Chapitre 10

Sécurité des communications en multicast

Avec l'évolution d'Internet, les communications en multicast semblent particulièrement bien adaptées pour des applications de distribution commerciale de contenu à grande échelle, comme, par exemple, des chaînes à péage. Un inconvénient qui freine ou ralentit cette démarche est le critère crucial de sécurité qui est différent du cas de l'unicast. Dans ce chapitre, nous allons tout d'abord donner une classification des applications multicast pour ensuite traiter leurs deux principaux besoins en termes de sécurité : l'authentification et la confidentialité des données. Après l'analyse détaillée de ces problèmes, nous présenterons les principales solutions existantes en analysant leurs avantages et limites en termes de performance et robustesse.

10.1. Introduction à la sécurité multicast

10.1.1. Les applications multicast et leurs caractéristiques

L'IP multicast [DEE 89] est un mécanisme qui permet à une source de distribuer des données à un groupe prédéfini de récepteurs de façon optimale. Au lieu de générer une copie des données à chaque membre du groupe, la source n'envoie qu'une seule copie et ce sont les autres éléments intermédiaires du réseau qui se chargent de dupliquer et de transmettre les copies au reste du groupe quand cela est nécessaire. L'identification du groupe se fait par une unique adresse IP appelée adresse IP multicast. Les récepteurs désirant faire partie de ce groupe doivent se manifester auprès de leur routeur multicast local par le protocole IGMP [FEN 97].

Aujourd'hui, les applications multicast sont regroupées sous trois catégories principales selon les relations existantes entre émetteur et récepteur [QUI 01] :

– *les applications 1-à-*N : elles sont définies par la présence d'une seule ou d'un petit nombre de sources qui émettent une très grande masse de données à un très grand nombre de récepteurs. La plupart des applications commerciales appartiennent à cette catégorie. Certaines de ces applications sont les suivantes :

- la diffusion de télévision à péage,
- la diffusion de contenu audio de haute qualité,
- la distribution de cotations boursières,
- la mise à jour automatique de logiciels ;

– *les applications* N-*à*-N : pour des applications de cette catégorie, tous les récepteurs peuvent avoir le rôle d'émetteur. Les applications suivantes sont les plus utilisées :

- la vidéo ou audio conférence,
- les bases de données distribuées,
- les jeux interactifs ;

– *les applications* N-*à-1* : il s'agit des applications où plusieurs sources envoient des données à un unique ou à quelques récepteurs. Contrairement aux applications des deux catégories précédentes, les protocoles à récepteur unique n'utilisent pas la couche IP.

Dans ce chapitre, nous nous intéressons plus particulièrement aux applications de la première catégorie et supposerons que le mécanisme de transport est peu fiable afin de couvrir le plus grand nombre de scénarios. En effet, la plupart des applications multicast utilisent UDP [POS 80] comme couche de transport, avec parfois l'addition d'un protocole de fiabilité supplémentaire comme par exemple RTP [SCH 96]. Cela implique que l'on n'a pratiquement aucune garantie de fiabilité sur le transport de données. Dans le cadre des communications 1-à-*N*, les applications souvent multimédias sont construites pour tolérer des pertes ou des délais comme ceux que l'on peut observer dans un réseau non fiable.

Suite à la classification de la majorité des applications multicast, nous pouvons alors définir les trois différents éléments indispensables d'une topologie générale d'une application multicast :

– *la source* qui se charge de l'émission des données aux membres du groupe et de la gestion de l'adhésion ou de la révocation d'un récepteur au groupe. Cette source sera localisée à la racine de l'arbre de routage ;

– *les éléments intermédiaires*, qui dans ce contexte sont les routeurs ou proxies implémentant un protocole de routage multicast. Leur nombre peut varier selon l'architecture du réseau ;

– *les feuilles*, qui représentent l'ensemble des membres d'un groupe qui y sont connectés *via* le protocole IGMP.

La topologie de la plupart des applications multicast est basée sur la présence de ces trois éléments. Néanmoins, il existe des applications où la source est directement connectée aux membres du groupe et donc où il n'existe pas d'éléments intermédiaires. Les protocoles de sécurité doivent donc prendre en compte l'architecture du réseau.

10.1.2. *Principaux besoins en sécurité*

Les besoins de sécurité d'une application multicast dépendent fortement de sa nature. Il est clair qu'une distribution commerciale nécessite de fournir des mécanismes qui restreignent l'accès au contenu distribué en multicast pour une durée déterminée aux seuls clients légitimes de l'application. Ce genre d'applications présente plus de contraintes que les applications non commerciales. Nous proposons donc d'analyser leurs besoins spécifiques en étant convaincus que les solutions apportées pour ces applications peuvent être utilisées pour des applications dont le besoin en sécurité est plus souple.

Pour illustrer les problèmes essentiels de sécurité, nous allons nous baser sur un scénario de diffusion de télévision à péage avec un très grand nombre de clients. Ces derniers doivent s'abonner à ce service pour une durée variable selon leurs besoins.

Dans ce contexte, l'entreprise qui aura donc le rôle de source dans la topologie, rencontre deux problèmes majeurs :

– *la confidentialité* : seuls les membres ou abonnés à ce service (qui ont donc déjà payé) doivent avoir accès au contenu des données. De plus, ils ne doivent pouvoir y accéder que pour la durée qui a été convenue selon le contrat.

– *l'authentification* : la source doit pouvoir fournir un mécanisme prouvant l'origine des paquets émis contre les possibles risques d'usurpation d'identité. Ce besoin est primordial surtout pour les applications où les données sont critiques. Le client doit également pouvoir s'assurer que les données reçues n'ont pas été modifiées pendant leur transmission. Il s'agit alors de *l'intégrité*. Tout au long de ce chapitre, nous considérerons que ce service est fourni avec les solutions d'authentification.

La confidentialité des données peut être assurée par l'utilisation des algorithmes cryptographiques de chiffrement. Ceux-ci se regroupent en deux catégories selon les propriétés de la clef de chiffrement utilisée [MEN 96]. Un algorithme de chiffrement est *symétrique* (ou à clef secrète) si la clef utilisée pour le chiffrement est identique à

celle du déchiffrement. Les algorithmes de chiffrement *asymétrique* (ou à clef publique) sont conçus de telle manière que la clef de chiffrement soit différente de celle du déchiffrement. La clef de chiffrement est appelée clef publique et comme son nom l'indique elle peut être rendue publique. La clef de déchiffrement appelée clef privée n'est connue que par l'entité pouvant déchiffrer le message et ne peut être calculée (du moins en temps raisonnable) à partir de la clef publique.

De façon similaire, les algorithmes fournissant un service d'authentification se différencient selon les propriétés des fonctions de génération et de vérification de l'empreinte [MEN 96]. Les *signatures numériques* sont basées sur des algorithmes à clef publique. Par conséquent, les fonctions de génération et de vérification utilisent différents paramètres d'entrée. Dans le cadre des *codes d'authentification de messages*, les fonctions de génération et la vérification sont basées sur une clef secrète.

Que ce soit pour la confidentialité ou l'authentification, le choix de l'algorithme à utiliser dépend fortement de la nature de l'application, de ses besoins non seulement en termes de sécurité mais également en termes d'efficacité et enfin de la nature de la communication.

10.1.3. *Limites des solutions adaptées aux communications en unicast*

La confidentialité et l'authentification ne sont pas des besoins nouveaux en sécurité. Pour les communications en unicast, il existe un certain nombre de protocoles qui fournissent ces deux services de façon souple et puissante.

Les algorithmes de chiffrement à clef publique sont souvent très coûteux en termes de mémoire et de calcul. Ceux-ci ne sont donc pas utilisés pour assurer la confidentialité des communications intégrant l'échange de données en masse. Dans le cadre des communications en unicast, deux entités se mettent d'accord pour l'utilisation d'une clef symétrique K et à la fin de la communication celle-ci n'est plus utilisée.

Dans le cadre des communications en multicast, la source ne peut pas se permettre de définir une clef secrète pour chaque membre et chiffrer les mêmes données avec chacune des clefs des membres du groupe. Elle doit donc définir une clef de chiffrement de données K (ou un petit nombre) qui ne doit être connue que par les membres ayant payé pour le service. La distribution de cette clef doit être efficace. Par ailleurs, cette clef doit être mise à jour lors de chaque adhésion ou révocation d'un membre. Si K est l'unique information partagée entre tous les

membres du groupe et la source et que sa modification est indispensable, la distribution de la nouvelle clef doit être efficace.

De même, pour des communications en unicast, en raison de l'important coût des algorithmes à clef publique, la signature numérique n'est pas utilisée pour l'authentification des données massives. Deux entités se mettent d'accord pendant une première phase sur une clef secrète qui est ensuite utilisée pour la génération de codes d'authentification de messages (MAC).

Cette solution ne peut cependant pas directement être appliquée aux communications en multicast. En effet, si tous les membres partagent une même clef de vérification d'empreinte, en raison de la propriété de symétrie des codes d'authentification de message, ces membres peuvent également générer des empreintes valides et se faire passer pour la source.

10.2. Authentification Multicast

10.2.1. *Définition et Besoins*

Dans le cadre de la plupart des applications en multicast de type 1-à-*N* comme celle de la distribution de cotations boursières, les données ne doivent pas être falsifiées. C'est pour cela que chacun des membres du groupe doit pouvoir s'assurer de l'origine des paquets qu'il reçoit. Il s'agit de l'authentification de l'origine d'un paquet.

Pour définir un schéma d'authentification efficace, il est important d'analyser le type de données à traiter et la nature de l'application. En effet, pour des données préenregistrées destinées au play-back, il est possible de calculer les empreintes à l'avance, sans contrainte de temps, et les insérer ensuite dans les données. Pour les données émises en temps réel, comme la retransmission d'un événement sportif en direct, la source doit être capable de faire les calculs cryptographiques nécessaires pour authentifier les données en temps réel et insérer l'information dans le flux immédiatement. Naturellement, un schéma qui fonctionne pour le direct, fonctionne *de facto* pour le play-back. Il est donc intéressant de fournir des solutions efficaces pour le direct.

Comme décrit dans le paragraphe 10.1.3, dans le cadre des communications en unicast, deux entités communicantes font appel aux codes d'authentification de messages. Pour des raisons de sécurité, ces techniques ne peuvent pas directement être appliquées dans le multicast. En effet, dans le cadre des applications où la source génère des empreintes pour un très grand nombre de récepteurs**,** l'utilisation

des codes d'authentification des messages engendre le problème d'usurpation de l'identité par n'importe quel membre du groupe. Pour adapter les solutions destinées aux communications en unicast à celles en multicast, il est indispensable de prendre en compte les critères suivants :

– *le délai* : dans un cas idéal, un membre de groupe doit pouvoir authentifier les paquets individuellement dès leur réception. Dans la pratique, la vérification de l'authenticité d'un paquet peut dépendre d'autres paquets qui n'ont pas encore été reçus. Le membre ne pourra vérifier l'empreinte qu'après la réception de certains autres paquets. Le délai doit être bien sûr optimisé ;

– *le coût* : il est indispensable d'optimiser le coût de la génération de l'empreinte du côté de la source et celui de la vérification du côté du membre en termes de ressources de calcul. D'autre part, pour authentifier les paquets il est nécessaire de leur ajouter une certaine information d'authentification. Si celle-ci est coûteuse en termes d'espace, elle réduit d'autant la bande passante disponible ;

– *la robustesse* : pour réduire le coût de la génération de la vérification de l'empreinte d'un paquet, la plupart des solutions créent une dépendance logique entre les paquets. Cette dépendance fait naître un nouveau problème qui est celui de la tolérance aux pertes de paquets. Une perte de paquets ne doit pas provoquer la perte de l'authentification pour un paquet qui a déjà été reçu ;

– *la modularité* : pour certaines applications, il n'est pas souhaitable qu'un membre du groupe soit présent depuis le début du flux. Celui-ci doit pouvoir authentifier les données à partir d'un point arbitraire dans le flux.

Tout au long de cette partie du chapitre, nous allons passer en revue les différents systèmes d'authentification proposés pour le multicast que nous regroupons selon la symétrie des algorithmes implémentés. Il existe donc deux approches où certaines des solutions sont basées sur l'utilisation des codes d'authentification de message qui sont efficaces en termes de ressources de calcul et d'autres sur l'utilisation des signatures numériques où une réduction du coût de génération et de vérification et de celui en termes d'espace doit être envisagée.

10.2.2. *Techniques utilisant des algorithmes symétriques*

10.2.2.1. *Les codes d'authentification de messages multicast* (*MMAC*)

Ce premier schéma [CAN 99] adapte l'utilisation des codes d'authentification de messages (MAC) pour les communications en multicast. Les auteurs proposent de générer l différentes empreintes pour un paquet et de fournir un sous-ensemble de clefs différent à chaque membre du groupe. Le nombre de clefs à distribuer à chacun des membres dépend d'un paramètre de sécurité w.

Le schéma est décrit dans le tableau 10.1. A l'aide de chacune des *l* clefs, la source produit *l* MAC et chaque membre ne vérifie que les empreintes générées avec les clefs en sa possession. Si au moins l'un des MACs ne correspond pas à celui envoyé, le paquet est considéré comme falsifié.

Initialisation :
Définir *l* clefs MAC $\{K_1, ..,.K_l\}$
Pour chaque membre M_i distribuer un ensemble R_i *de* $l/(w + 1)$ *clefs*
Authentification d'un paquet P :
Diffuser $(P, MAC(K_1,P), ..., MAC(K_l, P))$
Vérification du paquet P par le membre M_i :
Vérifier $(P, \{MAC(K_j, P)\})$ tel que $\{K_j\} = R_i$
Si au moins un MAC n'est pas correct, alors P est falsifié

Table 10.1. *Principe du schéma MMAC*

Du point de vue robustesse et délai, ce schéma a les mêmes avantages que l'utilisation d'un simple MAC pour les communications en unicast. En effet, la vérification de l'empreinte d'un paquet ne dépend pas de la réception d'autres paquets et est immédiate dès sa réception. Cette propriété fournit la possibilité pour un membre d'être partiellement présent lors de la transmission d'un flux. Par ailleurs, ce schéma est également très efficace en termes de coût de ressources de calcul puisque les MAC sont basés sur l'utilisation de fonctions de hachage.

En termes de sécurité, l'efficacité de ce schéma dépend du choix de la valeur du paramètre *w*. Ce paramètre représente le nombre limite de membres pouvant faire collusion afin de pouvoir usurper l'identité de la source. La valeur de ce paramètre doit par conséquent être assez large. Cette contradiction a malheureusement un impact très important sur le coût du schéma en termes d'espace par paquet. Dans [PAN 02c], l'auteur effectue une comparaison de ce schéma avec une simple génération de signature et réalise que pour $w = 10$, la taille du résultat reste non négligeable.

Pour conclure, l'implémentation de ce schéma présente des limitations assez importantes. Son utilisation n'est envisageable que dans le cadre des groupe avec peu de membres et où la source a confiance à la majorité des membres.

10.2.2.2. *TESLA*

Le schéma TESLA [PER 00] est également basé sur l'utilisation des techniques de codes d'authentification de messages (MAC). Un seul MAC est généré pour chaque paquet du flux selon une méthode particulière. La sécurité du schéma provient de la divulgation de la clef utilisée pour le calcul du MAC qui s'effectue après l'émission des empreintes. L'établissement d'une synchronisation entre la source et chacun des membres du groupe est donc indispensable.

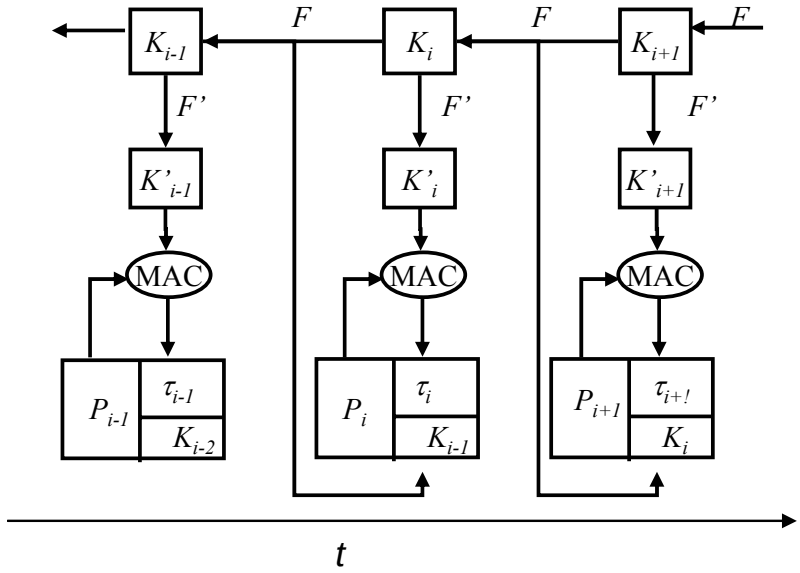

Figure 10.1. *Le schéma TESLA simplifié*

Dans ce schéma, la source divise le temps en un certain nombre d'intervalles de durée fixe et définit une chaîne de clefs en se basant sur les techniques de chaînage de clefs. Chaque clef de la chaîne prédéfinie est utilisée pour générer les MAC d'un unique intervalle dédié.

Avant l'émission des données, chaque membre s'authentifie auprès de la source et se synchronise de façon qu'un membre ne puisse pas usurper l'identité de la source. De son côté, la source définit la chaîne de clefs à l'aide d'une clef K_n et une fonction pseudo-aléatoire (ex : fonction à sens unique). Chacune des clefs de la chaîne est définie selon la méthode ci-dessous :

$$K(i) = F(K(i + 1)) \text{ pour } i = 1\ldots, n - 1 \qquad [10.1]$$

Lors de l'émission des données, au paquet *Pi,* la source concatène son empreinte calculée à partir de la clef K_i qui ne sera divulguée qu'après un certain nombre *d* d'intervalles. Lorsqu'un membre reçoit ce paquet, il ne peut vérifier son authenticité que lorsque la clef K_i sera en sa possession, c'est-à-dire à l'intervalle $i + d$. Le déroulement du protocole est illustré dans la figure 10.1 avec $d = 2$.

La sécurité de ce schéma dépend très fortement de la synchronisation des membres avec la source. Son coût est très faible : en termes de ressources de calcul, la source et les membres n'appliquent qu'une opération de MAC et une opération de hachage par paquet ; le coût en termes d'espace est presque identique à celui des protocoles d'authentification pour les communications en unicast. A l'empreinte s'ajoute seulement un champ contenant la clef utilisée pour la génération d'autres empreintes qui ont déjà été envoyés. Grâce à la propriété des fonctions de hachages, une perte d'un paquet ne provoque aucune contrainte pour l'authentification d'un autre paquet. En effet, si une clef K_i n'est pas reçue pour des raisons de pertes dans le réseau, grâce à la propriété de dépendance entre les clefs, un membre est capable de la retrouver dès la réception des prochaines clefs. L'un des paramètres les plus importants à définir est *d*, le délai d'annonce de la clef. Si cette valeur est petite, des paquets violeront la condition de sécurité. Si elle est très grande, le délai de vérification de l'empreinte augmentera proportionnellement.

Pour conclure, après l'établissement d'une synchronisation forte entre chaque membre et la source, l'efficacité de TESLA dépend seulement du choix du paramètre *d* et ne provoque aucune augmentation du coût supplémentaire à celui d'une fonction de hachage.

10.2.3. *Combinaison des algorithmes asymétriques et symétriques*

Les solutions comme [ROH 99] imposent la génération d'une empreinte pour chaque paquet et définissent de nouvelles méthodes de signature numérique qui ont un coût en termes de ressources de calcul plus faible que les méthodes classiques. Cependant, ces techniques génèrent un coût assez important en termes d'espace. D'autres travaux de recherche proposent de réduire ce coût avec la génération d'une signature pour un bloc de paquets consécutifs. Nous allons nous concentrer sur ces techniques et en présenter quelques-unes.

10.2.3.1. *Les arbres de hachage*

Dans ce schéma proposé par [WON 99], la source représente un bloc de paquets consécutifs par un arbre binaire équilibré dont chaque feuille correspond à la valeur du hachage d'un paquet. Les nœuds intermédiaires représentent le résultat du hachage de la concaténation de leurs fils (gauche et droit). Pour des raisons d'optimisation de coût, seule la valeur à la racine de l'arbre regroupant des informations sur tous les paquets est signée. A chaque paquet, la source ajoute cette signature ainsi que quelques autres informations complémentaires nécessaires pour que les récepteurs puissent recalculer la valeur de la racine de l'arbre et vérifier la signature.

Un exemple de l'utilisation de ce schéma pour un bloc de 8 paquets $\{P_1 ,.., P_8\}$ est illustré dans la figure 10.2. Pour le paquet P_3, les informations complémentaires que rajoute la source sont $\{h_4, h_{12}, h_{58}\}$. A la réception de ce paquet, un membre calcule d'abord h_3, puis récursivement $h_{34} = h(h_3|h_4)$, $h_{14} = h(h_{12}|h_{34})$ et $h_{18} = h(h_{14}|h_{58})$ avec « | » correspondant à l'opération la concaténation. Il vérifie ensuite l'authenticité du paquet à l'aide de l'empreinte reçue avec le paquet.

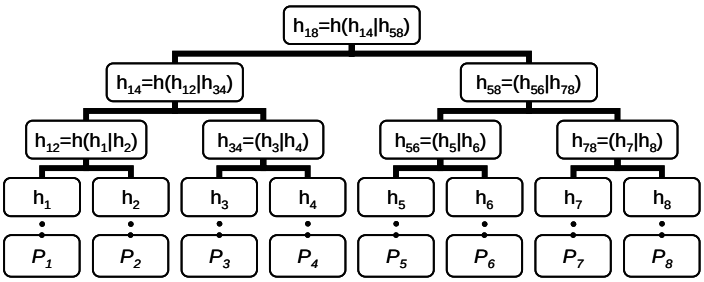

Figure 10.2. *Arbre de hachage pour 8 paquets*

Du point de vue robustesse, ce schéma est parfaitement tolérant aux pertes et ne provoque aucun délai de vérification car chaque paquet regroupe toutes les informations nécessaires pour la vérification de son authenticité. Cependant, il présente un inconvénient en termes d'espace. Si la taille du bloc *b* est importante, le nombre d'informations complémentaires sera important et le calcul de la racine de l'arbre pour chaque paquet ne sera pas efficace. Si la taille du bloc est petit, il y aura un plus grand nombre de vérifications de signatures.

10.2.3.2. *Chaînes de hachages*

Les auteurs de ce schéma [GOL 01] proposent une autre méthode de représentation d'un bloc de paquets. La source construit ici un graphe acyclique dirigé (DAG) en insérant la valeur du hachage d'un paquet dans un ou plusieurs autres paquets. La fonction de hachage est évaluée non seulement sur le paquet original mais également sur les valeurs de hachages qui ont été insérées.

Selon les propriétés des DAG, si un paquet P' est signé, alors l'authenticité de tous les paquets P_j pour lesquels il existe un chemin menant à P' peut être vérifiée. Concrètement, le schéma le plus simple qui se base sur cette construction définit pour chaque paquet P_i un ajout de sa valeur de hachage h_i dans le prochain paquet P_{i+1} et dans un autre paquet futur $P_{i+1+\beta}$. La génération de la signature s'effectue sur les valeurs de hachage des $\beta + 1$ derniers paquets du bloc. La transmission de la signature n'est pas clairement adressée. La valeur β est arbitrairement définie par la source selon la probabilité de perte du réseau qui souvent est modélisée par une chaîne de Markov à deux états [YAJ 96].

La figure 10.3 illustre ce schéma dans lequel le bloc est construit à partir de 16 paquets et $\beta = 6$. La valeur de hachage d'un paquet P_i apparaît dans les paquets P_{i+1} et P_{i+7}. La source génère la signature sur les valeurs de hachages des 7 derniers paquets, soit par exemple $\{P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}, P_{16}\}$. Pour vérifier l'authenticité d'un paquet quelconque de ce bloc, un membre doit d'abord réévaluer les valeurs $\{h_{10}, h_{11}, h_{12}, h_{13}, h_{14}, h_{15}, h_{16}\}$ à partir des paquets qu'il reçoit et vérifier ensuite la signature.

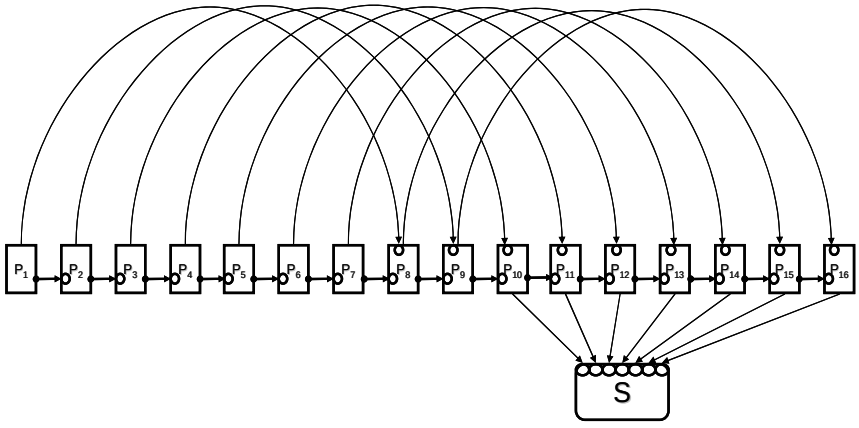

Figure 10.3. *Chaîne de hachage pour 16 paquets avec $\beta = 6$*

Du point de vue efficacité du schéma, chaque paquet compte deux valeurs de hachage additionnelles. La vérification d'une signature correspond à celle de l'authenticité d'un bloc de b paquets. Chaque membre doit donc recevoir le bloc (avec des pertes) pour ensuite évaluer le hachage de chaque paquet et reconstruire le bloc de hachage des β derniers paquets et vérifier la signature.

Le schéma tolère également les pertes de β paquets consécutifs. Cependant, le choix de ce paramètre n'est pas détaillé. Le délai avant la vérification de l'authenticité dépend de la durée de réception du bloc et de sa signature.

Pour conclure, la technique de chaînage de hachage reste l'une des techniques les plus efficaces du point de vue robustesse et coût en termes d'espace par paquet. D'autres auteurs [PER 00] ont repris ce schéma en définissant les relations entre les paquets aléatoirement. La définition d'un bloc s'effectue après l'établissement de ces relations pour la génération des signatures. La valeur de hachage d'un paquet peut alors apparaître dans plusieurs blocs.

10.2.3.3. Utilisation des codes correcteurs d'erreur

Dans [PAN 02a], les auteurs proposent l'utilisation des codes de reconstruction [RIZ 97] pour garantir un certain degré de robustesse. Par définition, un générateur de code $C_{k,r}$ prend un ensemble de k paquets en entrée et produit $(k + r)$ paquets dont les r derniers représentent des paquets de redondance. Un algorithme de reconstruction de données D_k permet de reconstruire les k paquets de données d'origine à partir de n'importe quel sous-ensemble de k paquets reçus.

Comme dans les techniques précédentes, les paquets consécutifs sont regroupés en un bloc. Pour chaque paquet P_i, la source évalue sa valeur de hachage h_i et la signature est générée à partir du bloc regroupant ces valeurs. La source construit ensuite des étiquettes selon la méthode décrite dans la figure 10.4. Selon la probabilité de perte du réseau p, la source utilise un premier générateur de code pour le bloc regroupant les valeurs de hachage.

Contrairement aux techniques précédentes, les auteurs proposent une méthode de transmission robuste de la signature grâce à l'utilisation d'un deuxième générateur de code prenant en entrée la signature ainsi que les paquets de redondance générés par le premier code. Le résultat est ensuite divisé en b étiquettes.

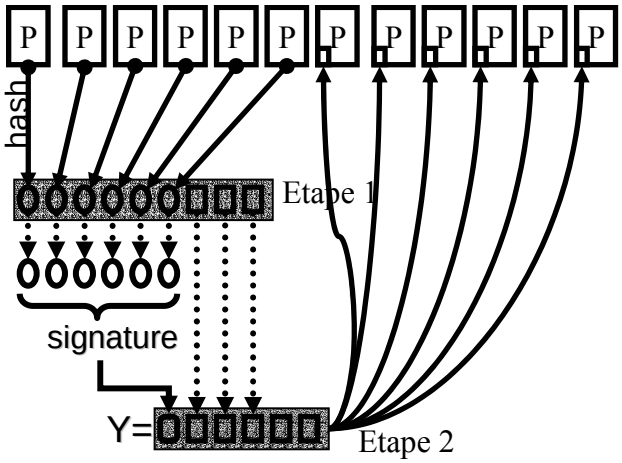

Figure 10.4. *Schéma d'authentification utilisant les codes de reconstruction*

Chacune de ces étiquettes est alors ajoutée à un paquet appartenant soit au bloc suivant, soit au bloc précédant ou tout simplement au bloc courant. Le choix de la configuration dépend de la nature des applications.

Les paramètres cruciaux que doit définir la source pour offrir une bonne efficacité sont la taille du bloc k et la valeur r. La valeur k dépend essentiellement du délai d'authentification maximal permis, car naturellement, plus k est grand, plus le délai d'authentification augmente. Si k est petit, cela demande plus de ressource de calcul. Le choix de r dépendra surtout de la probabilité de perte du réseau et du choix du degré de robustesse par la source.

Pour conclure, du point de vue calcul, cette architecture ne demande que l'évaluation d'une fonction de hachage pour chacun des paquets, une opération de vérification de signature et deux opération de reconstruction de codes en cas de pertes. La taille des étiquettes dépend du taux d'erreur et de la longueur du bloc. Elle reste optimale par rapport aux techniques de chaînes de hachages. L'avantage de ce schéma par rapport aux précédents est que la source n'envoie pas la valeur de hachage d'un paquet, ce qui réduit le coût de façon non négligeable.

10.2.4. *Conclusion*

Le problème de l'authentification dans le contexte des communications en multicast peut se résumer en deux points. Il existe d'une part une asymétrie implicite dans l'architecture du schéma. Les données sont générées par une ou un petit nombre d'entités pour ensuite être vérifiées par un très grand nombre de récepteurs. Cet aspect définit les propriétés de sécurité pour la conception d'un nouveau schéma. D'autre part, la quantité de données à authentifier est très importante au sein des applications comme celles de diffusion de télévision à péage. Il est donc important de tenir compte des contraintes en termes du délai d'authentification et des pertes dans le réseau pour définir un schéma efficace.

Pour répondre à ces besoins, s'il est possible d'établir une synchronisation sécurisée entre la source et chaque membre, TESLA reste le schéma le plus simple et efficace à intégrer. Si la synchronisation n'est pas envisageable, le schéma utilisant les codes correcteurs d'erreurs reste le moins coûteux en termes de calcul et d'espace parmi les schémas hybrides combinant l'utilisation des signatures numériques avec des fonctions de hachage qui sont peu coûteuses.

10.3. Confidentialité multicast

10.3.1. *Définition et besoins*

La confidentialité des données prend une place très importante dans le cadre des applications en multicast où le nombre de récepteurs est presque illimité. Seuls les membres d'un groupe défini doivent avoir la possibilité d'accéder aux données. Comme indiqué dans le paragraphe 10.1.2, le meilleur moyen pour protéger ces données est l'utilisation des algorithmes de chiffrement et de déchiffrement. Par conséquent, pour qu'un membre d'un certain groupe puisse avoir accès aux données en clair destinées à ce groupe, il doit recevoir une (ou plusieurs) clef de déchiffrement. Il est donc important d'étudier et d'optimiser le problème de la distribution de clefs.

Pour étudier ce problème, il est nécessaire de définir les propriétés du groupe multicast. Celui-ci peut avoir d'une part un statut statique dans lequel les membres adhèrent tous au début de la session et sont supposés ne pas quitter le groupe avant la fin de la session. Il peut aussi y avoir d'autre part un statut dynamique dans lequel les arrivées et départs de membres s'observent de façon fréquente. Dans la plupart des applications, il est fort probable que le groupe défini ait un statut dynamique. Nous nous intéressons aux groupes de ce type. Les solutions offertes pour ce type d'applications peuvent également être implémentés pour des groupes ayant un statut statique.

Les solutions apportées doivent valider les critères de sécurité énumérés ci-dessous :

– *le secret antérieur et ultérieur* : dans le cadre des groupes où les arrivées et départs de membres sont très fréquents, la source doit pouvoir assurer ces deux propriétés. Le secret antérieur est assuré lors de l'adhésion d'un ou plusieurs membres. Un nouveau membre ayant reçu les clefs nécessaires pour le déchiffrement des données actuelles ne doit en aucun cas avoir accès aux données transmises au groupe avant son adhésion. Symétriquement, le secret ultérieur est fourni lors de la révocation d'un membre du groupe. Un membre quittant le groupe ne doit plus avoir accès aux données transmises après son départ. Les secrets antérieur et ultérieur sont les besoins primaires pour permettre un accès temporel aux membres du groupe. Dans ce cas, à chaque changement de statut, la source doit au moins modifier la clef de chiffrement de données pour un certain ensemble de membres si ce n'est pour le groupe total ;

– *endiguement* : dans le cadre des groupes qui admettent un très grand nombre de membres il est fort probable que le secret que détient un certain membre soit divulgué par le membre lui-même à des récepteurs n'ayant pas l'autorisation d'accès. Lors de la révocation d'un membre suite à ce problème, le changement de certaines informations doit affecter seulement un petit nombre de membres. Lorsqu'un secret divulgué n'affecte qu'une partie du groupe et que cette divulgation reste temporelle, on parlera d'endiguement ;

– *résistance à la collusion* : dans le cas où un certain nombre de membres s'échangent leurs données confidentielles, ceux-ci ne doivent pas avoir plus de privilèges que ce que la source leur a assignés. Cet échange est la définition de la collusion. Si ces membres sont capables d'aboutir à un accès illimité, il est clair que le schéma est faible.

Par ailleurs, contrairement aux techniques classiques, les nouvelles solutions offrant de la confidentialité pour les communications de groupe doivent tenir compte des critères algorithmiques qui sont résumés ci-dessous :

– *indépendance au facteur d'échelle* : quelle que soit la solution offerte, la source doit être capable de gérer un très grand nombre de clients à n'importe quel moment pendant la session. Elle doit également être capable de traiter des arrivées ou départs en masse de certains membres sans réduire l'efficacité de la communication ;

– *coût* : la solution offerte doit tenir compte des caractéristiques des ressources (de calcul et de mémoire) de la source et surtout du récepteur. En effet, un membre ne doit pas consommer beaucoup de ressources pour le déchiffrement des données. L'architecture doit également optimiser l'utilisation des ressources de mémoire qui sont considérés comme limitées. Enfin, le coût d'une mise à jour de clefs ne doit pas saturer la bande passante. On parlera alors de coût de communication ;

– *robustesse* **:** cette propriété offre à un membre de pouvoir accéder au contenu en clair des paquets multicast dès leur réception ou après un très court délai.

Comme dans le cas de l'analyse du problème d'authentification, nous avons regroupé les solutions offrant de la confidentialité en deux catégories selon le rôle des nœuds intermédiaires dans le réseau. Dans certaines solutions, la méthode tient compte de la topologie du réseau et attribue un rôle de chiffrement et/ou de déchiffrement aux éléments intermédiaires. D'autres solutions se définissent indépendamment de la topologie du réseau. La présence des éléments intermédiaires n'est alors pas importante. Certains de ces schémas sont décrits dans le prochain paragraphe.

10.3.2. *Les arbres de rechiffrement*

10.3.2.1. *Iolus*

Ce schéma est considéré comme le plus représentatif de cette catégorie [MIT 97]. Les éléments intermédiaires y jouent un rôle important. Le groupe est divisé en sous-groupes où chacun d'entre eux est représenté par un élément intermédiaire. A chaque sous-groupe est associée une clef de chiffrement unique. Les éléments intermédiaires se chargent de transmettre les données chiffrées correctement.

La transmission des données chiffrées est illustrée par la figure 10.5. L'élément intermédiaire EI_1 fait partie des sous-groupes SG_1 et SG_2 avec SG_2 étant le sous-groupe que EI_1 est supposé gérer. EI_1 connaît donc les clefs K_1 et K_2 respectivement définies pour ces deux sous-groupes. A la réception d'un paquet P_i chiffré avec la clef K_1, EI_1 déchiffre tout d'abord ce paquet à l'aide de K_1 puis le rechiffre avec K_2 pour le transmettre aux membres du sous-groupe SG_2. Cette procédure se poursuit jusqu'à ce que tous les membres du groupe global reçoivent et déchiffrent correctement P_i.

Lorsqu'un membre adhère au groupe, il est directement affecté à un sous-groupe prédéfini et reçoit la clef de déchiffrement correspondante. Seuls les membres de ce sous-groupe doivent mettre à jour leur clef de chiffrement de données. De même, pour la suppression d'un membre, seule la clef du sous-groupe doit être modifiée et transmise aux membres. Les secrets antérieurs et ultérieurs sont alors fournis et la propriété d'endiguement est respectée. Du point de vue performance, ce schéma reste le plus efficace en termes d'utilisation de ressources (de mémoire et de calcul). En effet, tous les membres ne possèdent qu'une clef de déchiffrement et ne doivent

effectuer qu'une seule opération de déchiffrement symétrique pour accéder aux données en clair.

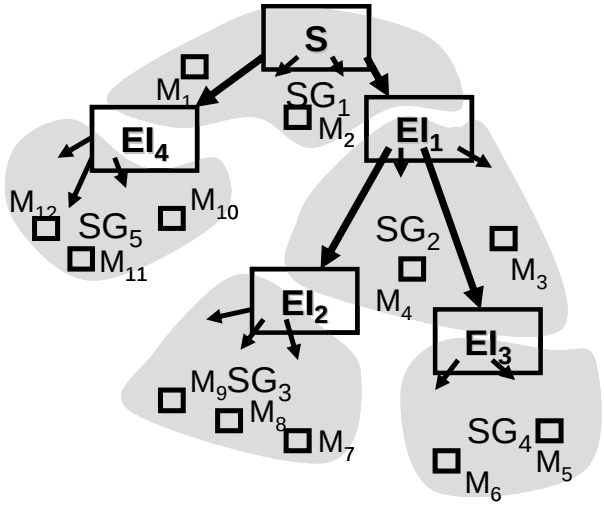

Figure 10.5. *Les arbres de rechiffrement : Iolus*

L'inconvénient majeur de ce schéma est celui de la confiance attribuée à tous les éléments intermédiaires. Ces derniers ont accès à toutes les données en clair puisqu'ils déchiffrent puis rechiffrent les données pour les transmettre à leur sous-groupe.

10.3.2.2. Les suites de chiffrement

Suivant le même principe des arbres de chiffrement, deux nouveaux schémas [MOL 00, PAN 02b] ont été proposés pour réduire la confiance dans les éléments intermédiaires. L'un est plus adapté à la distribution de clefs et l'autre à celle de données massives. Nous décrirons ici le schéma adapté au transport de données.

Dans le cas unicast, il existe un mode de chiffrement dit *counter-based* [BEL 97] qui permet de chiffrer des données en générant une suite pseudo-aléatoire d'octets qui sont combinés avec le texte en clair par un *ou exclusif* binaire. Ce mode de chiffrement est aussi celui que l'on utilise de façon typique avec les algorithmes de chiffrement en continu [MEN 96]. Soit f_a une primitive de chiffrement comme DES ou AES, opérant sur k bits et dont la clef est K_a. Le chiffrement et le déchiffrement d'un message M selon cet algorithme est décrit dans le tableau 10.2.

Chiffrement $E_a(\sigma,x)$	**Déchiffrement $D_a(\sigma,y)$**
$x = x_1x_2..x_n$	$y = y_1y_2...y_n$
pour $i = 1..., n$	pour $i = 1...,n$
$y_i = f_a(\sigma + i) \oplus x_i$	$x_i = f_a(\sigma + i) \oplus y_i$
retourner $(\sigma, y_1, y_2, ..., y_n)$.	*retourner* $x = x_1x...x_n$

Table 10.2. *Chiffrement et déchiffrement en mode* « *counter-based* »

Pour chiffrer un message x, la source utilise non seulement une primitive de chiffrement mais définit également un compteur σ qui est incrémenté à chaque fois qu'il est utilisé. La source applique un *ou exclusif* binaire pour chaque bit du message et une valeur du compteur.

La sécurité de cet algorithme a été justifiée dans [BEL 97]. Pour adapter cet algorithme de chiffrement aux communications en multicast, les auteurs du schéma [PAN 02b] proposent d'utiliser différentes primitives de chiffrement. Dans ce schéma illustré dans la figure 10.6, le chiffrement et le déchiffrement se font en l couches indépendantes les unes des autres. Chaque élément intermédiaire substitue une couche par une autre (une opération de chiffrement et une opération de déchiffrement). Les récepteurs déchiffrent les l couches pour accéder aux données. Les différentes clefs de couche ne sont connues que par le sous-ensemble de membres supposés déchiffrer au nœud précis.

Selon la figure 10.6, la source se trouvant à la racine de l'arbre applique à un message en clair X trois couches de chiffrement différentes avec les clefs respectives k_1, k_2 et k_3. Chaque nœud intermédiaire déchiffre premièrement le paquet reçu avec une des clefs de ces trois couches pour ensuite les rechiffrer avec une nouvelle clef. A l'arrivée du paquet, le membre possédant les clefs k_3, k_4 et k_5, déchiffre les trois couches correspondantes et peut ensuite accéder au message X en clair.

Cette approche permet d'avoir une forme d'endiguement similaire à ce que nous trouvons dans Iolus mais avec un avantage de taille : les éléments intermédiaires n'accèdent pas directement aux données, et se contentent de faire une transformation de celles-ci. Le coût de ce schéma en termes de ressources de calcul reste efficace car un membre utilise l opérations de déchiffrement qui sont basées sur l'utilisation de l'opération du *ou exclusif* binaire qui ne coûte rien.

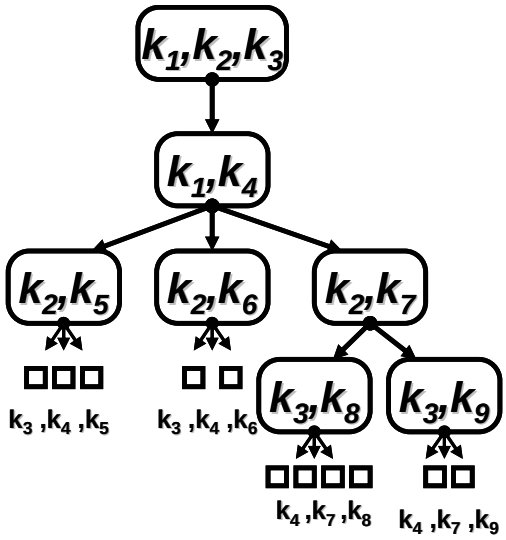

Figure 10.6. *Suite de chiffrements efficace en trois couches*

10.3.3. *Les arbres de hiérarchie de clefs logiques : LKH*

Les arbres de hiérarchie de clefs logiques [WAL 99, WON 98] ne prennent pas en compte la topologie du réseau. Dans ces schémas, la clef de groupe est la même pour tous les membres et doit donc être modifiée pour tous les membres à chaque changement de statut du groupe. Dans le cas du schéma LKH [WON 98], pour minimiser le coût de la mise à jour de clefs, la source définit un arbre logique de clefs hiérarchiques où la valeur de chaque feuille correspond à une clef secrète partagée entre un membre et la source et la valeur de la racine correspond à la clef de chiffrement des données. Un membre possède toutes les clefs appartenant au chemin à partir de la feuille à laquelle il correspond jusqu'à la racine (la clef du groupe).

D'après l'exemple de la figure 10.7, k_0 représente la clef de chiffrement des données. Au début de la session, le membre M_1 reçoit les clefs $\{k_o, k_1, k_3, k_7\}$. Pour révoquer un membre du groupe, toutes les clefs partagées avec d'autres membres du groupe doivent être modifiées. Pour cela, la source définit un processus de mise à jour des clefs. Par exemple, lorsque le membre M_1 quitte le groupe, la source met à jour les clefs k_3, k_1 et k_0 et envoie l'ensemble de messages suivant :

$$\{E_{k_8}(k_3'), E_{k_4}(k_1'), E_{k'_3}(k'_1), E_{k'_1}(k'_0), E_{k_2}(k'_0)\}$$

Pour ajouter un nouveau membre à l'arbre, la source ajoute une feuille. Les nœuds correspondant à ceux sur le chemin entre la feuille en question et la racine doivent encore une fois être modifiés.

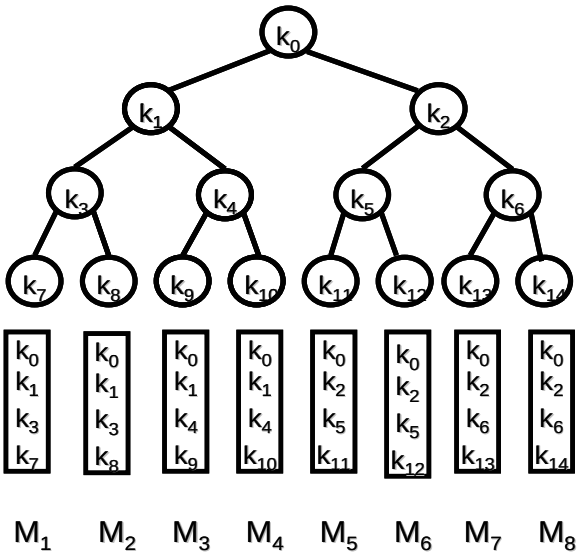

Figure 10.7. *LKH pour un groupe de 8 membres*

Cet algorithme a été prouvé optimal dans [SNO 01]. Du point de vue sécurité, la seule limite de ce schéma est l'endiguement.

Suite à la conception de LKH, beaucoup de travaux [LIX 01, SET 02, WON 00, YAN 01] ont analysé les faiblesses de robustesse du schéma et essayé de résoudre de nouveaux problèmes comme l'impact de la dynamicité du groupe ou la fiabilité du transport des clefs. En effet, lorsque les arrivées ou départs sont fréquents, la mise à jour individuelle de clefs reste inefficace. Les auteurs de [LIX 01] proposent de regrouper les opérations de gestion de groupe en un lot, pour ensuite le traiter en un intervalle. Le coût en nombre de messages se réduit alors de façon importante.

10.3.4. *Conclusion*

Les algorithmes de confidentialité pour les communications en multicast dépendent très fortement du choix du rôle des éléments du réseau. S'il existe des éléments intermédiaires, la source peut envisager d'implémenter les arbres de rechiffrement utilisant les algorithmes de chiffrement en mode *counter-based* pour réduire le problème d'endiguement. Dans le cadre des architectures où il y a peu d'éléments intermédiaires, comme les communications *via* satellite ou les applications dans lesquelles les éléments intermédiaires ne peuvent pas avoir un rôle actif dans le schéma de distribution de clefs, LKH reste le schéma le plus efficace, d'autant plus qu'il a été prouvé comme étant le schéma optimal.

10.4. Fiabilité des protocoles de distribution de clefs

10.4.1. *Besoins*

La plupart des applications multicast utilisent UDP comme couche de transport, avec parfois l'addition d'un protocole de fiabilité supplémentaire comme par exemple RTP. Cela implique que l'on n'a pratiquement aucune garantie de fiabilité sur le transport des données. Dans le cadre des applications 1-à-*N*, les applications souvent multimédias sont souvent construites pour tolérer des pertes ou des délais comme ceux que l'on peut observer dans un réseau non fiable. Se basant sur l'utilisation de ce type d'applications, il est nécessaire d'offrir une fiabilité parfaite pour le transport des messages de distribution de clefs.

LKH étant prouvé fournir un coût optimal, nous nous baserons sur ce schéma pour traiter les problèmes de fiabilité issus de la distribution de clefs. Ce schéma présente des propriétés particulières impliquant la forte nécessité d'un transport fiable des messages de distribution de clefs.

En premier lieu, à chaque opération de changement de clefs, tous les membres doivent au moins recevoir la nouvelle clef de chiffrement de données. Par ailleurs, la fiabilité du transport de toutes les clefs est un besoin important en raison de l'existence de deux types de dépendances entre les clefs :

– *dépendance des clefs d'un même intervalle* : lors d'un intervalle de mise à jour de clefs, LKH définit un schéma de distribution où les clefs nouvelles sont la plupart du temps chiffrées avec les clefs se situant dans les nœuds fils. Par conséquent, si un membre ne reçoit pas une clef de la chaîne, il ne peut pas avoir accès à la clef de chiffrement de données qui est définie à la racine de l'arbre. Si nous reprenons la figure 10.7 et l'exemple de la révocation du membre M_1, le membre M_2 ne peut pas

accéder à la nouvelle clef de chiffrement de données k'_0 sans avoir reçu les clefs k'_1 et k'_3 ;

– *dépendance des clefs entre plusieurs intervalles consécutifs* : selon les requêtes de départ ou d'arrivée au groupe, une clef définie pour un certain intervalle peut être utilisée pour chiffrer une autre clef mise à jour à l'intervalle précédent. Si nous reprenons le même exemple de la révocation du membre M_1, si le membre M_2 ne reçoit pas la clef k'_1 et si cette clef est utilisée lors d'une future mise à jour, M_2 doit contacter la source pour être synchronisé.

Il est donc indispensable pour un membre de recevoir la clef de chiffrement de données d'un certain intervalle avant la réception des données chiffrées avec cette clef. Pour cela, il existe deux classes de protocoles de transport de clefs proposés pour le schéma LKH. Ils sont respectivement basés sur l'utilisation des techniques de réplication de paquets et de codes de reconstruction.

10.4.2. Solutions basées sur des techniques de réplication

Les auteurs de [SET 02] définissent le schéma WKA-BKR qui se base sur la réplication des clefs les plus importantes. Deux phases sont définies pour ce protocole :

– *la phase de transmission* des clefs pendant laquelle les paquets contenant les clefs chiffrées sont générés selon l'algorithme WKA ;

– *la phase de retransmission* pendant laquelle la source génère de nouveaux paquets avec l'algorithme BKR en réponse aux acquittements négatifs émis par les récepteurs. Cette phase dure jusqu'à ce que tous les membres reçoivent toutes les clefs nécessaires.

Dans cette partie, nous ne décrirons que l'algorithme WKA, qui exploite la propriété que certaines clefs ont plus de valeur que d'autres. Plus la clef est près de la racine de l'arbre, plus le nombre de membres intéressés par cette clef augmente. A chaque intervalle de mise à jour de clefs, tous les membres ont besoin de recevoir au moins la clef de chiffrement des données qui est commune pour tous. L'idée de WKA est de définir un degré de réplication plus important pour les clefs se situant proche de la racine.

Dans cet algorithme, la source définit trois ensembles S_1, S_2 et S_3 regroupant des clefs selon leur degré d'importance. Pour établir une répartition des clefs dans chacun des ensembles, la source définit également deux valeurs de seuil h_1 et h_2 qui représentent la profondeur départageant les clefs de chaque ensemble. Les clefs dont la position est à une profondeur inférieure à h_1 sont regroupées dans l'ensemble S_1 et le reste dans l'ensemble S_2 (sauf les clefs des membres). Pour des raisons

d'efficacité les clefs de profondeur supérieure à h_1 mais inférieure à h_2 sont également regroupées dans S_3. Les clefs appartenant aux ensembles S_1 et S_3 sont distribuées plusieurs fois avec un degré de réplication prédéfini alors que celles de l'ensemble S_2 ne sont distribuées qu'une seule fois.

Selon la figure 10.7, la clef k_0 est celle qui est requise par tous les membres et donc celle qui présente le plus haut degré d'importance. Si pour cet arbre nous définissons $h_1 = 0$ et $h_2 = 2$ les trois ensembles seront définis ainsi : $S_1 = \{k_0\}$, $S_2 = \{k_1, k_2, k_3, k_4, k_5, k_6\}$ et $S_3 = \{k_1, k_2\}$.

L'efficacité de ce protocole se base sur la définition des valeurs h_1 et h_2 Augmenter ces deux valeurs augmente le coût de la communication dans la première phase de transmission de clefs. Mais cette augmentation réduit la probabilité qu'un membre perde ses paquets et donc le nombre d'acquittements négatifs et donc le coût de communication pendant la phase de retransmission du protocole. Il existe donc un dilemme pour la définition de ces deux valeurs. Pour définir une valeur optimale, la source doit pouvoir ajuster ces paramètres de façon à minimiser le coût de la communication.

Par ailleurs, la source doit également définir les degrés de réplication r_1 et r_2 pour chacun des ensembles S_1 et S_3. Cela dépend de la probabilité de perte du réseau et du nombre de membres d'un groupe. Pour un groupe de taille N et un réseau ayant une probabilité de perte p, la source envisageant de recevoir en moyenne au plus m messages d'acquittement négatifs, définit r par rapport à cette limite en suivant l'équation suivante :

$$m \geq Np^r \qquad [10.2]$$

10.4.3. *Solutions basées sur l'utilisation des codes de reconstruction*

Il existe des solutions offrant une fiabilité du transport des clefs en se basant sur l'utilisation des codes de reconstruction. Nous avons déjà décrit le principe de ces codes dans la section 10.2.3.3.

Les auteurs de [YAN 01] proposent un protocole dont le principe est similaire à WKA-BKR décrit dans la section précédente. Il existe deux phases dans ce protocole :

– dans la première phase, la source regroupe premièrement les données en blocs et définit ensuite un certain nombre de paquets de redondance pour chacun de ces blocs grâce aux codes de reconstruction ;

– dans la seconde phase, les membres n'ayant pas reçu un nombre suffisant de paquets pour reconstruire le bloc qui leur a été destiné, envoient à la source des messages d'acquittement négatifs. Dès la réception de ces derniers, la source déduit le nombre maximal a_{max} de paquets de redondance et transmet ces nouveaux paquets. Cette phase se termine lorsque tous les membres ont reçu tous leurs paquets.

Pendant la première phase, le nombre de paquets de redondance pour chaque bloc de données est déterminé selon la probabilité et le modèle des pertes dans le réseau. Durant la phase de retransmission, le nombre d'acquittements négatifs enrichit les informations nécessaires pour la détermination de ce nombre.

L'évaluation de cette quantité est indispensable d'une part pour assurer une fiabilité quasi-totale pendant la première phase de transmission et d'autre part pour utiliser efficacement les ressources de communication.

10.4.4. *Conclusion*

LKH étant prouvé fournir un coût optimal, plusieurs applications multicast visent à implémenter ce schéma. L'existence d'une dépendance très importante entre les clefs implique une forte nécessité d'un transport fiable des messages de distribution de clefs. Dans cette section, nous avons brièvement décrit les deux classes de protocoles fournissant de la fiabilité de transport qui sont basés respectivement sur des méthodes de réplication de paquets ou des techniques de codes de reconstruction. Le choix de la solution à implémenter et des paramètres correspondants dépendra très fortement de la nature de l'application et du réseau.

10.5. Conclusion générale

Les communications en multicast font naître de nouveaux besoins importants de sécurité en termes de confidentialité et d'authentification. Les solutions destinées à fournir ces deux propriétés pour les communications en unicast ne peuvent pas être utilisées dans le contexte de celles en multicast, non seulement pour des raisons d'efficacité mais surtout pour des raisons de sécurité.

En ce qui concerne l'authentification, les solutions actuelles pour les communications en multicast se regroupent principalement en deux catégories selon la nature de l'algorithme d'authentification utilisé. Les applications intégrant TESLA exigent une synchronisation sécurisée entre la source et chaque membre.

Celles qui utilisent les signatures numériques proposent de réduire leur coût en les générant sur un ensemble de paquets et non pour chaque paquet.

Pour la confidentialité, la plupart des solutions proposées se caractérisent par la présence ou non des éléments intermédiaires qui favorisent l'assouplissement du problème d'endiguement. Dans les applications où il n'y a pas d'éléments intermédiaires ou lorsque ceux-ci n'ont pas un rôle actif dans le protocole de distribution de clefs (ils ne font que transmettre les messages sans les modifier), LKH reste la meilleure solution. Certains travaux comme [ONE 04] proposent de reformuler le problème de l'endiguement spécifique au schéma LKH et définissent un degré de fidélité pour chaque membre dans le but de ne pas pénaliser les membres « fidèles » par la fréquence des arrivées et départs d'autres membres volatiles.

D'autres propriétés de LKH, comme la dépendance entre les clefs, impliquent une forte nécessité d'un transport fiable des messages de distribution de clefs. Les auteurs de [SET 02] et [YAN 01] proposent des protocoles de transport de clefs basés respectivement sur la réplication de paquet et les codes de reconstruction.

La sécurité des communications en multicast est donc un problème pour lequel il existe de nombreuses solutions, qui dépendent des besoins et de l'architecture de chaque cas. Chaque application a des besoins spécifiques, et, dans ce chapitre, nous avons passé en revue les différentes réponses possibles à la question de leur satisfaction.

10.6. Bibliographie

[BEL 97] BELLARE M., DESAI A., JOKIPII E., ROGAWAY P., « A concrete security treatment of symmetric encryption », *IEEE Symposium on Foundations of Computer Science*, 1997.

[CAN 99] CANETTI R., GARAY J., ITKIS G., MICCIANCIO D., NAOR M., PINKAS B., « Multicast security: A taxonomy and some efficient constructions », *Proceedings of IEEE Infocom,* 1999.

[DEE 89] DEERING S. E., « Host extensions for IP multicasting », *RFC 1112,* 1989.

[DEE 91] DEERING S. E., « Multicast Routing in a Datagram Internetwork », *PhD thesis*, Standford University, 1991.

[FEN 97] FENNER D., « Internet group management protocol, version 2 », *RFC 2236*, 1997.

[GOL 01] GOLLE P., MODADUGU N., « Authenticated Streamed Data in the Presence of Random », *Proceedings of NDSS,* 2001.

[LIX 01] LI X.S., YANG Y.R., GOUDA M.G., LAM S.S., « Batch rekeying for secure group communications », *Proceedings of the Tenth International World Wide Web conference,*p. 525-534, 2001.

[MEN 96] MENEZES A., VAN OORSCHOT P.C., VANSTONE S.A., « Handbook of Applied Cryptography », *CRC PRESS*, 1996.

[MIT 97] MITTRA S., « Iolus: A framework for scalable secure multicasting», *Proceedings of the ACM SIGCOMM,* Cannes, 1997.

[MOL 00] MOLVA R., PANNETRAT A., « Scalable multicast security with dynamic recipient groups », *ACM Transactions on Information and System Security*, 2000.

[ONE 04] ÖNEN M., MOLVA R., « Group Rekeying with a Customer Perspective », *Proceedings of ICPADS,* Newport Beach, California, 2004.

[PAN 02a] PANNETRAT A., MOLVA R., « Efficient Multicast Packet Authentication », *Proceedings of NDSS,* 2003

[PAN 02b] PANNETRAT A., MOLVA R., « Multiple layer encryption for multicast groups », *Sixth IFIP Communications and Multimedia Security Conference,* Portoroz, Slovénie, 2002.

[PAN 02c] PANNETRAT A., « Secure Multicast Communications », *PhD thesis*, Institut Eurécom, 2002.

[PER 00] PERRIG A., CANETTI R., TYGAR J., SONG D., « Efficient authentication and signing of multicast streams over lossy channels », *IEEE Symposium on Security and Privacy,* 2000.

[POS 80] POSTEL J., « User Datagram Protocol », *RFC 768,* 1980.

[QUI 01] QUINN B., ALMERMOTH K., « IP multicast applications : Challenges and solutions », *RFC 3170,* 2001.

[RIZ 97] RIZZO L., « Effective erasure codes for reliable computer communication protocols », *ACM CCR: Computer Communication Review,* 1997.

[ROH 99] ROHATGI P., « A compact and fast hybrid signature scheme for multicast packet authentication », *Proceedings of the 6th ACM conference on Computer and Communications Security,* Singapore, 1999.

[SCH 96] SCHULZRINNE H., CASNER S., FREDERICK R., JACOBSON V., « RTP: A transport protocol for real-time applications », *RFC 1889,* 1996.

[SET 02] SETIA S., ZHU S., JAJODIA S., « A Comparative performance analysis of reliable group rekey transport protocols for secure multicast », *Performance,* Rome, 2002.

[SNO 01] SNOEYINK J., Suri S., VARGUESE G., « A lower bound for multicast key distribution », *IEEE Infocom,* Anchorage, 2001.

[WAL 99] WALLNER D.M., HARDER E.J., AGEE R.C., « Key management for multicast: Issues and architectures », *RFC 2627,* 1999.

[WON 98] WONG C.K., GOUDA M., LAM S.S., « Secure group communications using key graphs », *ACM SIGCOMM*, p. 68-79, 1998.

[WON 99] WONG C.K., LAM S.S., « Digital signatures for flows and multicasts », *IEEE/ACM Transactions on Networking*, 1999.

[WON 00] WONG C.K., LAM S.S., « Keystone: a group key management system », *Proceedings of International Conference in Telecommunications*, 2000.

[YAJ 96] YAJNICK M., KUROSE J., Tosley D., « Packet loss correlation in the Mbone multicast network », *IEEE Global Internet Conference*, Londres, 1996.

[YAN 01] Yang Y.R., LI X.S., ZHANG B., LAM S.S., « Reliable group rekeying: A performance analysis », *ACM SIGCOMM,* 2001.