ;
- les coordonnées de l'émetteur du certificat ;

– la clé publique, objet du certificat ;
– les paramètres de sécurité utilisés ;
– la période de validité du certificat ;
– la signature numérique de l'autorité émettrice pour valider le certificat.

Avant tout échange, il convient donc de se procurer un certificat auprès d'une autorité de certification. Le partenaire doit fournir son identité.

– on y adjoint sa clé publique ;
– l'autorité ajoute ses propres informations dont sa propre clé publique ;
– l'autorité calcule l'empreinte du tout et chiffre avec sa clé privée ;
– l'autorité signe un certificat pour le partenaire à l'aide de cette empreinte.

Le récepteur peut récupérer ce certificat, recalculer l'empreinte correspondante pour vérifier l'intégrité du certificat, déchiffrer la signature du certificat avec la clé publique de l'autorité et vérifier que les deux empreintes sont identiques. On peut donc faire la preuve de son identité, produire sa clé publique avec le certificat associé qui est une assurance de sécurité entre un nom de personne et sa clé publique associée. Quiconque peut ainsi vérifier la validité de la relation entre la clé publique et le nom associé.

Les infrastructures de gestion de clés (PKI [1])

Le mécanisme de gestion de ces certificats est mis en place dans les infrastructures de gestion de clés qui sont des infrastructures de confiance sur les réseaux pour vérifier l'identité des partenaires dans une communication ou une transaction.

Les PKI sont des infrastructures matérielles et logicielles dont le déploiement et les procédures sont en définitive assez lourdes. Une PKI comprend donc :

– une autorité d'enregistrement : cette autorité recueille en différé les demandes de certificats et prépare les certificats à valider ;
– une autorité de certification : cette autorité signe les certificats à l'aide de sa clé privée ;
– une autorité de dépôt et de séquestre : cette autorité permet de conserver et éventuellement de régénérer un certificat délivré à un utilisateur pour déchiffrer des messages quand le certificat n'est plus valable ou s'il a été perdu.

1. Public Key Infrastructure

A.2 Cryptanalyse

La cryptanalyse des algorithmes est essentielle pour trouver les failles des algorithmes. Faire la cryptanalyse d'un algorithme de chiffrement, c'est étudier sa sécurité en tentant de résoudre (casser) les fonctions cryptographiques qui le composent à l'aide d'attaques. Il est donc avantageux pour un algorithme d'être complètement publié (code source) dans le but que des cryptographes étudient sa sécurité.

Dans cet ordre d'idées, il existe plusieurs attaques qui sont souvent utilisées et qui ont été répertoriées

D'abord il est important de connaître les vulnérabilités des systèmes cryptographiques. Souvent, les attaques fructueuses sur un système de chiffrement ne touchent aucunement les algorithmes de chiffrement en cause. Il peut s'agir notamment de l'exploitation d'une erreur de conception, d'une erreur de réalisation ou d'une erreur d'installation.

Liste des types de vulnérabilités :

- Vulnérabilités des architectures cryptologiques
- Vulnérabilités des réalisations
- Vulnérabilités liées aux mots de passe
- Vulnérabilités du matériel
- Vulnérabilités des modèles de la confiance
- Vulnérabilités des utilisateurs
- Vulnérabilités de la reprise après incident
- Vulnérabilités cryptographiques

Les types d'attaques sont nombreux et diffèrent selon le type de d'algorithme utilisé (symétrique, asymétrique, fonction de hachage, etc.).

La liste de types d'attaques sur les algorithmes est la suivante :

A.2.1 L'attaque en force brute

Brute force attack, Exhaustive key search attack

Le cryptographe essaie toutes les combinaisons de clés possibles jusqu'à l'obtention du texte clair. Avec des ordinateurs de plus en plus performants et avec les mé-

thodes de calculs distribués (avec cette méthode et un grand nombre d'ordinateurs travaillant ensemble, le DES a été cassé en 1999 en 22 heures seulement), l'attaque en force brute restera toujours un moyen de casser des systèmes de chiffrement.

A.2.2 L'attaque à l'aide de l'analyse statistique

Statistical analysis attack

Le cryptographe possède des informations sur les statistiques du message clair (fréquences des lettres ou des séquences de lettres, voir les statistiques d'utilisation des lettres).

A.2.3 L'attaque à l'aide de texte chiffré seulement

Ciphertext-only attack

Le cryptographe dispose de message chiffré par l'algorithme et fait des hypothèses sur le texte clair (des expressions, des mots, le sens du message, etc.).

A.2.4 L'attaque à l'aide de texte clair

Known-plaintext attack

Le cryptographe dispose des messages ou parties de message clairs et de leur version chiffrée.

A.2.5 L'attaque à l'aide de texte clair choisi

Chosen-plaintext attack

Le cryptographe dispose des messages clairs et de leur version chiffrée. Il a aussi la possibilité de tester des messages et d'obtenir le résultat chiffré. Les chiffrements asymétriques sont notamment vulnérables à cette attaque.

A.2.6 L'attaque d'une tierce personne

Man-in-the-middle attack

Dans une transaction entre deux entités, une troisième entité s'interpose entre les deux et termine la transaction normalement en captant les messages et en transmettant d'autres messages. Ainsi il peut changer les messages concernant l'échange de clés sans que les deux entités s'en aperçoivent. Cette attaque peut être évitée en utilisant des techniques d'authentification telles que la signature numérique.

A.2.7 L'attaque à l'aide du temps d'opération

Timing Attack

Elle est fondée sur la mesure répétitive du temps d'exécution exacte d'un groupe d'opérations exponentielles. L'attaque affecte RSA, Diffie-Hellman et la méthode des Courbes Elliptiques.

Bibliographie

[1] ISO/IEC 14496 :1999. Coding of moving pictures and audio, 1999.

[2] ADOBE. Portable Document Rights Language (PDRL) Specification, Juillet 2006. `http://www.adobe.com/devnet/livecycle/policyserver/articles/pdrl.pdf`.

[3] J.G. Apostolopoulos. Architectural principles for secure streaming & secure adaptation in the developing scalable video coding (svc) standard. *Image Processing, 2006 IEEE International Conference on*, pages 729–732, Oct. 2006.

[4] John Apostolopoulos. secure media streaming & secure adaptation for non-scalable video. *International Conference on Image Processing, ICIP'04*, 3, 2004.

[5] C.W.A. Arora and KS Pabla. jxme : JXTA Platform Project. *Web : http://www.jxme.org, February*, 2005.

[6] Mariam Kimiaei Asadi. *Adaptation de Contenu Multimédia avec MPEG-21 : Conversion de Ressources et Adaptation Sémantique de Scènes.* PhD thesis, École Nationale Supérieure des Télécommunications - Paris, 2005.

[7] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. volume 9, pages 1–30. ACM New York, NY, USA, 2006.

[8] A. Barbir, O. Batuner, B. Srinivas, M. Hofmann, and H. Orman. Security Threats and Risks for Open Pluggable Edge Services (OPES). Technical report, RFC 3837, August 2004.

[9] A. Berglund, S. Boag, D. Chamberlin, M.F. Fernandez, M. Kay, J. Robie, and J. Simeon. XML Path Language (XPath) 2.0. *W3C Recommendation*, 15, 2007.

[10] Harini Bharadvaj, Anupam Joshi, and Sansanee Auephanwiriyakul. An active transcoding proxy to support mobile web access. In *Seventeenth IEEE Symposium on Reliable Distributed Systems (SRDS '98)*, pages 118–126, Washington - Brussels - Tokyo, october 1998. IEEE.

[11] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[12] A. Boldyreva, A. Palacio, and B. Warinschi. Secure Proxy Signature Schemes for Delegation of Signing Rights. *At : http://eprint.iacr.org/2003/096*, 2003.

[13] S. Boll. *ZYX : Towards Flexible Multimedia Document Models for Reuse and Adaptation.* University of Vienna, Austria, 2001.

[14] Julien Bourgeois, Emmanuel Mory, and François Spies. Video transmission adaptation on mobile devices. *Journal of Systems Architecture*, 49(10-11) :475–484, November 2003.

[15] J. Chakareski, J. Apostolopoulos, S. Wee, W. Tan, and B. Girod. Rate-Distortion Hint Tracks for Adaptive Video Streaming. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY*, 15(10) :1257, 2005.

[16] J. Chapweske and G. Mohr. Tree Hash EXchange format (THEX), 2003. `http://www.open-content.net/specs/draft-jchapweske-thex-02.html`.

[17] C. Concolato, JC. Moissinac, and JC. Dufourd. Representing 2d cartoons using svg. In *SMIL Europe 2003 Conference*, pages 12–14, February 2003.

[18] W3C Consortium. Composite capability/preference profiles (cc/pp) : Structure and vocabularies, 2001. `http://www.w3.org/TR/CCPP-struct-vocab/`.

[19] Web 3D Consortium. X3d draft specification, February 2002. `http://www.web3d.org/x3d/`.

[20] I. ContentGuard. eXtensible rights Markup Language (XrML) 2.0 Specification, 2001. `http://www.xrml.org`.

[21] K. Coyle. Rights expression languages. In *A Report for the Library of Congress*, 2004.

[22] E. Damiani and S.D.C. di Vimercati. SECURING XML-BASED MULTIMEDIA CONTENT. *Security and Privacy in the Age of Uncertainty : IFIP TC11 18th International Conference on Information Security (SEC2003), May 26-28, 2003, Athens, Greece*, 2003.

[23] JH Davenport, Y. Siret, and E. Tournier. *Computer algebra : systems and algorithms for algebraic computation.* Academic Press Ltd. London, UK, UK, 1988.

[24] Parrott David. Requirements for a Rights Data Dictionary and Rights Expression Language. Technical report, Reuters, June 2001.

[25] S. Davoine, F. et Pateux. *Tatouage de documents audiovisuels numériques.* Hermes, 2004.

[26] G. Doërr. *Security issue and collusion attacks in video watermarking.* PhD thesis, EURECOM, 2004.

[27] J.C. Dufourd. LASeR : The lightweight rich media representation standard [Standards in a Nutshell]. *Signal Processing Magazine, IEEE*, 25(6) :164–168, 2008.

[28] A. M. Eskicioglu. Multimedia security in group communications : Recent progress in wired and wireless networks. In *Proceedings of the IASTED International Conference on Communications and Computer Network s*, volume 87 - 7, pages 125–133, November 4-6 2002.

[29] E. Fernandez-Medina, G. Ruiz, and S. De Capitani di Vimerati. Implementing an Access Control System for SVG Documents. *Lecture Notes in Computer Science, Catania, Italy, PP*, pages 741–753, 2003.

[30] D. Ferraiolo, J. Cugini, and D.R. Kuhn. Role-Based Access Control (RBAC) : Features and Motivations. *Proceedings of the Eleventh Annual Computer Security Applications Conference*, 1995.

[31] Borko Furht and Darko Kirovski. *Multimedia Security Handbook.* CRC Press, Inc., Boca Raton, FL, USA, 2005.

[32] D. Furht B. Muharemagic, E. Socek. *Multimedia Encryption and Watermarking.* Springer, 2005.

[33] R. Furuta, J. Scofield, and A. Shaw. Document Formatting Systems : Survey, Concepts, and Issues. *ACM Computing Surveys*, 14(3) :417–472, 1982.

[34] P.R. Gallagher. A guide to understanding discretionary access control in trusted systems. *Proc. National Computer Security Center Fort George G. Meade, Maryland 20755*, 6000, 1987.

[35] B. Gassend, E. Suh, D. Clarke, M. van Dijk, and S. Devadas. Caches and merkle trees for efficient memory authentication. *Proceedings of the 9th International Symposium on High Performance Computer Architecture*, pages 295–306, 2003.

[36] M. Gehrke, A. Pfitzmann, and K. Rannenberg. Information Technology Security Evaluation Criteria (ITSEC)-a Contribution to Vulnerability ? *IFIP Congress, Vol. 2*, pages 579–587, 1992.

[37] Craig Gentry, Alejandro Hevia, Ravi Jain, Toshiro Kawahara, and Zulfikar Ramzan. End-to-end security in the presence of intelligent data adapting proxies : the case of authenticating. In *IEEE Journal on Selected Areas in Communications*, volume 23-2, pages 464–473, 2005.

[38] JD Gibson, A. Servetti, H. Dong, A. Gersho, T. Lookabaugh, and JC De Martin. Selective encryption and scalable speech coding for voice communications over multi-hop wireless links. In *IEEE Military Communications Conference, 2004. MILCOM 2004*, volume 2.

[39] L. Gong. JXTA : A network programming environment. *IEEE Internet Computing*, 5(3) :88–95, 2001.

[40] Ted Hardie, Scott Hollenbeck, et al. Ietf open pluggable edge services (opes) working group. 2005. `http://www.ietf.org/proceedings/05mar/opes.html`.

[41] X. He and Q. Zhang. Image Encryption Based on Chaotic Modulation of Wavelet Coefficients. *Image and Signal Processing, 2008. CISP'08. Congress on*, 1, 2008.

[42] R. Iannella. Open Digital Rights Language (ODRL) Version 1.1, 2002. `http://www.odrl.net`.

[43] IBM. Websphere transcoding publisher, 2007. `http://www-01.ibm.com/software/pervasive/transcoding_publisher/`.

[44] Macromedia Inc. Macromedia flash mx 2004, October 25 2003. `http://www.macromedia.com/software/flash/`.

[45] Information Technology Security Evaluation Criteria (ITSEC). Provisional harmonised criteria. V 1.2, Jun. 1991.

[46] Java-Sun. Java secure socket extension. `http://java.sun.com/products/archive/jsse/`.

[47] Java-Sun. Myjxta. `https://myjxta.dev.java.net/`, 2006.

[48] Mathias Johanson. A RTP to HTTP video gateway. pages 499–503. ACM Press New York, NY, USA, 2001.

[49] Muriel Jourdan. Madeus, an authoring environment for interactive multimedia documents. In *Sixth ACM International Conference on Multimedia*, September 1998.

[50] Ahmed Reda Kaced and Jean-Claude Moissinac. Multimedia content authentication for proxy-side adaptation. In *Proceedings of the IEEE International Conference on Digital Telecommunications, ICDT 06*, 2006.

[51] Ahmed Reda Kaced and Jean-Claude Moissinac. Semafor : a framework for authentication of adaptive multimedia content and delivery for heterogeneous networks. In *Proceedings of ICISP 06*, 2006.

[52] Z.I Kazi-Aoul. *Une architecture orientée services pour la fourniture de documents multimédia composés adaptables*. PhD thesis, École Nationale Supérieure des Télécommunications - Paris, 2008.

[53] Z.I. Kazi-Aoul, I. Demeure, and J.C. Moissinac. PAAM : A Web Services Oriented Architecture for the Adaptation of Composed Multimedia Documents. In *proceedings of Parallel and Distributed Computing and Networks*, page 597, 2008.

[54] D. Khan. *The Codebreakers*. Scribner, 1996.

[55] N. Kodali, C. Farkas, and D. Wijesekera. An authorization model for multimedia digital libraries. *International Journal on Digital Libraries*, 4(3) :139–155, 2004.

[56] N. Kodali and D. Wijesekera. Regulating access to SMIL formatted pay-per-view movies. *Proceedings of the 2002 ACM workshop on XML security*, pages 53–60, 2002.

[57] M. Kropfberger, P. Schojer, et al. ViTooKi–The Video ToolKit, 2004. `http://vitooki.sourceforge.net/`.

[58] Jean-Christophe. Lapayre and Fabien. Renard. Appat : A new platform to perform global adaptation. In *the 1st IEEE International Conference. DFMA'2005*, France, 2005.

[59] Mpeg Laser. Laser homepage, 2004. `http://www.mpeg-laser.org`.

[60] Oussama Layaida, Slim Ben Attalah, and Daniel Hagimont. Adaptive media streaming using self-reconfigurable proxies. In *7th IEEE International Conference, HSNMC 2004*. Springer, 2004.

[61] Tieyan Li and Yongdong Wu. Adaptive Stream Authentication for Wireless Multimedia Communications. *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pages 2613–2618, 2007.

[62] Tieyan Li, Yongdong Wu, Di Ma, Huafei Zhu, and Robert H. Deng. Flexible verification of MPEG-4 stream in peer-to-peer CDN. In *Proceedings of the 6th International Conference on Information and Communications Security (ICICS)*, pages 79–91, 2004.

[63] Tieyan Li, Huafei Zhu, and Yongdong Wu. Multi-Source Stream Authentication Framework in Case of Composite MPEG-4 Stream. *Lecture Notes in computer science*, 3783 :389, 2005.

[64] C.S. Lu. *Multimedia Security : Steganography and Digital Watermarking Techniques for Protection of Intellectual Property.* Idea Group Inc (IGI), 2005.

[65] M. MAMBO and E. OKAMOTO. Proxy Cryptosystems : Delegation of the Power to Decrypt Ciphertexts. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 80(1) :54–63, 1997.

[66] P.C. Vanestone S.A Menezes, A.J. Van Oorschot. *Handbook of applied cryptography.* CRC Press, Juillet 1999.

[67] Ralph C. Merkle. A certified digital signature. In *CRYPTO'89 : Lecture Notes on Computer Science*, volume 0435, pages 218–238. Springer-Verlag, 1989.

[68] R.C. Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, 1979.

[69] T. Meyer-Boudnik and W. Effelsberg. MHEG Explained. *IEEE Multimedia*, 2(1) :26–38, 1995.

[70] R. MOLVA and A. PANNETRAT. Scalable Multicast Security with Dynamic Recipient Groups. *ACM Transactions on Information and System Security*, 3(3) :136–160, 2000.

[71] J.L. Muñoz, J. Forne, O. Esparza, and M. Soriano. Certificate revocation system implementation based on the Merkle hash tree. *International Journal of Information Security*, 2(2) :110–124, 2004.

[72] NATO. Nato : Trusted computer system evaluation criteria(blue book). NATO AC/35-D/1027, 1987.

[73] ORACLE. Oracle application server wireless : Complete mobile plateform. 2002, 2002.

[74] S. Osborn. Mandatory access control and role-based access control revisited. *Proceedings of the second ACM workshop on Role-based access control*, pages 31–40, 1997.

[75] FAP. Petitcolas and Markus G. Anderson, R.J. Kuhn. Information hiding - a survey. In *Proceedings of the IEEE*, volume 87 - 7, pages 1062–1078, 1999.

[76] PlanetLab. `http://www.planet-lab.org/`.

[77] M. Portmann and A. Seneviratne. The problem of end-to-end security for proxy-based systems. In *Procedings of Protocols for Multimedia Systems, PROMS2000, Cracow, Poland, oct*, 2000.

[78] Kai Rannenberg. Recent development in information technology security evaluation - the need for evaluation criteria for multilateral security. In *Proceedings of the IFIP TC9/WG9.6 Working Conference on Security and Control of Information Technology in Society on board M/S Illich and ashore*, pages 113–128, 1994.

[79] D. Recordon and D. Reed. OpenID 2.0 : a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management*, pages 11–16. ACM New York, NY, USA, 2006.

[80] Fabien Renard. *Conception d'une plate-forme d'adaptation globale : application au télédiagnostic médical.* PhD thesis, université de Franche-Compté, 2004.

[81] Z. Sakr and N.D. Georganas. A Novel Algorithm for Authentication and Protection of SMIL Scenes. *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*, pages 304–307, 2007.

[82] Z. Sakr and N.D. Georganas. Robust content-based MPEG-4 XMT scene structure authentication and multimedia content location. 2007.

[83] P. Samarati and S.D.C. di Vimercati. Access control : Policies, models, and mechanisms. *Foundations of Security Analysis and Design, LNCS*, 2171, 2001.

[84] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-Based Access Control Models. *COMPUTER*, pages 38–47, 1996.

[85] B. Schneier. *Applied cryptography.* Wiley New York, 1996.

[86] Y. Song, K. Beznosov, and V.C.M. Leung. Multiple-channel security architecture and its implementation over SSL. *EURASIP Journal on Wireless Communications and Networking*, 2006(2) :78–78, 2006.

[87] M. Stefik. Letting Loose the Light : Igniting Commerce in Electronic Publication. *Xerox PARC, Palo Alto, CA) 1994-1995, 35 pages.*

[88] M. Stefik. Trusted Systems. *Scientific American*, 276(3) :78–81, 1997.

[89] STREAMEZZO. `http://www.streamezzo.com`.

[90] Q. Sun and S.F. Chang. A secure and robust digital signature scheme for JPEG2000 image authentication. *Multimedia, IEEE Transactions on*, 7(3) :480–494, 2005.

[91] Lemlouma Tayeb. *Architecture de Négociation et d'Adaptation de Services Multimédia dans des Environnements Hétérogènes.* PhD thesis, Institut National Polytechnique de Grenoble - INPG, 2004.

[92] H.F. Tipton and M. Krause. *Information Security Management Handbook.* Auerbach Publications, 2004.

[93] K. Topley and R. Eckstein. *J2ME in a Nutshell.* O'Reilly & Associates, Inc. Sebastopol, CA, USA, 2002.

[94] W. Trappe, M. Wu, ZJ Wang, and KJR Liu. Anti-collusion fingerprinting for multimedia. In *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, volume 51, pages 1069–1087, 2003.

[95] Maria Simona Trocan. *Décompositions spatio-temporelles et allocation de débit utilisant les coupures de graphe pour le codage vidéo scalable.* PhD thesis, École Nationale Supérieure des Télécommunications - Paris, 2007.

[96] W3C. W3c recommendation. xml-encryption, December 2002. `http://www.w3.org/TR/xmlenc-core/`.

[97] W3C. W3c recommendation. xml-signature syntax and processing, February 2002. `http://www.w3.org/TR/xmldsig-core`.

[98] W3C. Scalable vector graphics (svg) 1.2 specification, April 2005. `http://www.w3.org/TR/SVG12/`.

[99] W3C. W3c recommendation. synchronized multimedia integration language (smil 2.1), December 2005. `http://www.w3.org/TR/2005/REC-SMIL2-20051213/`.

[100] Xin Wang and Henning Schulzrinne. Comparison of adaptive internet multimedia applications, mars 1999.

[101] Susie Wee and John Apostolopoulos. Secure scalable streaming enabling transcoding without decryption. *Image Processing, 2001. Proceedings. 2001 International Conference on*, 1, 2001.

[102] Susie Wee and John Apostolopoulos. Secure scalable streaming and secure transcoding with JPEG-2000. *Image Processing, 2003. Proceedings. 2003 International Conference on*, 1, 2003.

[103] W. Zeng, J. Lan, and X. Zhuang. Security for Multimedia Adaptation : Architectures and Solutions. *IEEE MultiMedia*, 13(2) :68–76, 2006.

[104] BB Zhu, C. Yuan, Y. Wang, and S. Li. Scalable protection for MPEG-4 fine granularity scalability. *Multimedia, IEEE Transactions on*, 7(2) :222–233, 2005.