Securing multicast communications in satellite networks: A user-centric approach

Suna Melek Önen

Thèse

présentée pour obtenir le grade de docteur

de l'Ecole Nationale Supérieure des Télécommunications

Spécialité : Informatique et Réseaux

Suna Melek Önen

La sécurité des communications multipoints pour les réseaux satellitaires:

Une approche centrée sur la satisfaction des utilisateurs

soutenue le 5 juillet 2005 devant le jury composé de

Pascal Urien	**Président**
Abdelmadjid Bouabdallah	**Rapporteurs**
Ludovic Mé	
Ernst Biersack	**Examinateurs**
Yves Deswarte	
Laurence Duquerroy	**Invitée**
Refik Molva	**Directeur de thèse**

Ecole Nationale Supérieure des Télécommunications

PhD thesis

Ecole Nationale Supérieure des Télécommunications

Computer Science and Networks

Suna Melek Önen

Securing multicast communications in satellite networks:

A user-centric approach

Defense date: July, 5th 2005

Committee in charge:

Pascal Urien	**Chairman**
Abdelmadjid Bouabdallah	**Reporters**
Ludovic Mé	
Ernst Biersack	**Examiners**
Yves Deswarte	
Laurence Duquerroy	**Invited**
Refik Molva	**Advisor**

Ecole Nationale Supérieure des Télécommunications

16 sene sonra sevgili Babama
ve / et
à ma Mère

Acknowledgments

At the end of my three and a half years of study, I will try again to “take some distance” as my supervisor Professor Refik Molva always suggested me and maybe sometimes go into details in order to thank everybody that really helped me during this period.

First and foremost, I would like to thank my thesis advisor Professor Refik Molva for his guidance, encouragements and constructive criticism. I learned a lot from him especially about writing and presenting technical papers.

I am also grateful to AlcatelSpace who partially funded this thesis and in particular to Sébastien Josset who gave me good knowledge about satellite networks. I also would like to thank Isabelle Buret and Laurence Duquerroy.

I would like to thank all the members of my thesis commitee for their feedback and discussion : Abdelmadjid Bouabdallah, Ludovic Mé, Yves Deswarte, Pascal Urien and special thanks go to Professor Ernst Biersack since although he was an examiner his feedbacks were those of a reviewer.

I wish to express my gratitude to Professor Claude Girault for his never ended support and guidance during the whole period beginning from my Masters studies.

I also would like to thank Institut Eurécom for giving me the opportunity of performing my research in very good conditions. I highly appreciate the unlimited help of Gwenaelle Le Stir, Delphine Gouaty, Anne Duflos and Christine Mangiapan who took care of all the administrative stuff in the organization of the defense.

Thanks to Institut Eurécom, I also had the opportunity to make good friends from all over the world. I am grateful to Serkan Yeşildağ, Gaurav Mudgal, Anwar Al Hamra, Fabien Pouget, Walid Bagga and Matti Siekkinen with whom I shared my office and had great times, Jerôme Haërri whom I consider to be part of the office and all the members of the NS-TEAM. I would like to thank Ahmet Başt˜g for the fruitful discussions that we often shared. I also do not forget to thank my friends from Galatasaray University and especially Gülfem Işıklar who is also preparing her PhD degree, for her support.

I owe special thanks to Cédric whose support, patience and love had no limit during this long and sometimes hard period. I guess he now knows my thesis by heart.

Last, but not least, I am indebted to my parents and my sisters Esra and Zehra. They were always encouraging and supporting me while being physically far far away.

Résumé

Introduction

Les applications multicast

L'IP multicast [19] 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 pour 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.

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

- **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.
- **les applicaitons N-à-N** : pour les applications de cette catégorie, tous les récepteurs peuvent avoir le rôle d'émetteur.
- **les applications N-à-1** : il s'agit des applications où plusieurs sources envoient des données à un unique ou à quelques récepteurs.

Dans cette thèse, nous nous intéressons plus particulièrement aux applications de la première catégorie en raison de leurs étroites liaisons avec le commerce et leurs exigences en terme de sécurité.

Les réseaux satellitaires

Les applications de type 1-à-N peuvent être fournies soit par la voie terrestre, soit par la voie satellitaire. Les réseaux satellitaires présentent des avantages considérables pour les communications multipoints comme leur couverture géographique large. Nous présentons dans cette section les caractéristiques des réseaux satellitaires et les différentes architectures.

Un réseau de télécommunications par satellite est constitué de deux parties distinctes : **le segment sol** et **le segment spatial**.

Le segment sol est principalement composé des équipements terrestres que nous appellons terminaux satellites. Ces terminaux peuvent être directement connectés aux équipements d'utilisateurs. Le réseau des terminaux satellites peut avoir deux principales architectures :

- **architecture en étoile** : les échanges se font par l'intermédiaire d'une station centrale appelée **passerelle**. Ces échanges sont alors le plus souvent dissymétriques, avec des débits d'information élevés de la passerelle vers les autres terminaux, et des débits plus faibles des terminaux satellites vers la passerelle.

- **architecture maillée** : les échanges se font directement entre terminaux et le plus souvent de façon symétrique.

Le segment spatial constitué du seul satellite géostationnaire est composé d'une porteuse montante ("uplink") et d'une porteuse descendante ("downlink"). Le satellite réceptionne les ondes radioélectriques de la porteuse montante, les transpose et les amplifie pour les transmettre aux terminaux satellites à travers la porteuse descendante. Un satellite géostationnaire est dit **transparent** si celui-ci ne met en place aucune fonctionnalité d'un niveau autre que physique. La plupart du temps un tel système se contente de répéter les signaux vers la Terre, en les amplifiant. Les satellites traditionnels sont le plus souvent transparents et proposent des solutions simples qui permettent de répéter le signal sur les faisceaux descendants. Toutefois, pour améliorer les performances des systèmes, de nouveaux satellites dits **regénératifs** intégrent la technologie OBP et offrent des capacités de traitement à bord.

Le satellite géostationnaire est considéré depuis sa création comme un support privilégié pour les communications multipoints. Malgré des coûts d'investissement élevés, ses avantages les plus significatifs sont sa grande surface de couverture (à l'échelle d'un continent) et l'absence de noeuds intermédiaires (la communication s'effectue en un ou deux bonds au maximum).

Par ailleurs, la capacité à diffuser des informations sous forme de paquets de données à un nombre élevé de terminaux physiquement distants est à la fois son plus fort atout face aux réseaux terrestres et un point faible au regard de la sécurité. En effet, en raison de cette caractéristique inhérente de diffusion, n'importe quel utilisateur peut recevoir un paquet transmis depuis le satellite, même s'il n'est pas membre du groupe multicast. De plus, le délai de transmission de bout-en-bout dans les réseaux satellitaires est important face à celui des réseaux terrestres et celui-ci est incompressible. Ces deux inconvénients doivent donc être pris en compte dès la conception des solutions de sécurité destinées aux réseaux satellitaires.

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 d'implémenter des mécanismes de sécurité pour restreindre l'accès au contenu distribué en multicast pour une durée déterminée aux seuls clients légitimes de l'application. Nous proposons d'analyser les besoins spécifiques des applications commerciales à large échelle en étant convaincus que les solutions apportées pour ces applications peuvent être utilisées pour des applications dont le besoin en sécurité est plus souple.

Les principaux besoins des applications multicast en terme de sécurité peuvent être résumés en trois points :

- **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 récepteur doit également pouvoir s'assurer que les données reçues n'ont pas été modifiées pendant leur transmission (on parle alors d'intégrité).
- **la confidentialité** : seuls les client autorisés doivent avoir accès au contenu des données et cet accès n'est autorisé que pour la durée convenue préalablement.
- **la protection contre le déni de service** : les attaques de déni de service ont pour but d'empêcher les clients légitimes d'accéder au service. Le fournisseur de service doit donc pouvoir minimiser l'impact de ce type d'attaques sur les clients légitimes.

Ces trois besoins en terme de sécurité ne sont pas nouveaux et plusieurs solutions ont déjà été proposées et sont utilisées dans le cadre des communications unicast. Cependant les solutions actuelles ne peuvent pas directement être appliquées dans le cadre des communications en multicast pour les réseaux satellitaires pour des raisons décrites dans la prochaine section.

Limites des solutions actuelles

Authentification

Les algorithmes fournissant un service d'authentification se différencient selon les propriétés des fonction de génération et de vérification de l'empreinte. Les *signatures numériques* [64] sont basées sur des algorithmes à clef publique : les fonctions de génération et de vérification utilisent différents paramètres d'entrée. Pour les *codes d'authentification de message*, les fonctions de génération et de vérification sont basées sur une même clef secrète.

Dans le cadre 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 préalablement d'accord sur une clef secrète qui est ensuite utilisée pour la génération d'empreintes grâce aux codes d'authentification de message (MAC) [7]. 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.

Confidentialité

La confidentialité des données est 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. Un algorithme de chiffrement est *symétrique* (ou à clef secrète) si la clef utilisée pour le chiffrement est identique à celle utilisée pour le 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.

Les algorithmes de chiffrement à clef publique sont souvent très coûteux en termes de mémoire et de calcul. Ils 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 tous les paquets échangés entre ces deux entités sont chiffrés et déchiffrés avec K jusqu'à la fin de la communication. Dans le cadre des communications multipoints, 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 du groupe multicast. La distribution de cette clef doit être efficace : 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, le coût de la distribution de la nouvelle clef doit également être optimisé.

Protection contre le déni de service

Les attaques de déni de service visent à empêcher les entités légitimes d'accéder aux services auxquels elles ont droit dans des circonstances normales. Les solutions actuelles de protection contre le déni de service deviennent inefficaces dans le contexte des réseaux satellitaires pour deux raisons principales : leur caractéristique de diffusion naturelle et le délai bout-à-bout qui est assez important. En effet, les techniques de filtrage sont la plupart du temps trop coûteuses pour ces réseaux. La technique anti-clogging basée sur un échange d'informations courtes (les cookies) ne peut également pas être utilisé dans le cadre des réseaux satellitaires. En effet, les cookies sont transmis uniquement à l'adresse de destination de la requête pour qu'un serveur puisse s'assurer de l'existence d'une entité légitime crorespondant à cette adresse. Or, de part la capacité native de diffusion des réseaux satellitaires, un intrus peut avoir accès aux cookies et donc réussir son attaque.

Contributions et présentation de la thèse

Dans cette thèse, nous analysons en premier lieu les problèmes majeurs de sécurité dans le cadre des communications multipoints et proposons des solutions destinées aux réseaux satellitaires. Le problème de l'authentification étant globalement analysé et plusieurs solutions proposées pouvant être implémentées sans modification pour les réseaux satellitaires, nous nous intéresserons principalement aux problèmes de la confidentialité multicast et de la protection contre le déni de service.

Les solutions actuelles offrant de la confidentialité multicast traitent seulement les aspects en terme de sécurité et de passage à l'échelle. Ces solutions souffrent encore des problèmes liés à la fiabilité. En effet, les protocoles de distribution de clefs présentent des propriétés et besoins particuliers qui ne sont pas adressés par les protocoles de fiabilité actuels [6, 26, 37, 51, 34, 73]. La perte d'une seule clef peut provoquer la suppression définitive d'un membre du groupe sans son accord. Nous nous intéressons donc premièrement au problème de la fiabilité de la distribution de clefs. Par ailleurs, dans la plupart des protocoles de renouvellement de clefs, chaque opération de renouvellement de clefs nécessite la mise à jour du matériel pour chacun des membres du groupe. En se basant sur cette observation, nous définissons une nouvelle approche, "la

satisfaction des utilisateurs", où le serveur partitionne les membres selon un critère prédéfini (durée d'abonnement, rôle hiérarchique, etc.) et offre une meilleure fiabilité pour les membres définis comme étant privilégiés.

En tenant compte de ces deux nouveaux besoins cruciaux qui sont la fiabilité et la satisfaction des utilisateurs, nous proposons deux protocoles de renouvellement de clefs adaptés aux réseaux satellitaires qui tiennent compte du comportement des membres du groupe et favorisent les membres privilégiés lors de l'opération de renouvellement de clefs.

Par ailleurs, nous analysons les solutions actuelles de protection contre le déni de service destinées aux réseaux terrestres et montrons leurs limites lors de leur implémentation pour les réseaux satellitaires. Enfin, nous proposons une solution originale qui permet à la source d'identifier les attaques de déni de service presque immédiatement, ceci limitant ainsi l'impact de ces attaques sur les clients légitimes.

Confidentialité multicast et renouvellement de clefs

Besoins

La confidentialité 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 prédéfini doivent avoir la possibilité d'accèder aux données en clair. 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. Dans la plupart des applications multicast, il est fort probable que le groupe défini ait un statut dynamique dans lequel les arrivées et départs de membres s'observent de façon fréquente. Nous nous intéressons aux groupes de ce type. Les solutions offertes pour ce type d'applications peuvent être implémentées pour des groupes ayant un statut statique.

Nous regroupons les besoins d'un schéma de renouvellement de clefs en quatre catégories:

- **sécurité** : de nouveaux problèmes de confidentialité se posent lors du passage à un contexte multicast où la dynamicité du groupe impose un besoin de renouvellement de clefs selon des arrivées ou des départs;
- **passage à l'échelle** : pendant l'implémentation d'un protocole de renouvellement de clefs, toutes les ressources liées au contexte multicast doivent être optimisées;
- **fiabilité** : un membre doit obligatoirement recevoir son matériel de clefs pour pouvoir déchiffrer les données chiffrées;
- **satisfaction des utilisateurs** : le serveur de clefs doit minimiser l'impact de la fréquence des opérations de renouvellement de clefs sur les membres privilégiés.

Sécurité

Pour pouvoir offrir un accès aux données seulement pour les clients qui y sont autorisés, la source doit utiliser des algorithmes de chiffrement symétrique. Dans le contexte des communications multipoints, le groupe multicast étant considéré avoir un statut dynamique et les arrivées et départs de membres étant très fréquents, la source doit pouvoir assurer les deux propriétés suivantes :

- **la confidentialité antérieure** est assurée 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.
- **la confidentialité ultérieure** est fournie 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 confidentialités antérieure et ultérieure 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.

Par ailleurs, un autre facteur souvent pris en compte dans le cadre des communications sécurisées est celui de 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.

Enfin, certains protocoles de sécurité peuvent faire intervenir des éléments intermédiaires du réseau situés dans le chemin entre la source et les récepteurs. Le degré de confiance attribué aux éléments intermédiaires est un autre facteur important de sécurité.

En résumé, le serveur de clefs doit pouvoir assurer les propriétés suivantes en terme de sécurité :

- la confidentialité,
- la confidentialité antérieure et la confidentialité ultérieure,
- la résistance aux collusions,
- la confiance minimale attribuée aux différents composants du réseau.

Passage à l'échelle

Contrairement aux techniques classiques, les nouvelles solutions offrant de la confidentialité pour les communications de groupe doivent tenir compte des critères algorithmiques. Quelle que soit la solution offerte, la source doit être capable de gérer un très grand nombre de clients. Elle doit également être capable de traiter les arrivées ou départs en masse sans réduire l'efficacité de la communication. La solution offerte doit donc tenir compte des coûts suivants:

- **coût de calcul** : dans la plupart des applications, les données étant transmises en masse et en temps réel, les opérations de chiffrement et de déchiffrement des données ne doivent pas imposer une utilisation importante du CPU. Il en va de même pour le calcul exigé par le schéma de distribution de clefs.

- **coût de communication** : les mécanismes de sécurité ne doivent pas imposer une duplication du contenu ; la souce ne peut pas se permettre de définir une clef de chiffrement pour chaque membre du groupe, de chiffrer un paquet multicast avec chacune de ces clefs et de le transmettre par conséquent un grand nombre de fois. De plus, le protocole de distribution de clefs ne doit pas également saturer la bande passante.

- **coût de mémoire** : l'architecture doit optimiser l'utilisation des ressources mémoire qui sont considérées comme limitées non seulement du côté des membres mais également pour la source.

Fiabilité

La plupart des applications 1-à-N tolèrent les pertes du réseau. Les protocoles de renouvellement de clefs présentent des propriétés spécifiques en terme de fiabilité qui ne sont pas prises en compte par les protocoles de fiabilié multicast actuels. En premier lieu, nous observons que lorsque le serveur de clefs envoie un certain nombre de clefs au groupe, chaque membre n'utilise qu'un sous ensemble de ces clefs. De part cette propriété deux nouveaux besoins se définissent en terme de fiabilité :

- **la fiabilité éventuelle** : un membre doit pouvoir recevoir toutes les clefs nécessaire pour déchiffrer les données;

- **la fiabilité temps-réel** : la distribution de clefs d'un certain intervalle doit être terminée avant le début du prochain intervalle.

Si le serveur de clefs n'assure pas ces deux besoins, un membre pouvant perdre au moins une clef peut être automatiquement éliminé du groupe et par conséquent n'aura pas accès aux données alors qu'il y est autorisé.

Satisfaction des utilisateurs

Un aspect particulièrement important qui est souvent négligé par les protocoles de sécurité multicast est le fait que la gestion des arrivées et départs affecte le groupe entier. Cet aspect renforce les besoins en terme de fiabilité. En effet, une arrivée ou un départ d'un membre imposant un renouvellement de clefs pour tous les membres du groupe, le serveur de clefs devrait au moins fournir une forte fiabilité pour des membres supposés rester dans le groupe durant toute la session multicast. En se basant sur cet exemple, nous proposons une nouvelle approche qui tient compte du comportement des membres pour fournir un meilleur service pour les membres privilégiés.

Etat de l'art

Dans le chapitre 1, nous avons regroupé les solutions offrant de la confidentialité en deux catégories selon le rôle attribué aux noeuds intermédiaires. 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. Le schéma le plus représentatif de la première catégorie est nommé Iolus [44] : 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 différente clef de chiffrement. Les éléments intermédiaires se chargent de déchiffrer les données reçues pour ensuite les chiffrer avec la clef définie pour leur sous-groupe et transmettre ces données chiffrées aux membres de leur sous-groupe.

Les arbres de hiérarchie de clefs logiques [78] nommées LKH ne tiennent pas compte de la topologie du réseau et n'attribuent aucun rôle aux éléments intermédiaires. Dans ce schéma, 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 de LKH, pour minimiser le coût de renouvellement de clefs, la source définit un arbre logique de clefs hiérarchiques où la valeur attribuée à la racine de l'arbre 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).

La catégorisation des solutions de confidentialité est basée sur leur efficacité en terme de sécurité et de passage à l'échelle. Les critères de fiabilité et de satisfaction des utilisateurs n'ont pas été pris en compte lors de la conception des solutions actuelles. Par ailleurs, dans le cadre des architectures où il y a peu d'éléments intermédiaires, comme les communications via satellite, LKH reste le schéma le plus efficace d'autant plus qu'il a été prouvé comme étant le schéma optimal [71]. Dans cette thèse, nous nous sommes donc plus particulièrement intéressés à ce schéma et l'avons analysé par rapport à ces nouveaux critères. Suite à cette analyse, nous avons proposé de nouveaux schémas fiables de distribution de clefs tenant compte du comportement des utilisateurs.

Renouvellement de clefs fiable basé sur la satisfaction des utilisateurs

Analyse de LKH : besoins en fiabilité et satisfaction des utilisateurs

LKH étant prouvé optimal, nous nous basons 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. LKH impose une dépendance forte entre les clefs de deux types :

- **une dépendance inter-intervalle** : selon les requêtes de départ ou d'arrivé au groupe, une clef définie pour un certain intervalle peut être utilisée pour chiffrer une clef future. Par conséquent, si un membre reçoit une nouvelle clef chiffrée avec une clef antérieure qu'il n'a pas reçue, il ne peut plus avoir accès à la nouvelle clef ni aux clefs futures. LKH présente donc une **dépendance temporelle** entre les clefs.

- **une dépendance intra-intervalle** : lors d'un intervalle de renouvellement de clefs, les nouvelles clefs sont la plupart du temps chiffrées avec les clefs se situant dans les noeuds fils. Par conséquent, si un membre ne reçoit pas une clef de l'arbre, il ne pourra pas avoir accès à la clef de chiffrement de données. LKH présente donc une **dépendance spatiale** entre les clefs.

Il existe deux classes de protocoles de distribution fiable de clefs qui ont été proposées pour LKH. Elles sont respectivement basées sur l'utilisation des techniques de réplication de paquets et de code de reconstruction. Les auteurs de [69] définissent le schéma WKA-BKR qui se base sur la réplication des clefs les plus importantes. Les auteurs de [81] proposent un schéma de fiabilité basé sur l'utilisation des codes de reconstruction proactifs. Le choix de la solution à implémenter et de ses paramètres dépend très fortement de la nature de l'application et du réseau.

Contrairement aux propriétés de sécurité, de passage à l'échelle et de fiabilité qui ont été étudiées par certains protocoles de renouvellement de clefs fiables, les besoins de la vie réelle comme la satisfaction des utilisateurs sont la plupart du temps négligés et pas prises en considération lors de la conception de ces protocoles. Or, la fréquence des opérations de renouvellement de clefs étant supposée très importante, ces dernières peuvent avoir un impact important sur tous les membres sans tenir compte de leur comportement. Le serveur de clefs doit au moins pouvoir assurer une meilleure fiabilité pour les membres privilégiés. La notion de "membre privilégié" dépend de la nature des applications et de leurs besoins spécifiques notamment en fonction du mécanisme de tarification. Dans le chapitre 3, nous proposons premièrement de différencier les applications sécurisées selon leur besoins en terme de sécurité et leur politique de tarification. Nous définissons ensuite la notion de "membre privilégié" pour chacune de ces catégories.

Classification des membres par rapport à leur durée de présence

Dans une première classification décrite dans le chapitre 3, le serveur de clefs différencie les membres par rapport à leur durée de présence dans le groupe. Cette classification résulte des observations faites par Almeroth [4] durant des sessions multicast. Les auteurs concluent que les membres quittent le groupe soit après un très court délai après leur arrivée soit à la fin de la session. Les membres forment deux catégories réelles selon leur durée de présence :

- les membres de durée de présence courte ;
- les membres de durée de présence longue.

Le serveur de clefs ne pouvant pas définir quel membre appartient à quelle catégorie, il définit une classification expérimentée en se basant sur un seuil prédéfini. Les membres sont alors classifiés par rapport à ce paramètre en deux ensembles :

- les membres dits **volatiles** sont ceux dont la durée de présence est inférieure au seuil prédéfini;
- les membres dits **permanents** sont ceux dont la durée de présence est suprérieure ou égale au seuil prédéfini.

Grâce à cette nouvelle classification, le serveur de clefs peut minimiser l'impact des opérations de renouvellement de clefs causées par les membres **volatiles** sur les membres **permanents** et leur assurer une meilleure fiabilité. En effet, pour assurer un meilleur service aux membres permanents, le serveur de clefs définit un arbre logique de clefs pour chaque catégorie. Un nouveau membre adhère à l'arbre des membres **volatiles** et lorsque sa durée de présence excède le seuil prédéfini, il est transféré à l'arbre des membres **permanents**. La mise à jour de chaque arbre se fait indépendamment l'un de l'autre. Les membres **permanents** étant supposés ne pas quitter le groupe, leur arbre est supposé avoir un statut quasi-statique. En se basant sur cette observation, l'intervalle de mise á jour de l'arbre représentant les membres **permanents** est plus long que celui de l'arbre des membres **volatiles**.

Par ailleurs, la clef de chiffrement de données étant commune pour tous les membres du groupe, elle doit par conséquent être modifiée à chaque mise à jour de l'arbre des membres volatiles. Pour gérer indépendemment les deux arbres définis et empêcher les membres permanents d'être affectés par la mise à jour de l'arbre des membres **volatiles**, nous proposons un algorithme de génération de clefs de chiffrement de données pour les membres **permanents** lorsque leur arbre n'est pas mis à jour. Par conséquent, lors du renouvellement de la clef de chiffrement de données, si l'arbre des membres permanents n'est pas mis à jour, ces derniers calculent la nouvelle clef de chiffrement de données sans recevoir aucune information du serveur.

Protocole de fiabilité pour les membres permanents

Pour pouvoir assurer un minimum de confidentialité antérieure et ultérieure, l'arbre des membres permanents doit être mis à jour. Cette mise à jour est évidemmnet moins fréquente que celle de l'arbre des membres volatiles. Pendant ces mises à jour, le serveur de clefs doit s'assurer de la réception des messages de renouvellement de clefs par les membres permanents. Nous proposons donc un nouveau protocole de fiabilité pour les membres permanents où le serveur définit deux paramètres α et β tel que α représente la proportion des membres permanents recevant leur clefs avec une probabilité minimale β. Le schéma de fiabilité combine les méthodes de codes de reconstruction et réplication proactives.

L'algorithme de construction de bloc FEC proposé utilise la hiérarchie définie dans LKH. Comme les membres d'un même sous-arbre partagent un certain nombre de clefs, nous proposons de diviser l'arbre en sous-arbres et de regrouper les clefs de chacun de ces sous-arbres dans un bloc. La taille du bloc dépendra alors de la profondeur des sous-arbres. Le bloc regroupera alors :

- la clef de chiffrement de données ;
- les clefs se situant entre la racine de l'arbre global et celle du sous-arbre (elles ne sont chiffrées avec un seule clef et partagées par tous les membres) ;
- les clefs se situant dans le sous-arbre (seul un sous-ensemble de ces clefs est nécessaire pour chaque membre).

Avec cette méthode de regroupement de clefs, certaines clefs vont apparaître dans plusieurs blocs. Par conséquent, la profondeur des sous-arbres va définir le paramètre de réplication des

clefs. Par ailleurs, le nombre de paquets de redondance pour chaque bloc se définit par rapport aux paramètres α et β et la probabilité de perte du réseau.

Classification des membres selon leur temps d'arrivée

Nous proposons également une deuxième méthode de classification pour des sessions à courte durée où la tarification des applications est basée sur le temps d'arrivée des membres. Les membres sont supposés s'enregistrer juste après le début de la session. Pour éviter les problèmes de pertes dues aux arrivées abondantes, nous proposons ici une nouvelle méthode de restructuration de l'arbre des clefs selon le moment d'arrivée des membres.

Selon les résultats d'observation des sessions courtes, il a été réalisé que la majorité des nouveaux membres s'enregistrent au groupe juste après le début de la session et sont censés ne pas quitter le groupe jusqu'à la fin de la session. En se basant sur ces résultats, nous modélisons le temps d'arrivée des membres par une distribution Zipf [85] qui par définition représente le fait qu'une grande proportion des échantillons sont concentrés au début d'une échelle et que le reste des membres sont dispersés dans le reste de l'échelle.

Suite à cette modélisation, nous proposons de construire un nouvel arbre de clefs à chaque intervalle de renouvellement de clefs. Chaque arbre regroupe seulement les membres arrivés à l'intervalle correspondant. Ce protocole présente trois avantages :

- comme les nouveaux arrivés sont regroupés dans un même arbre de clefs, lors de l'opération de renouvellement de clefs, les membres enregistrés préalablement ne recevront qu'une seule clef : la clef de chiffrement de données;
- de façon similaire, la mise à jour de certains arbres ne provoque le changement que d'une seule et unique clef (la clef de chiffrement des données) pour les membres ne faisant pas partie des arbres en questions;
- enfin, le fait de regrouper les membres dans différents arbres réduit la profondeur de ceux-ci et donc leur coût de mise à jour.

Suite à de nombreuses et différentes simulations, nous avons conclu deux résultats majeurs. Pendant la période proche du début de la session multicast, la structuration de l'arbre n'ajoute pas de coût supplémentaire par rapport à l'utilisation du schéma LKH classique. Par conséquent, il est plus efficace d'implémenter ce protocole pour éviter que des membres soient affectés par l'arrivée d'autres membres. Nous avons également implémenté une méthode de réplication pour assurer la fiabilité du transport de clefs et nous avons conclu que le coût de notre protocole est inférieur à celui du schéma classique LKH.

Protection contre le déni de service

Les attaques de déni de service ont pour but d'empêcher une entité légitime d'accéder à un service auquel elle y est autorisée dans des circonstances normales. Pour cela, les attaques visent à consommer les ressources limitées de la cible, pour éviter le déroulement du service en cours. Les principales ressources ciblées sont :

- la mémoire,
- la bande passante,
- le CPU.

La première approche pour protéger le réseau contre ce type d'attaques serait d'éliminer au plus vite les requêtes provenant des entités non identifiées. Pour cela, il est nécessaire d'authentifier l'origine de chaque paquet reçu. Cependant, cette approche ne résout pas le problème du déni de service car les opérations cryptographiques utilisées pour l'authentification coûtent très cher en terme de CPU et de mémoire. Par conséquent, il faudra définir un protocole d'authentification efficace en terme de consommation de CPU, de mémoire et de bande passante.

Dans le cadre des communications terrestres, le protocole de sécurité IPSEC [45], utilise la technique anti-clogging [33] avant le traitement intensif du paquet reçu. Cette technique est basée sur un échange de cookies qui sont des informations générées de façon efficace en fonction d'une clef secrète locale (connue par le générateur) et les adresses des communicants. En effet, au début de chaque connection client-serveur, un échange de cookies est faite avant toute réservation de ressources intensives. Ce cookie est le résultat d'une fonction de hachage à clef contenant les identités des entités, une information sur le temps pour empêcher le rejeu et la clef secrète de l'entité générant le cookie. Lorsque le serveur reçoit une requête d'un client, il calcule un cookie dépendant de l'adresse figurant dans l'entête du paquet IP. Sans aucune allocation de ressources, le serveur renvoie ce cookie à l'adresse en question. Si ensuite il reçoit une requête avec le cookie, après la vérification de ce dernier, le traitement du paquet et les allocations nécessaires peuvent s'effectuer.

Dans le cadre des réseaux satellitaires, cette technique ne peut pas être utilisée en raison de la capacité native de diffusion du système : les cookies peuvent être interceptés par toute entité. Un intrus ayant envoyé un paquet avec une identité falsifiée peut poursuivre l'envoi de ce paquet après avoir intercepté le cookie nécessaire provenant du serveur. De plus, le délai de bout-en-bout étant très important pour un réseau satellitaire, l'échange des cookies engendre une latence très longue.

Proposition d'une méthode d'authentification efficace

Après avoir étudié les méthodes de protection contre les attaques de déni de service dans le cadre des réseaux terrestres et dressé leurs avantages et inconvénients, nous avons proposé dans le chapitre 7 un nouveau protocole d'authentification efficace entre un terminal et le serveur en tenant compte des caractéristiques du réseau satellite. Il s'agit d'un algorithme de vérification incrémentale à plusieurs étapes (ici deux) où à chaque étape la probabilité qu'un message est légitime augmente et que le taux de messages falsifiés diminue. Puisque le nombre de messages à vérifier diminue, le serveur peut se permettre d'augmenter le taux d'utilisation de ressource pour chaque étape suivante.

Cette méthode tire profit de l'existence d'un contrôle centralisé dans les réseaux satellitaires. En effet, chaque terminal partage une clef avec le serveur, facilement calculable du côté de celui-ci. Par ailleurs le protocole se définit par une communication à sens unique où un terminal envoie un seul message (sa requête) contenant les informations d'authentification également.

Grâce à cette propriété l'utilisation de la bande passante et le délai de traitement de requête sont optimisés.

La requête est un message dans lequel le terminal ajoute deux champs à son entête. Le premier champ est le résultat d'une fonction de hachage regroupant la clef du terminal, le nonce correspondant à l'intervalle d'envoi et le hachage du message qui également constitue le deuxième champ de l'entête. A la réception d'une requête envoyée par un terminal, le serveur procède à la vérification de son premier champ qui lui permet de s'assurer de l'identité du terminal et de l'absence de rejeu du message. Si cette vérification échoue, la requête n'est pas traitée et est immédiatement rejetée. Dans le cas contraire, une deuxième vérification est nécessaire pour s'assurer de l'intégrité du message. Le deuxième champ de la requête représente le résultat d'une fonction de hachage sur le message. Dans la deuxième étape, le serveur calcule donc cette valeur et la compare avec la valeur correspondante de l'entête.

Les calculs des fonctions de hachage n'étant pas des opérations coûteuses, ce processus de vérification à deux étapes permet au serveur de rejeter les requêtes falsifiées le plus rapidement possible.

Nous proposons également une deuxième version de ce protocole où lorsque le serveur reçoit et vérifie avec succès une certaine requête provenant d'un terminal, il garde certaines informations concernant cette requête légitime dans un tableau qui est remis à zéro à chaque intervalle. Pour chaque terminal ayant envoyé une requête légitime, le serveur garde en mémoire, son identité, la clef secrète partagée avec ce terminal et le numéro de séquence de la requête pour identifier facilement les rejeux. La différence entre la première version du protocole et le nouveau est que pour chaque nouvelle requête envoyée pendant un intervalle, le terminal incrémente le numéro de séquence initialisé au nonce. Lorsqu'un serveur reçoit une deuxième requête par un terminal légitime, le processus de vérification consiste à tout d'abord vérifier le numéro de séquence puis l'authenticité du message. Certaines informations étant déja stockées dans le tableau, ce processus est moins coûteux en terme de calcul.

Les évaluations de performance fournies à la fin du chapitre 7 montrent que le protocole proposé diminue l'utilisation du CPU de 50% dès la première étape.

Conclusion

Les réseaux satellitaires sont de très bon candidats pour les applications multicast. Grâce à leur capacité de diffusion et leur large couverture, une source peut atteindre un très grand nombre d'entités en transmettant une seule copie d'un paquet au segment satellite qui va le transmettre directement sans passer par d'autres éléments intermédiaires. Cependant, les avantages de ces réseaux augmentent leur besoin en terme de sécurité. Quelques solutions de sécurité ont été proposées mais restent propriétiares.

Dans cette thèse, nous avons analysé et proposé des solutions pour deux principaux besoins de sécurité visant les applications multicast pour les réseaux satellitaires : la confidentialité multicast et la protection contre le déni de service.

Nous avons en premier lieu analysé les protocoles de renouvellement de clefs ciblant les réseaux terrestres et conclu que le schéma LKH est celui le plus convenable pour les réseaux satellitaires en raison de l'absence de noeuds intermédiaires. Ce schéma souffrant encore en terme de fiabilité et de satisfaction des utilisateurs, nous définissons ses propriétés spécifiques

par rapport à ces deux besoins et analysons les performances des solutions actuelles ayant pour but de fournir des protocoles de renouvellement de clefs fiables. Suite à cette analyse, nous avons montré que ces solutions présentent encore des faiblesses en terme d'efficacité. En effet, l'arrivée ou le départ d'un membre de groupe affect le groupe en entier. Face à ce problème, nous avons proposé une nouvelle approche dans laquelle le fournisseur de service regroupe les entités selon certains critères, définit un ensemble de membres "privilégiés" et offre à ces membres un meilleur service. La notion de "privilège" peut varier d'une application à l'autre. Nous avons donc premièrement proposé une classification des applications selon leurs besoins en terme de sécurité et leur politique de paiement. En se basant sur cette classification, nous avons défini un premier protocole tenant compte de la durée de présence des membres pour offrir une transmission fiable pour les membres permanents. Ce protocole combine l'utilisation des codes de reconstruction et de réplication. Grâce à l'évaluation de la performance de notre protocole basée sur des simulations, nous avons conclu que celui-ci minimise l'impact des opérations de renouvellement de clefs sur les membres permanents en augmentant légèrement le coût d'utilisation de la bande passante. Nous avons également proposé une autre méthode de partitionnement de membres destinée aux sessions multicast de courte durée. Dans ce schéma, les membres sont regroupés selon leur temps d'arrivée. Grâce à ce deuxième protocole, le serveur réduit le nombre de pertes par rapport au schéma classique LKH en ayant un coût similaire.

Dans la deuxième partie de cette thèse, nous avons analysé le problème de protection contre le déni de service. Nous avons tout d'abord défini les attaques de déni de service et analysé les techniques de protection destinées aux réseaux terrestres. Ces techniques ne pouvant pas être implémentés dans le cadre des réseaux satellitaires, nous avons proposé deux versions d'un algorithme d'identification à deux étapes où le serveur rejette un nombre maximum de requêtes falsifiées dès la première étape sans consommer une quantité importante de ressources.

En conclusion, cette étude nous montre que le problème de la sécurité ne peut pas être traité sans prendre d'autres paramètres cruciaux en considération. Ces paramètres dépendent de la configuration du réseau et des attentes réelles. Les mécanismes de sécurité peuvent demander une utilisation importante des ressources comme la mémoire, la bande passante et le CPU. Par conséquent, l'implémentation de ce type de mécanismes ne doit pas introduire de nouvelles faiblesses contre les attaques de déni de service. Ils peuvent également avoir un impact fort sur la fiabilité de la transmission des données multicast. En effet, une opération de renouvellement de clefs peut induire la révocation d'un membre légitime du groupe dans un réseau présentant des pertes. Comme il existe toujours un dilemme entre les besoins de sécurité, d'efficacité, de fiabilité, nous avons proposé une classification des membres selon des critères de vie réelle pour pouvoir optimiser les paramètres et offrir un meilleur service aux membres "privilégiés".

Abstract

Multicast communications allow a network source to send data to multiple destinations simultaneously while transmitting only a single copy of the data on the network. For such communications and particularly for one-to-many communications, satellite networks provide some particular benefits such as a geographically extended coverage and an efficient delivery to a very large number of recipients. Applications intended for multicast communications strongly need some security solutions in terms of confidentiality, authentication and availability. In the last ten years, many security solutions were proposed in order to deal with these crucial problems and have been analyzed in terms of scalability. These solutions yet remain to be implemented because they still are severely lacking with respect to real-life requirements such as reliability and customer satisfaction.

The goal of this thesis is to study and provide secure and reliable solutions that take into account customers' expectations. The dissertation is divided in two themes that are relevant for satellite networks: **multicast confidentiality** and **denial of service prevention**. In the first part of the thesis, we start with a detailed study of the problem of multicast confidentiality and define the problem of group rekeying. We then analyze existing group rekeying solutions and highlight their neglected aspects in terms of reliability and customer expectations. We further suggest a new approach whereby the service provider partitions recipients with respect to some criteria, defines a set of privileged members and offers them a better service. In the second part of the thesis, we deal with denial of service attacks aiming at preventing legitimate recipients from accessing the service they are authorized to access in normal circumstances. We review existing solutions intended for terrestrial networks and show the limitations of their implementation in satellite networks. We then come up with an original solution that provides to the source the advantage of identifying bogus attacks almost immediately thus limiting the impact of such attacks on legitimate recipients.

Contents

List of Figures

List of Tables

Introduction

In Next Generation Networks (NGN) integrating a satellite segment, packet network technologies will have to support real-time multicast services such as tele-conference and pay-per-view applications. These applications need some security solutions in order to ensure that only authorized clients can access the service and that the service provider guarantees availability for these clients. The purpose of this thesis is to provide a detailed analysis of existing proposals dealing with multicast security and suggest new solutions that can be adapted to satellite networks.

We first briefly introduce multicast communications, then discuss general satellite networking architectures and technologies and their main differences from terrestrial networks. We then present security issues in a multicast setting and show that most existing multicast security techniques affect the behavior of the applications in a number of aspects.

Multicast communications

Many commercial applications such as audio/video distributions are destined to a very large number of recipients. In this case, two major scalability issues need to be taken into account:

- **the communication overhead**: the network bandwidth usage should not depend on the number of recipients;
- **the processing load**: the CPU utilization at the source should not depend on the number of recipients.

Since implementing such applications over unicast channels did not solve the scalability problem, Steve Deering introduced multicast communications [18] that allow a network source to send data to multiple destinations simultaneously while transmitting only a single copy of the data on the network. The network then takes care of replicating data packets at proper branching

points. Based on this new communication paradigm, at most one copy of each packet is transmitted over any link in the network (the total network load is significantly reduced compared to unicast communications). Since the source host does not need to keep track of individual connections with each recipient, the processing load at the source therefore is also reduced with respect to an equivalent solution using unicast.

In multicast communications, the source defines a group identified by a group address and the set of recipients that are members of the group receiving multicast packets destined to the group address. If a recipient wants to receive the multicast data, it joins the group following the Internet Group Management Protocol (IGMP) [23] by sending a message to the group manager that usually is implemented by the source itself.

There are three general categories of multicast applications:

- **One-to-Many**: a single host sends data to two or more receivers;
- **Many-to-Many**: any member of the group sends data to the whole group;
- **Many-to-One**: any number of receivers send data back to a source via unicast or multicast.

Since most of commercial applications such as audio/video distribution involve a single source and a very large number of recipients, in this thesis we investigate the security of One-to-Many applications.

Satellite networks

One-to-Many multicast applications can be provided either over terrestrial networks or satellite networks. Satellite networks exhibit some particular advantages for multicast communications such as their geographically extended coverage. We first describe the satellite networking architecture and then present particular aspects of these networks with respect to terrestrial networks.

Satellite networks are characterized by two segments: the **ground segment** and the **satellite segment** [40].

The ground segment consists of all the earth stations that are also called satellite terminals (STs). These stations are assumed to be directly connected to the end-user's equipment. Some stations are both transmitters and receivers and others are only receivers. There are two main topologies:

- a **star topology**: characterized by a large **gateway earth station** that has a high data rate broadcast channel (30/40 Mbps) to a large number of STs. In the return link, STs

transmit in bursts at low to medium data rates to the gateway. In this specific topology, all communications pass through the gateway.

- a **mesh topology**: in this topology, the communications are directly performed between two STs without any forwarding through the gateway.

Figure 1 illustrates these two topologies.

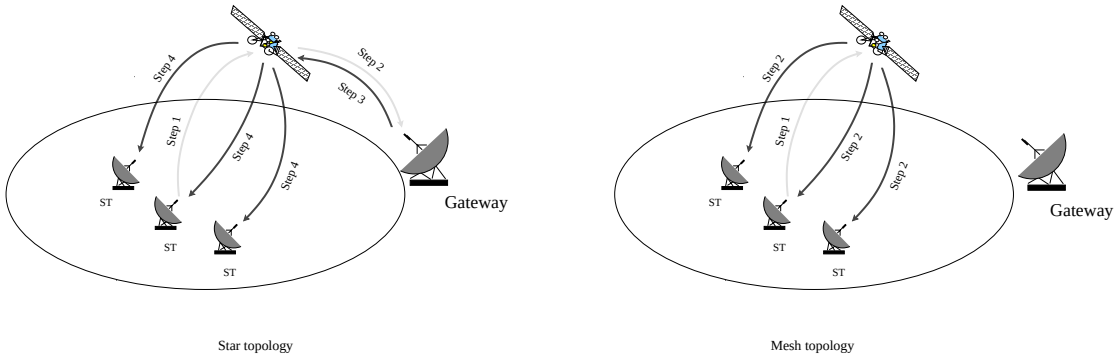


Figure 1: A Satellite network illustrating star and mesh topologies

The next important segment is the satellite segment. The radio waves transmitted by the earth stations are received by the satellite from the *uplink* and the satellite transmits to the receiving earth stations through the *downlink*. To fulfill its functions, a geostationary satellite can either operate in a transparent mode as a simple relay, or in a regenerative mode whereby On-Board Processing (OBP) will direct packets to each appropriate downlink spot beam. Satellite systems in transparent mode provide a simple solution to support a star topology for networking whereby in regenerative mode they can support both topologies.

GEO satellite systems, in general, are well suited for multicast applications thanks to their broadcast capability and especially, their extended coverage (where a spot beam can cover a continent). Regenerative satellites with on-board processing will be able to integrate broadcast and interactive services by combining DVB-S [21] and DVB-RCS [22] standards. The GEOCAST project [2] includes designs for both transparent and regenerative satellites. For regenerative satellites, the forward link is based on DVB-S and the return link on DVB-RCS or ATM [70].

However, satellite networks raise new issues [32] such as:

- **ease of eavesdropping**: any user located within the coverage of the spot beam of the satellite can receive any packet transmitted by this satellite;
- **high delay**: the end-to-end transmission delay is about 300ms for a geostationary satellite [30].

Such architectures require specific security solutions different from those intented to terrestrial communications.

Security issues in Multicast Communications

Commercial applications destined to a very large number of recipients mostly are implemented over multicast communications and are faced with several security issues that we can summarize in three items:

- **authentication**: the content provider needs to prevent its impersonation by another entity attempting to generate content on behalf of the content provider;
- **confidentiality**: the content should only be accessible to authorized clients and only for the duration of the authorization;
- **denial of service prevention**: denial of service attacks aim at preventing legitimate recipients from accessing a service they are authorized to access in normal circumstances. The content provider should minimize the impact of such attacks on legitimate recipients.

Several proposals give solutions to the confidentiality and authentication problem in the case of unicast communications. However, these techniques turn out to be ineffective when implemented in a multicast setting.

In terms of authentication, since the content provider must guarantee the origin of the data and its integrity to a very large number of members, symmetric schemes such as Message Authentication Codes (MAC) [35] turn out to be inefficient due to the inherent requirement for asymmetric schemes in multicast authentication. In order to alleviate the need for asymmetry and the resulting computational per packet overhead, classical signatures schemes based on asymmetric cryptographic algorithms such as RSA [64] must be combined with fast operations such as hashing or error correction techniques. Several solutions have been proposed to deal with this problem: [12, 59] suggest extensions of symmetric algorithms and other papers propose to adapt the use of digital signatures either by creating efficient signature schemes [67] or by combining some efficient schemes such as hash functions with them [27, 79, 29, 43, 58, 56].

As to confidentiality, the source needs to use some symmetric encryption algorithms such as AES [49] in order to let only group members access the content of the data. These algorithms are based on the use of a secret information called **key** that is used both for encryption and decryption. The source needs to share this key, that we call the group data encryption key, with all members of the group that may join or leave at any time. This dynamic nature of the group involves frequent updates of this group data encryption key. Consequently, the source or the group manager has to be able to distribute a new group data encryption key to all group

members, in an efficient way, at any time. We define this problem as the problem of **group rekeying**.

In multicast communications, intruders may perform denial of service attacks by targeting the source and exhausting its resources. Since a multicast group is assumed to be very large, attacks against a single entity (the source) have a strong impact on the whole group. Moreover, multicast confidentiality and authentication mechanisms may introduce some new weaknesses and can become new targets for DoS attacks (since such mechanisms require additional CPU and memory utilization).

Contributions

The goal of this thesis is to first analyze the main security problems of multicast communications and then suggest solutions intented for satellite networks. The problem of multicast authentication has been widely analyzed and most of the existing solutions can be implemented over satellite networks. Therefore, we do not further investigate this problem.

Existing multicast confidentiality solutions were only proposed to deal with security and scalability issues and are severely lacking with respect to reliability. Group rekeying protocols raise special requirements that are not addressed by classical reliability solutions. Indeed, the loss of a single rekeying packet can provoke the revocation of a legitimate recipient from the group. In this thesis, we first deal with the problem of reliable group rekeying. Having analyzed the specific relationships between rekeying packets transmitted during a secure multicast session, we show that existing reliable multicast protocols cannot be considered as suitable to the delivery of the keying material.

Moreover, in existing group rekeying solutions, each rekeying operation requires the update of the keying material for all members alike. Based on this observation, we address a new requirement, that is **customer satisfaction**, whereby the key server partitions members with respect to some criteria (such as membership duration, subscription options, etc.) and offers a better service to a set of members determined by these criteria.

To meet these two additional real-life requirements that are reliability and customer satisfaction, we suggest two different reliable group rekeying protocols suitable to satellite networks based on a new approach that takes into account group members' behavior:

- In the first proposed protocol, the group manager partitions members with respect to their membership duration and offers a strongly reliable rekeying protocol for long-lived members;
- In the second solution, we deal with another class of multicast applications whereby most of the members joining the group are assumed to stay until the end of the session. In this

case, the group manager partitions members with respect to their arrival time and thus minimizes the impact of frequent arrivals on members already belonging to the group.

Another interesting requirement in terms of customer expectations in secure multicast is the availability of the service provider. A service provider should always be able to treat recipients' requests on time. However, denial of service attacks aim at preventing legitimate recipients from accessing the service. Existing denial of prevention techniques designed for terrestrial networks are inefficient for satellite networks because of their inherent broadcast capability. Hence, the anti-clogging technique adopted by the Internet Security Protocol (IPSec [5]) involves the exchange of some **cookies** between the source and a legitimate recipient and allows the verification of the recipient's presence at the claimed address. In the context of satellite networks, cookies generated by the server would still be received by the intruder who would then be able to transmit the expected replies including server's cookies. Based on these observations and the special characteristics of the satellite environment, we propose an efficient identification protocol that allows satellite servers to quickly discard bogus requests.

Outline of the thesis

The first part of this thesis deals with the problem of reliable group rekeying. In chapter 1, we first establish the main requirements of group rekeying protocols including those of reliability and customer satisfaction. We then analyze the performance of a specific rekeying scheme (Logical Key Hierarchy) with respect to the first two requirements and show why this protocol fits for satellite networks. In chapter 2, we show the weaknesses of this protocol in terms of reliability and analyze recent solutions dealing with this requirement. In chapter 3, we finally deal with the customer satisfaction requirement and show that any rekeying operation affects all members alike. We then propose a reliable group rekeying scheme whereby the group manager classifies members with respect to membership duration and offers a high reliability to long-lived members and study its efficiency with simulations in chapter 4. Finally, in chapter 5 we propose a different reliable rekeying protocol destined to short sessions where during any rekeying operation, members having already subscribed to the service are not penalized from abundant further arrivals.

The second part of this thesis deals with the problem of denial of service: in chapter 6, we define the problem of denial of service and analyze the existing prevention techniques in terrestrial networks. We then show in chapter 7 that existing techniques cannot be implemented in satellite networks due to the inherent broadcast nature of the satellite segment and then propose an identification protocol that allows the satellite servers to quickly discard bogus requests and thus provide availability for legitimate recipients.

Part I

Group Rekeying in Satellite Networks

Chapter 1

Multicast Confidentiality and Group Rekeying

1.1 Introduction

As explained in the introduction of this thesis, the main advantage of satellite networks over terrestrial networks with respect to IP-multicast [18] is that a packet that passes through the satellite segment does not need to be replicated and can reach a very important number of recipients at almost the same time. With this approach, the content provider saves resources compared to an equivalent set of unicast communications. Many large-scale one-to-many commercial applications such as audio/video broadcasting or stock quote distribution that could benefit from multicast and broadcast mechanisms to reach many receivers in a scalable way are faced with the problem of **confidentiality**: the content should only be accessible to the clients who payed for the service and only for the duration corresponding to the payment.

This security issue has already been addressed in the case of unicast communications where symmetric encryption algorithms [42] are used to ensure secrecy. However, when implemented in a multicast setting, these techniques turn out to be ineffective in terms of security and scalability.

If the content provider uses some symmetric encryption algorithms, the multicast data is encrypted with a single key K_{enc} common to all recipients who paid for the service. The content provider then should prevent others from accessing the data. When a client is joining the service or when the subscription time of one of the clients expires, K_{enc} must be updated and its new value must be distributed to all remaining members in a scalable way. This problem is defined as the problem of **group rekeying**. An alternative to this approach is to define an individual secret key K_{enc_i} shared between recipient R_i and the source and encrypt each packet P with each of these secret keys. This approach clearly is not scalable. Finally, replacing symmetric encryption algorithms by asymmetric ones will increase the computational cost and the communication

overhead.

In the sequel of this chapter, we focus on the problem of group rekeying and define the entity that distributes the relevant keying material to members as the **key server**. Existing solutions were first proposed in order to offer some security solutions in an efficient way intended for large groups. In the following sections, we first define the main requirements for the design of a rekeying protocol. We then present a brief overview of existing proposals such as [44, 47, 57, 76, 78] and analyze them in terms of the predefined requirements. Since we aim at offering a confidentiality solution in satellite networks, we focus on an efficient rekeying scheme, that is the Logical Key Hierarchy [78] and show that yet this solution, as other existing solutions, still is lacking with respect to real-life requirements.

1.2 Requirements for group rekeying

In this section, we define the main requirements of a group rekeying scheme. We regroup them in four categories:

- **security requirements**: the key server faces new secrecy problems that do not exist in the context of unicast communications;
- **scalability requirements**: when implementing a group rekeying protocol, all resources related to the multicast setting should be optimized;
- **reliability requirements**: members of a group must receive their keying material in order to decrypt the corresponding encrypted multicast data;
- **customer satisfaction**: the key server should minimize the impact of frequent rekeying on members.

1.2.1 Requirements in terms of security

As explained in the introduction, in order to prevent the unauthorized access to the secret data, the content provider must use some encryption algorithms and distribute the corresponding keying material to all members of a secure group (ie., recipients that have paid for the service). Recipients authorized to access the content of protected data should only be able to decrypt this data for the assigned amount of time.

Moreover, we qualify the status of the group as being dynamic because we consider that the set of members evolves through time as members join or leave the group during a session. When a member joins the group, it first needs to contact the membership manager through a secure authenticated unicast channel. If this recipient is allowed to become a member, then it

receives the decryption keying material corresponding to its membership policy. On the other hand, a member is removed from the group when its ability to access the encrypted multicast content is suppressed by the membership manager that can decide its removal at any time during the session. This dynamicity of the group inherently implies two basic security requirements that are **backward** and **forward secrecy**. They are defined as follows:

Definition 1.1 *`Backward secrecy` is the requirement that a new member should not be able to access the multicast content transmitted prior to its joining the multicast group.*

Definition 1.2 *Symmetrically, `forward secrecy` is the requirement that a member leaving the group should not be able to further access multicast content.*

Multicast applications that are commercial may or may not require backward and/or forward secrecy. This requirement will depend on the pricing model of the application. Indeed, in some actual broadcast applications, like current digital TV platforms, customers first subscribe to the service and then pay a fixed fee periodically. In this case, the service provider ignores customers' behavior and let the clients pay for the whole application. In this kind of applications, since the customers pay for the whole session, the service provider does not need to ensure forward and backward secrecy. On the other hand, some other applications are not subscription based and offer customers the possibility to pay only the service they really had access to. In this kind of pricing model, the content provider should ensure forward and backward secrecy at each join or leave of a member. Backward and forward secrecy require the update of the keying material at each join or leave.

Another factor that is often taken into consideration in security is the **collusion**. The main requirement related to collusion is that members should not be able to get more privileges through collusion than they originally were granted by the system. For example, consider a scenario where there are several registration options for a service. Members choosing the less expensive service should not have access to the service that is more expensive by colluding with other members.

Finally, some security protocols may involve intermediate elements located on the path between the source and the recipients. The degree of trust that needs to be granted to intermediate elements might be another point of vulnerability.

In summary, the rekeying service should meet the following specific security requirements:

Requirement 1 *secrecy*

Requirement 2 *backward and forward secrecy*

Requirement 3 *resistance to collusion*

Requirement 4 *minimal trust in network components*

1.2.2 Requirements in terms of scalability

Multicast applications are destined for groups that are assumed to have a very large number of members and aim at keeping the overall costs as low as possible. In this case, the group rekeying solution must lend itself to efficient operations regardless of the number of members.

The costs that have to be taken into account are:

- **the computational overhead**: Since the application scenarios typically involve real time transmission of large amount of data, data encryption and decryption should not require an important amount of CPU consumption. Additionally, the retrieval of the keying material during any rekeying interval should not affect the processing load of the members.
- **the communication overhead**: Security mechanisms should not impose the duplication of the content: the service provider cannot afford to define one secret key for each member, encrypt a multicast packet with each of these keys and send it N times. Moreover, while rekeying a group, the rekeying cost (ie. the number of keys to be distributed in multicast) should also not depend on the group size and be optimized.
- **the memory usage**: Secure memory is a somewhat less important resource but should also be minimized. Members may have memory limitations. Symmetrically, the number of keys that are shared with each member and kept by the service provider should again be optimized with respect to the service provider's memory configuration.

The cost supported by an individual component, be it the source, a member or an intermediate network component, should scale to any potential group size and membership duration. The key server needs thus to ensure the following three scalability requirements both at the source and the receivers side:

Requirement 5 *minimize the computational overhead*

Requirement 6 *minimize the communication overhead*

Requirement 7 *minimize the memory usage*

1.2.3 Requirements in terms of reliability

Most of one-to-many applications can tolerate losses. Although many reliable multicast protocols have been proposed and analyzed [37, 26], rekeying protocols raise special properties and requirements that are not addressed by classical reliability solutions. First, we observe that while a key server sends a set of encrypted keys, each member only needs to receive a small fraction of these keys. This specific property is defined as the `sparseness` property in [81].

Beside this property, same authors define two further requirements to the problem of reliable rekeying:

Requirement 8 ***eventual reliability****: a receiver should be able to receive all of its encrypted keys.*

Requirement 9 ***soft real-time reliability****: the reception of the keying material is finished before the start of the next rekeying interval.*

If the key server does not ensure these two requirements, a member losing at least one rekeying packet can be automatically removed from the group and prevented from accessing the content of multicast data he is authorized to access in normal circumstances.

1.2.4 Requirements in terms of customer satisfaction

In [44], Mittra defines the **1 affects n** scalability failure which occurs when the action of a member affects the entire group. This type of failure enlarges the reliability requirement. For example, if each arrival or departure of a single member causes a rekeying operation for all members alike, then the key server should at least provide an almost fully reliable delivery for members assumed to stay until the end of the session. Based on this example, we suggest a new approach that takes into account members' "loyalty" where some members may be seen as more "loyal" then others with respect to some metrics such as the membership duration or the members' arrival time. The key server should provide a better reliable delivery to such members. We thus define the following requirement:

Requirement 10 ***customer satisfaction****: the key server should minimize the impact of frequent rekeying operations on a pre-determined set of "loyal" members.*

1.2.5 Summary

We began the chapter by defining the specific requirements of a rekeying protocol in terms of security, scalability, reliability and customer satisfaction. While designing a rekeying protocol, all these requirements must be taken into account together. However, existing solutions mainly focused on security and scalability issues. We first propose to review them and then analyze their weaknesses in terms of reliability and customer satisfaction in the subsequent chapters.

1.3 Classification of protocols offering group secrecy

Existing solutions for group rekeying were intented to optimize the rekeying cost while keeping a high security level. Reliability issues and customer expectations were not taken into account. In [55], Pannetrat classified these protocols into two main categories with respect to the dependence of the protocol on the architecture of the network and the role given to intermediate elements in the security protocol itself. Hence, if the network is meshed and intermediate elements are assumed to play a role in the security protocol in order to reduce the communication overhead, the service provider must define the degree of trust placed on intermediate elements. These elements may or may not have access to the content of the data. If on the other hand, there are no intermediate elements as in satellite networks or if intermediate elements are not involved within the security protocol, the service provider needs to optimize the communication overhead as well as the memory usage.

1.3.1 Re-encryption trees

The first protocol offering group secrecy while taking into account the network architecture and involving intermediate elements was proposed by Mittra and is called `Iolus` [44]. This framework physically partitions the group of members into several subsets and creates a tree hierarchy between them. Each subset of members is managed by a distinct entity called a Group Security Intermediary (GSI_i) and members of a common subset share the same decryption key, K_i. All encrypted data must pass through GSIs which first decrypt the data with the key associated to the parent subset and encrypt it with the key shared with the subset of members associated with each GSI. As illustrated in figure 1.1, when GSI_2, being the manager of members R_7, R_8 and R_9, receives an encrypted multicast packet P, from its parent GSI_1, it first decrypts the packet with the key K_1, then reencrypts it with K_2 and multicasts it to members of his subset. Since this method is simple and uses only symmetric encryption, it can be applied either for the transmission of the data packet itself or for the transmission of a short term key in case of group rekeying.

In order to ensure the two main security requirements that are backward and forward secrecy, the Group Security Controller (GSC) modifies some control plane data at each arrival or removal of a member. Hence, when a member is removed from a subgroup, the GSI chooses a new key for its corresponding subgroup and sends this information via unicast to each remaining member of its subgroup. For example if member R_9 leaves the group, GSI_2 defines a new key K_2' and sends this new secret information in separate secure unicast channels to R_7 and R_8. Symmetrically, if a recipient becomes a member of a certain subgroup, the corresponding GSI should again define a new subgroup key and send this new subgroup key to all members in a broadcast channel by encrypting it with the old subgroup key.

Table 1.1 summarizes the specific properties of Iolus with respect to the requirements defined in the previous section. Since the group is physically partitioned into several subgroups,

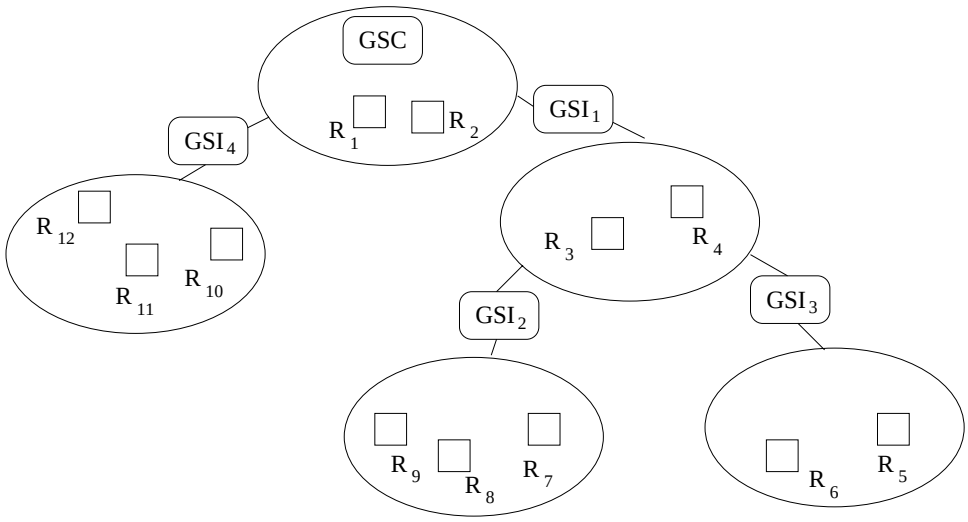


Figure 1.1: An illustration of IOLUS

if the size of these subgroups is small enough then this scheme provides scalability in terms of computation and communication. Thanks to the distributed management of the group, each GSI needs to know only two keys and each member a single.

While Iolus is efficient in terms of security and scalability, all the intermediate elements can have access to the content of the data. Since some applications cannot afford to place any trust in intermediate network components, in [46, 57], authors define a scalable multicast confidentiality scheme involving intermediate elements as in Iolus but do not let intermediate nodes access the content of the data.

1.3.2 Group Rekeying protocols for flat architectures

Some applications are implemented in an environment where the network architecture is flat, ie. where there are not many intermediate elements. Hence, in satellite networks, the only intermediate element is the satellite segment and this element can cover a very large area such as a continent in one network. In such environments, there exist some protocols that do not take the physical configuration of the network into account and independently define a group rekeying scheme. The data is assumed to be encrypted with only one secret key K_{enc} that is shared by all members of a pre-determined group. As opposed to reencryption trees, when an arrival or departure of a member occurs, the group manager must define a new data encryption key K'_{enc} and send this new key to all members in a scalable and reliable way in order to ensure backward and forward secrecy.

A well-known technique that offers a group rekeying method independently of the network architecture is the Logical Key Hierarchy (LKH) scheme where the group manager defines a logical key tree in order to minimize the communication overhead and the memory usage. This

Security:
The scheme offers secrecy, backward secrecy and forward secrecy;
the key server completely trusts its intermediate nodes;
Scalability:
computational overhead:
server side: one encryption operation;
receiver side: one decryption operation;
intermediate elements: one encryption and one decryption operation;
communication overhead:
reduced to the subgroup where there is arrival or departure. Rekeying done in unicast;
memory usage:
server side: the group key;
receiver side: one decryption key;
intermediate elements: two keys;
Reliability
not taken into account;
Customer Satisfaction
the set of affected members during a rekeying operation is minimized.

Table 1.1: Iolus summary

scheme is proven to be optimal in [71]. We present it in detail in the next section and show its efficiency in terms of security and scalability.

1.4 Logical Key Hierarchy: LKH

1.4.1 Description of the scheme and Security Evaluation

The Logical Key Hierarchy (LKH) scheme, which was independently proposed by Wong et al. [78] and Wallner et al. [76], offers a rekeying scheme that is independent of the network architecture. Hence, unlike Iolus, LKH does not rely on any intermediate elements.

In this scheme, the group manager constructs and maintains an almost balanced logical tree G with N leaves where N is the group size. Let d_G and h_G respectively denote the outdegree and the depth of G . These parameters are defined by the following equation:

$$h_G = log_{d_G} N + 1 \tag{1.1}$$

A random key is attributed to each node where each leaf node corresponds to a unique

member of the group. The key corresponding to the root node is the data encryption key. Each member receives the set of keys corresponding to the path from the root of the tree to its corresponding leaf. Referring to the example in figure 1.2 where the group size is 8, the key server defines a key tree with outdegree 2 and height 4. All members receive the keys associated with the nodes located on the path from the root node to the corresponding leaf. Hence, R_1 would receive the key set $\{k_0, k_1, k_3, k_7\}$ where k_0 is the data encryption key.

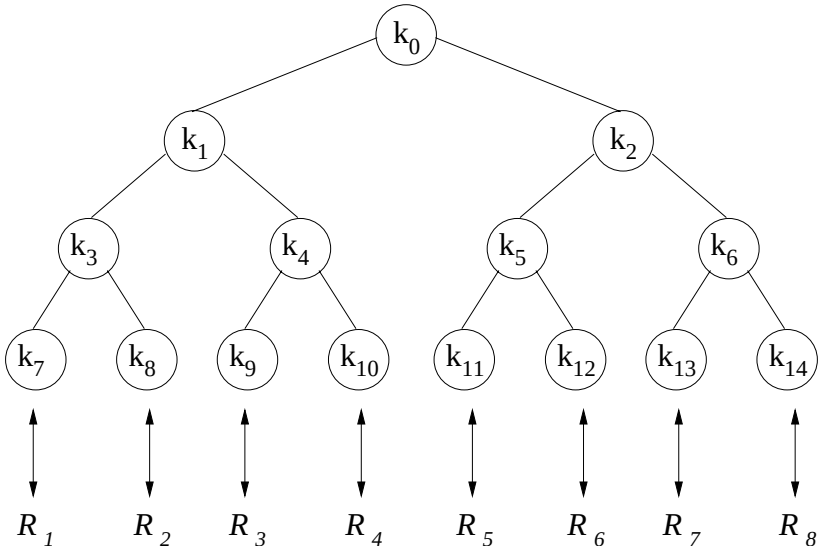


Figure 1.2: An Example of the LKH Scheme

In order to offer backward and forward secrecy, the key server performs a rekeying operation each time a member joins or leaves the group. We thus define the rekeying mechanisms at each join or leave event. In the sequel of the chapter, we use the notation $E_{k_x}(k_y)$ when a key k_y is encrypted using another key k_x.

Join

In order to include a new member to the group, the key server extends the tree with an additional leaf. The server marks all keys associated with the vertices on the path from the additional leaf to the root as invalid. A random key is assigned to the new leaf and transmitted with a secure unicast channel to the new member. Subsequently, the keying material of each of the other nodes on the path from this new leaf node to the root node are updated: each of the new keys associated with a node is encrypted with the actual key associated with the same node. As illustrated in figure 1.3, when member R_8 joins the group, the key server assigns to it a new key k_{14} and defines an additional leaf node in the key tree G. In order to update other nodes in the path from this leaf to the root, the group manager broadcasts $E_{k_0}(k'_0), E_{k_2}(k'_2), E_{k_6}(k'_6)$ to the remaining members so that they can receive all the keys required for the decryption of future encrypted data. R_8 receives $E_{k_{14}}(k'_0)$, $E_{k_{14}}(k'_2)$ and $E_{k_{14}}(k'_6)$.

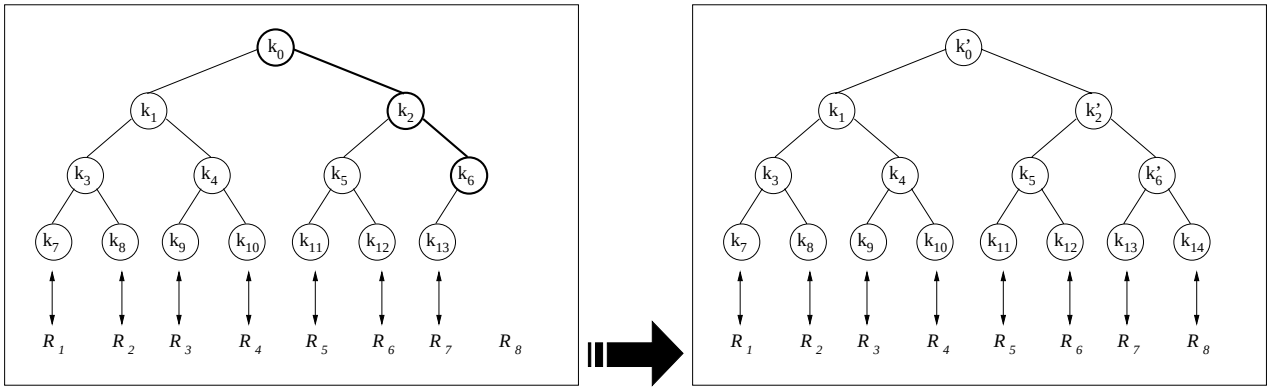


Figure 1.3: Addition of member R_8 to G

Leave

When a member leaves the group, all keys associated with the vertices of the path from the root to the leaf corresponding to the leaving member are marked as invalid. The rekeying operation then consists of substituting these invalidated keys with new values and broadcasting the new values in key envelopes encrypted under keying material that is known by remaining members.

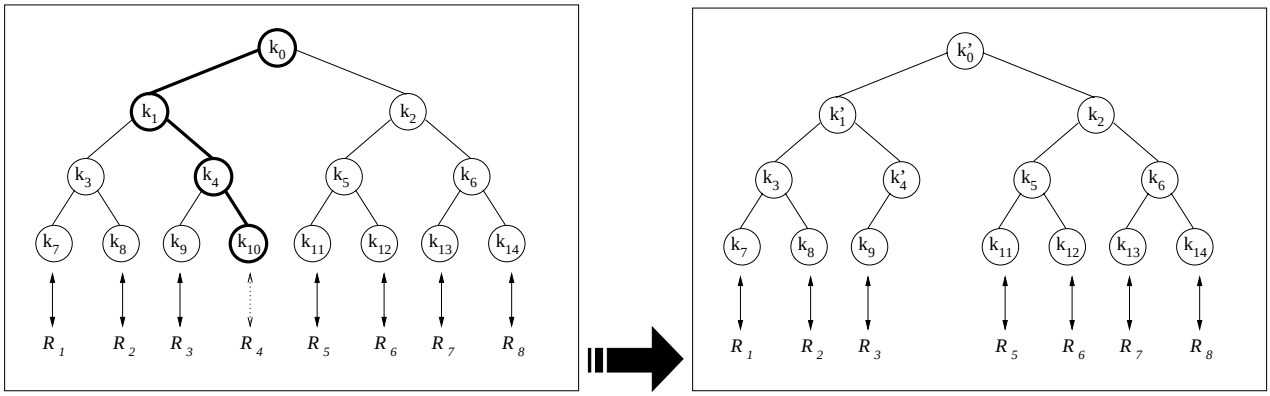


Figure 1.4: Removal of member R_4 from G

As depicted in figure 1.4, if member R_4 leaves the group, k_4, k_1 and k_0 are updated with k_4', k_1' and k_0', respectively. The key server then broadcasts:

$$\{E_{k_9}(k_4'), E_{k_3}(k_1'), E_{k_4'}(k_1'), E_{k_1'}(k_0'), E_{k_2}(k_0')\}$$

Security Evaluation

Since at each arrival or departure, the group manager updates the key tree in order not to give the knowledge of old/new keys to joining/leaving members, LKH offers backward and forward secrecy. Hence, when a member joins the group, all keys corresponding to the path from the assigned leaf to the root node are updated and encrypted with the actual keys such that the new member that was not a part of the group at the previous rekeying interval does not have access to the old information. Symmetrically, when a member leaves the group, the group manager updates the keys assigned to the leaving member by encrypting them with keys shared only by remaining members.

1.4.2 Scalability Evaluation

For a group with N members that are regrouped in a logical key tree $\mathcal{G}$ of degree d and depth h such that $N = d^{h-1}$, the memory cost of each member in number of keys is:

$$Memcost(\mathcal{G}, member) = h \tag{1.2}$$

The server has to store in its memory all the keys in $\mathcal{G}$. The memory cost of the server is defined as follows:

$$Memcost(\mathcal{G}, server) = \sum_{i=0}^{h-1} d^i = \frac{d^h - 1}{d - 1} \tag{1.3}$$

During a rekeying operation, in order to reach the new data encryption key, each member needs to perform a certain number of decryption operations. This number depends on the location of the member and the location of the updated keys. Hence, back to the example of figure 1.4 whereby member R_4 leaves the group, member R_3 needs to do three decryption operations before the decryption of the data and member R_8 receives only one key which is the data encryption key. We thus define here the maximum number of operations performed when all the keying material of the corresponding member needs to be updated:

$$Compcost_{max}(\mathcal{G}, member) = h - 1 \tag{1.4}$$

Similarly, since the number of encryption operations that the server performs during a specific rekeying operation also depends on the location of updated keys and joining or leaving members, we define the maximum computational cost as:

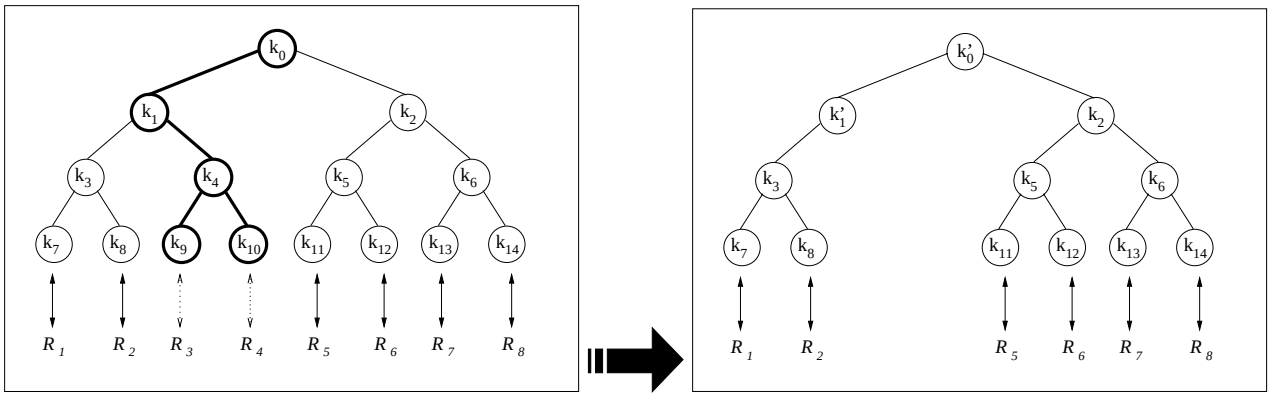


Figure 1.5: Removal of R_3 and R_4 in a batch

$$Compcost_{max}(G, server) = d * \sum_{i=0}^{h-2} d^i = \sum_{i=0}^{h-2} d^{i+1} \tag{1.5}$$

Finally, the communication overhead corresponds to the number of encrypted keys that a key server broadcasts to the members of the group. This rekeying cost is equal to the number of operations performed by the key server. We therefore have:

$$Comcost_{max}(G) = \sum_{i=0}^{h-2} d^{i+1} \tag{1.6}$$

1.4.3 An improvement of LKH: Batch rekeying

At each arrival or departure of a member, the key server needs to immediately rekey the whole group to ensure backward and forward secrecy. However, individual rekeying is relatively inefficient in large groups where join/leave requests happen very frequently. For example, referring to the key tree in figure 1.5, if members R_3 and R_4 leave the group one after the other with a very short delay between the two departures, the key server will need to modify twice the keys located at same vertices in the tree. If on the contrary, the key server had regrouped these two departures in one rekeying operation, the rekeying cost would be reduced by half.

In order to overcome the scalability problems raised by individual rekeying, batched rekeying algorithms have been proposed where leave and join requests collected during an interval are processed by rekeying operations performed during the subsequent interval. Thanks to these batching algorithms, the key server can improve efficiency at the expense of delayed group access control, because a new recipient has to wait longer to be accepted by the group and a departed (or expelled) member can stay longer within the group.

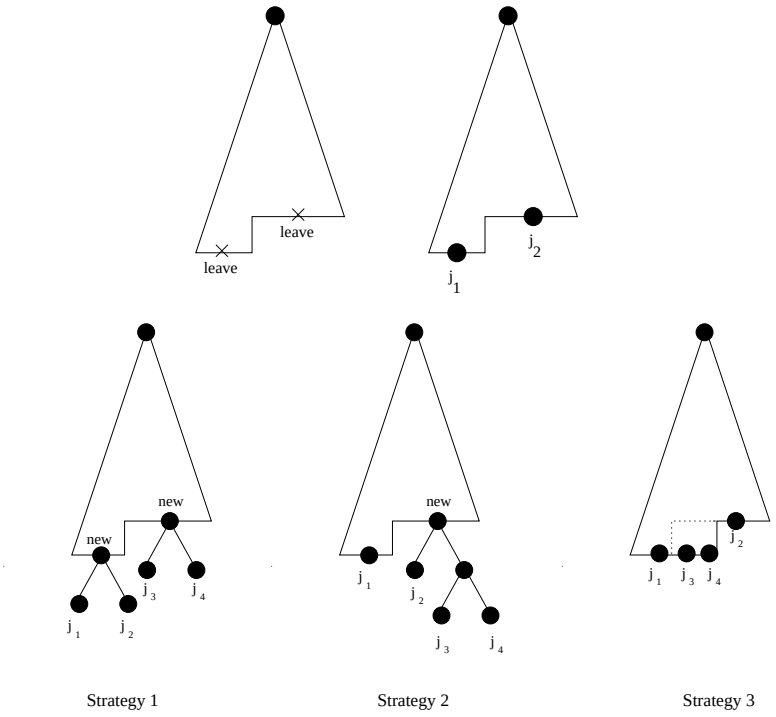


Figure 1.6: An illustration of batch rekeying with 2 leaving and 4 joining members

Authors in [81] describe three main batch algorithms where the choice depends on the need of the service provider:

- the key server processes both join and leave requests in a batch;
- the key server processes each join request immediately and leave requests in a batch;
- the key server processes each leave request immediately and join requests in a batch.

This paper further investigates how to process join and leave requests in a batch. There exist three different strategies for collecting join and leave requests. In all these strategies, the key server first assigns the maximum number of leaf nodes corresponding to leaving members to newly joining members. Assuming that the number of joining members J is greater than the number of leaving members L, the three strategies differ in how to place the remaining $J-L$ members in the updated key tree.

- Strategy 1: In this strategy, the key server splits the L replaced nodes to add the remaining $J-L$ members. If this is still not sufficient then nodes are split from left to right.
- Strategy 2: This strategy was first proposed and analyzed in [36]. At the beginning of a rekeying interval, the key server collects all the joining members in a small key tree and

after processing the leaving members, the key server grafts the new subtree to the node with the smallest height.

- Strategy 3: This strategy was proposed in [82] and was only designed to make it efficient for members to identify the rekeying packets that they need. With this strategy, the remaining $J - L$ members are assigned to the children null-nodes whose parent node's IDs are the most recent ones. The key server multicasts this ID in order to let remaining members derive their updated node-IDs.

These three strategies are illustrated in figure 1.6 where at a given rekeying interval, there are two leaving and four joining members. First of all, in all cases, the first two joining members are assigned to nodes corresponding to leaving members. The location of the two remaining joining members depends on the chosen strategy.

An evaluation of the batch rekeying scheme in [81] shows a clear advantage over individual rekeying. Considering a group of 4096 members regrouped in a key tree of degree 4, in the case of 400 leaving members, batch rekeying requires approximatively 2147 encrypted keys while individual rekeying requires 9600 keys.

Discussion

As other existing solutions, LKH was first proposed in order to deal with security and scalability issues. The other requirements presented in sections 1.2.3 and 1.2.4 (reliability and customer satisfaction) were not taken into consideration while designing this scheme. In this section, we therefore gave the evaluation of LKH in terms of security and scalability. We summarize LKH's specific properties with respect to these two issues in table 1.2. This scheme has been proven to be optimal in terms of rekeying cost in [71], where Snoeyink et al. show that the lower bound of a rekeying operation is $O(log_d(N))$, which is the bound already achieved by LKH.

1.5 Conclusion

Most multicast security requirements differ from their unicast equivalents and cannot be met by existing unicast security solutions. In this chapter, we defined the new challenges raised by secure multicast communications and described four requirements for multicast confidentiality that are security, scalability, reliability and customer satisfaction. Existing solutions were designed to provide only the first two requirements. The other issues were not taken into account at the design phase of the protocol. These solutions fall in one of two categories with regards to the relationship between the security protocol and the network architecture. The first category that requires the involvement of intermediate nodes (Iolus) does not seem suitable for satellite networks, whereas the second category encompassed by the umbrella of protocols known as

Security:
The scheme offers backward and forward secrecy;
Intermediate elements are not involved in LKH;
Scalability for a group of size N:
computation overhead:
server: $O(Nlog_d(N))$;
member: $O(log_d(N))$;
communication overhead:
$O(Nlog_d(N))$;
memory usage:
server: $\sum_{i=0}^{h-1} d^i$;
member: $h = 1 + log_d(N)$.

Table 1.2: LKH scheme summary

Logical Key Hierarchy (LKH) offers a suitable basis for further study. We thus analyzed in detail the LKH scheme in terms of memory, bandwidth and CPU utilization. Other protocols were designed to deal with this problem like in [10, 75, 25, 72], but since LKH is proven optimal, we did not describe them. In the next chapter, we will analyze LKH with respect to the two other requirements: reliability and customer satisfaction.

Chapter 2

Reliable Group Rekeying Protocols

2.1 Introduction

Several reliable multicast protocols have been proposed and analyzed in [6, 26, 37, 51, 34, 73]. These protocols are designed with respect to the nature of the multicast application and it cannot be expected that a specific reliable multicast protocol will be used for all multicast applications [6]. Hence, some applications such as audio/video distributions can tolerate losses: for example, audio might have a short gap of lower fidelity but will remain intelligible despite some data loss; some other applications like file distribution or caching strongly require reliable data delivery: the delivered information must have reached the destination as complete.

Existing reliable multicast protocols cannot be considered as suitable to the delivery of the keying material where the information is assumed to be sensitive. Hence, rekeying protocols, including LKH, raise special properties and requirements that are not addressed by existing solutions. First, as explained in section 1.2.3, rekeying protocols raise the `sparseness` property: while a key server sends a set of encrypted keys, each member only needs a small fraction of these keys. In addition to this sparseness property, there is a strong relationship between rekeying packets: since updated keys are transmitted while being encrypted with some other keys, members should have received these encryption keys. Unlike classical multicast applications, the loss of one rekeying packet would prevent the recipient from accessing the updated data encryption key and the corresponding encrypted multicast data. The key server should thus ensure that each member receives all of its updated keying material (**Requirement 8**) before the beginning of the next rekeying interval (**Requirement 9**). In this chapter, considering that LKH is proven to be optimal in terms of scalability, we analyze the relationship between rekeying packets in order to show why existing reliable multicast protocols are not suitable for it. We then present existing protocols ensuring the reliable delivery of the keying material.

2.2 Problem Statement: Relationship between keys in LKH

LKH exhibits a strong relationship between rekeying packets being transmitted. We consider two different classes of relationships between keys that explain the need for a strongly reliable delivery of the keying material:

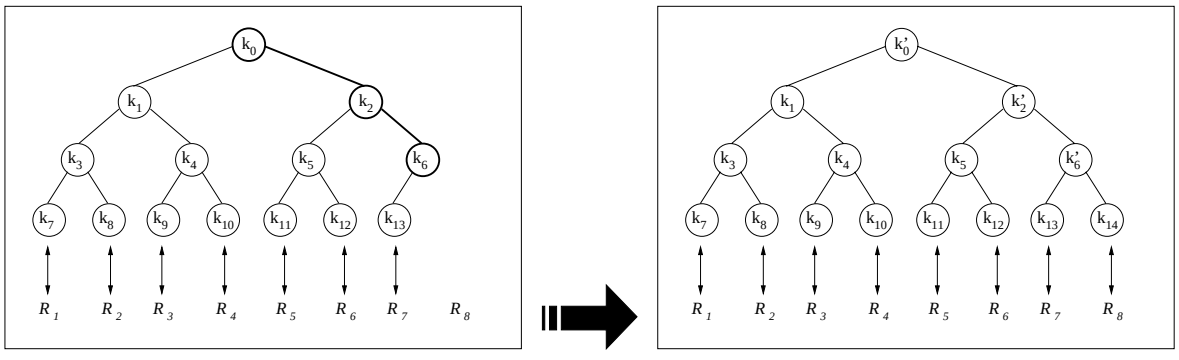
- the inter-dependency (time dependency) among keys raising the relationship between keys of different intervals;
- the intra-dependency (spatial dependency) among keys raising the relationship between keys transmitted within a single interval.

In this section we define and illustrate these relationships with examples. For the sake of clarity to the reader, we denote $k_i^{(j)}$ as the key associated with node i at the rekeying interval t_j.

Inter-dependency among keys

Referring to the example illustrated in figure 2.1, when member R_8 joins the group at interval t_{j+1}, $k_0^{(j)}$, $k_2^{(j)}$ and $k_6^{(j)}$ are marked as invalid and need to be updated. The key server must thus define a new key for each of the invalid nodes and respectively encrypt each of them with the existing ones. The key server thus broadcasts the following three encrypted keys:

$$E_{k_6^{(j)}}(k_6^{(j+1)}), E_{k_2^{(j)}}(k_2^{(j+1)}), E_{k_0^{(j)}}(k_0^{(j+1)})$$



(a) interval j (c) interval t_{j+1}

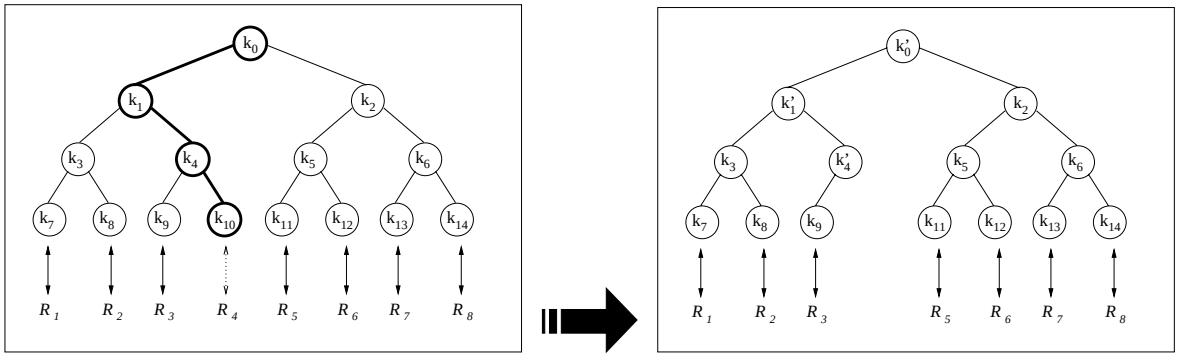
Figure 2.1: Addition of member R_8 from G

Having received these three encrypted keys, if the recipient R_7 did not receive for example $k_6^{(j)}$ during the previous interval, it cannot decrypt the new key $k_6^{(j+1)}$ that can be used to encrypt

keys of subsequent intervals. Based on this observation, we define the **inter-dependency** among keys as follows:

Definition 2.1 *Inter-dependency among keys: A key $k_i^{(j)}$ associated with node i in the key tree at time t_j can be used to encrypt a new key $k_i^{(j+d)}$ associated with same node i defined at a future interval t_{j+d}. Therefore, a member can have access to $k_i^{(j+d)}$ only if it has previously received $k_i^{(j)}$.*

2.2.1 Intra-dependency among keys



(a) interval t_j (c) interval t_{j+1}

Figure 2.2: Removal of member R_4 from G

In figure 2.2, when member R_4 leaves the group at interval t_{j+1}, the key server needs to transmit $k_4^{(j+1)}$, $k_1^{(j+1)}$ and $k_0^{(j+1)}$ while encrypting each of them with keys associated with their child nodes. The key server thus transmits:

$$E_{k_9^{(j)}}(k_4^{(j+1)}), E_{k_4^{(j+1)}}(k_1^{(j+1)}), E_{k_3^{(j)}}(k_1^{(j+1)}), E_{k_1^{(j+1)}}(k_0^{(j+1)}), E_{k_2^{(j)}}(k_0^{(j+1)})$$

If member R_3 does not receive at least one of these encrypted keys, it cannot have access to the new data encryption key $k_0^{(j+1)}$ and thus cannot decrypt the multicast data. **Intra-dependency** among keys is defined as follows:

Definition 2.2 *Intra-dependency of keys: During a rekeying interval t_{j+1}, an updated key $k_i^{(j+1)}$ associated with node i is encrypted with keys associated with the children nodes of i. Therefore, a member can have access to $k_i^{(j+1)}$, only if it has access to the keys associated with the child nodes of node i.*

Conclusion

Based on these specific relationships between the keys and the reliability requirements defined in chapter 1, it is now obvious that the key server needs to define a specific delivery protocol for rekeying protocols. Hence, for each member, the loss of a single rekeying packet can raise the removal of this member from the group. In this case, affected members should contact the key server in order to be re-synchronized with the group.

Recently, some studies have focused on this issue and different reliability schemes have been proposed to reduce the probability of losses in rekeying. In order to describe these reliability schemes, we first present in the next section the basic techniques that are essential for the design of any reliable protocol.

2.3 Background

2.3.1 Basic techniques

Coding theory distinguishes two types of errors [38]:

- **corruption of data**, where bits are corrupted
- **erasure data**, where the whole packet is lost.

As in [50], we consider a corrupted packet as being lost and investigate packet recovery features.

Reliability techniques can be implemented in two main modes:

- **reactive mode**, whereby the sender first transmits the original packets and the transmission of additional information is triggered by a feedback from the receiver;
- **proactive mode**, whereby the transmission of original packets and of the additional information takes place without waiting for a feedback from the receiver.

In the case of the delivery of the keying material, in order to ensure the specific requirements described in the previous chapter such as the **eventual reliability** (**Requirement 8**), the most suitable method is the use of a proactive method combined with reactive techniques used in subsequent rounds to guarantee the delivery. Moreover, the inherent characteristics of satellite

networks such as the end-to-end latency and the restricted capability of members to send some feedback messages are the other main reasons to use proactive reliability methods.

The most frequently used proactive reliability techniques are: **proactive replication** and **proactive FEC**.

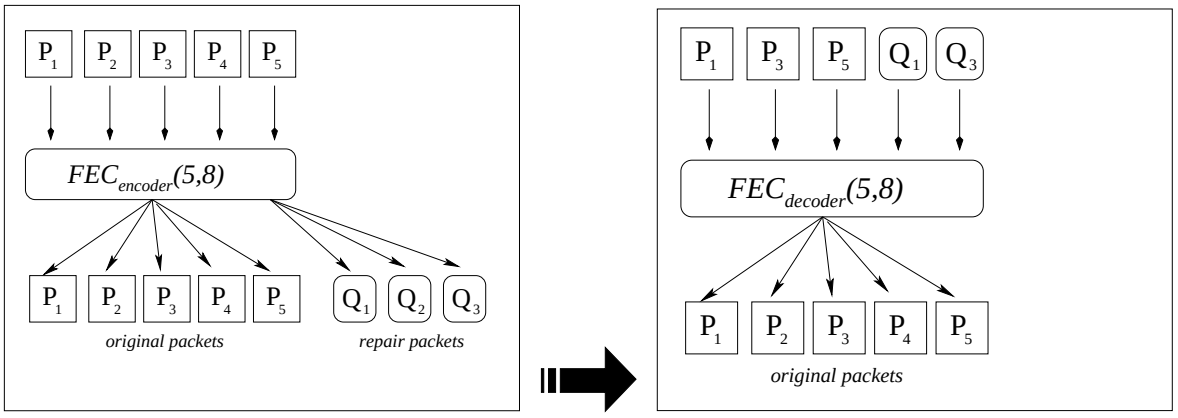
On one hand, the key server can replicate the data and transmits these replications as many times as required. We formalize this technique in definition 2.3:

Definition 2.3 ***Proactive Replication****: Given a packet P and a replication degree r, we define* $Replicate(P,r)$*, the function that replicates r times packet P.*

On the other hand, **proactive FEC** involves the transmission of the original data along with additional information called **parity packet** which can be used to reconstruct the original data in the event of losses. The sender constructs a block of *k* packets and uses an algorithm (the FEC_encode) to generate some **parity packets**.

Definition 2.4 ***Proactive FEC****: Given a block of k packets, the key server uses* $FEC\_encode(k,n)$ *that returns n packets such that* $n \geq k$ *and from any k packets, a receiver can reconstruct the original k packets.*

An example of the usage of proactive FEC is illustrated in figure 2.3. In this example, the key server generates 3 parity packets, that are Q_1, Q_2 and Q_3 from 5 original packets $\{P_1, P_2, P_3, P_4, P_5\}$. A member only receives the following packets: $\{P_1, P_3, P_5, Q_2, Q_3\}$. Since this member did not receive all original packets, it uses the decoder function in order to recover the remaining packets that are P_2 and P_4.



(a) server side (c) receiver side

Figure 2.3: An example of the use of FEC for 5 packets

The widely used techniques are Reed Solomon Codes [63, 66] and Tornado Codes [39]. Unlike Tornado Codes, the Reed Solomon encoder includes the *k* original packets in the resulting

packets without modification. Hence, on the case when the k original packets were received without loss, the recipient does not even need to perform the decoding operation at all.

2.3.2 Keystone: The first basic reliable key delivery protocol

The need for reliability for group rekeying protocols was first discussed by Wong et al. [77] shortly after the design of LKH. In order to increase the reliability of the transport of rekeying messages, the authors implemented their own key management system, Keystone, where the key distribution method is based on LKH and where proactive FEC is used for rekeying operations due to the addition or removal of members.

In this scheme, after each join or leave, the key server defines the set of encrypted keys to transmit to the group and split it over k packets. The key server then uses an FEC encoder to generate the additional r parity packets. Remaining members may recover the encrypted key set if they at least receive any k packets out of $k+r$ packets. The number of parity packets r is computed based on k and the packet loss probability p that is assumed to be independent in this context. The probability that a member does not receive its k packets is defined as follows:

$$P(k,r,p) = \sum_{i=r+1}^{k+r} \binom{k+r}{i} p^i (1-p)^{k+r-i} \tag{2.1}$$

Thanks to this equation, the key server can set the number of parity packets to a value such that the message loss probability defined in equation 2.1 does not exceed a given threshold value. Moreover, since there still is a probability for a remaining member not to recover its keying material, Keystone also provides for clients a re-synchronization mechanism that allows a member to contact the key server and update its keying material with regards to the rekeying interval. Each time a group member sends a `re-synchronization` message, instead of retransmitting the lost rekeying packets, the key server only sends the required keying material encrypted with the corresponding members' individual key.

While designing Keystone, authors did not analyze LKH's specific properties and its relevant requirements in terms of reliability. This technique turns out to be very basic in the case where the group is assumed to be very large and dynamic [82]. In order to enhance the rekeying protocol together with reliable delivery, these specific requirements must be taken into consideration with the scalability requirements. The group manager should be able to offer a strongly reliable delivery of the keying material without increasing the communication overhead.

In the sequel of this chapter, we present three protocols that were proposed after the design of Keystone and take into account the three main requirements that are security, scalability and reliability. While the first protocol, ELK, includes the reliability features within the design of the rekeying protocol itself, the two others are respectively based on the use of proactive replication and proactive FEC.

2.4 ELK

Perrig et al. proposed ELK [60], standing for ***E**fficient **L**arge Group **K**ey Distribution* which is a key distribution protocol originating from LKH but slightly modified in order to improve the scalability and the reliability features of the basic scheme. In this particular scheme, as in LKH, the key server constructs and maintains a binary key tree where each leaf node corresponds to a unique member of the group. ELK is composed of two basic mechanisms:

- **the key update mechanism**, which is a new key distribution protocol that optimizes the rekeying cost;
- **the key recovery mechanism**, which deals with reliability issues.

We first summarize rekeying with ELK in case of a join and a leave respectively and then describe the measures taken against network losses.

2.4.1 The rekeying mechanism

Notations

We use the following notation:

- as in the previous chapter, a message M encrypted with a key K is denoted by $E_K(M)$;
- $PRF_K^{m->n}(M)$ denotes pseudo random functions using key K on input M of length m and output n bits;
- $LSB^{(n)}(M)$ returns the n least significant bits of M.

Join

When a member joins the multicast group, as opposed to LKH, the key server does not need to broadcast any rekeying packet. At every rekeying interval, all keys of the whole key tree are updated using the procedure described in table 2.1.

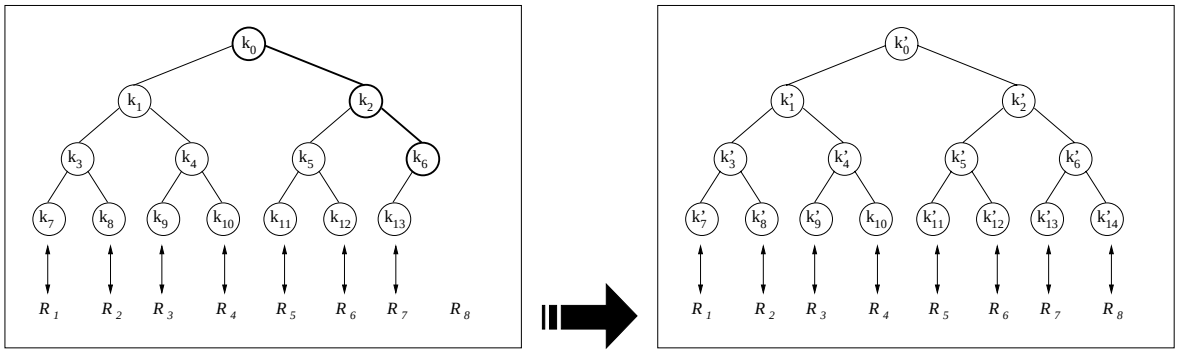
Each member automatically updates its keying material, that is the set of keys in the path from the root node to the corresponding leaf node. Since all auxiliary keys depend on the data encryption key K_0 of the previous interval that is known by all group members, the key server

At each rekeying interval t_{j+1}:

if $(i = 0)$ then
$K_i^{(j+1)} = K_0^{(j+1)} = PRF_{K_0^{(j)}}(0)$
else
$K_i^{(j+1)} = PRF_{K_i^{(j)}}(K_0)$

Table 2.1: The key update mechanism in case of a join

does not need to send any rekeying packet during the corresponding rekeying interval. As illustrated in figure 2.4, at the beginning of the rekeying interval t_{j+1}, if member R_8 joins the group, without receiving any broadcast messages from the key server, R_7 updates $\{K_0, K_2, K_6, K_{13}\}$ using the algorithm defined in table 2.1. The key server sends in unicast to R_8 its required keying material, that is $\{K'_0, K'_2, K'_6, K'_{14}\}$.



(a) Rekeying interval t_j (c) Rekeying interval t_{j+1}

Figure 2.4: Addition of member R_8 with ELK

Although no keying material is broadcasted to the members of the group, the key server still needs to send a few unicast messages as some members might be moved to new locations in the key tree and new nodes are added to the key tree. When the key server needs to add new nodes, it first picks a node N_j and generates a new parent node N_P for N_j and the node corresponding to the new member. The new key K_P corresponding to the node N_P is defined as in equation 2.2:

$$K_P = PRF_{K_j}(1) \tag{2.2}$$

All members located below the node N_P need to be informed of this modification. The rekeying mechanism in the case where there are several join events remains the same.

Leave

As in all key management schemes, dealing with a member leaving the group is more complicated than with a joining member. Indeed, ensuring forward secrecy is much more difficult than backward secrecy, since a new joining member initially does not share any key with any other member of the group. All the keys that the leaving member knows need to be replaced with new keys that the leaving member must not be able to compute. During this event, as in the LKH scheme, the server deletes the leaf node corresponding to the leaving member, marks all remaining nodes on the corresponding key path as invalid and these keys need to be updated. In this scheme, unlike in LKH, the update mechanism of any key involves keys represented by its left and right child nodes.

Let $K_{i,L}$ and $K_{i,R}$ be the respective left and right child keys of a key K_i of length n. The key update mechanism in the case of member leave events is described in table 2.2. K_i' is computed with the contribution of n_1 bits of $K_{i,L}$ and n_2 bits of $K_{i,R}$ such that $n_1 \leq n_2$.

At each rekeying interval t_{j+1}:

Compute the left and right contributions of each $K_i^{(j)}$:

$C_{i,L}^{(j)} = PRF_{K_{i,L}^{(j)}}^{(n->n_1)}(K_i^{(j)})$

$C_{i,R}^{(j)} = PRF_{K_{i,R}^{(j)}}^{(n->n_2)}(K_i^{(j)})$

$C_{i,LR}^{(j)} = C_{i,L}^{(j)} || C_{i,R}^{(j)}$

Compute $K_i^{(j+1)}$:

$K_i^{(j+1)} = PRF_{(C_{i,L}^{(j)} || C_{i,R}^{(j)})}^{n_1+n_2->n}(K_i^{(j)})$

Table 2.2: The key update mechanism in a leave event

In order to update $K_i^{(j)}$, the key server broadcasts $E_{K_{i,L}^{(j)}}(C_{i,R}^{(j)})$ and $E_{K_{i,R}^{(j)}}(C_{i,L}^{(j)})$ so that members who know $K_{i,L}^{(j)}$ or $K_{i,R}^{(j)}$, can respectively recover $C_{i,R}^{(j)}$ or $C_{i,L}^{(j)}$ and thus compute the new key $K_i^{(j+1)}$.

As illustrated in figure 2.5, when member R_4 leaves the group, the key K_4 is not used anymore and only K_0 and K_1 are updated using the algorithm described in table 2.2. With respect to our definition we have: $K_{0,L} = K_1$, $K_{0,R} = K_2$, $K_{1,L} = K_3$ and $K_{1,R} = K_4$. Therefore, the key server broadcasts:

$$\{E_{K_1^{(j)}}(C_{0,R}^{(j)}), E_{K_2^{(j)}}(C_{0,L}^{(j)}), E_{K_3^{(j)}}(C_{1,R}^{(j)}), E_{K_4^{(j)}}(C_{1,L}^{(j)})\}$$

Knowing K_0, K_1, K_4 and K_9, member R_3 can only compute $C_{0,L}^{(j)}$ and $C_{1,R}^{(j)}$. When R_3 receives

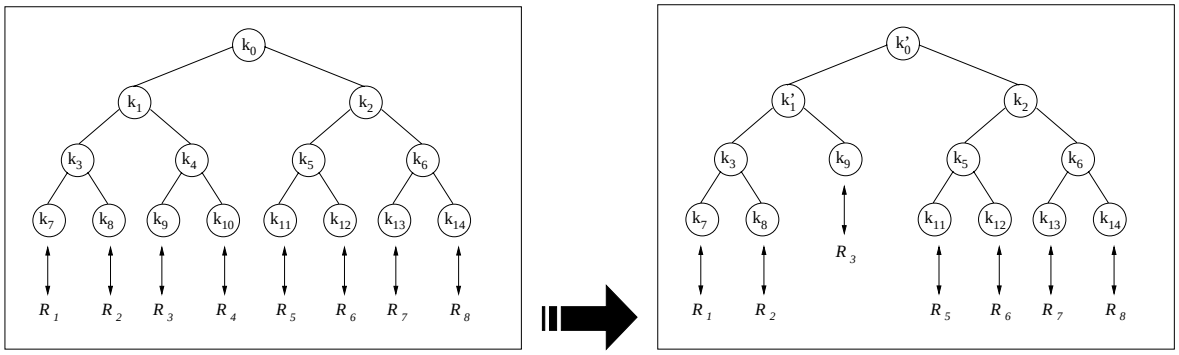


Figure 2.5: Removal of member R_4 with ELK

the broadcast message, it decrypts $C_{0,R}^{(j)}$ with K_1 and $C_{1,L}^{(j)}$ with K_4. Knowing now the left and right contributions of K_0 and K_1 it can compute $K_0^{(j+1)}$ and $K_1^{(j+1)}$ and decrypt future data encrypted with $K_0^{(j+1)}$.

As with the case of multiple joins, in case of multiple leaves occurring in the same interval, the key server aggregates all the leaving members and creates a joint leave key update message. This aggregation can save 50% of the communication overhead [60].

2.4.2 The key recovery mechanism

In ELK, in order to ensure a reliable delivery of the keying material to all group members, some additional information called hints are attached to the encrypted data packets. A member losing an updated key during its delivery still has a chance to retrieve it from the hints. These hints contain some checksum information $h_i^{(j)}$ of length n_3 which is defined by the following algorithm:

$$h_i^{(j)} = V(K_i^{(j)}) = PRF_{K_i^{(j)}}^{(n->n_3)}(0) \tag{2.3}$$

Since any member is able to compute either $C_{i,L}$ or $C_{i,R}$ with regards to its location in the key tree, thanks to the key verification algorithm V defined in equation 2.3, in order to reconstruct a lost key $K_i^{(j)}$, the member first performs a brute-force method to retrieve the missing part of $C_{i,LR}$ and then verifies if $V(K_i^{(j)}) = h_i^{(j)}$. Authors assume that any member can only perform 2^{n1} computations. Since the left and right contributions do not have the same length, in addition to $h_i^{(j)}$, the hint message also contains the least $n_2 - n_1$ significant bits of the right contribution, denoted by $LSB^{(n_2-n_1)}(C_{i,R}^{(j)})$ encrypted with $K_{i,L}^{(j)}$, so that left-hand members only need to perform a brute force search for the first n_1 bits of $C_{i,R}^{(j)}$. The key recovery mechanism is summarized in table 2.3.

At each rekeying interval t_{j+1}:

For each updated key $K_i^{(j+1)}$:

If member knows $K_{i,R}^{(j)}$:

Compute $C_{i,R}^{(j)} = PRF_{K_{i,R}^{(j)}}^{(n->n_2)}(K_i^{(j)})$;

try all 2^{n_1} candidates of x_i until

$x_i = C_{i,L}^{(j)}$;

Compute $K_i^{(j+1)}$;

If member knows $K_{i,L}^{(j)}$:

Compute $C_{i,L}^{(j+1)} = PRF_{K_{i,L}^{(j)}}^{(n->n_1)}(K_i^{(j)})$;

Decrypt $LSB^{(n_2-n_1)}(C_{i,R}^{(j)})$;

try all 2^{n_1} candidates of x_i until

$x_i | LSB^{(n_2-n_1)}(C_{i,R}^{(j)}) = C_{i,R}^{(j)}$;

Compute $K_i^{(j+1)}$;

Table 2.3: The key recovery mechanism

If we reconsider the example illustrated in figure 2.5, when member R_4 leaves the group, R_3 needs to update $K_0^{(j)}$ and $K_1^{(j)}$ and needs to receive $C_{0,R}^{(j)}$ and $C_{1,L}^{(j)}$ from the key server. If we suppose that R_3 does not receive $C_{1,L}^{(j)}$ from rekeying packets, it computes a candidate x_i at most 2^{n_1} times for the right contribution. It then retrieves the corresponding checksum information $h_1^{(j+1)}$ and finds the correct key $K_1^{(j+1)}$ such that $V(K_1^{(j+1)}) = h_1^{(j+1)}$.

2.4.3 Evaluation of ELK

Security evaluation

ELK's rekeying operations for each join and leave offer backward and forward secrecy. The security of the scheme depends on the security of the encryption algorithm and the security of the pseudo-random functions.

Scalability evaluation

Although in [60] authors assume that the memory and computational overhead at the key server are of lesser concern and only focus on the communication overhead, we evaluate all the costs related to the scalability requirements defined in the previous chapter.

First, as with LKH, the key server constructs a binary key tree $\mathcal{G}$ of height h. In this case, the server needs to store all keys present in the key tree. Thus the memory usage is defined by the following equation:

$$Memcost_{ELK}(\mathcal{G}, server) = \sum_{i=0}^{h-1} 2^i = 2^h - 1 \tag{2.4}$$

The memory usage in terms of number of keys for each member corresponds to the height of the key tree. We thus have:

$$Memcost_{ELK}(\mathcal{G}, member) = h \tag{2.5}$$

As with LKH, since we cannot estimate the exact number of encryption/decryption operations performed by either the server or a member, we define the maximum number of such operations that corresponds to the special case when all the keying material need to be updated. In this case, the server first computes the right and left contributions of each key, then it encrypts all of these contributions. Thus, the maximum computational cost of this rekeying operation in terms of PRF and encryption operations is:

$$Compcost(\mathcal{G}, server) = 2 * (2^{h-1} - 1) = 2^h - 2 \tag{2.6}$$

Moreover, the key server performs $2^{h-1} - 1$ additional PRF operations in order to compute the hint of each updated key. If a member does not lose any rekeying packet, then this member performs $(h-1)$ PRF to compute the possible contributions and $h-1$ decryption operations for decrypting the remaining contributions. Besides, if a member loses all of its keys during the rekeying interval, it then needs to perform at most 2^{n_1} computations for each key in order to retrieve the key from the hints integrated in the rekeying packet. The maximum total computational cost for the retrieval of keys in terms of PRF functions is then:

$$Recover\_cost^{max}_{ELK}(\mathcal{G}, member) = (h-1)2^{n_1} \tag{2.7}$$

The communication overhead is optimized for joins, since no rekeying packets need to be broadcast. In case of a leave, the key server sends the left and right contributions of each updated key, that is $n_1 + n_2$ bits per key. In addition to this overhead, the key server broadcasts n_3 additional bits per key added in the encrypted data packets. In the following equation, we define the maximum communication overhead in bits. We get:

$$Comcost^{(max)}_{ELK}(\mathcal{G}) = (2^{h-2} - 2)(n_2 + n_1) + (2^{h-1} - 1)n_3 \tag{2.8}$$

Evaluation of ELK in terms of reliability

Most of key management protocols separate key update messages and encrypted data packets. The receiver must receive both of them to read the encrypted data. In ELK, the rekeying operation resulting from member joinings reduces the degree of the inter-dependency between keys. Hence, when a recipient joins the group, each member computes its updated keying material without receiving any rekeying packet from the key server. Moreover, in order to offer **soft-time reliability**, a small amount of key update information is added within the encrypted data. Thanks to this method, in case of losses, members are able to recover their keys at the time they receive the encrypted data.

Conclusion

Table 2.4 summarizes ELK's properties with respect to the requirements described in the previous chapter. Although authors of ELK assume that the computational overhead is a lesser concern than the communication overhead, we observe that the communication overhead decreases whereby the computational overhead of each member increases with respect to LKH (additional PRF operations are performed in order to compute contributions of each updated key).

2.5 WKA-BKR

The remaining two techniques presented in this chapter deal with the reliability enhancements suited to LKH. Since LKH is proven to be the most efficient technique in terms of scalability, some studies have focused on this rekeying protocol and proposed different reliability schemes using either proactive replication or proactive FEC. The classical proactive replication where each packet is replicated with a certain degree r might be highly inefficient for multicast applications where the group is large since replicating each packet several times before any feedback can be very costly [66]. In the case of LKH, this technique even becomes less efficient because some encrypted packets only are required by a small number of members and some others can be required by almost all members. In order to construct an efficient reliable key delivery protocol, the key server first needs to separate keys with respect to their popularity and regroup them in different packets.

Authors in [69] have designed the **WKA-BKR** protocol standing for "**Weighted Key Assignment - Batched Key Retransmission**". This protocol uses proactive replication where the degree of replication of each packet depends on the position in the key tree of the keys included in the packet. The key delivery protocol is divided into two phases:

- **the key transmission phase** where the key server first defines the algorithm to construct

Security:
ELK offers backward and forward secrecy;

Scalability:
Memory usage:
$O(log(N))$ for a member, $O(N)$ for the key sever;

Communication overhead:
no messages for join events;

leave event: $((n_1+n_2)+k(n_3+n_2-n_1))$, k being the number of updated keys;

Computation overhead:
server:
$O(N)$ encryption and PRF operations;
member:
$O(log(N))$ encryption and PRF operations;
at most 2^{n_1} PRF operations for each lost key;

Reliability:
offers eventual reliability and soft-time reliability.

Table 2.4: ELK summary for a group of size N

packets containing some encrypted keys. It then associates each packet with a replication factor. This process is called the **Weighted Key Assignment (WKA)**. Original packets and replicates resulting from this process are transmitted;

- **the retransmission phase** during which the key server generates new packets with the BKR algorithm in response to NACKs received from clients and multicasts them to the group. This phase continues until all members have received their keys.

In the next section, we present the WKA algorithm which exploits the property that some keys are more valuable than others and defines the replication degree of a key based on its location in the key tree.

2.5.1 The Weighted Key Assignment

As opposed to classical data transmission schemes where packets are intented to all members of a multicast group, in the LKH scheme, some encrypted keys are not required by a certain number of members. This number depends on the location of the key in the key tree. Hence, the higher a key in the key tree, the more the number of members that need this key. For example,

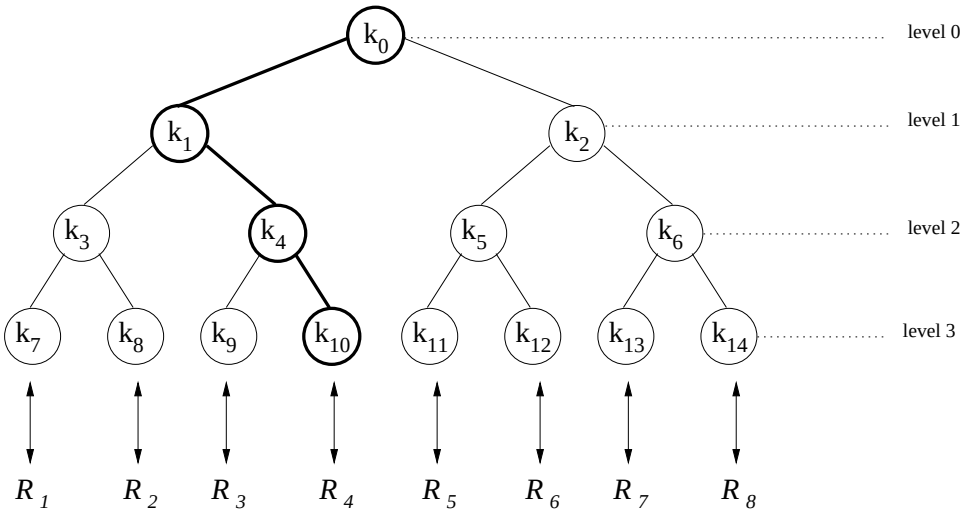


Figure 2.6: A simple LKH tree

at every rekeying operation, all members need to receive the new data encryption key. Since the number of members needing to receive a certain key increases with regards to the localization of this key in the key tree, the expected number of NACKs proportionally increases also. The basic idea behind WKA is that keys residing at higher levels of the key tree are regrouped in packets that are multicasted several times while keys at lower levels are multicast just once.

Description

At each rekeying interval, the key server first defines three sets S_1, S_2 and S_3 that regroup encrypted keys with regards to their degree of importance and two arbitrary threshold levels l_1 and l_2. Encrypted keys that reside at or above level l_1 are placed in S_1 and remaining keys in S_2. Additionally, S_3 regroups keys localized at levels between l_1 and l_2. The level of a key K_i is computed following equation 2.9 where d denotes the outdegree of the key tree and i the id of the corresponding node.

$$level(K_i) = \lceil i/d \rceil \tag{2.9}$$

This classification allows the key server to distinguish popular keys from others that are less popular. These sets being defined, the key server associates a weight w_i for each set S_i that corresponds to the replication degree of corresponding packets. Since the number of members interested to keys depends on the level of these keys, the key server determines w_i such that $w_1 > w_2 > w_3$. These parameters should be optimal since the rekeying cost depends on them.

Based on the example in figure 2.6 where R_4 leaves the group and assuming that the key server sets $l_1 = 0$ and $l_2 = 2$, the key server defines the following three sets:

- $S_1 = \{E_{k'_1}(k'_0), E_{k_2}(k'_0)\}$
- $S_2 = \{E_{k_3}(k'_1), E_{k'_4}(k'_1), E_{k_9}(k'_4)\}$
- $S_3 = \{E_{k_3}(k'_1), E_{k'_4}(k'_1)\}$

After this step, for each specific set, the key server needs to construct rekeying packets. For efficiency reasons, authors propose to regroup keys in sets by assigning encrypted keys to a packet in a way such that each member will need to receive at most two packets (one packet regrouping its corresponding keys from S_1 and one packet from S_2 or S_3 to obtain all its keys). However, this assignment algorithm is not defined in **WKA**.

Batched Key Retransmission

At the end of the first round where the key server sent rekeying packets with the specific replication factors, it collects the feedback from all members. In order to again reduce the communication overhead, when it receives a NACK from a receiver, the key server defines the keys needed by the specific member. Since each packet defined at the transmission phase contains several keys where most of them are not needed by the particular receiver, it is not required to retransmit the same packet to the member. The keys needed by all the receivers who have missed a packet in the first round can be regrouped together in a single packet and multicast to them.

Analysis of the protocol

Since this protocol does not modify the LKH scheme in terms of rekeying and only deals with the delivery of the keying material, it does not decrease the security of LKH and offers backward and forward secrecy.

The rekeying architecture being defined, we now turn to the crucial problem of determining the system parameters that are the threshold levels l_1 and l_2 that define the sets S_1, S_2 and S_3 and the weights w_i corresponding to each of these sets. On one hand, increasing l_1 and l_2 increases the communication overhead since the size of S_1 and S_3 increases and the corresponding keys are replicated. On the other hand, when the maximum number of keys are replicated, the number of losses at the end of this round decreases, which reduces the bandwidth consumed in the phase of retransmission. As a result, l_1 and l_2 should be as large as possible for reliability reasons but small enough for efficiency reasons.

Moreover, the replication factors should also be considered optimal. In order to offer an optimized solution, the key server needs to adjust these parameters by computing all possible

rekeying costs including the reliability factor and choose the optimal value that covers the two phases.

Let s_1, s_2 and s_3 be the respective number of rekeying packets constructed from S_1, S_2 and S_3. These values depend on the levels l_1 and l_2 and the keys needed to be updated at each rekeying interval. If n_i denotes the number of keys that need to be modified as part of S_i, we have $s_i \leq n_i * d$. The rekeying cost of the first round is thus defined by:

$$Cost_{WKA} = \sum_{i=1}^{3} w_i s_i \tag{2.10}$$

We denote the expected number of NACKs received from group members at the end of the first round by $Cost_{NACK1stround}$. In their solution, authors define the system parameters such that the sum of $Cost_{1stround}$ and the cost originated from the feedback is optimal. In order to define $Cost_{NACK1stround}$, we define the following events:

- E = "R loses at least one packet at the end of the first round";
- E_1 = "R only needs to receive one packet from S_1 and loses at least one packet";
- E_2 = "R needs to receive one packet from S_2 that is not included in S_3 and loses at least one packet";
- E_3 = "R needs to receive one packet from all sets and loses at least one packet";

In order to define the probability corresponding to each of these events, we also define the following additional parameters:

- p_1: the probability that a member only needs to receive one packet from S_1;
- p_2: the probability that a member does not need to receive a packet from S_3;

Based on these parameters and assuming that p denotes the network loss probability, we have:

$$p(E_1) = p_1 p^{w_1} \tag{2.11}$$
$$p(E_2) = (1-p_1)p_2(1-(1-p^{w_1})(1-p^{w_2})) \tag{2.12}$$
$$p(E_3) = (1-p_1)(1-p_2)(1-(1-p^{w_1})(1-p^{w_2+w_3})) \tag{2.13}$$

Since E_1, E_2 and E_3 are disjoint events we get:

$$p(E) = p(E_1) + p(E_2) + p(E_3) \tag{2.14}$$

The expected number of NACKS received from a group of size N at the end of the first round is then determined by $Cost_{BKR} = N * p(E)$. Based on the rekeying cost defined in equation 2.10 and this expected number of NACKs, the key server finds the required values for threshold levels l_1 and l_2 and the assigned weight for each set of keys w_1, w_2 and w_3 such that the communication overhead is minimized. Since these values do not highly vary, the key server can compute in real time optimal values for each rekeying interval.

2.5.2 Conclusion

WKA aims at offering a reliable rekeying method using proactive replication while optimizing the rekeying cost. The replication degree of a key being updated depends on its location in the key tree with replication degrees increasing with the key's level in the tree. This technique also lends itself to the definition of optimal system parameters based on network loss probability. Thanks to WKA-BKR, the key server minimizes the impact of the **inter-dependency** and the **intra-dependency** among keys on the delivery of rekeying packets and keeps the communication overhead optimized.

2.6 proactive FEC

Based on the results in [51], where authors show that round-based proactive FEC can reduce the delivery latency, Yang et al. propose to use this technique in order to deal with reliable group rekeying [81] as in the Keystone scheme [77]. Because of the specific dependency between the keys of LKH, as explained in section 2.2, the key server must ensure that members receive their updated keying material. Moreover, from all the rekeying packets sent by the key server at a rekeying interval, each member needs only to receive those packets that contain its encrypted keys. Thus, as in the WKA-BKR scheme, in proactive FEC, the key server first needs to define an algorithm to assign keys into packets. We first present the key assignment algorithms proposed in the same paper and then address the reliable delivery of resulting packets with proactive FEC.

2.6.1 Key assignment algorithms

In addition to the assignment algorithm described in section 2.5 used by WKA-BKR where a member can retrieve all of its required updated keying material from at most two packets, in [77], authors suggest the UKA (User oriented Key assignment Algorithm) where each member needs to receive only one rekeying packet regrouping all of its required encrypted keys. Although the aim of this algorithm is clearly defined, the process of regrouping is not described at all. The disadvantage of these two assignment algorithms (WKA and UKA) is that an important number of encrypted keys are duplicated into several packets and such duplications can increase the communication overhead, especially when receiver loss rates are low. In order to reduce this cost as a further requirement, in [81], authors define three different key assignment algorithms where each encrypted key is assigned to only one rekeying packet and is never replicated:

- Depth First Assignment (DFA)
- Breadth First Assignment (BFA)
- Recursive BFA (R-BFA)

With the use of one of these three key assignment algorithms, each member needs to receive a set of rekeying packets in order to retrieve all of the updated keying material it needs. In order to present these algorithms in detail, we use a common example depicted in figure 2.7. In this specific example, members R_1 and R_8 are leaving at the same rekeying interval and thus k_0, k_1, k_2, k_3 and k_6 need to be updated. The server needs to broadcast the following encrypted keys:

$$\{E_{k_8}(k'_3), E_{k_4}(k'_1), E_{k'_3}(k'_1), E_{k'_1}(k'_0), E_{k'_2}(k'_0), E_{k_5}(k'_2), E_{k'_6}(k'_2), E_{k_{13}}(k'_6)\}$$

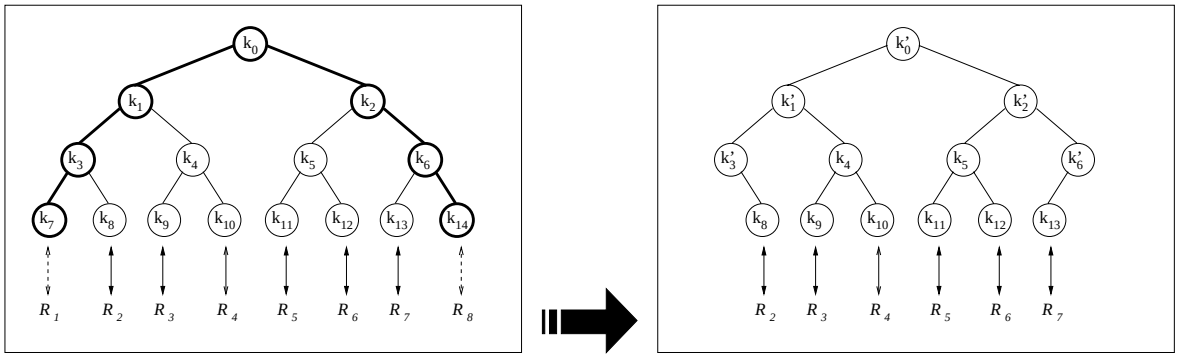


Figure 2.7: Removal of member R_1 and R_8 from G

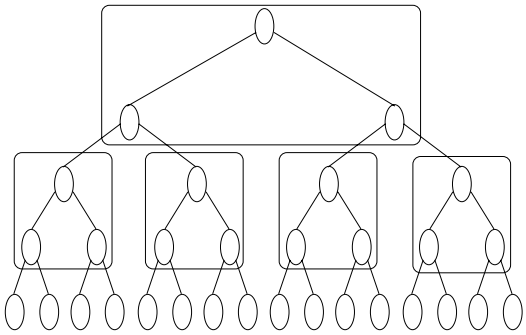


Figure 2.8: An illustration of the R-BFA algorithm

Depth First Key Assignment algorithm (DFA)

As its name implies, this algorithm is based on the Depth-First search method [17] that consists of searching for updated keys vertically. In this case, the key server first collects all the keys required for one member. Based on the example depicted in figure 2.7 where each edge in bold corresponds to an encryption (parent node encrypted by child node), the sequence of encrypted keys a key server will follow to generate rekeying packets is as follows:

$$\{E_{k'_1}(k'_0), E_{k'_3}(k'_1), E_{k_8}(k'_3), E_{k'_1}(k_4), E_{k'_0}(k'_2), E_{k_5}(k'_2), E_{k'_6}(k'_2), E_{k_{13}}(k'_6)\}$$

Breadth First Assignment algorithm (BFA)

Similar to DFA, BFA is based on the Breadth-First search method [17] that consists of searching for updated keys horizontally: the key server first collects encrypted keys of a given level in the tree before it switches to the next level below. Referring again to figure 2.7, the sequence that a key server follows in order to generate rekeying packets is as follows:

$$\{E_{k'_1}(k'_0), E_{k'_2}(k'_0), E_{k'_3}(k'_1), E_{k_4}(k'_1), E_{k_5}(k'_2), E_{k'_6}(k'_2), E_{k_8}(k'_3), E_{k_{13}}(k'_6)\}$$

Recursive Breadth First Assignment Algorithm (R-BFA)

This algorithm is designed in order to gain the benefits of both DFA and BFA. With this algorithm, the key server begins to run the BFA algorithm until the packet is full. Although the scheme is not well defined in [81], authors claim that when a packet is full, the key server runs BFA in the local subtree. The packet construction is illustrated in figure 2.8. In this figure, we observe that BFA is run recursively, that is, each time a new packet has to be constructed.

Referring again to the figure 2.7, the sequence that a key server follows to construct rekeying packets will depend on the capacity of each packet. If we assume here that a rekeying packet

only can transport four encrypted keys then we observe that the order of the keys are the same as in the BFA algorithm:

$$\begin{aligned} P_1 &= \{E_{k'_1}(k'_0), E_{k'_2}(k'_0), E_{k'_3}(k'_1), E_{k_4}(k'_1)\} \\ P_2 &= \{E_{k_5}(k'_2), E_{k'_6}(k'_2), E_{k_8}(k'_3), E_{k_{13}}(k'_6)\} \end{aligned}$$

In [81], authors show that compared to **BFA** and **DFA**, **R-BFA** reduces the average number of packets needed by each user while keeping the variance low.

2.6.2 The reliable delivery of the keying material

The rekeying protocol at the first round

Once the key assignment algorithm is defined, the proactive FEC scheme needs to assure a reliable key delivery. To that effect, a solution based on a combination of FEC and retransmission technique is proposed. Let k be the number of packets a key server has generated by applying the key assignment algorithm and ρ the proactivity factor that will define the number of parity packets. The key server generates $k(\rho - 1)$ parity packets with the encoder $FEC_{encode}(\rho * k, k)$ (refer to definition 2.4) and transmits them with the k original packets.

The retransmission protocol

At the end of this first round, the key server collects the feedbacks from receivers and needs to retransmit some additional packets until all members received their keying material. A member R_i having received neither its required updated keys nor at least k different packets, determines the number of packets required to reconstruct the block and sends this number within a NACK packet. In order to minimize the bandwidth consumption, the key server computes a_{max}, the maximum number of packets lost by receivers, generates a_{max} additional FEC parity packets and multicasts them to all receivers. If at the end of subsequent rounds, member R_i receives at least k distinct packets, it recovers its encrypted keys. This protocol is repeated until all members received their corresponding encrypted keys.

2.6.3 Analysis

The number of parity packets is computed based on the proactivity factor ρ. In order to define this parameter, the key server should know the packet loss probabilities with respect to each receiver which is not possible in the real world. Authors propose to let the key server dynamically adjust ρ at each rekeying interval such that a maximum number of recipients receive their keying material at the first round while keeping the bandwidth overhead low. As in the WKA-BKR scheme, the key server aims at keeping both the expected number of feedbacks and the number of parity packets at a minimum.

Moreover, since *k* may vary from an interval to another one, the key server determines ρ by choosing the largest proactivity factor that still yields the lowest bandwidth overhead. When ρ is large, the key server will send more `parity` packets in the first round and therefore more receivers will receive their packets in the first round and the expected number of feedback packets will decrease.

2.7 Conclusion

In this chapter, we defined two main relationships between keys in LKH that involves a strong requirement in terms of reliability. In order to deal with this problem, we first presented and analyzed **ELK** that takes into account the reliability features within the design of the protocol. In this particular protocol, the key server associates with the multicast data segments some hints in order to ensure that members losing their rekeying packets during the rekeying interval have an additional possibility to recover these keys from the hints. However, this protocol is inefficient in terms of computational overhead since if a member loses a rekeying packet it must perform at most 2^{n_1} operations in order to retrieve the updated key. We then presented and analyzed existing reliable delivery protocols depending on either proactive replication or proactive FEC. In the **WKA-BKR** protocol, each member needs to receive at most two rekeying packets and these packets are replicated with respect to their popularity, while in [81] the key server uses a different assignment algorithm, that is **R-BFA**, where the number of packets required by a receiver slightly increases but the overall bandwidth utilization is low. In this case, the key server uses an FEC encoder in order to generate parity packets in addition to the original packets.

Unfortunately, all the solutions described in this chapter still suffer from the “one affects all” failure due to the fact that each arrival or departure of a single member causes the update of at least one key (ie. the data encryption key) for all group members. Consequently, members who do not leave the group during the entire session can strongly be affected by frequent membership changes. From a commercial point of view, it is unfair for a member who will stay until the end of the session to be treated the same way as with short-lived members. In the next chapter, we deal with this problem and define a new reliable rekeying protocol intended for satellite networks that minimizes the impact of rekeying caused by short-lived members on long-lived members.

Chapter 3

Reliable Group Rekeying with a Customer Perspective

3.1 Customer Satisfaction requirement

While security, scalability and reliability are throughly addressed by existing reliable rekeying approaches, real life requirements such as customer satisfaction are overlooked or not even addressed by most solutions. As explained in section 1.2.4, frequent rekeying operations may have a strong impact on all members alike, regardless of their behavior. Although this problem has partially been analyzed in the case of re-encryption key trees, mainstream solutions based on LKH where the key server does not involve any intermediate node do not take this aspect into account.

Hence, for re-encryption trees, as in [44, 46, 57], the key server physically partitions the group of members into many subsets and defines a hierarchy between them. Each subset of members is managed by a distinct entity and members of a common subset share the same encryption key K_i. In such protocols, when a member joins or leaves the group, it is assigned to a specific subset with respect to its physical location. In this case, only members assigned to this subgroup need to update their keying material and the corresponding Group Security Intermediary (GSI) chooses a new key for this subgroup and sends this new secret to members of its subgroup. During the update of a subgroup's keying material, the keying material of members located at other subgroups is not modified at all and thus these members are not affected by this rekeying operation. In this case, the key server can define a strongly reliable key delivery protocol for the small subset of affected members.

However, in the case of LKH as in other rekeying schemes that do not involve any intermediate element in the security protocol, the same data encryption key is shared by all members. In order to provide backward and forward secrecy, the data encryption key needs to be updated at each arrival or departure of a recipient and all members need to receive the new key. In order

to deal with scalability requirements and minimize the rekeying cost, the key server defines a logical key hierarchy that does not depend on the physical configuration of the network. As described in the previous chapter, members share a subset of their keying material with some other members, including the data encryption key that is located at the root node of the defined key tree. Thus, at each rekeying interval, all members need at least to receive the data encryption key.

Since all members need to update their keying material at each rekeying interval, the key server must minimize the impact of rekeying due to frequent arrivals or departures on members. While designing a rekeying algorithm, the key server must take into account the nature of the service and its policy in terms of customer satisfaction and reliability. As explained in section 1.2.4, we suggest a new approach that takes into account group members' "loyalty" and define the requirement of customer satisfaction (**Requirement 10**) whereby the key server provides a strongly reliable key delivery in order to minimize the impact of rekeying operations caused by mass arrivals or departures of less "loyal" members. In order to implement this approach, we need to define a "loyalty degree" for each member. This definition will depend on the nature of the application and on its specific requirements with respect to the pricing mechanism. In this chapter, we first give a classification of secure multicast applications, define the notion of "loyalty" for each set of applications and then focus on a particular set of applications where "loyal" members are defined with respect to their membership duration. We then propose a new reliable rekeying protocol for such members.

3.2 Classification of secure multicast applications

3.2.1 A taxonomy of multicast applications

In [62], a taxonomy of multicast applications is presented including a classification of multicast applications with respect to the number of servers or receivers (one-to-many, many-to-many, many-to-one). The authors define five categories for one-to-many multicast applications:

- (a) scheduled audio/video distributions: lectures, meetings, ...
- (b) push-media: news headlines, weather updates, ...
- (c) file distribution and caching: web site content, executable binaries, ...
- (d) announcements: network time, multicast session schedules, ...
- (e) monitoring: sensor equipments, manufacturing, ...

Given this classification, authors characterize some application service requirements and position each type of application with respect to these characteristics as depicted in table 3.1.

Applications	Bandwidth requirement	Delay tolerancy	Loss tolerancy
(a) A/V distributions	high	medium	medium
(b) Push-media	low	tolerant	tolerant
(c) File distribution	medium	tolerant	intolerant
(d) Announcements	low	medium	medium
(e) Monitoring	medium	intolerant	intolerant

Table 3.1: Characteristics of multicast applications

This taxonomy is very general in nature and does not highlight the differences between applications with respect to security. We thus propose in the next section a more specific classification of one-to-many multicast applications based on their security requirements and pricing models. We first briefly overview these characteristics and then describe our classification.

3.2.2 The proposed classification with respect to the security requirements

As explained in chapter 1, in the case of dynamic multicast groups where recipients are supposed to join/leave the group frequently, we came up with two security requirements that are not found in traditional secure unicast communications: **backward and forward secrecy**.

Multicast applications may or may not require backward and/or forward secrecy. This requirement will depend on the nature and pricing model of the application. Indeed, in some actual broadcast applications, like current digital TV platforms, customers first subscribe to the service and then periodically pay a fixed fee. In this case, the service provider does not have to know about customers' behavior and customers pay for the whole application regardless of their actual use patterns. On the other hand, some other applications are not subscription based and offer to clients the possibility to pay only the service they really had access to. This kind of pricing model better fits customers' needs. By combining the security requirements and the possible pricing models, we end up with three classes of secure multicast applications [54]:

- (i) **Subscription based applications** where clients pay a fixed fee for the entire session no matter their membership duration or arrival time. In this type of applications, the service provider needs to ensure neither forward nor backward secrecy and rekeying only occurs when there is a need for updating keys to prevent cryptanalysis.
- (ii) **Applications where the pricing mechanism is based on members' arrival time** where clients pay at their arrival for the remaining time until the end of the session. Here, the service provider does not need to strongly ensure forward secrecy, since clients pay for the entire remaining session. However, it needs to provide backward secrecy in order to prevent clients from accessing the portion of the session prior to their arrivals.
- (iii) **Applications where the pricing mechanism is based on members' membership duration** where clients only pay for the service they really had access to. In this kind of

applications, the service provider needs to ensure both backward and forward secrecy.

Category	Backward secrecy requirement	Forward secrecy requirement
(i)	low	low
(ii)	strong	low
(iii)	strong	strong

Table 3.2: Security requirements of secure multicast applications

In this chapter, we deal with applications where the service provider must ensure backward and forward secrecy (ie. applications of the category **(iii)**). Consequently, the degree of "loyalty" of any member depends on its membership duration. The service provider should offer a better reliability assurance for long-lived members.

In order to provide customer satisfaction, we propose in [52] to partition members with respect to their membership duration and provide a strongly reliable key delivery for long-lived members. In section 3.3, we describe the proposed partitioning method and the resulting new rekeying protocol. In section 3.4, we propose a new hybrid reliable delivery protocol that ensures that long-lived members receive their keying material before the reception of the corresponding encrypted multicast data. We then discuss the choice of the system parameters that are critical for the implementation of the proposed protocol. Finally, in section 3.6, we present two proposals that are somehow related to our strategy and compare them with respect to the proposed protocol.

3.3 Partitioning members with respect to membership duration

3.3.1 Partitioning

In [4], Almeroth et al. observed the group members' behavior during an entire multicast session. The authors realized that members leave the group either very shortly after their arrival or at the end of the session. We thus define two real categories to distinguish members:

- **short-duration members** are supposed to leave the group very shortly after their arrival.
- **long-duration members** are on the opposite supposed to stay in the group during the entire session.

Based on these observations, for each real category, membership duration can be represented by an exponential distribution [68] where the mean of the distribution of membership for **short-duration** and **long-duration** members are denoted by M_s and M_l respectively, where $M_l >> M_s$. In the following equations, we define the probability density functions of the membership duration with respect to each member category. For a **short-duration** member we have:

$$f(t) = \frac{1}{M_s} e^{-t/M_s} \tag{3.1}$$

Similarly, the probability density function of the membership duration for **long-duration** members is defined by the following equation:

$$f(t) = \frac{1}{M_l} e^{-t/M_l} \tag{3.2}$$

Since the key server cannot predict the time a member will spend in a multicast session, it cannot determine if a member belongs to the short-duration category or the long-duration one. Thus, we propose to partition members into two monitored categories, based on the key server's subjective perceptions, called **volatile** and **permanent**.

In this proposed partitioning, we define a certain threshold time that separates these two categories. A new coming member is first considered to be **volatile**. If this member spends more than this threshold time in the group, then it becomes **permanent**. Let t_{th} denote the defined threshold time. The key server thus assigns members to one of the two separate sets as follows:

- **volatile** members whose membership duration is less than t_{th};
- **permanent** members whose membership duration has exceeded t_{th}.

Thanks to this partitioning, the key server tries to avoid or at least minimize the impact of rekeying operations caused by **volatile** members on **permanent** members. **Permanent** members will not be affected by arrivals or departures of **volatile** members but only by departures of members from their subgroup which is supposed to be quasi-static. The reliability processing of each monitored category will be different and in order to ensure customer satisfaction (**requirement 10** described in chapter 1), the key server must guarantee to almost all **permanent** members the delivery of the keying material with a very high probability and before the reception of the multicast data encrypted with these keys. In the following section, we explain the rekeying mechanism with respect to this partitioning scheme.

3.3.2 Rekeying the two monitored sets

As explained in the previous section, members are separately regrouped in two disjoint sets. Since the key server cannot predict the time a new coming member R_i will spend in a multicast session, R_i first is considered as a **volatile** member until its membership duration reaches the defined threshold time t_{th} where R_i joins the logical set of **permanent** members. **Volatile** and **permanent** members are respectively regrouped in two key trees denoted by G_v and G_p, with $k_{v,0}$ and $k_{p,0}$ being keys located at the root of each tree (see figure 3.1). Unlike the classical key tree approach, $k_{v,0}$ and $k_{p,0}$ are different from the data encryption key k_0 that is shared by all members regardless of their membership duration.

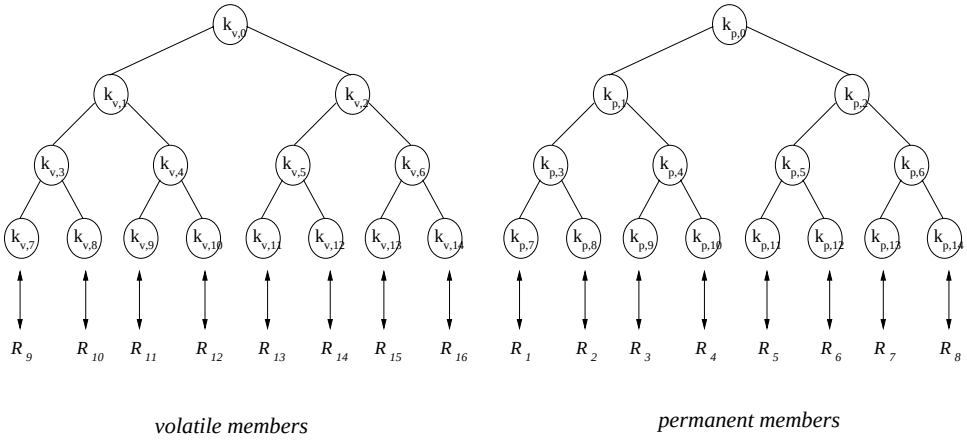


Figure 3.1: The proposed rekeying scheme

G_v and G_p are managed and updated separately and periodically. Requests are thus collected in batch. Assuming that **volatile** members' arrivals and departures will happen very frequently, the key server sets the corresponding rekeying interval T_v to a value as short as possible. On the other hand, since **permanent** members are assumed to stay longer in the group, the corresponding rekeying interval T_p will be set to be much longer than T_v. Consequently, **permanent** members are not involved in the rekeying of G_v.

A new coming member R_i first joins the tree representing **volatile** members G_v and receives the actual data encryption key and its keying material. Every T_v, R_i receives the new data encryption key and the necessary information to update its keys. When R_i's membership duration reaches t_{th}, it is directly transfered to the key tree representing **permanent** members and it receives the new K_{data} and its new set of keys without waiting the next T_p. After its transfer to G_p, R_i updates its keying material every T_p.

However, since k_0 is shared by all members, this key needs to be modified while rekeying **volatile** members in order to ensure forward and backward secrecy. **Permanent** members still could be affected by losses resulting from such rekeying operations at each T_v. Thus, during each T_p whereby no rekeying for **permanent** members takes place, an additional feature of our scheme allows these members to retrieve new data encryption keys, resulting from rekeying

operations at each T_v, from their local keying material and without receiving any information from the key server. In order to implement such an algorithm we define $T_p = \delta * T_v$, δ being a positive integer.

In addition to the classical **Join** and **Leave** events, we also consider a third event that we define as the **Transfer** event, where a **volatile** member becomes **permanent** and is transfered to the corresponding key tree. This event may happen each time the key tree regrouping **volatile** members is updated. In the sequel of this section, we describe the rekeying mechanism for each key tree.

Rekeying volatile members

The update mechanism of the key tree G_v is exactly the same as with the classical LKH scheme whereby rekeying operations occur as a result of batched join/leave events. Consequently, at each T_v, some of the keys in G_v are updated, encrypted and multicasted with required keys with respect to joining and leaving members. If the membership duration of a certain member R_i reaches the defined threshold time value, then R_i has to be transfered in G_p. The key server treats the **transfer** as a **leave** because when a member is transfered to the **permanent** key tree, if it is not revoked from the **volatile** key tree, it will still have access to the keying material of the **volatile** key tree. With these keys, it could still have access to the multicast data even if he subsequently left the **permanent** key tree.

Rekeying permanent members

The update mechanism of G_p is slightly different than LKH's one. Rekeying operations take place at each T_p and the key server does not need to strongly ensure backward secrecy for this specific key tree. Since transfered members have already registered to the service for at least t_{th} time, they can have a restricted right to access to the previous keys in G_p when becoming **permanent**. Thus, members becoming **permanent** do not wait until the end of the corresponding T_p, but can join the key tree at each T_v and receive their rekeying material related to G_p. In this case, the key server sends to these transfered members their new keying material at the time of their transfer and does not perform any rekeying operation. Thus remaining **permanent** members are not affected at all by joining members. However, in order not to cause an exposure with respect to backward secrecy in a larger time interval, at each T_p, after having treated **leaves**, all members update unchanged keys by computing new corresponding values based on the following operation:

$$k_{p,i}^{(j+\delta)} = PRF(k_{p,i}^{(j)}) \tag{3.3}$$

The key server thus does not need to send any of those keys.

The update of the data encryption key k_0

Since the data encryption key k_0 is included in the keying material of all members alike, this key needs to be modified at each T_v. Volatile members leaving the group should not have access to the content of encrypted data more than $2 * T_v$. Since **permanent** members are also affected by this update, the key server determines for them the data encryption key retrieval algorithm so that during a rekeying interval T_p they do not need to receive any rekeying packet at all. The key retrieval algorithm for a permanent member at each T_v during one T_p is described as follows:

$$k_0^{(j+1)} = PRF(k_{p,0}^{(j)}, k_0^{(j)}) \tag{3.4}$$

PRF denotes a pseudo-random function (see [28] for further details) and thanks to the use of such functions this algorithm provides forward secrecy for **volatile** members since they do not have the knowledge of $k_{p,0}$.

Summary

Figure 3.2 summarizes the rekeying mechanism that the key server implements for secure multicast applications. Thanks to the partitioning scheme, the key server offers a better quality of service to **permanent** members since the corresponding key tree is less frequently updated and the set of **permanent** members is assumed to be quasi-static. The proposed rekeying protocol does not decrease the security of basic LKH scheme, since backward and forward secrecy are provided every T_v for volatile members and every T_p for **permanent** members. For the sake of clarity for the reader, we recall that a **transfer** is considered as a **Leave** for G_v and a **Join** for G_p.

3.4 Reliability features

3.4.1 Introduction

As explained in the previous chapter, any key distribution scheme should provide a reliable delivery mechanism. Each member of the group should be able to receive its entire keying material in order to be able to access the content of multicast data. In the case of the LKH scheme, this reliability issue becomes a more important problem because of the **inter-dependency** and **intra-dependency** among keys defined in the previous chapter. In the protocol that we proposed and described in the previous section, we define a separate key tree for **volatile** and **permanent** members and to ensure **Requirement 10**, our goal is that almost every permanent member receives all of its keying material before the end of the corresponding rekeying interval, ie. before

At each $T_p = \delta * T_v$:
collect **Member Leave** events from **permanent** members;
update G_p;
Compute $k_0^{(T_p)}$;
multicast $E_{k_{p,0}^{(T_p)}}(k_0^{(T_p)})$ and $E_{k_{v,0}^{(T_p)}}(k_0^{(T_p)})$;
At each T_v:
collect **Member Join** and **Member Leave** events from volatile members;
update G_v;
collect and treat **Member transfer events**;
$k_0^{(T_v)} = PRF(k_{p,0}^{(T_v-1)}, k_0^{(T_v)})$;
multicast $E_{k_{v,0}^{(T_v)}}(k_0^{(T_v)})$ to volatile members;

Figure 3.2: The proposed rekeying protocol with two membership classes

the reception of the corresponding encrypted multicast data. In this section, we deal with the reliability scheme specific to permanent members.

In the previous chapter, we presented two techniques that deal with the reliability features related to LKH based either on proactive replication or proactive FEC. We cannot directly implement these techniques for the following reasons:

- in satellite networks, packets in a control plane mostly are defined as being ATM cells whose length is 53 bytes [74]. Therefore, the key assignment algorithms used by previously described reliability techniques cannot be implemented in satellite networks;
- in existing reliability techniques, the key server does not take into account the customer satisfaction and chooses system parameters in order to minimize the rekeying cost. In the proposed reliability scheme, we aim at offering a strongly reliable delivery of the keying material for **permanent** members.

Moreover, in the previously described protocols, a corrupted packet is considered as lost. However, the greater the number of keys in a packet, the more the probability of corruption in this packet. Since one packet targets a certain number of members, many members can be affected from only one packet loss. Based on these observations, we propose to set the number of keys assigned in a packet to one.

We need to propose a new reliable delivery mechanism while ensuring a very low probability of loss for the maximum number of permanent members. The key server must thus ensure a strongly proactive reliable delivery for permanent members, in order to reduce the number of losses. In the proposed solution, since there is always a probability of loss, the key server first defines some bounds on the expected losses. It defines α, β, such that α denotes the portion of

permanent members that receive their keying material with probability at least as high as β. Given the number of **permanent** members N_p, the key server must ensure that the probability that more than $(1-\alpha)N_p$ of **permanent** members will not receive their corresponding keying material must not exceed $(1-\beta)$. Assuming X is a random variable representing the number of **permanent** members that do not receive all of their corresponding rekeying packets, the previous requirement can be expressed by the following inequality:

$$P\{X > (1-\alpha)N_p\} \leq (1-\beta) \tag{3.5}$$

Based on this equation and the specific characteristics required for satellite networks, we propose in the next section a reliability scheme for **permanent** members that both uses proactive FEC and proactive replication.

3.4.2 Hybrid reliability scheme

The proposed hybrid reliability scheme is based on proactive FEC. Since we consider that only one key is assigned to a packet, we do not implement any key assignment algorithm for packets. Besides, we need to define the structure of FEC blocks. In order to improve performance at the member's side, we assume that a member should perform at most one FEC_{decode} operation per rekeying interval. Therefore, any **permanent** member will receive its complete required updated keys from one FEC block of rekeying packets. In the following section, we formally define our specific key assignment algorithm for blocks.

User-oriented Key Assignment for FEC blocks

Since LKH provides a hierarchical structure between keys and since members that are in a same subtree share a certain number of keys, we propose to split G_p into disjoint subtrees and define a specific block for members of each subtree. We denote d_p and h_p as the respective outdegree and depth of G_p. A member needs to receive at most $h_p - 1$ keys from G_p and the data encryption key. With respect to the chosen subtree, members share the keys associated with the nodes that are in the path from the root of G_p to the parent of the root of the resulting subtrees. We define the block size of a rekeying packet based on the parameter a such that $(d_p)^a$ defines the number of members of the specific subtree. In this case, the block regroups:

- the data encryption key encrypted with the actual $k_{p,0}$;
- all the encrypted keys in the path from the parent of the root node of the subtree to the root of G_p ($(h_p - 1 - a)$ keys). They are shared by all the corresponding members and thus are encrypted only once for this specific block;

- all the keys from the subtree except those at leaf nodes, encrypted with keys at children nodes: $\sum_{i=1}^{a}(d_p)^i$.

The block size is then summarized by the following equation (for the sake of simplicity, h and d respectively denote h_p and d_p):

$$\begin{aligned} b(a) &= 1+h-1-a+\sum_{i=1}^{a} d^i \\ b(a) &= h-a+\frac{d^{a+1}-1}{d-1}-1 \end{aligned} \tag{3.6}$$

The real size of the FEC block of keys will depend on the location of leaving members but will not exceed the value determined in equation 3.6.

For example, in figure 3.3 where G_p is a binary key tree of depth 4, when the key server sets $a = 1$, it defines one block per two members. In this case, the key server partition members in four separate subtrees:

- $Block_1^{(a=1)}$ regroups all rekeying packets needed by R_1 and R_2;
- $Block_2^{(a=1)}$ regroups all rekeying packets needed by R_3 and R_4;
- $Block_3^{(a=1)}$ regroups all rekeying packets needed by R_5 and R_6;
- $Block_4^{(a=1)}$ regroups all rekeying packets needed by R_7 and R_8;

With this key regrouping method, some rekeying packets will appear in several blocks. Hence, if for example k_0 needs to be updated, members R_1, R_2, R_3 and R_4 must receive $E_{k_1}(k'_0)$. If the key server sets $a = 1$ as illustrated in figure 3.6, this rekeying packet will appear in $Block_1^{(a=1)}$ and $Block_2^{(a=1)}$. Therefore, if R_1 loses this specific rekeying packet from its corresponding block, it can get it from $Block_2^{(a=1)}$ without performing any additional operation. Thus, our scheme inherently combines proactive FEC and proactive replication techniques, and lets members who have not received their rekeying packets from their block, retrieve remaining keys from other blocks.

Further to the assignment of keys to blocks, the key server uses the $FEC_{encode}(b, b+r)$ and generates r parity packets for each block. For example, if the key server sets $a = 2$, in the rekeying interval t_{j+1}, $Block_1^{(a=2)}$ regroups all the keys from the corresponding subtree encrypted with the keys located in children nodes and the data encryption key. We thus have:

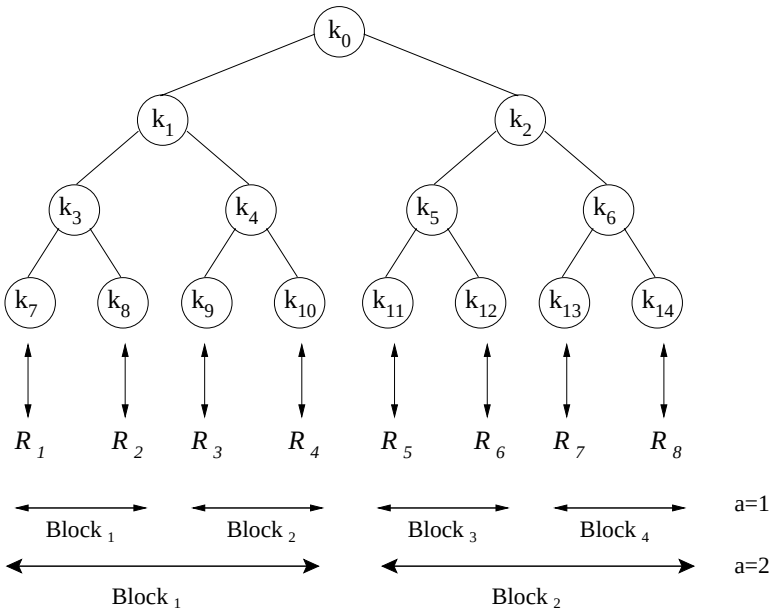


Figure 3.3: An illustration for the User Oriented key assignment algorithm where $a = 1$ and $a = 2$

$$
\begin{aligned}
Block_1^{(a=2)} \;=\; & \{E_{k_p^{(j+1)}}(k_0^{(j+1)}), E_{k_{p,1}^{(j+1)}}(k_p^{(j+1)}), E_{k_{p,3}^{(j+1)}}(k_{p,1}^{(j+1)}), \\
& E_{k_{p,4}^{(j+1)}}(k_{p,1}^{(j+1)}), E_{k_{p,7}^{(j)}}(k_{p,3}^{(j+1)}), E_{k_{p,8}^{(j)}}(k_{p,3}^{(j+1)}), E_{k_{p,9}^{(j)}}(k_{p,4}^{(j+1)}), E_{k_{p,10}^{(j)}}(k_{p,4}^{(j+1)})\}
\end{aligned}
$$

Then, the key server encodes $Block_1^{(a=2)}$ and as illustrated in figure **??** generates r parity packets. The whole block including the parity packets are then transmitted to the corresponding members.

Providing Customer Satisfaction

Thanks to this user-oriented key assignment algorithm, we can now redefine the equation 3.5. We remind that the key server needs to assure the required degree of reliability to **permanent** members independently of the number of leaving members in both subgroups. The worst rekeying cost corresponds to the case where all keys of the **permanent** key tree, except those associated with leaf nodes need to be modified. In this case, for every d members, one member leaves the group and thus each of the $(d-1)$ remaining members needs to receive all h keys located on the path from the root to the corresponding leaf except the individual key. We define the following event:

- E = “member R receives its h packets from its corresponding block”

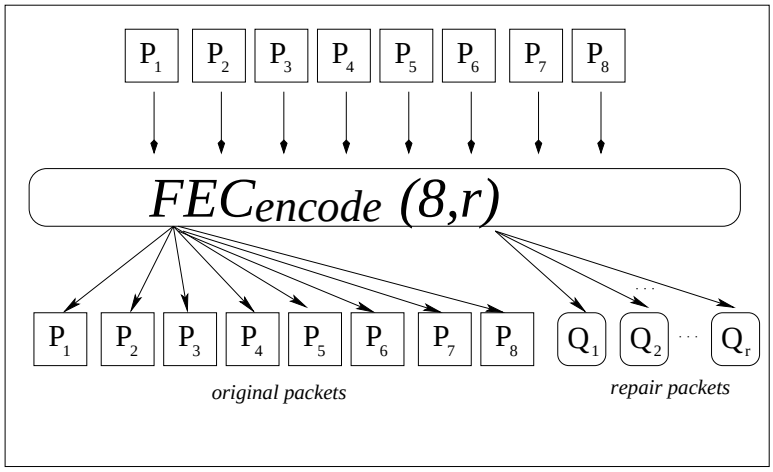


Figure 3.4: The reconstruction of $Block_1^{(a=2)}$ after the use of FEC_{encode} where $b = 8$

Assuming that the packet loss probability per member is independent and identical for each member, we can translate the equation 3.5 to the equation 3.7 stating that the probability that at most $(1-\alpha)N_p$ members lose their keys is at least β:

$$\sum_{i=0}^{(1-\alpha)N_p} \binom{N_p}{i} (1-p(E))^i p(E)^{N_p-i} \geq \beta \tag{3.7}$$

$p(E)$ depends on the independent packet loss probability p, and the block size defined during the user oriented key assignment algorithm. The resulting value must follow the source's reliability requirements defined in equation 3.7. Let b be the original block size and r the number of parity packets for one FEC block. In order to compute $p(E)$ we define the following events:

- E_1 = "R receives at least b packets from its block and thus can recover all its h packets"
- E_2 = "R receives less than b packets but receives all its h packets "

We have:

$$p(E_1) = \sum_{l=b}^{b+r} \binom{b+r}{l} (1-p)^l p^{b+r-l} \tag{3.8}$$

$$p(E_2) = (1-p)^h \sum_{l=h}^{b-1} \binom{b+r-h}{l-h} (1-p)^{l-h} p^{b+r-l} \tag{3.9}$$

Since E_1 and E_2 are disjoint events we get

$$p(E) = p(E_1) + p(E_2) \tag{3.10}$$

Given this definition, in order not to increase the communication overhead, the key server chooses the lowest r value satisfying the inequality 3.7. Besides, since some keys inherently are replicated in other blocks and their degree of replication depends on their localization in the key tree, the probability that a member receives all required keying material is even higher than $p(E)$.

The definition of the block size

Given α and β, the key server first needs to choose the partitioning degree of **permanent** members in order to define the corresponding block size and the relevant number of repair packets. However, there exists a tradeoff in the choice of b. We illustrate this tradeoff with the following example: we assume that $N_p = 65536$, $d = 4$ and $p = 0.1$ and the key server needs to assure that 99.99% of **permanent** members receive their keys with a probability greater than 99.99%. We then have $h = log_4(65536) + 1 = 9$ and $\alpha = \beta = 0.9999$. Table 3.3 gives a comparison of the rekeying cost with different block sizes in the worst case where the initial rekeying cost is defined by the following equation:

$$Cost_{init} = \frac{d^h - 1}{d - 1} \tag{3.11}$$

Initial Cost	Cost with the proposed scheme			
	b(1) = 12	b(2) = 27	b(3) = 90	b(4) =345
87382	344064	167936	119808	105728

Table 3.3: Rekeying cost where $N = 65536$, $p = 0.1$,$\alpha = \beta = 0.9999$

In this table, we realize on one hand that the rekeying cost decreases when the block size increases. However, in order to avoid excessive buffering at members, the block size needs to be chosen as small as possible. Hence, in this specific example, member R needs to receive at least its $h = 9$ rekeying packets. If R does not receive all of its rekeying packets, it needs to receive at least b packets in order to use the FEC_{decode}. If the key server sends blocks of $b(3) = 90$ packets, this means that if R does not receive all rekeying packets destined to it, it has to receive at least 90 packets in order to reconstruct the 9 packets required for rekeying.

3.5 How to define system parameters?

3.5.1 Introduction

Now that the global rekeying architecture is defined, we turn to the crucial problem of determining the values of T_v, T_p and t_{th}. On one hand, to increase the quality of service, the key server needs to increase as much as possible T_v and T_p to be able to offer almost fully reliable delivery of keying material. On the other hand, increasing these values implies to let more extra-time to leaving members since rekeying is processed in a batch for efficiency reasons. As a result, T_v and T_p should be as small as possible for security reasons but large enough to offer a better service to **permanent** members. In order to offer to them an almost fully reliable delivery of keying material, the key server needs to adjust these parameters by computing the rekeying cost of each category (including additional packets to offer reliability). We first introduce the evaluation of the rekeying cost for **volatile** and **permanent members** and then present the method used to determine T_v, T_p and t_{th}.

3.5.2 Restrictions on T_v and T_p

In order to determine T_v and T_p, the key server first must have some knowledge about the rekeying cost per interval in advance. Let L_v and L_p be the number of respectively **volatile** and **permanent** members. Since the rekeying cost in one rekeying interval should not exceed the reserved bandwidth, we can write the following inequality where B denotes the bandwidth limited for rekeying operations and $cost(L_v)$ the rekeying cost resulting from L_v members leaving the **volatile** key tree:

$$cost(L_v) < B \times T_v \tag{3.12}$$

Symmetrically, the key server needs to adjust T_p in order to assure a high degree of reliability for **permanent** members. Let $cost(L_p)$ denotes the rekeying cost for a given interval T_p. Assuming that $T_p = \delta T_v$ the key server also needs to satisfy the following inequality which yields a lower bound on T_p:

$$\delta \times cost(L_v) + cost(L_p) < B \times T_p \tag{3.13}$$

In order to satisfy equations 3.12 and 3.13, the key server must compute or estimate $cost(L_v)$ and $cost(L_p)$ with respect to requirements related to customer satisfaction.

We remind that membership durations are represented by an exponential distribution where the mean duration of membership for short-duration and long-duration members are denoted

by M_s and M_l. In order to define T_v, we propose to define $cost(L_v)$ as the average rekeying cost for a mean number of **volatile** members leaving G_v. Symmetrically, the key server needs to adjust T_p in order to assure an arbitrarily high degree of reliability to **permanent** members independently of the number of leaving members in this group. Hence, the key server adjusts T_p based on the maximum rekeying cost ($cost(L_p)$) in order to take into account the worst case. We first give the evaluation of the average rekeying cost for volatile members and then define the worst rekeying cost for permanent members.

Evaluation of the rekeying cost for volatile members

The mean number of members leaving G_v is the sum of the average number of leaving members from the two real categories. In the case of an exponential distribution with mean M, the probability that a member leaves at T_v is:

$$p(t \leq T_v) = 1 - e^{-T_v/M} \tag{3.14}$$

Given the ratio of short-duration members over the total group size that is denoted by γ, the mean number of **volatile** members leaving G_v therefore is:

$$L_v = \gamma N(1 - e^{-T_v/M_s}) + (1-\gamma)N(1 - e^{-T_v/M_l}) \tag{3.15}$$

In order to define the average rekeying cost, we need to compute the probability that a key needs to be updated and encrypted with one of the keys at children nodes. For the sake of simplicity, we define the position of a key in the key tree by its level l and its lateral offset s that is incremented from left to right. The probability that a key $k_{l,s}$ is encrypted with one of the keys at level $l+1$ is the probability that at least one member is leaving the subtree whose root node is the child node of $k_{l,s}$. We thus first define $p(l,s)$ as the probability that at least one member needs to receive one of the following rekeying packet:

$$E_{k_{l+1,s+1}}(k_{l,s}), E_{k_{l+1,s+2}}(k_{l,s}), ..., E_{k_{l+1,s+d}}(k_{l,s})$$

$p(l,s)$ also is the probability that at least one member assigned to a node in the corresponding subtree leaves the group. However, if all members assigned to this subtree leave the group, then the corresponding rekeying packets do not need to be sent. We again define two events in order to compute $p(l,s)$:

- $E_3^{(l,s)}$: "At least one member leaves the subtree whose root node's level is $l+1$"

- $E_4^{(l,s)}$: "All members of the subtree whose root node's level is $l+1$ leave the group"

Based on these two events we have $p(l,s) = p(E_3) - p(E_4)$. E_3 is the complement of the event "No member leaves the subtree whose root node's level is $l+1$". We thus have:

$$p(E_3) = 1 - \frac{\binom{N-d^l}{L_v}}{\binom{N}{L_v}} \tag{3.16}$$

If $d^l > L_v$, then there is always a probability that a member stays at the subtree. Therefore $p(E_4)$ is defined by the following equation:

$$\begin{aligned} p(E_4) &= 0 \text{ if } d^l > L_v \\ p(E_4) &= \frac{\binom{L_v}{d^l}}{\binom{N}{d^l}} \text{ if } d^l \leq L_v \end{aligned} \tag{3.17}$$

Finally, $p(l,s)$ is:

$$p(l,s) = 1 - \frac{\binom{N-d^l}{L_v}}{\binom{N}{L_v}} - \frac{\binom{L_v}{d^l}}{\binom{N}{d^l}} \tag{3.18}$$

The average rekeying cost without reliability mechanism is then:

$$cost(L_v) = \sum_{l=0}^{h-1} \sum_{s=0}^{d^l-1} 1 - \frac{\binom{N-d^l}{L_v}}{\binom{N}{L_v}} - \frac{\binom{L_v}{d^l}}{\binom{N}{d^l}} \tag{3.19}$$

Evaluation of the rekeying cost for permanent members

As opposed to the case of **volatile** members, the key server defines T_p such that it follows the equation 3.13 even in the worst case. Thus, we need to define the worst rekeying cost of G_p. This value corresponds to the case where all the keys except those at leaf nodes need to be updated. Given a, $b(a)$ and r, this value is determined by the following equation:

$$cost(L_p) = (b(a) + r) * \frac{N_p}{d^a} \tag{3.20}$$

Example

We assume that $N = 65536$ where 50% of the group are short-duration members with $M_s = 3$ minutes and $M_l = 3$ hours. The bandwidth reserved for rekeying is limited to 1 Mbps and the loss probability of a rekeying packet for each member is independent and equal to $p = 0.1$. Based on the optimization method, we then compute system parameters for an objective defined by a target probability for the rekeying rate as perceived by a large fraction of **permanent** members. We set the block size b = 27. The following settings for the rekeying intervals assures a quasi-certain rekeying rate for **permanent** members, that is 99.99 % of **permanent** members have 99.99% of probability to receive all rekeying packets:

$$\begin{aligned} T_v &\geq 46s \\ T_p &\geq 3726s \end{aligned}$$

Table 3.4 compares the computed minimum value of T_p with other block sizes implemented in the same conditions. Hence, T_p should be as small as possible in order to let the minimum extra-time to leaving members.

b(1)=12	b(2)=27	b(3)=90	b(4)=345
4600s	3726s	2806s	2530s

Table 3.4: Comparison on T_p

3.5.3 How to determine the threshold time t_{th}?

In order to define t_{th} which is the time at which a **volatile** member is transfered to the **permanent** key tree, the key server should on one hand regroup long-duration members in G_p as soon as possible and on the other hand, minimize the number of short-duration members considered as **permanent**. The probability that a member does not leave the group before t_{th} where the membership duration time is distributed exponentially with a mean M is:

$$p(t \geq t_{th}) = e^{-t/M} \tag{3.21}$$

Following this equation, the average number of short-duration members considered as **permanent** members is $\gamma * N * e^{-t_{th}/Ms}$, γ corresponding to the ratio of short-duration members in the real partitioning. The key server should limit this number while choosing $(1-\gamma) * N * e^{-t_{th}/M_l}$ as large as possible.

Based on the same example described in the previous section, we are able to compute the threshold value t_{th} that would best fit the real partitioning (50% short duration members). Using the resulted value ($t_{th} = 1000$), the protocol will eventually identify 10% of short-duration members as **permanent** members.

3.6 Related Work

Having described a new rekeying protocol partitioning members with respect to their membership duration and offering a strongly reliable delivery of the keying material for **permanent** members, we present in this section two pieces of related work that are somehow similar to ours with regards to some of the strategies that we have taken. We then compare their efficiency with our protocol. We first present a protocol that reduces the number of members losing their keying material and then deal with a second protocol that divides the set of members in two categories based on the observations in [4], in order to reduce the rekeying cost.

3.6.1 Group Rekeying with Limited Unicast Recovery

Description

In [83], as in our hybrid reliability scheme for permanent members, authors investigate how to limit unicast recovery to a small fraction of the user population and guarantee that almost all members of the group receive their updated keying material within a single multicast round. As in our scheme, in order to provide reliability, they propose the use of FEC and analyze the expected ratio of losing members based on the packet loss rate, the duration of the rekeying interval and the number of parity packets defined for each block. As in our proposed protocol, they first analyze this value in the case where the packet loss probability is considered to be independent (namely, the Bernouilli model). They conclude that the expected number of losses decreases by increasing the number of parity packets and the rekeying interval. Moreover, authors also consider Markov losses. Hence, in [80], authors studied about Internet loss patterns and realized that:

- Packet losses are not independent. When a packet is lost, the probability that the next packet will be lost increases, which means that losses in the Internet are often `bursty`;
- the majority of bursts are small (from one to six-seven packets);
- there are some very long bursts, lasting up to a few seconds. These bursts could be attributed to network disruption or maintenance [9].

Thanks to these observations, authors conclude that all the packets should be equally spaced in each rekeying interval. If the key server has multiple blocks to send, the packets belonging to the same block should be equally spaced. However, the packets from different blocks should be sent in an interleaved fashion. That is, the i^{th} packet from each block should be sent together without spacing. Though these packets are likely to experience burst losses, the effect on the overall key distribution is harmless since each particular user needs packets only from one specific block. Figure 3.5 illustrates how the key server should space packets when there are three blocks of four packets including parity packets. $P_{i,j}$ denotes a packet number i from block j.

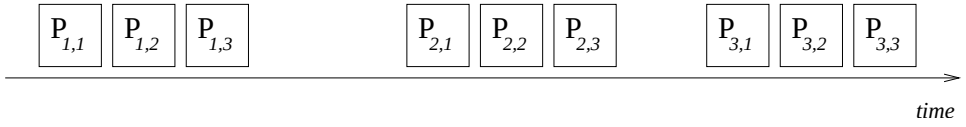


Figure 3.5: An illustration of the sending strategy of packets

Since the key server cannot indefinitely increase T, authors additionally suggest an adaptive FEC protocol to determine the rekeying interval for any specified loss ratio bound. In order to determine T, the key server collects *NACKs* from the previous rekeying interval and if this number is greater than the given bounded value, the key server increases the number of parity packets and thus the rekeying interval. Similarly, the key server decreases the number of parity packets based on at least three samples using exponentially weighted average of the expected number of losing members.

Discussion

The proposed protocol which aims at decreasing the number of members losing their keys at the first round of a rekeying interval seems to be similar to our proposed hybrid reliability scheme. However, as in their previous work [77], their packet generation follows the User-oriented Key assignment algorithm which guarantees that all of the encrypted keys needed by any member will be contained in a single packet. In our protocol, a packet contains a single key, and the keying material required by each member can be retrieved from packets of a single block. In addition to this advantage, thanks to our user oriented key assignment algorithm, most of the keys located at high levels in the key tree are replicated in several blocks. Consequently, if a member does not even receive packets required to reconstruct its FEC block, it can complete it using rekeying packets from other blocks.

Moreover, authors conclude that the rekeying interval needs to be increased when the ratio of members losing their keys is low. However the key server cannot indefinitely increase this value since members sending leaving requests should leave the group as soon as possible. In a similar scenario, our protocol behaves much more efficiently, since it treats **permanent** members differently from **volatile** members: the key server grants an additional delay for **permanent** members and offers a more strongly reliable delivery.

3.6.2 A Two-Partition Algorithm for Group Rekeying

In [84], similarly to our protocol, authors based their research on the observations made in [4] and propose to divide the key tree into two partitions: *S-partition* and *L-partition* regrouping members with respect to their membership duration. Based on this partitioning idea, the authors propose and compare two different constructions for the two-partition algorithm where the data structures defined in each partition are different. The key server defines a balanced key tree for the **L-partition** and the two constructions differ with respect to the data structure defined for the **S-partition**:

- **QT-scheme**: in their first scheme called the *QT-scheme*, the key server uses a linear queue to represent the **S-partition**. Each member of this set stores only one individual key and the new data encryption key needs to be encrypted with each member's individual key.

- **TT-scheme**: This alternative scheme defines a balanced tree for each partition. Consequently, in this scheme, each member stores $log(N)+1$ keys where N denotes the size of the corresponding partition and the key server follows the LKH scheme for group rekeying.

From their analytical results, they conclude that the two-partition schemes outperform the classical LKH when a group has a certain degree of dynamics. The authors showed that when N_s (size of the **S-partition**) is large, the *TT-scheme* is more scalable than the *QT-scheme* in terms of the number of messages to send by the key server for the rekeying.

Discussion

Although this work proposes a partitioning scheme similar to ours, they only focalize on the aspect of scalability. Hence, the aim of this work only was to optimize the overall communication overhead. We, on the contrary, proposed the partitioning scheme in order to deal with the problems of reliability and customer satisfaction. In addition to the basic partitioning scheme, for each set of members, we set different rekeying intervals and different reliability parameters for each partition. We provide a longer rekeying interval for **permanent** members during which the data encryption key can be automatically retrieved by these members. However in this related work, authors only provide the definition of the threshold time whereby a member is transfered from the **S-partition** to the **L-partition** and the rekeying operations occurs at the same time for both sets.

3.7 Conclusion

In this chapter, we proposed a new rekeying protocol that separately regroups members into two categories based on their membership duration. Members are first considered as **volatile** and when their membership duration reaches a predefined threshold value they become **permanent**. In order to distinguish between the two member categories, the key server defines a separate key tree for each class of members. Higher reliability is offered to **permanent** members thanks to the proposed hybrid reliability protocol that combines both proactive FEC and proactive replication techniques using a specific user-oriented key assignment algorithm for the definition of the FEC blocks. We then define some bounds with respect to the rekeying intervals and optimize the threshold value t_{th} in order to keep the partitioning of members as perceived by the key server as close as possible to the real categories. In the next chapter, we analyze and validate the efficiency of this protocol with simulations.

Chapter 4

Simulation-based validation of the proposed protocol

4.1 Introduction

In this chapter, we carry out a simulation based study of the rekeying protocol presented in the previous chapter. The features of the protocol are analyzed in terms of simulation metrics that we deem relevant to assess its basic advantages over the classical LKH scheme:

- the communication overhead;
- the number of losses.

Simulation results are used to understand that the proposed rekeying protocol offers an optimization in terms of scalability, reliability and answers to the requirement of customer satisfaction over LKH. We therefore first analyze the efficiency of the simple partitioning scheme, then deal with the performance of the hybrid reliability method based on the proposed User-oriented Key assignment algorithm.

4.1.1 Implementation

The proposed rekeying mechanism described in the previous chapter is implemented as a discrete-event simulator in C. The events represent the actions of a key server and operations performed by recipients such as join, leave or change of status to become **permanent**. In order to show the significance of the performance of our protocol, the original logical key tree is also implemented. Therefore, the simulator can run in two modes. One mode simulates the original LKH

scheme whereby only one key tree is defined to manage all group members and the other mode simulates the proposed partitioning scheme using two key trees: one for **volatile** members and the other one for **permanent** members.

With respect to scalability, we can set a different rekeying algorithm in each of the main modes. If individual rekeying is adopted, the key trees are updated after each join or leave event. Using batch rekeying, the duration of the rekeying interval is defined and all the joining and leaving members are collected in a queue. The corresponding rekeying operation is performed at the beginning of the subsequent interval.

In order to have results in terms of reliability, we simulate a packet loss probability which is independent for each client as an input simulation parameter. We therefore can analyze the number of members losing at least one rekeying packet during each rekeying operation. In both modes (classical LKH - Two key trees), three optional reliability mechanisms are available as follows:

- **No reliability**: the rekeying packets are sent without any additional packets;
- **Static replication**: every rekeying packet is transmitted a given number of times;
- **hybrid FEC**: this option illustrates the proposed hybrid reliability method where the key server first defines FEC blocks with respect to the proposed user oriented key assignment algorithm, then, based on the parameters α and β, transmits each block with the corresponding parity packets.

Type	**Distribution**	**Parameters**
1	Uniform	lower bound, upper bound
2	Exponential	mean value
3	Normal	mean value and standard deviation
4	Log Normal	mean value and standard deviation
5	Pareto	mean value and distribution shape
6	Hyperexponential	mean value and distribution covariance
7	Triangular	lower bound, upper bound, mode
8	Zipf	Zipf α and n

Table 4.1: The probability distributions implemented in the simulator

Finally, we can also simulate different customer behaviors. The main parameters for this aspect are the inter-arrival and service time distributions for each class of client (**short-duration** and **long-duration**). The arrival time and membership duration of each member can be defined as random variables. The simulator either uses a random variable generator, given the required parameters for each type of distribution or the client behavior can explicitly be specified. It is possible to specify the exact arrival times and service times of clients. Thanks to this option, one can simulate the efficiency of the proposed rekeying protocol with respect to real customer

behavior. The probability distributions that are implemented in the simulator are given in table 4.1.

4.1.2 Simulation parameters

Table 4.2 summarizes the choice of parameters for the simulator. We regroup system parameters into four main sections: the **key tree options** define the main parameters with respect to the key tree; the **customer behavior** where the user defines the inter-arrival and service time distribution of both short-duration and long-duration members; the parameters related to the **time**; the **reliability features** describing the method used for reliability.

Keytree Options	keytree mode	1: one key tree (SKT) 2: 2 key trees (2KT)
	keytree outdegree	the degree of the key tree with respect to keytree mode
	rekeying option	1: individual rekeying 2: batch rekeying
Customer behavior	inter-arrival_s	inter-arrival time distribution parameters for short-duration members (or the file)
	inter-arrival_l	inter-arrival time distribution parameters for long-duration members (or the file)
	service-time_s	service time distribution parameters for short-duration members (or the file)
	service-time_l	service time distribution parameters for long-duration members (or the file)
Time parameters	rekeying interval for members in SKT	T_s
	rekeying interval for volatile members	T_v
	rekeying interval for permanent members	$T_p = \delta T_v$
	threshold time value	t_{th}
	total simulation time	T
Reliability features	loss probability	p
	subtree height for hybrid FEC blocks	a
	reliability method	1: None 2: Replication (n: replication factor) 3: hybrid FEC (α, β)

Table 4.2: The simulator's system parameters

4.1.3 Simulation metrics

In this section we present the major simulation metrics that are relevant for the analysis of the proposed rekeying protocol.

Rekeying cost

Thanks to this simulator, given input parameters, we can analyze either the total number of rekeying packets for the whole session or the number of rekeying packets per rekeying interval. The simulator can output the rekeying cost with respect to the chosen reliability method. If the replication method is chosen with parameter n, then the total rekeying cost is:

$$total\_cost(n,t) = n * cost(t)$$

If on the other hand, the hybrid FEC scheme is chosen, the simulator first constructs the FEC blocks using the user oriented key assignment algorithm (refer to section 3.4.2) and then computes their corresponding number of parity packets. The total rekeying cost will be the sum of all the FEC block sizes including additional parity packets.

Finally, we also can distinguish the rekeying cost resulting from each key tree (single, volatile or permanent).

Number of joining and leaving members

The simulator can also output the number of joining and leaving members. We can distinguish the category of a member (**short-duration**, **long-duration**). We can also get the number of false positives, that is the number of **short-duration** members who have joined the **permanent** key tree. Thanks to this property, we can get the number of **long-duration** members treated as **permanent** and decide the threshold value t_{th} that better fits with our scheme.

Number of losses

Given the loss probability of a packet, the simulator outputs the number of members losing at least one rekeying packet per rekeying interval. We can again distinguish the number of losses with respect to the real categories (**short-duration** and **long-duration**) and the monitored categories (**volatile** and **permanent**). These values will allow us to compare the efficiency of the proposed protocol with LKH in terms of reliability in general and in terms of customer satisfaction.

Additional join/leave time

Finally the simulator also outputs the average time that a joining member waits for, before receiving its corresponding keying material and symmetrically the average additional time granted to a leaving member that is at most twice the rekeying interval's duration.

4.2 Simulation results

4.2.1 Simulation set-up

In this section, we describe the simulation parameters that are never modified in further simulations. In our simulation, we consider two real categories of members that are equally present in the group: their interarrival time are distributed exponentially with a mean of one second. Their membership duration are distributed exponentially with a mean M_s for short-duration members and M_l for long-duration members. We set the simulation time to $T = 10000s$ and the means to: $Ms = 1000s$ and $M_l = 100000s$.

Batch rekeying is performed and the rekeying interval is set to $T_s = 60s$ for the single key tree scheme and to $T_v = 60s$ and $T_p = 60 * 10 = 600s$ for the proposed partitioning scheme. The packet loss probability is set to $p = 0.1$

Impact of the threshold time t_{th}

As explained in the previous chapter, in order to define t_{th}, the key server should in one hand regroup a maximum number of long-duration members while trying to keep the number of short-duration members considered as **permanent** low.

Based on the previously defined parameters, we simulate the partitioning scheme with different values of t_{th} and analyze the number of short-duration members considered as being permanent (ie. the number of false positives). Figure 4.1 illustrates these simulations and compares the total number of short-duration members joining the group with the number of those joining the permanent key tree.

From this figure, we observe that the number of short-duration members joining the permanent key tree decreases exponentially. If the key server sets that at most 5% of short-duration members should be considered as **permanent**, we end up with $t_{th} = 3000$ and observe that the resulted proportion of short-duration members transfered to the permanent key tree is 4.98%.

4.2.2 Simulation scenarios

With the given simulation set-up parameters, we analyze the performance of the proposed rekeying protocol over the classical LKH scheme with respect to the number of losing members and the rekeying cost (being the number of rekeying packets) in an incremental way:

- **Case 1**: we only implement the partitioning scheme defining two different key trees re-

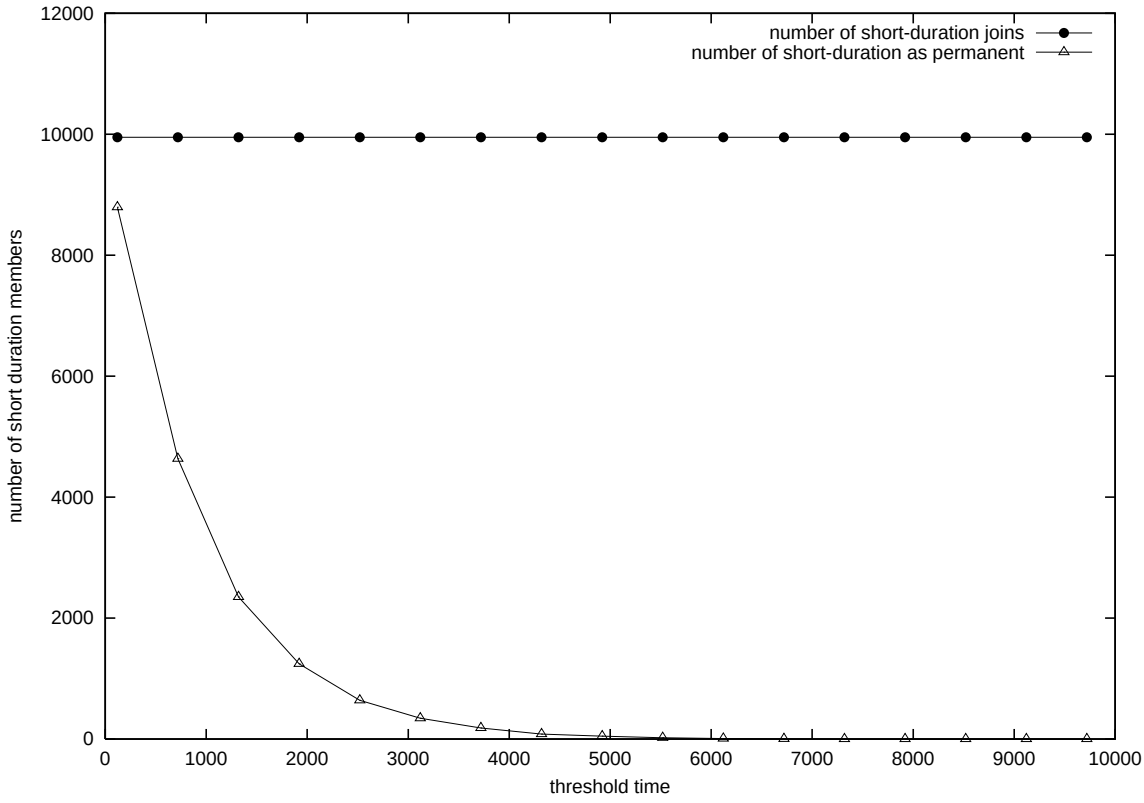


Figure 4.1: Number of short-duration members as being permanent with respect to t_{th}

spectively for **volatile** and **permanent** members and the key server transmits rekeying packets without any additional packets;

- **Case 2**: we apply our user oriented key assignment algorithm where the key server regroups rekeying packets such that any member only receives one block to retrieve all of its updated keying material;

- **Case 3**: we analyze the efficiency of the proposed hybrid reliability scheme which aims to offer a reliable delivery of the keying material to almost all permanent members given parameters α and β.

4.2.3 Case 1: Performance of the partitioning scheme with no reliability mechanism

In this section, we analyze and compare the performance of the partitioning scheme with respect to LKH, where there is no reliability method taken into account. We first analyze the number of long-duration members losing at least one rekeying packet and then compare the rekeying cost of both schemes.

Number of losses

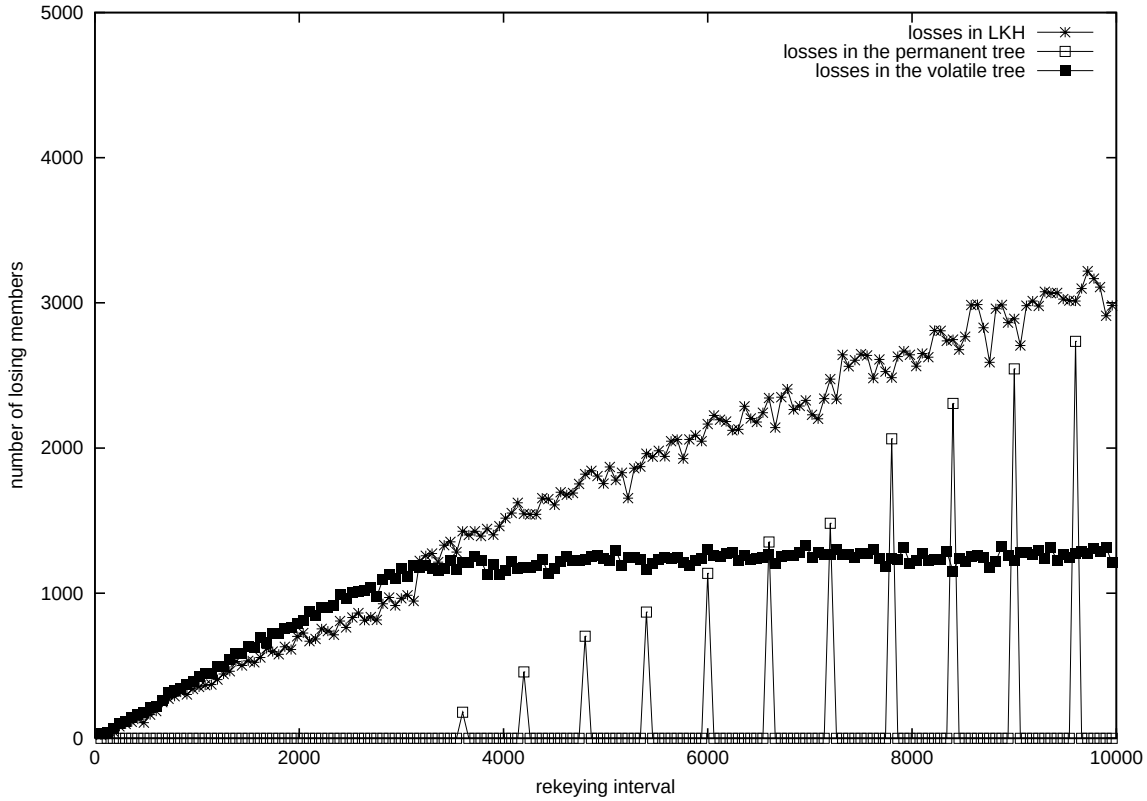


Figure 4.2: Number of losses experienced by long-duration members in Case 1

Given the simulation set-up defined in section 4.2.1, we analyze the number of losses experienced by long-duration members in both schemes. From figure 4.2, we realize that, before the threshold value $t_{th} = 3000$, the number of losses experienced by long-duration members in the volatile key tree is close to the one observed in the single key tree case. Beginning from $t = w$, this number does not vary in the volatile key tree. This is due to the fact that long-duration members are transfered to the permanent key tree when their membership duration reaches t_{th} and thus the number of long-duration members considered as volatile becomes constant. Symmetrically, since long-duration members can join the permanent key tree at each T_v, the number of losses experienced in the permanent key tree increases with the number of long-duration members transfered to this key tree. Moreover, since in the proposed partitioning scheme rekeying operations are only performed every T_p, we observe that members only lose their keying material at each T_p.

Single Key tree	Volatile key tree	Permanent key tree	Both key trees
282025	173729	15830	189559

Table 4.3: Total number of losses experienced by long-duration members in Case 1

From table 4.3 that presents the total number of losses experienced by long-duration members, we can derive that the proposed partitioning scheme reduces the number of losses by 33%

with respect to the classical LKH scheme.

Rekeying cost

Single key tree	Volatile Key tree	Permanent key tree	Both key trees
123030	142412	5309	147721

Table 4.4: Total rekeying cost in Case 1

Table 4.4 illustrates the total rekeying cost (number of rekeying packets) resulting from the classical LKH scheme and the proposed partitioning scheme. The proposed partitioning method reduces the rekeying cost for permanent members while increasing the total rekeying cost by 20%.

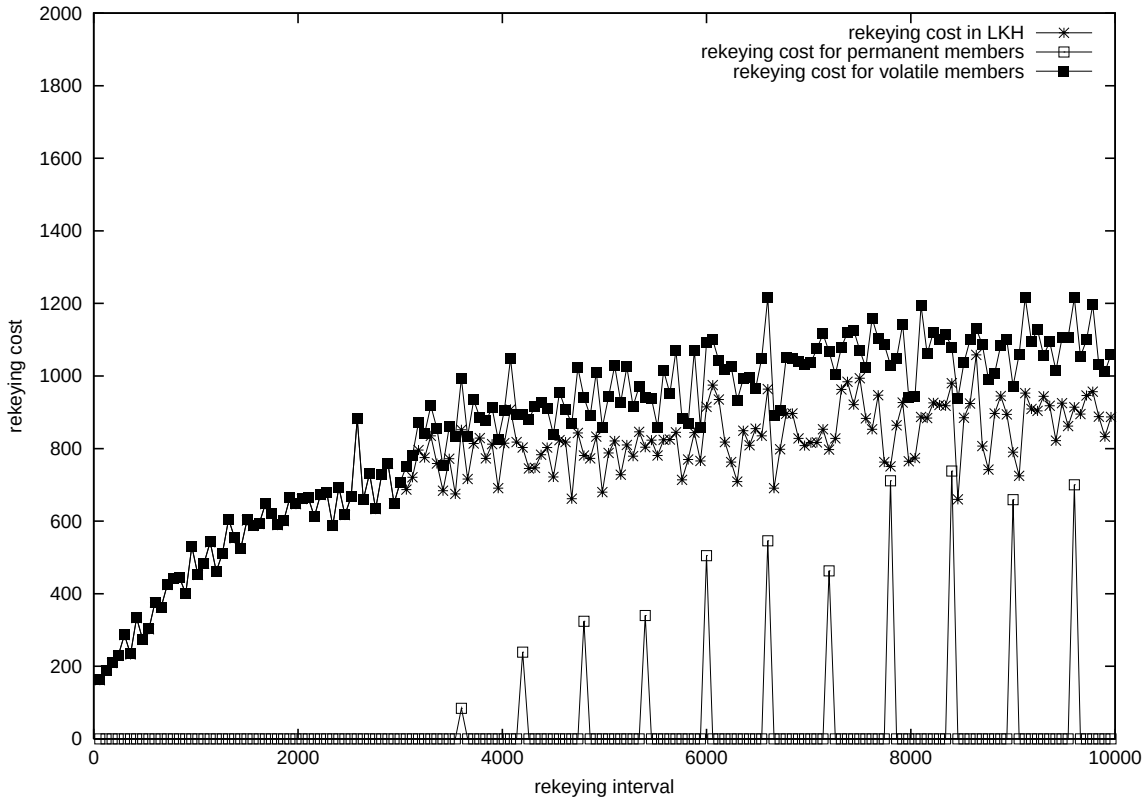


Figure 4.3: Rekeying cost in Case 1

From figure 4.3 illustrating the rekeying cost for each key tree at each interval, we observe that the rekeying cost for volatile members is slightly higher than the rekeying cost of the LKH scheme. This difference is in fact due to the rekeying messages transmitted in order to move members, whose membership duration reaches the threshold value t_{th}, from the volatile key tree to the permanent one.

In the same figure, the number of rekeying packets destined to permanent members is very low with respect to volatile members. From this observation, we can conclude that since the rekeying cost resulting from the update of the permanent key tree is low, given a bandwidth limitation, the key server can manage a strongly reliable delivery for members of this category.

4.2.4 Case 2: Impact of the User-oriented Key assignment algorithm

In this section, we analyze the efficiency of the proposed User-Oriented Key Assignment algorithm where the key server regroups rekeying packets such that any member only receives one block to retrieve all of its updated keying material. In this case, as described in section 3.4.2, the key server should first define the block size defined in equation 3.6 and based on the parameter *a*. We thus perform several simulations with different values of *a* and analyze and compare the performance of this algorithm with the previous case. We remind that this algorithm only is implemented for **permanent** members.

Number of Losses

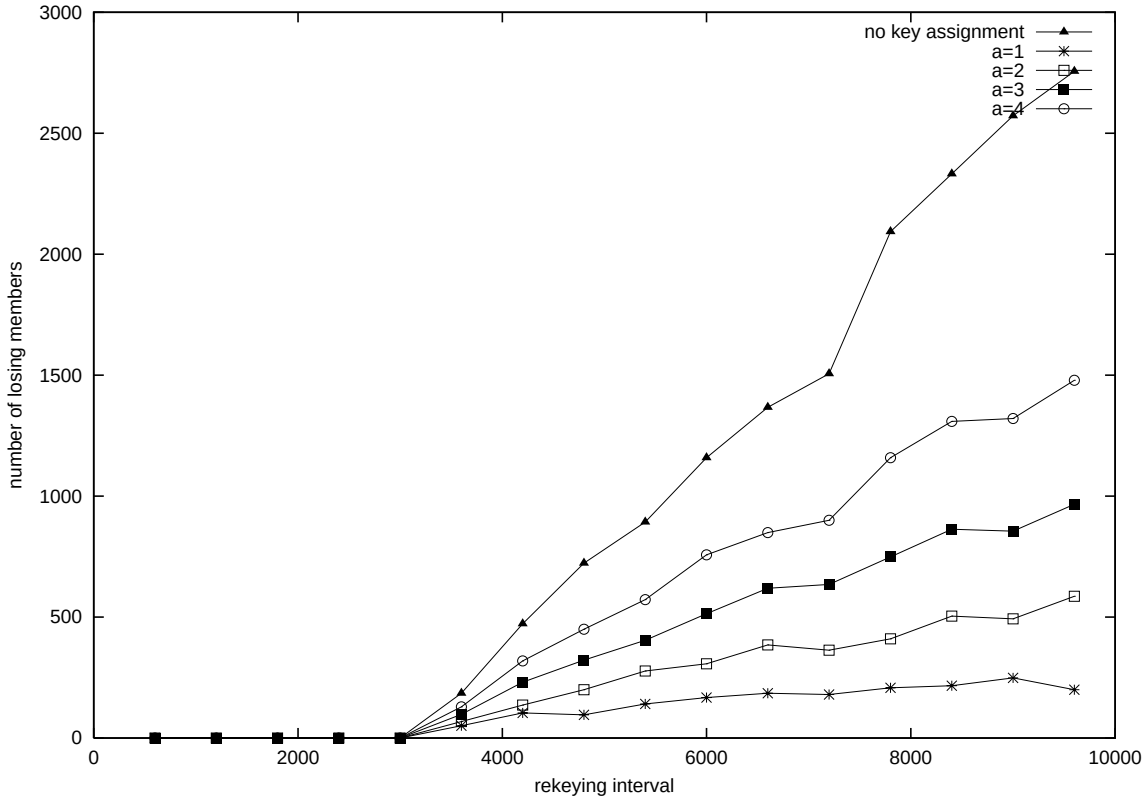


Figure 4.4: Losses experienced by permanent members in Case 2

Figure 4.4 illustrates the number of **permanent** members losing rekeying packets at each T_p. We observe that this number increases when a increases. This fact was expected since when a increases, the number of members needing packets from this block increases, and packets are less replicated. However, the number of losses resulting from Case 1 (no key assignment) is always greater than those when the proposed user-oriented key assignment algorithm is implemented. We thus can conclude that the proposed user-oriented key assignment algorithm reduces the number of losses even when there is no FEC implementation.

Rekeying cost

Figure 4.5 illustrates the number of rekeying packets sent by the key server to **permanent members**. We only show the rekeying cost at each T_p, since the key server does not send any rekeying packet apart from these intervals. We observe that the rekeying cost resulting from the user-oriented key assignment algorithm is greater than the rekeying cost in the permanent key tree in Case 1. This observation is not surprising since some keys are inherently replicated in different blocks.

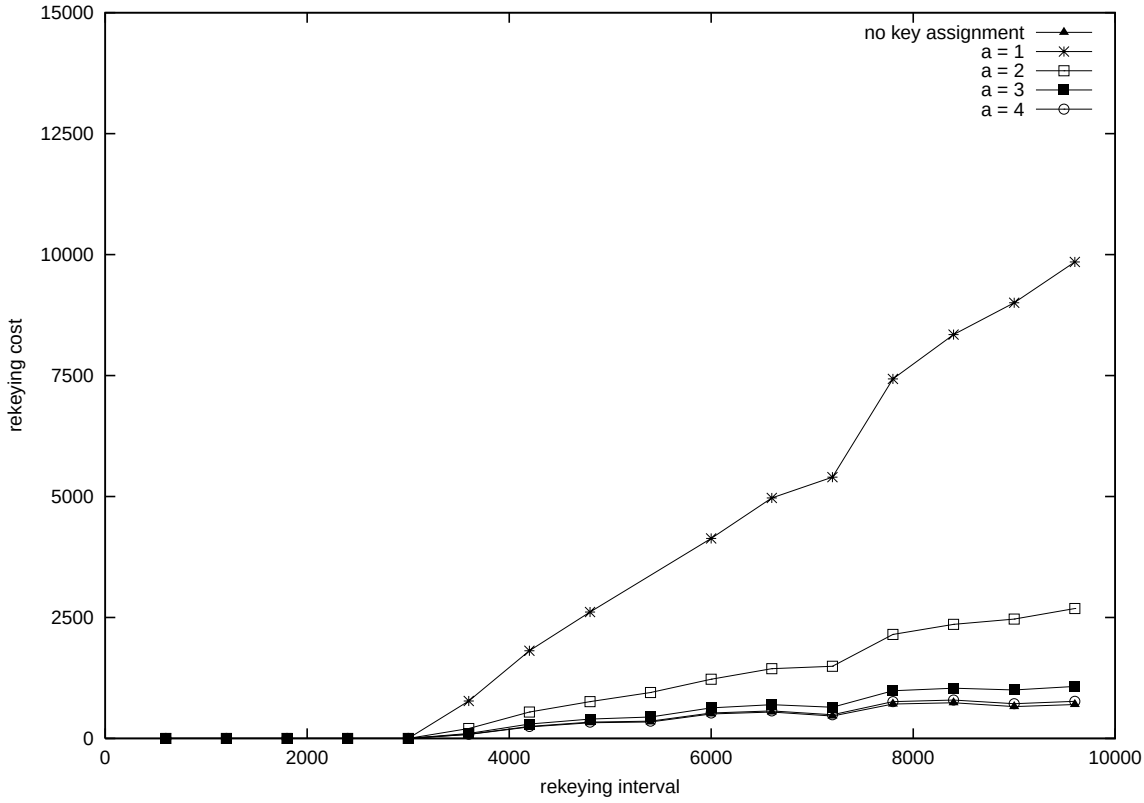
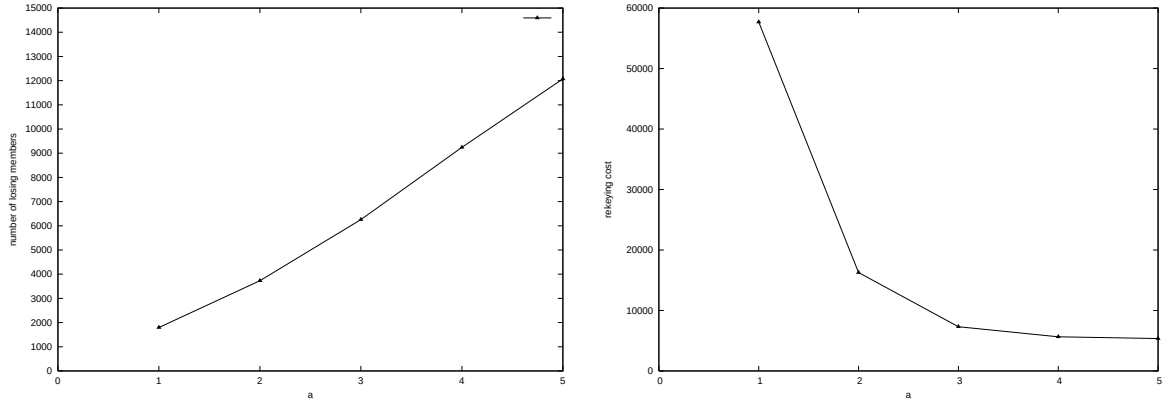


Figure 4.5: Rekeying cost in Case 2

Discussion

Figure 4.6 shows the behavior of the two simulation metrics (losses and rekeying cost) with respect to *a*. In figure 4.6**(a)** we observe that when *a* increases, the number of losses increases almost linearly. From this observation we can conclude that the key server must choose the minimum possible value for the block size. However, figure 4.6**(b)** shows that the rekeying cost decreases exponentially. We notice that between $a = 1$ and $a = 2$ the difference is very high and beginning from $a = 3$ the rekeying cost is almost stable. Since there is a tradeoff between these two simulation metrics, we can conclude that in order to achieve scalability with customer satisfaction, the key server can set $a = 2$ or $a = 3$.



(a) Losses experienced by permanent members in Case 2 with respect to *a*

(b) Rekeying cost in Case 2 with respect to *a*

Figure 4.6: Performance of the user oriented key assignment algorithm with respect to *a*

4.2.5 Case 3: Impact of the hybrid reliability scheme

Now that we analyzed the performance of the proposed user-oriented key assignment algorithm and the estimation of the optimal FEC block size based on *a*, we turn to the efficiency of the hybrid reliability scheme which defines the number of parity packets for each FEC block depending on α and β. We again analyze the number of losses and the rekeying cost for given values of α and β and compare the results with the Case 2 where the key server does not generate any parity packet at all (case where $\alpha = 0$ and $\beta = 0$). In the previous simulations, we showed that the key server needs to set either $a = 2$ or $a = 3$ in order to optimize the performance of the key assignment algorithm. We thus only take these two cases into account and set $\alpha = 0.99$ and $\beta = 0.99$.

Number of losses

Figure 4.7 illustrates the number of losses experienced by **permanent** members for different values of a and for (α, β) set to $(0,0)$ (Case 2) and $(0.99, 0.99)$. We observe that the implementation of the hybrid reliability scheme ($\alpha = 0.99$, $\beta = 0.99$) reduces the number of losses which almost reaches zero. Thanks to this simulation, we can conclude that the hybrid reliability scheme reduces the number of **permanent** members losing their keying material and achieves the requirement of customer satisfaction.

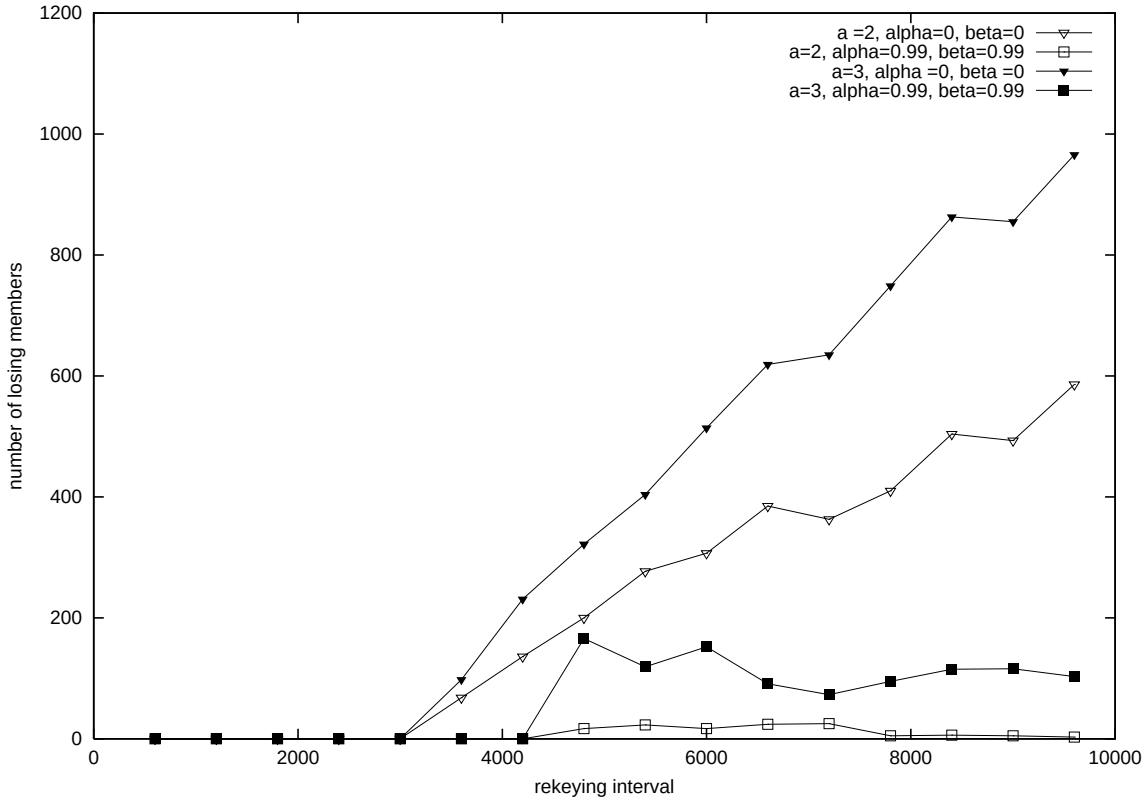


Figure 4.7: Losses experienced by permanent members in Case 2 and Case 3

Rekeying cost

Figure 4.8 displays the rekeying cost with and without the implementation of the hybrid reliability scheme. We notice that the rekeying cost slightly increases when the key server generates parity packets with $a = 3$ (based on $\alpha = 0.99$, $\beta = 0.99$) whereas for $a = 2$ the difference becomes much higher. Since the number of losses depicted in figure 4.7 remains acceptable when $a = 3$, we can conclude that this value achieves the best tradeoff between losses and rekeying cost.

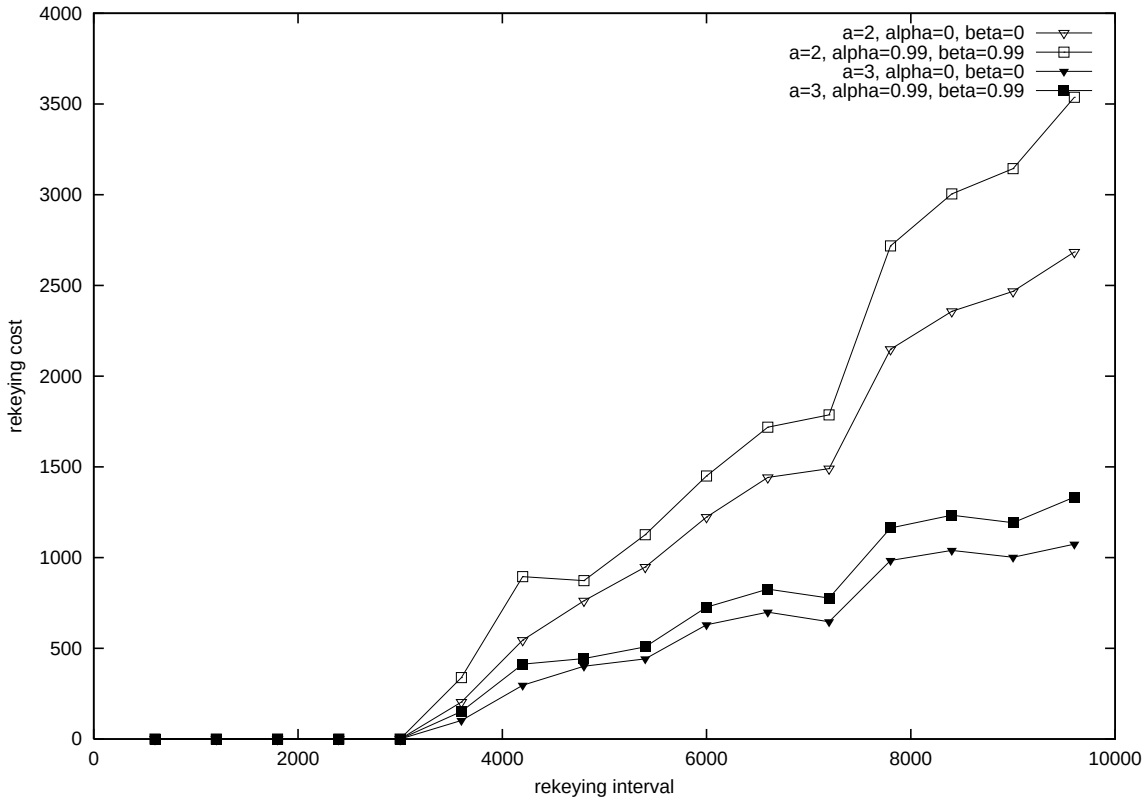


Figure 4.8: Rekeying cost in Case 2 and Case 3

Discussion

Table 4.5 summarizes the simulation results for each scenario. In this table, we illustrate the total number of losses and the total rekeying cost for the **permanent** key tree. We first realize that the use of the proposed User-Oriented Key Assignment algorithm significantly reduces the number of losses (77% for $a = 2$ and 61% for $a = 3$). Furthermore, the implementation of the hybrid reliability scheme shows a much better performance in terms of losses. Unfortunately, this advantage has an impact on the rekeying cost (about 100% of increase for $a = 3$ in Case 3). However, as shown in table 4.4, this cost still remains low with respect to the **volatile** key tree's one (142412). Consequently, this additional cost will have a small impact on the total rekeying cost regrouping both **volatile** and **permanent** members (less then 4%).

	Case 1	Case 2		Case 3	
		a=2	a=3	a=2	a=3
Total Losses	16058	3728	6256	286	1030
Total Rekeying Cost	5309	16270	7316	20592	10372

Table 4.5: Efficiency of the proposed rekeying protocol: Summary

4.3 Conclusion

In this chapter, we analyzed the performance of the proposed rekeying protocol over the LKH scheme in an incremental way. We first evaluated the efficiency of the partitioning scheme and showed that this scheme reduces the number of losses while slightly increasing the total rekeying cost. However, the rekeying cost resulting from the update of the permanent key tree being very low with respect to the one resulting from the update of the volatile key tree, the key server can transmit additional packets for members of this category in order to offer a reliable delivery. Based on this observation, we then proposed to analyze the efficiency of the hybrid reliability method for **permanent** members. We first showed that the use of the User-Oriented Key assignment algorithm reduces the number of losses and that the key server needs to define the block size as high as possible in order not to observe a high rekeying cost. We then came up with the definition of additional parity packets based on α and β and showed that although this method increases the rekeying cost, it remains efficient with respect to the total rekeying cost resulting from the whole group (**permanent** and **volatile** members).

Chapter 5

A Specific Reliable Group Rekeying Protocol for short sessions

5.1 Introduction

In this chapter, we deal with the short-duration applications of the second category (refer to section 3.2.2) where the pricing mechanism is based on members' arrival time and clients pay at their arrival for the time remaining until the end of the session. For such applications, the degree of "loyalty" of a member depends on its arrival time. We propose a new rekeying scheme where during rekeying operations, members having already subscribed to the service are not penalized from further arrivals and thus do not lose their keying material due to failures occurring during these rekeying operations. We first describe a scheme that partitions members with respect to their arrival time [53]. We then analyze the efficiency of this specific protocol with respect to security, scalability, reliability and customer satisfaction and give simulation results.

5.2 The proposed solution

For applications such as video-conferences, arrivals frequently happen at the beginning of the session and few recipients leave the group before the end of the session. A suitable pricing scheme for such applications consists of having clients pay at their arrival for the time remaining until the end of the session. We propose to restructure the LKH scheme by building different key trees for separate groups of members based on their arrival time. Thanks to this restructuring of key trees, the impact in rekeying operations on members having already subscribed to the session is drastically reduced.

5.2.1 Modeling Customer Behavior

In [4], Almeroth et al. observed group members' behavior during several multicast sessions. The authors realized that most short sessions exhibit an increasing amount of group join activity up to just after a program's scheduled start time. Based on these results, in order to model the arrival time of a member, we use a Zipf distribution representing the fact that a large percentage of samples are concentrated at the beginning of the range while the remaining percentage are widely dispersed over the remainder of the spectrum [85]. Let J_j denote the number of members joining the group at interval j defined as in equation 5.1:

$$J_j = \lfloor A/j^{\alpha} \rfloor \tag{5.1}$$

This value depends on the Zipf distribution parameters, A and α, that are defined by the following equation in order not to exceed the group size N, where the number of intervals is set to j_{max}:

$$\sum_{j=1}^{j_{max}} \lfloor A/j^{\alpha} \rfloor = N \tag{5.2}$$

5.2.2 Partitioning and rekeying process

As we assume that recipients join at different times up to the beginning of the session, we define as many key trees as rekeying intervals. Indeed, the key server collects arrivals during a rekeying interval j and builds a different key tree T_j that only regroups the corresponding J_j members.

The protocol is summarized in table 5.1.

For each interval j:
build the new key tree T_j for new members;
For each remaining key tree T_i ($i \in \{1..j-1\}$)
  If there is no leaving member:
    Send $\{E_{K_i^{(j-1)}}(K_{data_j})\}$;
  else
    Update $T_i^{(j-1)}$;

Table 5.1: Rekeying process at each rekeying interval

In order to ensure backward secrecy, when there is no leaving member, the data encryption key is updated at each rekeying interval, and the new data encryption key is separately encrypted with the keys localized at the root of each key tree that was already built before. If on the other hand, some departures occur in a key tree, then the key server sends the required additional keying material only to members assigned to the tree that is being updated.

5.2.3 Efficiency of the proposed scheme

While defining several key trees during the multicast session, the key server minimizes the set of members that could be affected from the loss of rekeying messages and thus offers a better customer satisfaction. Indeed, the proposed scheme presents several advantages:

- Since new members are regrouped in a new key tree, members in existing key trees only need to receive one key, that is, the updated data encryption key;
- If a departure occurs in a key tree and this key tree needs to be updated, members assigned to the other key trees only need to receive the new data encryption key;
- Since members are separately regrouped into several key trees, the number of members assigned to each key tree can be kept small. Thus, the number of rekeying messages per key tree can be small. Consequently, since the key server sends a small amount of messages to a small number of members, a good level of reliability can be assured for the delivery of keys.

5.3 Analysis of the protocol

5.3.1 Security issues

In this section, we analyze the security properties of the proposed scheme.

Backward secrecy

Since at each rekeying interval, a new key tree is generated for arrivals at the corresponding interval and there is no transfer of members from a tree to another one, the security of our scheme depends on the security of each key tree. When a new member joins the group at the rekeying interval j, a new key tree T_j is created and the member is assigned to T_j. Since the keys assigned to this new key tree are generated at rekeying interval j, the member cannot have access to the keys transmitted previously in the session. Moreover, at each rekeying operation,

the data encryption key is updated for all members. This new data encryption key is encrypted with the key located at the root of each key tree. Consequently, the proposed scheme ensures backward secrecy for all members.

Forward secrecy

Symmetrically, at each rekeying interval, if a member leaves the group, it is removed from the key tree to which it belonged and by the definition of LKH, this key tree is updated in order to prevent accessing to the future keys assigned to this key tree. The data encryption key is again updated and encrypted with keys that do not appear in the keying material of the leaving members. Therefore, the proposed scheme ensures forward secrecy for all members.

5.3.2 Scalability issues

Notation

In this section, we analyze the memory usage and the cost of rekeying and use the resulting figures in further simulations in order to evaluate the performance of the scheme as a whole. In order to define these overheads, we use the following notation:

- j the number of intervals left since the beginning of the session;
- d denotes the key tree degree and is assumed to be the same for all key trees;
- h_j denotes the depth of the key tree T_j;
- J_j denotes the number of members joining the group at interval j and is defined in equation 5.1.

Evaluation of the memory usage

We first evaluate the memory requirements of this scheme both at members and at the key server.

A new member arriving at interval j receives its keying material corresponding to the key tree T_j. This member must store h_j keys from this key tree, such that $h_j = \lceil log_{d_j}(J_j) \rceil + 1$ (refer to 1.2). In addition to the keys located at its corresponding key tree T_j, a member should also store the actual data encryption key. Therefore the total memory usage for a member in the proposed scheme is defined by the following equation:

$$Memcost(T_j, member) = h_j + 1 \tag{5.3}$$

The key server's memory usage increases at each rekeying interval, since a new key tree is constructed at each rekeying interval. The total number of keys defined for the whole session is the sum of the keying material defined for each key tree and the data encryption key. The number of keys defined for a key tree T_j depends on the number of joining members at this interval J_j, that is:

$$Memcost(T_j, server) = \sum_{i=0}^{h_j-1} \lceil J_j/d_j^i \rceil \tag{5.4}$$

The total memory usage for j_{max} intervals will then be:

$$Memcost(total, server) = 1 + \sum_{j=1}^{j_{max}} \sum_{i=0}^{h_j-1} \lceil J_j/d_j^i \rceil \tag{5.5}$$

Evaluation of the rekeying cost

We now evaluate the cost of rekeying at each rekeying interval. Authors in [31] show that while building a new key tree, the perfect approach from the memory utilization point of view consists of encrypting keys of one level of the tree only with keys located at the children vertices. We evaluate the cost of building the key tree T_j based on this approach as follows:

$$Cost_{T_j} = \sum_{i=0}^{h_j-2} \lceil J_j/d_j^i \rceil \tag{5.6}$$

Furthermore, let δ be the volatility degree of a member. In other words δ denotes the probability that a member leaves the group. Assuming that this number is very low in the targeted applications, at each rekeying interval, the rekeying cost caused by leaving members denoted as $cost_{L_j}$ can be defined as:

$$cost_{L_j} = \sum_{i=1}^{j-1} \lfloor \delta J_i \rfloor (d_i \times h_i - 1) \tag{5.7}$$

If there are no leaving members, the key server sends the required keying material for the new key tree T_j and the new data encryption K_{data_j} encrypted with each root key of all trees

constructed before the current interval. Hence, we have:

$$\begin{aligned} ComCost(j) &= j-1+Cost_{T_j} \\ ComCost(j) &= j-1+\sum_{i=1}^{h_j-1}\lceil J_j/d_j^i\rceil \end{aligned} \tag{5.8}$$

If on the other hand, some leave requests occur, the total cost will include $cost_{L_j}$:

$$\begin{aligned} ComCost(j) &= j-1+Cost_{L_j}+Cost_{T_j} \\ ComCost(j) &= j-1+\sum_{i=1}^{j-1}\lfloor \delta J_i\rfloor (d_i\times h_i-1)+\sum_{i=1}^{h_j-1}\lceil J_j/d_j^i\rceil \end{aligned} \tag{5.9}$$

5.3.3 Reliability and Customer satisfaction features

The key server handles several key trees. It can thus independently choose a different reliability method and degree for each key tree. When some members leave the group, the keying material of each member has to be updated. All members have to update the data encryption key and members belonging to the same trees as the leaving ones have to receive some more keys which could lead to more losses for them. Since members are regrouped in key trees according to their "loyalty" degree (which is here their arrival time), the key server can provide a better reliability to more "loyal members" (by more replicating packets sent to them for example). If leaving members belong to very "loyal" trees, the server will send the new keys with higher reliability for the remaining members of the tree. If the leaving members belong to less "loyal" trees, the key server will use less reliability for the remaining members, thus limiting bandwidth usage.

The proposed scheme can therefore properly manage the customer satisfaction requirement by better handling more "loyal" members. Moreover, this requirement can be provided with a high level of flexibility, parameters being adaptable to the needs of a particular application.

5.4 Performance evaluation based on simulations

The performance of the proposed rekeying protocol has been analyzed with respect to the number of losses for each interval and the cost of rekeying using a discrete event simulator. This performance evaluation shows that the proposed scheme provides better reliability than LKH without increasing the cost of rekeying.

5.4.1 Simulation environment

We consider a secure multicast session of a duration of three hours where the bandwidth limit specified for the rekeying operation is $B = 100Kbps$. We assume that the total number of members for this session is $N = 4096$ and for efficiency reasons, all key trees' degrees are set to $d = 4$ [78]. Given these parameters, in order not to exceed the defined bandwidth limit even in the worst case where all keys of the key tree need to be updated, we set the rekeying interval duration to three minutes.

Membership durations follow a probabilistic exponential distribution with an average of 20000 seconds (that guarantees that members do not leave the group before the end of the session). We assume that the loss probability of each rekeying packet for each member is independent and is equal to $p = 0.1$.

With these assumptions, we take two sets of parameters defined for the Zipf distribution that follow equation 5.2 where $N = 4096$ and $j_{max} = 60$ $\{(\alpha_1 = 1; A_1 = 875), (\alpha_2 = 1.5; A_2 = 1739)\}$. In order to show the efficiency of the proposed solution, we compare its performances with the case where members are regrouped in a single key tree (LKH).

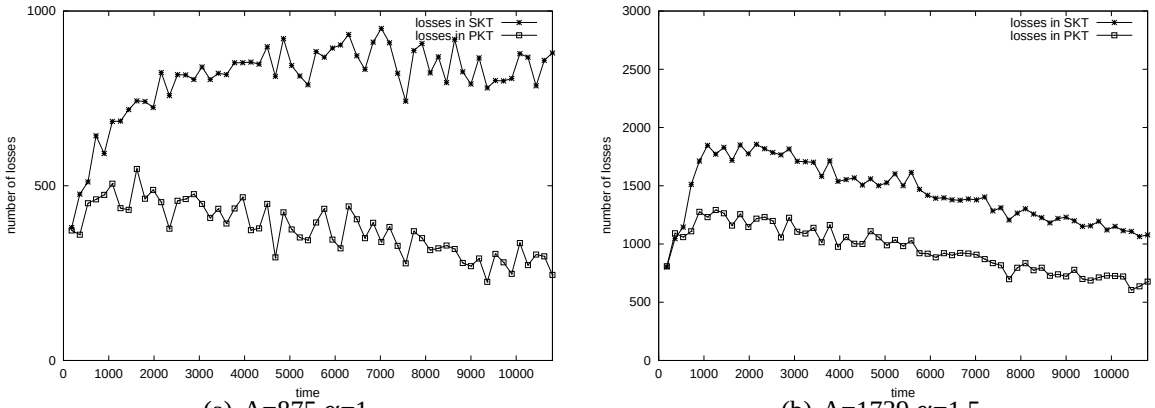
5.4.2 Evaluation of the number of losses

The proposed rekeying mechanism is implemented using the simulator presented in the previous chapter. The simulation parameters and metrics are described in section 4.1. Thanks to this implementation, we first have simulated the rekeying operations for both sets where the key server does not deal with reliability issues and sends the rekeying packets without any additional information. From figure 5.1, we observe that the total number of losses occurring with the classical single key tree scheme (SKT) is always greater than the one with our proposed scheme involving multiple key trees (PKT). Based on these results, we conclude that our scheme offers a better reliability even without implementing any prevention method.

We then evaluate the performance of our solution in the case where the key server offers a basic reliability service, that is replication, whereby all rekeying packets are sent twice to all members. We again observe from figure 5.2 that although the number of losses decreases, the remaining number of losses occurring with the SKT is again larger than in our scheme.

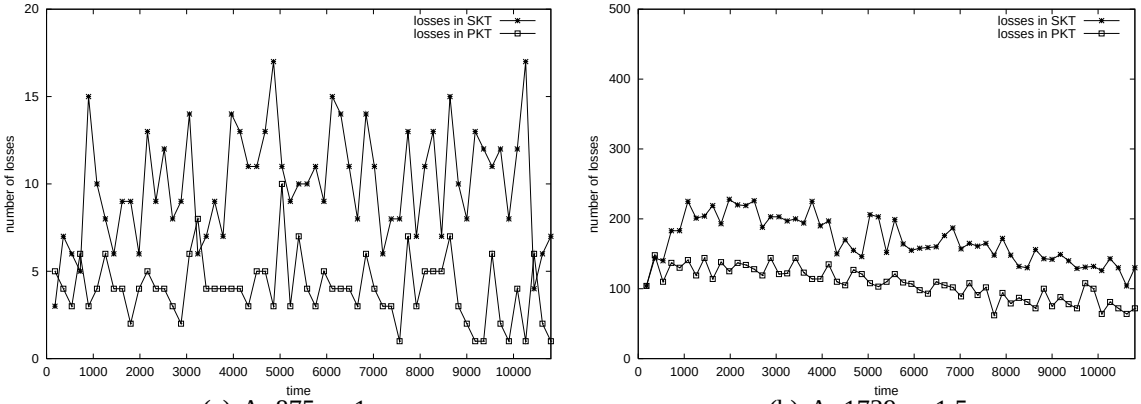
Finally, we evaluate the number of losses experienced by members arriving at the first rekeying interval (and thus supposed to be "loyal" and treated better than members arriving at subsequent intervals). From table 5.2, we again conclude that this number within this specific set of members decreases when a new key tree is defined at each rekeying interval.

From these results, we can conclude that our scheme offers a better reliable delivery of the keying material and the impact of new arrivals on existing members is very low.



(a) A=875,α=1 (b) A=1739,α=1.5

Figure 5.1: Losses during the multicast session where there is no prevention



(a) A=875,α=1 (b) A=1739,α=1.5

Figure 5.2: Losses during a multicast session with replication (n=2)

5.4.3 Memory usage

In this section, we analyze the overhead of this scheme in terms of memory. A straightforward observation is that the memory overhead of this scheme for a member is always lower than with the classical single tree approach. The number of keys that each member keeps in memory is equal to the depth of the key tree it is assigned to. Since the key server defines a different key tree for each rekeying interval, this value will be less than or equal to the one in the case where there is a single key tree.

We have evaluated the same overhead at the server's side with the same simulation parameters and from table 5.3 where the memory cost is computed in terms of number of keys, we conclude that our scheme still slightly outperforms the single key tree scheme.

Reliability options	A=875,α=1		A=1739,α=1.5	
	SKT	PKT	SKT	PKT
No reliability	12510	8674	29294	23764
replication (n=2)	1386	883	3430	2609

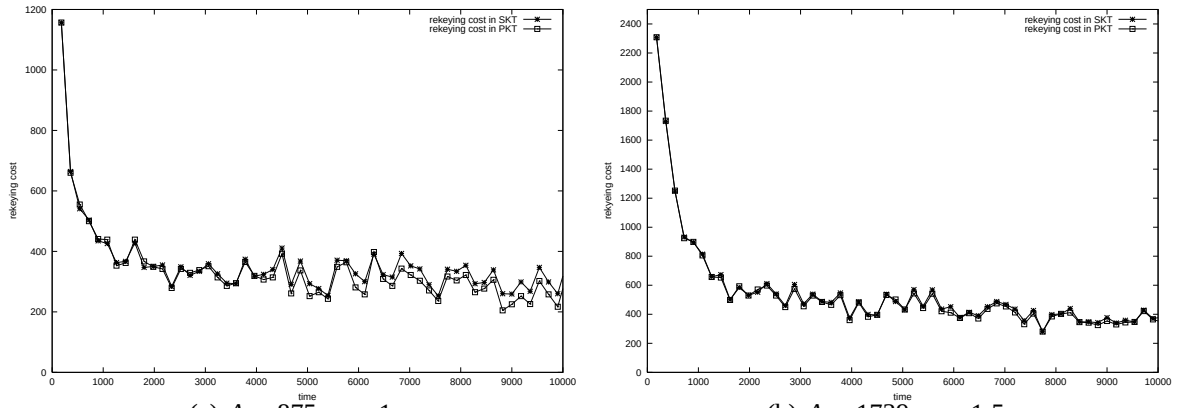
Table 5.2: The number of losses occurring for first arrivals

Memory Cost	SKT	PKT
$A = 875, \alpha = 1$	5461	5439
$A = 1739, \alpha = 1.5$	5461	5408

Table 5.3: The memory cost at the server side

5.4.4 Rekeying cost

In this section, we evaluate the rekeying cost of the proposed scheme with the same simulation parameters and compare this value with the rekeying cost resulting from the classical LKH (SKT). From figure 5.3(a) and 5.3(b), we conclude that the proposed partitioning scheme and the classical LKH scheme show the same behavior in terms of communication overhead.



(a) $A = 875$, $\alpha = 1$ (b) $A = 1739$, $\alpha = 1.5$

Figure 5.3: Evaluation of the rekeying cost

5.5 Conclusion

In the context of multicast applications such as video conferences, where the duration of the session is short, recipients tend to subscribe up to just after the beginning of the session and stay in the group until the end. For such applications, we propose to separately regroup members in different trees with respect to their arrival time. Thanks to this partitioning, the key server treats each set of members assigned to a different key tree independently. We show that the new scheme offers a better reliable delivery of the keying material. We also observe that the

rekeying cost of our scheme is similar to the one of the original LKH scheme. Thanks to this observation, we can conclude that integrating our solution is not costly and offers more reliability and satisfaction since it reduces the impact of the “one affects all” failure from a large set of recipients to a small number of members.

Part II

Denial of Service Prevention

Chapter 6

Denial of Service: State of the Art

6.1 Definition of Denial of Service attacks

6.1.1 Introduction

Denial of Service (DoS) attacks aim at preventing legitimate users from accessing a service they are authorized to access in normal circumstances. The CERT© Coordination Center [1] classifies three kind of DoS attacks in [15]:

- Physical destruction or alteration of network components;
- Destruction or alteration of configuration information;
- Consumption of scarce, limited, or non renewable resources.

First of all, network segments such as computers or routers should be guarded against unauthorized access. Physical security is a prime component in guarding against many type of attacks in addition to denial of service. Furthermore, an improperly configured computer may not perform well or may not operate at all. An intruder may be able to completely control a misconfigured computer or even a network. For example, an intruder can jeopardize the whole network operation by tampering with routing tables in routers.

In this thesis, we consider that the network and its components are properly configured and physically protected against unauthorized access. Consequently, the first two types of attacks will not be under the scope of this chapter and we focus on the last type of attacks where the target of an attacker is the victim's main resources. The intruder may be able to consume a significant portion of computer or network resources such as the memory, the disk space, the

bandwidth or the CPU. Some of the possible attacks against each specific resource are presented in the next section.

6.1.2 Targeted Resources

The memory and/or the disk space

In connection oriented or stateful scenarios, each protocol execution results in the reservation of a number of data structures. By causing a large number of such protocol instances to start with a legitimate node, an attacker can provoke the exhaustion of memory or disk space.

A typical example of such attacks is the "TCP SYN flood" attack described in [13]. When a host attempts to establish a TCP connection to a server, these two entities exchange a sequence of messages. The client sends a SYN message to the server. The server then acknowledges the SYN message by sending a SYN-ACK message to the client. The connection is established only if the client sends back an ACK message. After this step, data can be exchanged between the two peers.

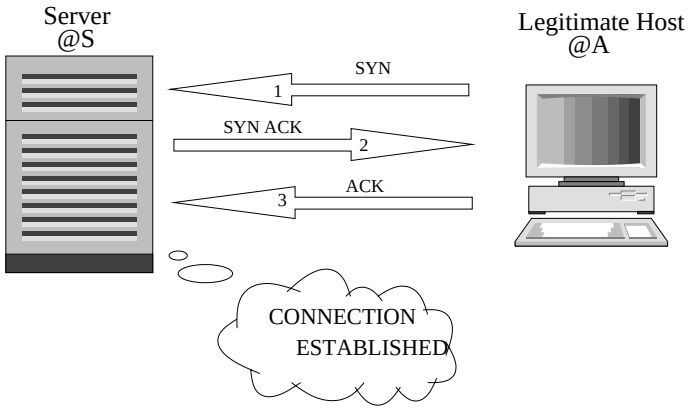


Figure 6.1: A TCP connection establishment

The potential for DoS attacks arises at the point where the server sends a SYN-ACK message but does not receive any ACK message later on. This problem is called the "half-open connection" problem [13]. While sending the SYN-ACK message, the server creates in its system memory a data structure describing all pending connections, that is of finite size. By creating several half-open connections, system memory on the victim server will eventually be exhausted and therefore force the server to reject further connection requests (SYN packets). As a result of such attacks, the target system may exhaust memory, crash, or be rendered otherwise inoperative with respect to legitimate clients.

The bandwidth

In other DoS attacks, intruders aim at consuming all the available bandwidth of the network by generating a large number of packets and preventing legitimate users to use the bandwidth. Typically, in the case of `smurf attacks` [14], intruders forge ICMP ECHO request packets sent to IP broadcast addresses. The Internet Control Message Protocol [61] is used to handle errors and to exchange control messages. If a host receives an ICMP ECHO request packet, it automatically returns an ICMP ECHO reply packet.

Attackers use ICMP ECHO request packets directed to IP broadcast addresses from remote locations. The result is that when all the intermediaries receive forged ICMP ECHO request packets and send replies to the victim, the victim is subject to network congestion that could potentially make the network unusable.

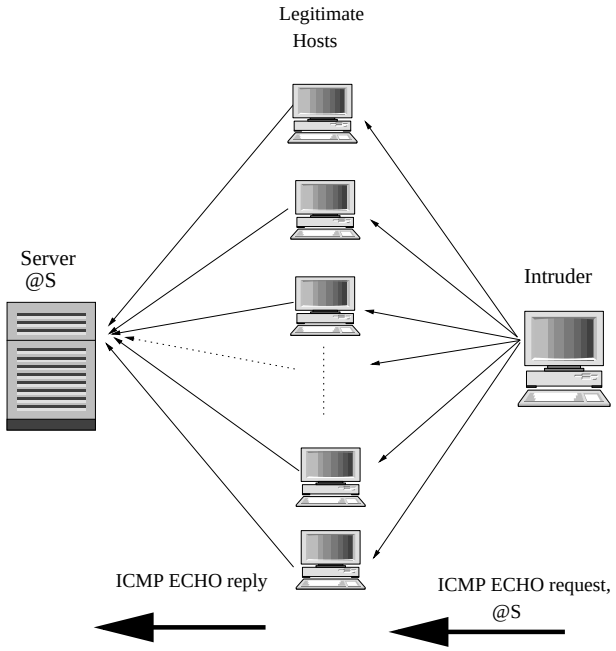


Figure 6.2: An example of a smurf attack

The CPU

Some services such as security may involve computationally intensive operations. Attacks against this kind of protocols may affect CPU utilization. For example, if during a protocol, a server requires the verification of a digital signature for each received packet, attackers sending a large number of bogus packets can provoke the exhaustion of the server's CPU. In order to illustrate the impact of such attacks, the performance of a victim server is analyzed in terms of CPU utilization. Table 6.1 depicts the maximum number of signature verification operations

that a server can perform with a Pentium 4 2.4GHz with 1GB of RAM. The corresponding CPU consumption has been evaluated with OPEN-SSL [3]. Consider an intruder using the total amount of the bandwidth set up to 10Mbps. Such an intruder can send at most 10240 packets of length 1024 bits per second based on these assumptions. However, with 100% of CPU a server can verify at most 4788 signatures. Consequently, this example illustrates a DoS attack against the CPU of the target.

	sign	verify	sign/s	verify/s
rsa 512 bits	0.0008s	0.0001s	1304.3	14650.0
rsa 1024 bits	0.0040s	0.0002s	251.9	4788.2
rsa 2048 bits	0.0245s	0.0007s	40.8	1372.2
rsa 4096 bits	0.1670s	0.0026s	6.0	386.0

Table 6.1: The performance of a server verifying RSA signatures with 100% of CPU utilization

6.1.3 The attack

In a typical DoS attack, the intruder may be able to consume a significant portion of the victim's resources by sending a very large number of attack packets specifically designed to consume resources. Since the rate of requests originating from a legitimate entity lays below a limit that is acceptable for the server, in order to justify an increased rate, DoS attacks persistently emulate several sources by either setting bogus source addresses in requests generated by a single intruder or by having recourse to several sources as distributed intruders, or both.

Definition 6.1 *An **impersonation attack** is an attack whereby an intruder sends to the target messages with a faked source address.*

Intruders can either generate from scratch some requests with bogus source addresses or eavesdrop some requests coming from legitimate sources and send them later on. We define such attacks as replay attacks.

Definition 6.2 *A **replay attack** is an impersonation attack whereby the intruder transmits a message that has already been exchanged between a legitimate party and the target.*

The first approach used by DoS prevention methods therefore consists of screening requests with bogus source identification. Even though a perfect countermeasure against this problem, a straightforward implementation of data origin authentication techniques unfortunately does not solve the DoS problem. As a computing intensive operation based on cryptographic algorithms, like digital signature or encryption, the authentication of the requests becomes the new target for the DoS attack (see figure 6.3).

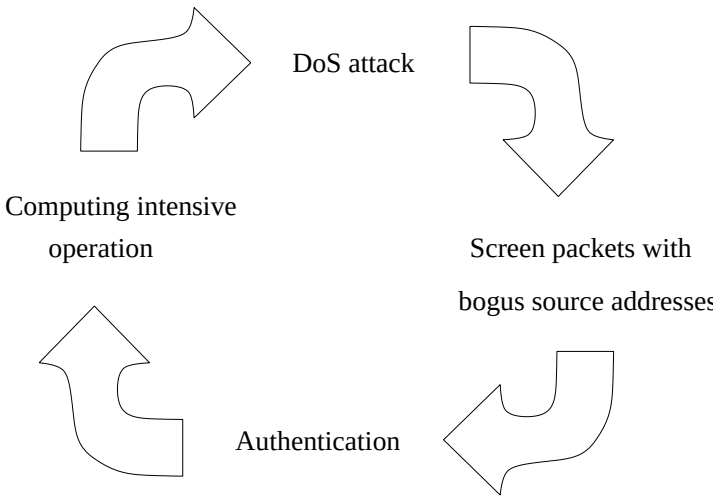


Figure 6.3: The problem of Denial of Service prevention with authentication

In the next section, we review existing techniques designed for terrestrial networks that prevent the victim computer or network from DoS attacks by screening requests with bogus source addresses.

6.2 Denial of Service Prevention for Terrestrial Networks

Existing approaches for Denial of Service prevention fall in one of four major classes: packet tracing, filtering based on forwarding tables, ingress filtering, using weak authentication techniques.

6.2.1 Packet Tracing

Traceback [11] aims at retrieving the actual source of malicious traffic based on path determination techniques used to determine the path(s) taken by the attack flows. Traceback solutions use two different approaches: marking packets with a signature of the routers they pass through, and sending samples of packets to a special collector that analyzes the path. Burch et al. introduced the concept of traceback by selectively flooding network links and monitoring the changes caused in attack traffic. In [8, 11], the proposed solutions implement some marking algorithms in the packet in order to get a history of the path of a packet.

Tracing IP packets with forged source addresses requires complex and expensive techniques to monitor traffic at routers and reconstruct the actual path taken by each packet. In order to prevent this technique from being the new target for DoS attacks, it should be efficient in terms

of bandwidth, memory and CPU utilization. However, tracing involves an important number of additional packets in order to identify the attack path. It also becomes ineffective when the volume of attack traffic is small or the attack is distributed.

Moreover, tracing is typically performed after an attack is detected, possibly too late to avoid damage.

6.2.2 Filtering based on forwarding tables

In another approach called the Reverse Path forwarding (RPF) [16], the router examines all packets received and checks to make sure that the source address appears in the routing table and matches the interface on which the packet was received. Thus, this feature does not allow receiving packets with source addresses that are not in the forwarding table. The router drops such packets.

This filtering approach assumes that the outgoing interface that a router uses to reach a given address, as specified by its forwarding table, is also the valid incoming interface for packets originating from that address. Unfortunately, routing asymmetry on the Internet is common, invalidating this assumption and causing many legitimate packets to be dropped. As a result, this feature is often disabled since it leads to erroneous packet dropping when asymmetric paths are used.

6.2.3 Ingress filtering

The ingress filtering allows a router to filter packets with source addresses that are not within the prefix which is assigned to the subnetwork they originate from. If we take the example described in [24] and depicted in figure 6.4, ingress filtering avoids an attacker whose next hop is `router2` from using "invalid" source addresses which reside outside of the prefix range 204.69.207.0/24. This prevention method ensures that packets leaving the domain of a periphery router have a source address from within the domain, and packets entering have an outside source address, effectively providing a special purpose incoming table only at network ingress.

This filtering technique can drastically reduce the success of source address spoofing if all domains use it. It can also assist service providers in locating the source of the attack. It is hard though, to deploy ingress filtering in all Internet domains. If there are some unchecked points, it is possible to launch DoS attacks from those points.

Moreover, although ingress filtering reduces the impact of IP spoofing attacks, it does not prevent attacks where the intruder uses a forged source address of another host within the permitted prefix filter range. This type of filtering can also break some type of services such as mobile IP. Indeed, traffic to the mobile node is tunneled, but traffic from the mobile node is not

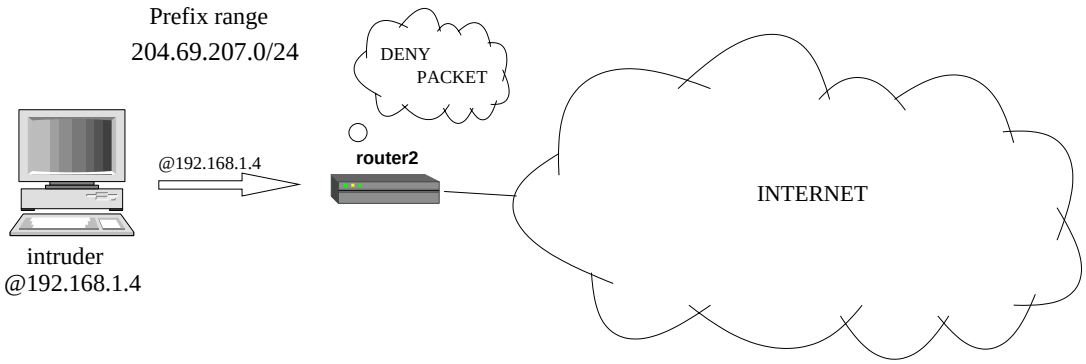


Figure 6.4: An illustration of the ingress filtering method

tunneled. This results in packets from the mobile node(s) which have source addresses that do not match with the network where the station is attached.

6.2.4 Using weak authentication: The anti-clogging technique

The IP security protocol (IPSEC) [5] proposes a framework for key exchange and negotiation of security associations (SA) denoted by ISAKMP [41]. This protocol allows peer entities to select and negotiate the SA with the incorporation of a mechanism to counter DoS attacks. ISAKMP is based on the anti-clogging technique [33] where an exchange of weak authentication information is performed before initiating any resource-intensive verification.

The anti-clogging technique is based on the exchange of some `cookies` that are generated by each communicating entity and have some special characteristics fostering a fast and efficient exchange as follows:

- each cookie is generated based on a local secret known only by its generator;
- each cookie depends on the addresses of the communicating entities;
- each cookie is verifiable by the generator;
- cookie generation and verification are efficient in CPU and memory consumption.

The cookie exchange protocol that is depicted in figure 6.5 allows the verification of the client's presence at the claimed IP address. The client initially sends a request to the server with its cookie. This cookie is used for self-defense reasons. Upon the reception of this request, the server sends without any verification and any resource consumption another cookie generated based on the claimed IP address of the client retrieved from the request. Because the server's

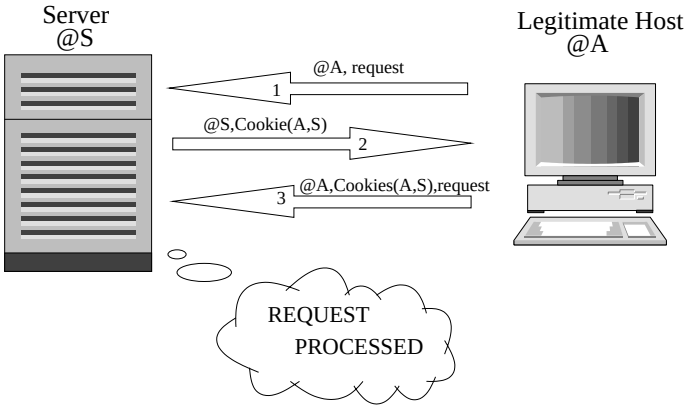


Figure 6.5: The anti-clogging technique: process of a legitimate request

reply including servers' cookie is sent to the claimed client address, an intruder using a bogus IP address cannot get the server's cookies.

In case of a request with a bogus source address, the server will not receive any subsequent message including both the client's and the server's cookies, hence the DoS attack intended by the bogus request will not succeed. Only requests originating from legitimate parties will thus reach the server.

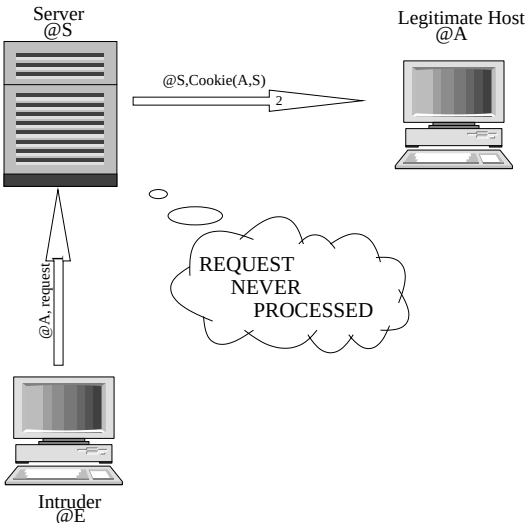


Figure 6.6: The anti-clogging technique:case with a faked request

Moreover, the computation and the verification of the cookie by the server are based on a simple keyed hash function [35] requiring low CPU usage in comparison with CPU intensive strong authentication and key generation operations [64]. No resource reservation takes place before the completion of the successful cookie exchange [45]. Each ISAKMP message contains the pair of cookies generated by the initiator and the responder based on the anti-clogging technique.

6.3 Conclusion

Assuring that each IP packet carries a valid source address would be a reasonable countermeasure for most DoS attacks. Existing approaches solve part of the problem but still present some shortcomings and are not very effective. Tracing packets' source is based on expensive operations and not always effective apart from the fact that it does not prevent the attack. Ingress filtering could significantly reduce packet spoofing in the Internet if it was deployed in all domains. The deployment of ingress filtering requires exhaustive involvement of Internet service providers that are reluctant to such an undertaking due to its high operational cost. Similarly, frequently asymmetric paths akin to Internet are the main reason behind the inefficiency of the reverse path forwarding technique.

IP spoofing attacks can also be foiled by authentication techniques used at the very beginning of any protocol exchange. Since strong authentication itself can be a target for DoS attacks, the approach that has been taken to resolve this conflict consists of authentication phase prior to the actual protocol exchanges in order to screen out DoS attacks based on source address spoofing. The anti-clogging technique involves the exchange of some information called cookies that are generated based on local secret information. By requiring in each request message the presence of a cookie that it generated during the initial cookie exchange, the server can eliminate bogus packets.

Cookies provide a very efficient DoS prevention method for meshed terrestrial networks. Unfortunately such techniques are not suitable to the satellite environment mostly due to the inherent broadcast capability of the satellite segment. In the next chapter, we first show the shortcomings of existing DoS prevention techniques when they are implemented in the satellite environment and then describe a new basic identification protocol aiming at preventing requests with bogus source address in this environment.

Chapter 7

Denial of Service Prevention in Satellite Networks

7.1 Requirements for Satellite Networks

7.1.1 Environment

In this section we focus on the main characteristics of networks integrating the satellite segment and analyze their advantages and drawbacks compared to terrestrial networks.

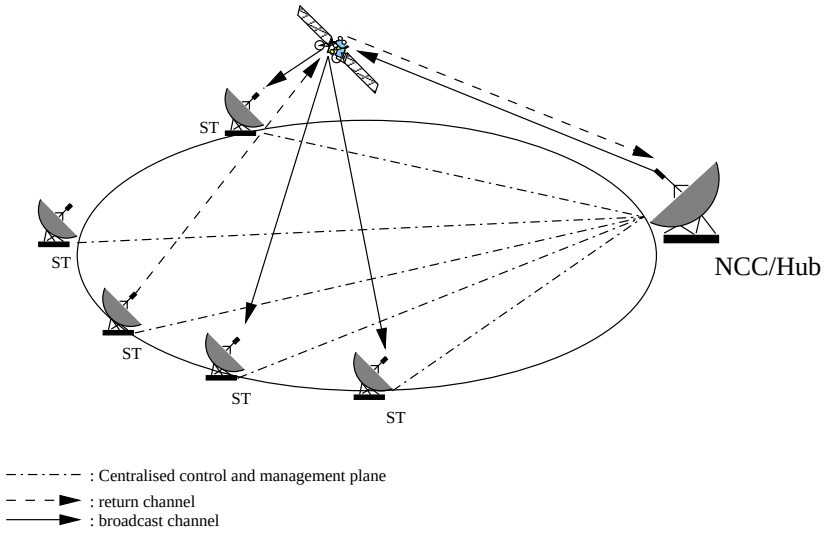


Figure 7.1: A Satellite System Architecture

As depicted in figure 7.1, in a typical satellite system architecture, network control and

management are centralized under the control of the network control center (NCC) which is usually co-located with the Hub-station responsible for the data-plane and some management functions such as satellite system address management. A satellite terminal (ST) is assigned a unique identity such as its physical address and sends control messages such as connections requests to the NCC. These messages are transmitted via the satellite channel using a protocol like DVB-RCS [22].

The first and most important characteristic of a satellite network is the inherent broadcast capability of the satellite segment over a large area. Satellites are the most natural and economical mean for providing broadcast and multicast services. However, this advantage brings a weakness with it in terms of security. Attacks such as eavesdropping imply a strong requirement of some security services including those that prevent denial of service.

The main drawbacks of a satellite architecture are the inherently **high end-to-end delay** which is close to 300 ms [30] and **the cost of the link**. Extra delay that would be caused by including some security services requiring some additional exchanges between satellite terminals and the server would be a significant drawback that is unacceptable for many applications and could also be a new target for DoS attacks. Furthermore, the satellite network infrastructure and bandwidth can be more expensive than terrestrial alternatives. Therefore, future satellite systems should be optimized for the transport and delivery of a wide range of services and intruders should be prevented from taking advantages of these two main weaknesses.

On the other hand, the inherent centralized management is a clear advantage of satellite networks over terrestrial networks and while defining a new security protocol (including DoS prevention protocols) one might take this aspect into consideration. This property can help to relax the constraints on the security services and have the way for the design of a solution with minimal resource usage.

These characteristics have a strong impact on the design and performance of a satellite system and should be carefully considered while defining a new DoS prevention technique adapted for satellite networks. In the next section, we review the shortcomings of existing DoS prevention techniques designed for terrestrial networks with respect to the requirements of satellite networks.

7.1.2 Shortcomings of existing DoS prevention techniques

DoS prevention techniques described in chapter 6 are ineffective in the satellite environment for mainly two reasons: the broadcast nature of communications and the end-to-end latency. In this section, we will review the weaknesses of each existing DoS prevention technique when adopted by satellite networks.

Packet traceback

Tracing packets is already an expensive approach for terrestrial networks. Sending samples of traffic packets to a special collector that analyzes the path becomes a more expensive approach when implemented in satellite networks. Since the cost of the bandwidth should be optimized, this technique cannot be chosen for such networks.

Ingress filtering

Ingress filtering could be the most efficient solution for terrestrial networks. However this technique does not prevent attacks when intruders use a forged source address of other hosts within the permitted prefix filter range. In the case of satellite networks, a satellite segment can cover a very large area such as a continent in one network. Consequently, intruders can forge a very large number of source addresses even when this approach is adopted. Therefore, ingress filtering is inefficient for satellite networks.

Filtering based on forwarding tables

Filtering based on forwarding tables cannot be implemented in satellite networks since there is no forwarding table in the segment. Moreover, some protocols such as UDLR [20] imply the use of some asymmetric paths.

The anti-clogging technique

The anti-clogging technique also turns out to be inefficient in case of satellite networks. Due to the inherent broadcast nature of satellite communications, the cookies generated by the server and intended for sources impersonated by the intruder would still be received by the intruder who will then be able to transmit bogus packets with the expected replies including the server's cookies, successfully masquerading as several different sources. The screening of bogus source addresses that is perfectly suitable to mesh networks would thus be totally ineffective in case of broadcast media like the satellite segment. Moreover, the additional delay that would be caused by including an anti-clogging phase with three additional messages in the satellite protocols will sharply affect the performance of the network.

Summary

Packet tracing, ingress filtering or filtering based on forwarding tables cannot be implemented in satellite networks for efficiency and architectural reasons. The anti-clogging technique is

not suitable either, mostly due to the inherent broadcast capability of the satellite segment. However, the concept behind this technique, that is, adding an inexpensive but secure operation before any resource consuming operation provided the main inspiration for our solution.

Based on this concept and the special characteristics of the satellite environment, we propose an efficient identification protocol, that mostly prevents *impersonation* attacks (see definition 6.1 before the beginning of any application.

7.2 The basic protocol

7.2.1 Preliminaries

By taking advantage of the existing centralized management and by reducing the impact of the main inherent weaknesses of satellite networks (the broadcast capability and the end-to-end latency), we propose an efficient identification protocol that allows satellite servers to quickly discard bogus requests. As in the anti-clogging technique, we introduce a preliminary weak authentication technique in order to prevent DoS attacks without lowering the other security properties.

The novelty of the suggested technique is that the process of message authentication is divided into several steps and at each of the successful step, the probability of legitimacy of a message increases. In the version presented here, the protocol includes mainly two steps. As illustrated in figure 7.2, most of faked messages are rejected at the first step of the protocol using an inexpensive screening method. In the second step of the protocol, a slightly expensive but strong verification method is used to assure the ultimate authentication of the message. Since most bogus messages are filtered out at the first step, the probability of several packets reaching the second step is very low, so is the likeliness of a successful DoS attack.

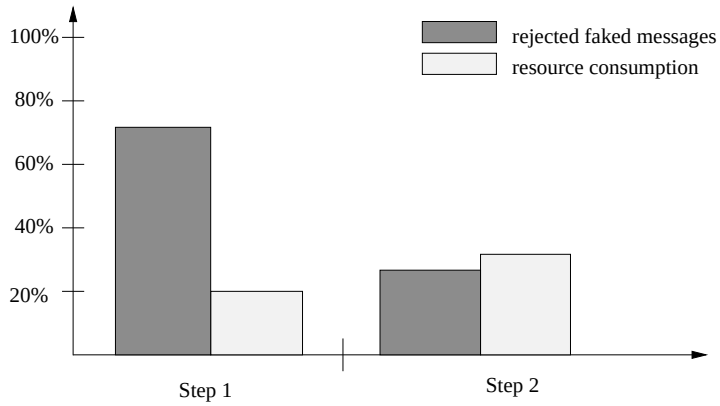


Figure 7.2: Concept of the identification protocol

In this section, we first describe a basic identification protocol requiring a strong synchronization mechanism between the satellite terminals and the network control center (NCC). We further enhance this protocol in an improved version that only requires loose clock synchronization between the NCC and the terminals. In both versions of the protocol, each satellite terminal ST_i is assigned a unique identity Id_i, such as the MAC address of the terminal. Initially, each ST_i shares a secret key K_{ST_i} with the NCC which is precomputed by the NCC based on Id_i and a local secret known only by the NCC denoted by K_{NCC} as follows:

$$K_{ST_i} = MAC(K_{NCC}, Id_i) \tag{7.1}$$

MAC denotes a cryptographic keyed hash function such as HMAC [35].

Moreover, the NCC subdivides the time in fixed intervals T_j during which it reliably broadcasts a different nonce N_j for the purpose of replay detection. Unlike DoS prevention techniques destined to terrestrial meshed networks, the proposed protocol requires only one message flow from the satellite terminal to the NCC.

7.2.2 The structure of the message

As explained in the previous section, the proposed protocol requires only one message flow from the satellite terminal to the NCC. The figure 7.3 depicts the structure of a request message based on the following notation:

- Id_i denotes the identity of the satellite terminal ST_i which is unique for each ST;
- *Seq* denotes the sequence number of the message sent by a ST and is always equal to the nonce N_j of the current interval T_j;
- h denotes a cryptographic hash function such as MD5 [65] or SHA1 [48] and *MAC* denotes a cryptographic keyed hash function such as HMAC [35];
- K_{ST_i} denotes the secret key shared between ST_i and the NCC defined as in equation 7.1;
- $a|b$ denotes the concatenation of a and b;
- M denotes the payload of the request message sent by a ST.

7.2.3 Verification protocol

Upon reception of a request sent by a satellite terminal ST_i during the interval T_j, the NCC first verifies that the sequence number *Seq* is equal to the actual nonce N_j. It then computes the secret

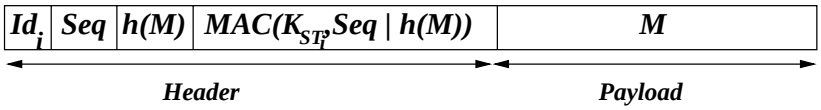


Figure 7.3: The structure of a message sent by ST_i at time T_j

shared key K_{ST_i}, using the identity Id_i retrieved from the header of the request message. In order to verify the authenticity of the request, the NCC computes the value $MAC(K_{ST_i}, Seq|h(M))$ using the $h(M)$ value retrieved from the header. If this computed value and the one retrieved from the header match, the NCC needs to further verify the integrity of the message by evaluating the hash value of the payload and comparing the result with the corresponding value received with the message.

This two-step verification process summarized in figure 7.4 allows for fast discarding of control messages with obvious inconsistencies. Once both verification steps of a request sent by a ST end with success, the NCC processes the request and allocates necessary resources for it.

For each interval T_j:

Define and reliably broadcast N_j

The verification of a request received during interval T_j:

Request = { Id_i**,** Seq**,** $h(M)$**,** $MAC(K_{ST_i}, Seq|h(M))$**,** M **}**

```
if Request.Seq ≠ N_j
  REJECT
else
  compute K_STi = MAC(K_NCC, Request.Id_i)
  compute MAC(K_STi, Request.Seq|Request.h(M))
  if MAC(K_STi, Request.Seq|Request.h(M)) ≠ Request.MAC(K_STi, Seq|h(M))
    REJECT
  else
    compute h(Request.M)
    if h(Request.M) ≠ Request.h(M)
      REJECT
    else
      ACCEPT
```

Figure 7.4: Verification process of a request with the basic protocol

7.2.4 Security Analysis of the protocol

In the previous chapter, we gave the type of DoS attacks prevented in the case of terrestrial networks with existing techniques. In order to differentiate several types of impersonation attacks in our specific protocol, we define the following three attacks. Since in order to detect replay attacks, the NCC broadcasts a different nonce N_j at each interval, we differentiate basic replay attacks from replay attacks processed within an interval.

Definition 7.1 *A basic replay attack is defined as a replay attack whereby during an interval,*

the intruder sends some legitimate messages exchanged by legitimate parties during previous intervals.

Theorem 7.1 *The protocol is secure against basic replays.*

Proof - This type of replays are easily detected by the NCC based on the difference between the current nonce value and the value included in the header of the replayed request. ∎

Definition 7.2 *We define a basic impersonation attack as an impersonation attack that is not a replay attack, that is, whereby the intruder creates a new message with faked source addresses.*

Theorem 7.2 *The protocol is secure against basic impersonation attacks*

Proof - K_{ST_i} is a secret key shared only by the corresponding ST_i and the NCC. Since its computation is based on the use of a MAC with K_{NCC} which is only known by the NCC, computing K_{ST_i} turns out to break the security of a MAC function. ∎

Definition 7.3 *Replay attacks within an interval are defined as attacks whereby the intruder sends some legitimate messages originating from the same interval.*

Theorem 7.3 *The protocol is secure against replays within an interval.*

Proof - We now discuss the case where the generation and replay of a request are performed within the same interval. For a satellite network, it is practically impossible to intercept a legitimate terminal's request only from the down-link, that is, before an end-to-end latency for a GEO-satellite system which is approximatively equal to 300 ms [30]. Thus, if the interval is set to a value less than the end-to-end latency and a different nonce is used in each interval, the intruder cannot successfully perpetrate this type of replay. ∎

Finally, before allocating any resource, the server needs to be sure about the authenticity of the payload itself. This operation is performed in the second step where the NCC computes the value $MAC(K_{ST_i}, Seq|h(M))$ using the $h(M)$ value retrieved from the header.

7.3 Extension of the basic protocol

In this section, we propose to improve the basic protocol by introducing a stateful verification mechanism that allows to relax the synchronization requirement. The improvement is based on the following idea: by relaxing the synchronization between the NCC and the terminals, some replays occurring under race conditions would go undetected; in order to detect those replays, the NCC will keep some state information about successful authentication attempts occurring in each interval.

7.3.1 Description of the second version

At the beginning of a verification interval with a new defined nonce N_j, when the NCC receives and successfully verifies the authenticity of the first request sent from a legitimate ST, it keeps some information about this request in a table reset at the beginning of each interval. The arguments of the table for each different satellite terminal authenticated within interval T_j are:

- Id_i, the identity of ST_i;
- K_{ST_i}, the secret key of ST_i shared with the NCC;
- *Seq*, the sequence number of the message (initially equal to N_j).

The difference between the previously described basic protocol and this improved one is that the ST will increment the sequence number *Seq* initially equal to the nonce of the period for every new request within the valid interval. Thus, when the NCC will receive another request from a terminal which previously has been added to the table, it will also verify that the sequence number sent by the ST is greater than the one in the table, and replace the value with the new one if the message is authenticated. Since an intruder cannot generate a valid request, it therefore will not be able to make replay attacks within a same interval where requests have been eavesdropped, because the sequence number changes for each message.

The verification process is described in detail in figure 7.5

7.3.2 Security analysis

In this section, we show that this protocol also presents all the secure properties offered by its old version.

Theorem 7.4 *The improved protocol is secure against basic replays and basic impersonation attacks.*

Proof - When the NCC receives some bogus messages originating from basic replays, it can almost immediately detect them thanks to the sequence number of the message which depends on the nonce of the actual interval. Moreover, as in the previous version of the protocol, generating a basic impersonation message is only possible if the intruder can compute the personal key of a legitimate ST from any previous authentication information. MAC functions are assumed to be secure and do not reveal any information about the secret key: thus the improved protocol is secure against basic impersonation attacks. ∎

For each interval T_j:

Preliminaries:

Define and reliably broadcast N_j
Define an initially empty table which arguments for each authenticated ST_i are:
the identity of the ST_i, Id_i
its shared key K_{ST_i} and,
the last received sequence number Seq_i

The verification of a request received during interval T_j:

Request = { Id_i**,** Seq**,** $h(M)$**,** $MAC(K_{ST_i}, Seq|h(M))$**,** M **}**
if $Request.Id_i \notin table$
if $Request.Seq \neq N_j$
REJECT
else
compute $K_{ST_i} = MAC(K_{NCC}, Id_i)$
compute $MAC(K_{ST_i}, Request.Seq|Request.h(M))$
if $MAC(K_{ST_i}, Request.Seq|Request.h(M)) \neq Request.MAC(K_{ST_i}, Seq|h(M))$
REJECT
else
compute $h(Request.M)$
if $h(Request.M) \neq Request.h(M)$
REJECT
else
add { Id_i , K_{ST_i}, Seq } to the table
ACCEPT
else if $Request.Idi \in table$
if the value $Seq_i \leq Request.Seq$
REJECT
else
compute $MAC(K_{ST_i}, Request.Seq|Request.h(M))$
if $MAC(K_{ST_i}, Request.Seq|Request.h(M)) \neq Request.MAC(K_{ST_i}, Seq|h(M))$
REJECT
else
compute $h(M)$
$h(M) \neq Request.h(M)$
REJECT
else
replace $Seq_i = Request.Seq$
ACCEPT

Figure 7.5: Verification process of a request with the improved protocol

Theorem 7.5 *Even if the NCC allows an extension of the duration of one interval, replays within an interval would not be possible.*

Proof - If two messages with the same source address and sequence number are received within the same interval, the NCC will necessarily reject one of them based on the expected sequence number. ∎

The new version also optimizes the CPU consumption because the NCC will not need to compute the shared key K_{ST_i} if this value is already present in the table defined below. We assume also that the NCC can allocate necessary resources for the dynamic table re-initialized at the beginning of each interval and that searching if a key is already existing in the table is much more efficient than computing its value using MAC algorithms.

7.4 Cost Evaluation

In order to evaluate the possibility of DoS attacks with our protocol, we consider the cost of memory and CPU usage due to an attack performed by an intruder taking advantage of all available bandwidth. Since the NCC only keeps track of authenticated terminals in an array that is reset at each interval, the resulting memory usage is optimal and cannot be exploited by an intruder.

As to the evaluation of the CPU usage, we first compare the cost of computing the MAC over the entire message and the MAC over the hash value of the message in order to validate the necessity of the first step of the verification of the proposed protocol. The table 7.4 shows the number of messages verified in one second and compares the evaluation of the two MAC values. CPU consumption has been evaluated with OPEN-SSL [3], where the HMAC and MD5 cryptographic functions are respectively used for MAC and hash functions. We consider that the message is 500 bytes long. With these assumptions we see that for a Pentium 4 2,4GHz with 1GB of RAM, 326435 packets are verified in one second in the case of the verification of h(M) as opposed to 206829 packets in the case where the input of the MAC function is the message M itself. Thus, the first step of verification increases the number of verified packets by 58%.

Workstation Properties	Number of verified packets per second	
	$MAC(h(M))$	$MAC(M)$
sparc v9 300Mhz RAM = 256MB	49783	26022
sparc v9 440Mhz RAM = 640MB	57846	33055
Pentium 3 1Ghz RAM = 896MB	141312	103280
Pentium 4 2,4Ghz RAM = 1GB	326435	206289

Table 7.1: CPU usage of the NCC per second for the compute of HMAC over M and $h(M)$

We then consider a scenario where the NCC is under a DoS attack by an intruder that uses the total amount of the bandwidth set up to 10Mbps. Such an intruder can send at most 2621 packets per second based on these assumptions. Table 7.4 shows the percentage of CPU usage by the NCC as required for message authentication. One can note that the CPU usage due to

the computation of a MAC over a hash value is approximatively 50% more efficient than the one over the message itself.

Workstation Properties	CPU usage per second for verification	
	$MAC(M)$	$MAC(h(M))$
sparc v9 300 Mhz RAM = 256MB	10,07%	5,26%
sparc v9 440 Mhz RAM = 640MB	7,93%	4,53%
Pentium 3 1Ghz RAM = 896MB	2,54%	1,85%
Pentium 4 2,4GHz RAM = 1GB	1,27%	0,80%

Table 7.2: CPU usage per second of the NCC for the verification in case of a DoS attack

7.5 Conclusion

An efficient DoS prevention approach taken for meshed terrestrial networks is the anti-clogging technique. This technique however turns out to be inefficient to thwart DoS attacks in satellite networks because of the inherent broadcast capability of the satellite system.

We proposed two versions of an efficient identification protocol preventing DoS attacks for centralized control plane protocols in satellite networks. Since the basic protocol requires a tight clock synchronization between the satellite terminals and the NCC, we proposed an improved protocol to relax the synchronization requirement assuming that the NCC keeps some state information about successful authentication attempts in each interval. Using this improved technique, we showed that intruders controlling the total bandwidth of the satellite link can only use a very small percentage of the NCC's CPU without requiring a strong synchronization mechanism.

Conclusions and Future work

Satellite networks are a very good candidate for multicast applications. Thanks to their natural broadcast capability and their geographically extended coverage, a network source can reach a very large number of recipients by transmitting a single copy of a packet to the satellite segment which will directly forward this packet to the end-users without traversing any intermediate nodes. Moreover, satellite broadcast channels are characterized by a high data rate. However, such networks remain to be deployed since their particular benefits raise strong requirements in terms of security. For example, any user located within the coverage of the spot beam of satellite can receive any packet transmitted by this satellite. Very few security solutions have been proposed in the context of satellite networks and most of them are proprietary. Since satellite networks are expected to be widely used in the near future, security mechanisms offering confidentiality, authentication and availability are strongly required.

In this thesis, we analyzed and suggested solutions for the two main security issues raised by multicast communications in satellite networks that are **group rekeying** and **denial of service prevention**. In the first part of the thesis, we investigated the problem of group rekeying and analyzed existing secure protocols targeting terrestrial networks. We showed that the Logical Key Hierarchy (LKH) scheme is the most convenient group rekeying protocol for satellite networks, since its execution does not depend on the topology of the network. However, this rekeying scheme still lacks from meeting reliability and customer satisfaction requirements. In chapter 2, we analyzed the reliability requirements of key delivery based on LKH and identified two specific relationships between rekeying packets sent during the secure multicast session. These strong relationships between rekeying packets force the use of reliability solutions. We presented existing studies that have focused on this issue and analyzed their performance.

In chapter 3, we showed that all existing reliable group rekeying protocols still suffer from the “one affects all” failure which occurs when the arrival or departure of a member affects the whole group. Indeed, in LKH, any rekeying operation requires at least the update of the data encryption key which is shared by all members of the group. We suggested a new approach that takes into account group members’ “loyalty” where the key server provides a strongly reliable key delivery to “loyal” members in order to minimize the losses they experience arising from the rekeying operations caused by less “loyal” members. The notion of “loyalty” can vary from an application to another. In order to investigate the relevance of this concept in real multicast applications, we proposed a classification of such applications with respect to their requirements in terms of security and their pricing mechanisms. Based on this classification, we first

proposed an approach that consists of separately treating members according to their membership duration and presented a strongly reliable key delivery protocol for permanent members that combines proactive FEC techniques with proactive replication techniques. This protocol is based on a new user-oriented key assignment algorithm whereby a group member only needs to receive one FEC block to recover all its required keying material. We then investigated the reliability parameters in order to ensure that almost all permanent members receive their keying material before the end of the rekeying interval. In chapter 4, we gave a performance evaluation of our protocol based on simulation and concluded that it minimizes the impact of rekeying operations over **permanent** members while slightly increasing the rekeying cost.

We also proposed, in chapter 5, another partitioning method for short multicast sessions such as video-conference applications. In this scheme, members are regrouped according to their arrival time in a new key tree defined at the beginning of the corresponding rekeying interval. Thanks to this partitioning scheme, the key server reduces the number of loss events when compared with the classical LKH scheme while keeping the rekeying cost close to the original one.

In the second part of the thesis, we investigated the problem of denial of service prevention. In chapter 6, we first defined denial of service attacks aiming at exhausting the resource of a target system. We then presented and analyzed existing DoS prevention techniques destined to terrestrial networks. In chapter 7, we showed that these techniques are not suitable to satellite networks mainly because of the inherent broadcast capability and the high end-to-end delay of these networks. In order to provide to the server the capability of screening bogus requests almost immediately, we defined two versions of a two-step identification algorithm where the key server rejects the maximum number of faked requests at the first step without consuming a large amount of resources.

In conclusion, this study reveals that security concerns cannot be addressed without taking into consideration some other crucial network parameters and real-life expectations. Security mechanisms may require an important amount of resources such as memory, bandwidth and CPU. Consequently, the integration of such mechanisms should not introduce new security concerns such as Denial of Service attacks. They also may have a strong impact on the reliability of multicast data transmission if they are not carefully integrated into applications. For example, a rekeying operation over lossy networks can automatically revoke a legitimate recipient from the group. Since there always exist some tradeoffs between security, scalability and reliability requirements, we suggested a classification of recipients with respect to some real-life criteria in order to optimize system parameters and offer a better service to “loyal” recipients.

Future Directions

An interesting direction that needs further research is the analysis of the problem of key authentication in group rekeying protocols. Hence, if rekeying packets are not authenticated, some

malicious recipients can generate some keys while impersonating the server and prevent legitimate recipients from accessing the service. Current group rekeying protocols use asymmetric authentication techniques in order to let the members verify the origin and the integrity of the keying material. As depicted in this thesis, in large groups, rekeying operations occur very frequently and may require a large amount of bandwidth. It is thus desirable to optimize the cost of signature schemes. For example, transmitting n authenticated key envelopes increases the bandwidth usage more than twice (a RSA signature length is much higher than a symmetric encryption key). By combining existing multicast data authentication schemes with the relationships between rekeying packets, it is certainly possible to reduce this cost.

Bibliography

[1] "CERT Coordination Center", http://www.cert.org.

[2] "GEOCAST: Multicast over Geostationary EHF satellites", IST-1999-11754.

[3] "The OpenSSL Project", http://www.openssl.org.

[4] K. Almeroth and M. Ammar, "Collection and modelling of the join/leave behavior of multicast group members in the mbone", In *Proceedings of High Performance Distributed Computing Focus Workshop (HPDC'96)*, Syracuse, New York USA, August 1996.

[5] R. Atkinson, "Security Architecture for the Internet Protocol", RFC 1825, August 1995.

[6] J. W. Atwood, "A Classification of Reliable Multicast Protocols", *IEEE Network,* 18(3), May 2004.

[7] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication", *Lecture Notes in CS*, 1109:1–15, 1996.

[8] S. Bellovin, M. Leech, and T. Taylor, "ICMP traceback messages", IETF Draft, February 2003, http://www.ietf.org/internet-drafts/draft-ietf-itrace-04.txt.

[9] M. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet Packet Loss: Measurement and Implications for End-to-End QoS", In *International Conference on Parallel Processing*, August 1998.

[10] B. Briscoe, "MARKS: Zero side-effect multicast key management using arbitrarily revealed key sequences", In *First International Workshop on Networked Group Communication*, November 1999.

[11] H. Burch and W. Cheswick, "Tracing anonymous packets to their approximate source", In *Systems Administration Conference*, December 2003.

[12] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions", In *Proceedings of IEEE Infocom'99*, 1999.

[13] CERT Coordination Center, "CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks", September 1996, http://www.cert.org/advisories/CA-1996-21.html.

[14] CERT Coordination Center, "CERT Advisory CA-1998-01 Smurf IP Denial-of-Service Attacks", January 1998, http://www.cert.org/advisories/CA-1998-01.html.

[15] CERT Coordination Center, "Denial of Service Attacks", February 1999, http://www.cert.org/tech_tips/denial_of_service.html.

[16] CISCO IOS, "Unicast Reverse Path Forwarding", 1999.

[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1992.

[18] S. E. Deering, "RFC 1112: Host extensions for IP multicasting", Aug 1989.

[19] S. E. Deering, *Multicast Routing in a Datagram Internetwork*, Ph.D. Thesis, Stanford University, 1991.

[20] E. Duros, W. Dabbous, H. Izumiyama, N. Fujii, and Y. Zhang, "A Link-Layer Tunneling Mechanism for Unidirectional Links", RFC 3077, March 2001.

[21] ETSI EN 300 421, "Digital Video Broadcasting (DVB), Framing structure, channel coding and modulation for 11/12GHz satellite servers", 1997.

[22] ETSI EN 301 790, "Digital Video Broadcasting (DVB), Interaction Channel for Satellite Distribution Systems", 2003.

[23] W. Fenner, "Internet Group Management Protocol, Version 2", Request For Comments 2236, November 1997, see also draft-ietf-idmr-igmpv3-and-routing-01.txt for IGMP v3.

[24] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service attacks which employ IP source address spoofing", RFC 2827, may 2000.

[25] A. Fiat and M. Naor, "Broadcast encryption", In D. R. Stinson, editor, *Advances in Cryptology - Crypto '93*, pp. 480–491, Berlin, 1993, Springer-Verlag, Lecture Notes in Computer Science Volume 773.

[26] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Lightweigth Session and Application Layer Framing", In *IEEE/ACM Trans. on Networking*, December 1997.

[27] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams", In B. S. K. Jr., editor, *Advances in Cryptology - Proceedings of CRYPTO 97*, volume 1294 of *Lecture Notes in Computer Science*, pp. 180–197, Santa Barbara, California, USA, August 1997, Springer Verlag.

[28] O. Goldreich, S. Goldwasser, and S. Micali, "On the Cryptographic Applications of Random Functions", In *CRYPTO*, pp. 276–288, 1984.

[29] P. Golle and N. Modadugo, "Streamed authentication in the presence of random packet loss", In *Proceedings of NDSS*, 2001.

[30] T. Henderson, *Networking over Next-Generation Satellite Systems*, Ph.D. Thesis, University of California at Berkeley, 1999.

[31] M. Howarth, S. Iyengar, Z. Sun, and H. Cruickshank, "Dynamics of Key Management in Secure Satellite Multicast", *IEEE Journal on Selected Areas in Communications*, 22(2), 2004.

[32] S. Josset, L. Duquerroy, M. Önen, M. Annoni, G. Boiero, and N. Salis, "Satellite IPSEC: An optimized way of securing multicast wireless communications", *Information Security Solutions Europe*, october 2002.

[33] P. Karn and W. Simpson, "Photuris : Session-Key Management Protocol", RFC 2522, march 1999.

[34] S. K. Kasera, J. Kurose, and D. Towsley, "A comparison server-based and reciver-based local recovery approaches for scalable reliable multicast", In *IEEE INFOCOM*, March 1998.

[35] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC : Keyed Hashing for Message Authentication", RFC 2104, February 1997.

[36] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, "Batch rekeying for secure group communications", In *Tenth International World Wide Web conference*, pp. 525–534, 2001.

[37] J. C. Lin and S. Paul, "A Reliable Multicast Transport Protocol", In *IEEE INFOCOM*, March 1996.

[38] S. Lin and D. J. Costello, *Error Correcting Codes: Fundamentals and Applications*, Prentice Hall, Englewwod Cliffs, NJ, 1983.

[39] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical Loss-Resilient Codes", In *ACM Symposium on Theory of Computing*, pp. 150–159, 1997.

[40] G. Maral and M. Bousquet, *Satellite Communications Systems: systems, techniques,technology*, John Wiley & Sons, 1993.

[41] D. Maughan, M. Schertler, M. Schneider, and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[42] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[43] S. Miner and J. Staddon, "Graph-Based Authentication of Digital Streams", In *2001 IEEE Symposium on Security and Privacy*, May 2001.

[44] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting", In *Proceedings of the ACM SIGCOMM'97*, Cannes, France, September 1997.

[45] R. Molva, "Internet Security Architecture", *Computer Networks : The International Journal of Computer and Tele communications Networking*, 31(9):787–804, 1999.

[46] R. Molva and A. Pannetrat, "Scalable multicast security in dynamic groups", In *Proceedings of the 6th ACM conference on Computer and Communications Security*, pp. 101–112, Singapore, November 1999, Association for Computing Machinery.

[47] R. Molva and A. Pannetrat, "Scalable Multicast Security with Dynamic Recipient Groups", *ACM Transactions on Information and System Security*, 3(3):3, 2000.

[48] National Institute of Standards and Technology, "Secure Hash Standard", 1995.

[49] National Institute of Standards and Technology, "Advanced Encryption Standard", 2001.

[50] J. Nonnenmacher, *Reliable Multicast Transport to Large Groups*, Ph.D. Thesis, Ecole Polytechnique Fédérale de Lausanne, 1998.

[51] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission", *IEEE/ACM Transactions on Networking*, 6(4):349–361, 1998.

[52] M. Önen and R. Molva, "Group Rekeying with a Customer Perspective", In *Proceedings of the 10th International Conference on Parallel and Distributed Systems (ICPADS'04)*, Newport Beach, California, July 2004.

[53] M. Önen and R. Molva, "How to satisfy both customers and service providers in secure multicast?", Research Report RR-04-102, April 2004.

[54] M. Önen and R. Molva, "Reliable Group Rekeying with a Customer Perspective", In *Proceedings of Globecom 2004*, Dallas, Texas USA, 29 November - 3 December 2004.

[55] A. Pannetrat, *Secure Multicast Communications*, Ph.D. Thesis, Université de Nice Sophia-Antipolis, 2002.

[56] A. Pannetrat and R. Molva, "Efficient Multicast Packet Authentication", In *NDSS 2003*, 2003.

[57] A. Pannetrat and R. Molva, "Multiple layer encryption for multicast groups", In *CMS'2002, Communication and Multimedia Security 2002 Conference, September 26-27, 2002, Portoroz, Slovenia / Also published in the book : Advanced Communications and Multimedia Security /Borka Jerman-Blazic & Tomaz Klobucar, editors, Kluwer Academic Publishers, ISBN 1-4020-7206-6, August 2002 , 320 pp*, Aug 2002.

[58] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization", In *2002 IEEE Symposium on Security and Privacy*, Berkeley, California, may 2002.

[59] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels", In *IEEE Symposium on Security and Privacy*, May 2000.

[60] A. Perrig, D. Song, and D. Tygar, "ELK, a New Protocol for Efficient Large-Group Key Distribution", In *IEEE Symposium on Security and Privacy*, May 2001.

[61] Y. Postel, "Internet Control Message Protocol", RFC 792, september 1981.

[62] B. Quinn and K. Almeroth, "IP Multicast Applications : Challenges and Solutions", RFC 3170, september 2001.

[63] I. S. Reed and G. Solomon, "Polynomial Codes over certain finite fields", *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, 1960.

[64] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, 21(2):120–126, 1978.

[65] R. Rivest, "The MD5 message-digest algorithm", April 1992.

[66] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols", *ACMCCR: Computer Communication Review*, 27, 1997.

[67] P. Rohatgi, "A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication", In *Proceedings of the 6th ACM conference on Computer and Communications Security*, pp. 93–100, Singapore, November 1999, Association for Computing Machinery.

[68] S. Ross, *A First Course in Probability*, Prentice Hall, 1998.

[69] S. Setia, S. Zhu, and S. Jajodia, "A Comparative Performance Analysis of Reliable Group Rekey Transport Protocols for Secure Multicast", In *Performance*, Rome, Italy, September 2002.

[70] K. Siu and R. Jain, "A Brief Overview of ATM: Protocol Layers, LAN Emulation, and Traffic Management", In *Computer Communications Review (ACM SIGCOMM)*, volume 25 of *2*, pp. 6–28, April 1995.

[71] J. Snoeyink, S. Suri, and G. Varghese, "A lower bound for multicast key distribution", In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[72] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean, "Self-Healing Key Distribution with Revocation", In *Proc. IEEE Symposium on Security and Privacy*, pp. 224–240, May 2002.

[73] D. Towsley, J. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols", *IEEE Journal on Selected Areas*, 15:398–406, April 1997.

[74] R. Vetter, "ATM concepts, architectures, and protocols", *Communications of ACM*, 38(2), 1995.

[75] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The Versakey Framework : Versatile Group Key Management", *IEEE Journal on Selected Areas in Communications*, 17(9):1614–1631, September 1999.

[76] D. M. Wallner, E. J. Harder, and R. C. Agee, "Key Management for Multicast: Issues and Architectures", RFC 26 27, june, 1999.

[77] C. Wong and S. Lam, "Keystone: a group key management system", In *Proceedings of International Conference in Telecommunications*, 2000.

[78] C. K. Wong, M. Gouda, and S. S. Lam, "Secure Group Communications Using Key Graphs", In *ACM SIGCOMM 1998*, pp. 68–79, 1998.

[79] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts", *IEEE/ACM Transactions on Networking*, 7(4):502–513, 1999.

[80] M. Yajnick, J. Kurose, and D. Tosley, "Packet Loss Correlation in the Mbone Multicast Network", 96-32., UMCASS CMPSCI, 1996.

[81] Y. R. Yang, X. S. Li, X. B. Zhang, and S. S. Lam, "Reliable group rekeying : A Performance Analysis", In *ACM Sigcomm*, San Diego, CA, August 2001.

[82] X. B. Zhang, S. S. Lam, D. Y. Lee, and Y. R. Yang, "Protocol design for scalable and reliable group rekeying", In *SPIE Conference on Scalability and Traffic Control IP Networks*, August 2001.

[83] X. Zhang, S. Lam, and D.-Y. Lee, "Group rekeying with limited unicast recovery", *The International Journal of Computer and Telecommunications Networking*, 44:855–870, April 2004.

[84] S. Zhu, S. Setia, and S. Jajodia, "Performance Optimizations for Group Key MAnagement Schemes for Secure Multicast", In *The 23rd IEEE International Conference on Distribued Computing (ICDCS)*, May 2003.

[85] G. Zipf, *Human Behavior and the Principle of Least Effort*, Addison-Wesley, Reading, MA, 1949.

Curriculum Vitae

Personal Information

Name	:	Suna Melek ÖNEN
Birth Date	:	October 22nd 1978, Ankara (Turkey)

Education

2001 - present	:	**PhD student in Computer Science** **"Securing multicast applications in satellite networks: A user-centric approach"** Supervisor: Prof. Refik Molva Institut Eurécom - Sophia-Antipolis in collaboration with Alcatel Space Industries, Toulouse
2000 - 2001	:	**MSc (DEA) in "Distributed Systems"** University of Pierre and Marie Curie (Paris VI)
1996 - 2000	:	**BSc in Computer Science** Galatasaray University (Istanbul - Turkey)

Publications

- Papers
  - **Satellite IPSec : An optimized way of securing multicast wireless communications**, *S. Josset, L. Duquerroy, M. Önen, M. Annoni, G. Boiero, N. Salis*, **ISSE** october 2002, Paris.
  - **Denial of Service Prevention in Satellite Networks**, *M. Önen, R. Molva,* In proceedings of the 2004 International Conference on Communications (**IEEE-ICC 2004**), Paris.
  - **Group Rekeying with a Customer Perspective**, *M. Önen, R. Molva.*, In proceedings of the tenth International Conference on Parallel and Distributed Systems (**IEEE-ICPADS 2004**), Newport Beach, California.
  - **Reliable Group Rekeying with a Customer Perspective**, *M. Önen, R. Molva.*, In Proceedings of the **IEEE Globecom'04** Dallas, TX.
  - **How to satisfy both customers and service providers in secure multicast?**, *M. Önen, R. Molva*, Research Report RR-04-102, Intitut Eurécom, 2004.
- Chapter in Books
  - Chapter 10: **"Sécurité des Communications en Multicast"**, *M. Önen, R. Molva , A. Pannetrat*, "Multicast Multimédia sur Internet", HERMES - LAVOISIER, ISBN: 2-7462-1063-0 (02/2005).
- Tutorials:
  - **La sécurité des communications multicast**, Atelier "Cryptographie pour les applications spatiales", July 4th 2002, CNES, Toulouse.
  - **Sécurité des communications en Multicast : Applications pour les liens satellites**, ECOTEL 2002, Logiciels pour Télécommunications, december 2nd-6th 2002, Golfe-Juan.
  - **Multicast Security Project at Eurécom** , The 57th IETF meeting, 16/07/2003, Vienna. (A Scalable key distribution scheme, A Multi-Layer Encryption Scheme, An Efficient Multicast Packet Authentication Scheme)