Sécurité dans les réseaux ad hoc (2)

Marine Minier

INSA de Lyon

PKI

PKI : Public Key Infrastructure

- But : distribution de clé publique avec sécurité et gestion de certificats

- Principe général et fonction :
  - Enregistrement de demandes et de vérifications des critères pour l'attribution d'un certificat
    - Id du demandeur vérifier
    - Possession de la clé secrète associée
  - Création des certificats
  - Diffusion des certificats avec publication des clés publiques

PKI : Public Key Infrastructure

- Archivage des certificats pour suivi de sécurité et de pérennité
- Renouvellement des certificats
- Suspension de certificats (pas de standard, peu aisé à mettre en œuvre,surtout administrative)
- Révocation de certificat sur date, perte, vol ou compromission des clés
- Création et gestion des listes de révocation des certificats
- Délégation de pouvoir à d’autres entités reconnues de confiance

Principaux problèmes

- Suspension de certificats : pas de standard
- Création et publication des listes de révocation des certificats
  - Pas de standard pour révocation automatique
  - Moyens administratifs : implémentés de façon sécurisée
    - Le propriétaire de la clé doit prouver que sa clé est inutilisable

Problème de gestion !

- Listes de révocations doivent
  - Être protégées pour ne pas être corrompues
  - Être accessibles en permanence et à jour
  - => synchronisation des horloges de toutes les pers. concernées
- Listes de révocations peuvent être très grande
  - Exemple : paiement des impôts par Internet
  - Maintenues en permanence
  - Enorme base de données, accessible à tout instant !

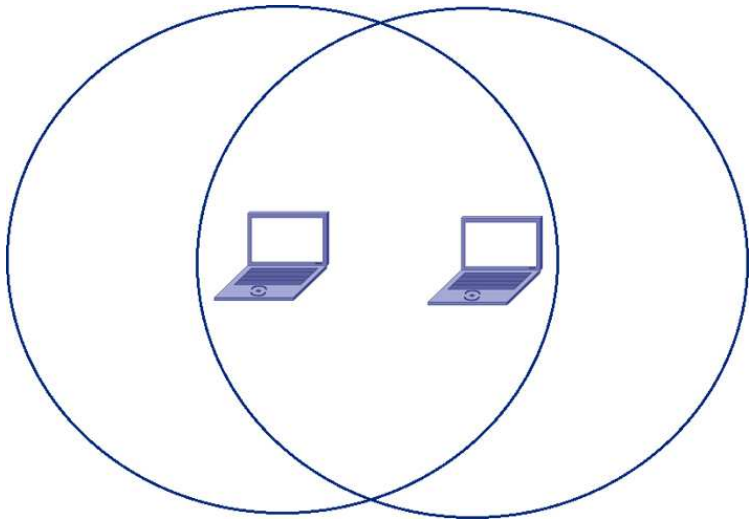
À Titre de comparaison

- Comme une carte d'identité national
  - Preuve de l'identité quand création de la carte
  - Unique, liée à une identité et non falsifiable
  - Déclaration en préfecture quand vol ou perte afin d'en obtenir une autre et pour ne pas qu'il y est une mauvaise utilisation de la disparue

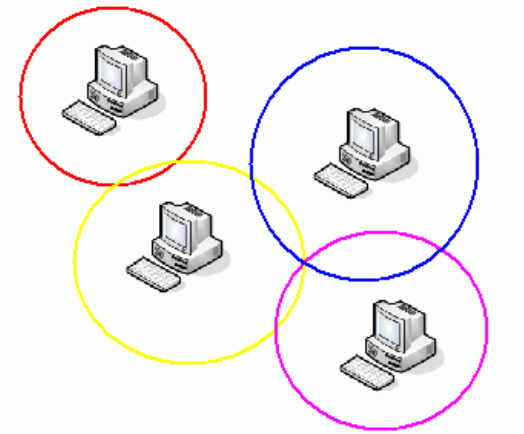
Conclusion PKI

- Important :
  - Étude de faisabilité préalable pour estimer
    - Besoins (matériels) selon le nombre de personnes concernées
    - La validation par des organismes de confiance
    - Le déploiement
- Un exemple : les impôts http://www.ir.dgi.minefi.gouv.fr/
- Plus d'informations : http://www.ssi.gouv.fr/

Les réseaux ad hoc

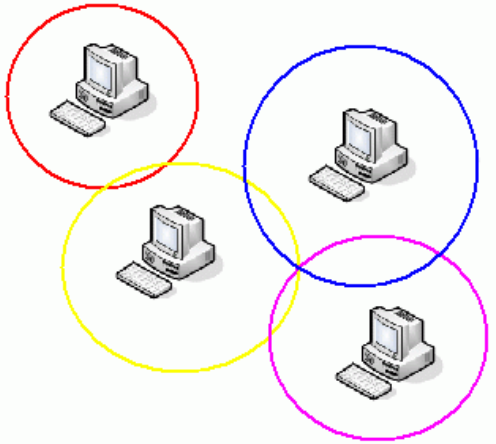


Plan



- Contexte, Introduction
- Accès au médium radio
  - 802.11, WiFi, Bluetooth,…
- Routage
  - Technique de routage dans les réseaux ad hoc

Introduction



- Histoire :
  - Année 90 : apparition des systèmes cellulaires numériques : GSM, DCS,…
  - Fin des années 90 :
    - Réseaux locaux sans fil (WLANs)
    - Réseaux personnels sans fil (WPANs) : Bluetooth, IEE 802.15
- Futur : convergence des systèmes radiomobiles : cellulaire, locaux, satellitaires
- Ubiquité : n'importe qui, n'importe où, n'importe quand…

Réseaux ad hoc

- Composés de nœuds sans fil
  - **autonomes**,
  - **mobiles** qui sont routeurs, jouent le double rôle client/serveur et **actives**
  - utilisant des interfaces radio de courte portée (802.11b, 802.11g)
  - avec peu de ressources
  - Sans infrastructure fixe, sans contrôle central

  => IETF : MANET (Mobile Ad-hoc NETwork)

Réseaux ad hoc (suite)

- A l'opposé du monde filaire
  - Utilisation des ondes radio (entre 9Khz et 300 Ghz) ou électromagnétiques
  - Ondes lumineuses : laser, infrarouge,…
  - Avantages : pas de câbles, facilement extensible, mobile,…
  - Inconvénients : chère, ressource rare, peu sûr,…

Caractéristiques des ondes radio

- Propagation dans l'environnement
- Atténuation de la puissance, fonction de la distance (Réflexion, Réfraction,…)
- Trajets multichemins

- Pbs : Bruits et interférences
  - Collisions
  - Bruit ambiant

Conséquences

- Pertes de données
- Débits plus faible que filaire
- Faible sécurité

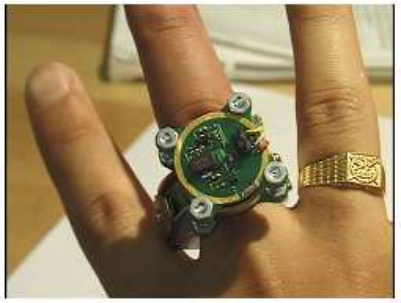
- Les nœuds eux-mêmes : souvent peu puissant, pb de consommation d'énergie (capteurs !)

Réseau ad hoc = multisauts !

- Portée limitée => multisauts
- Mobilité des stations + instabilité des ondes => topologie dynamique
- Arrivées et départs de nœuds => auto-organisation sans infrastructure fixe

Exemples d’équipements

- Capteurs



- Téléphones portables

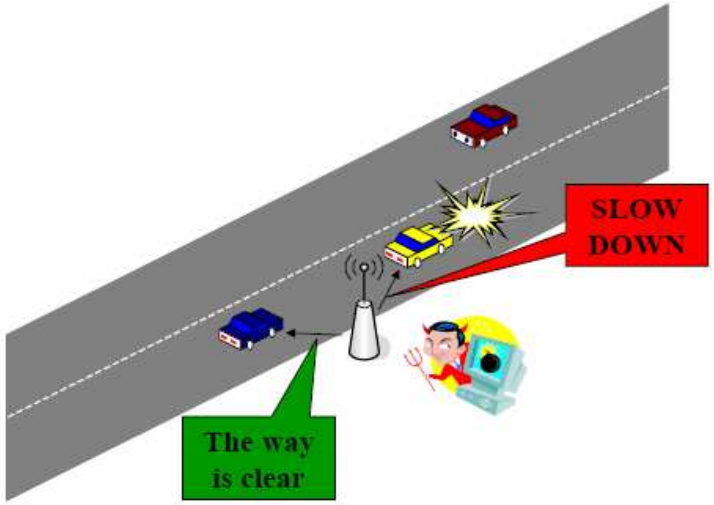
- PDA

- Laptop

puissance

Exemples d’utilisations

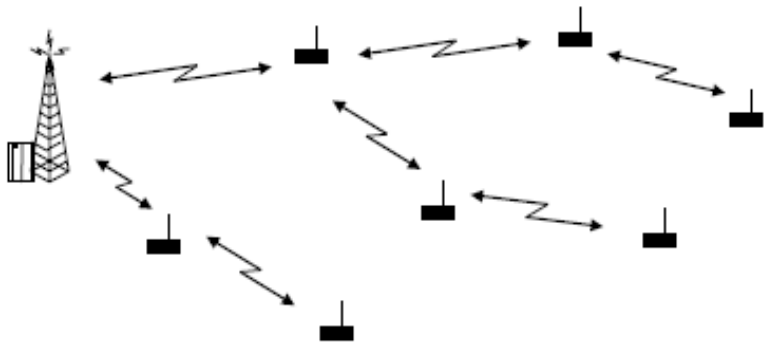
- Militaire…
- Catastrophe naturelle :
  - Coordination des secours,..
- Vehicular network



- Conférence, réunion : partage données,..

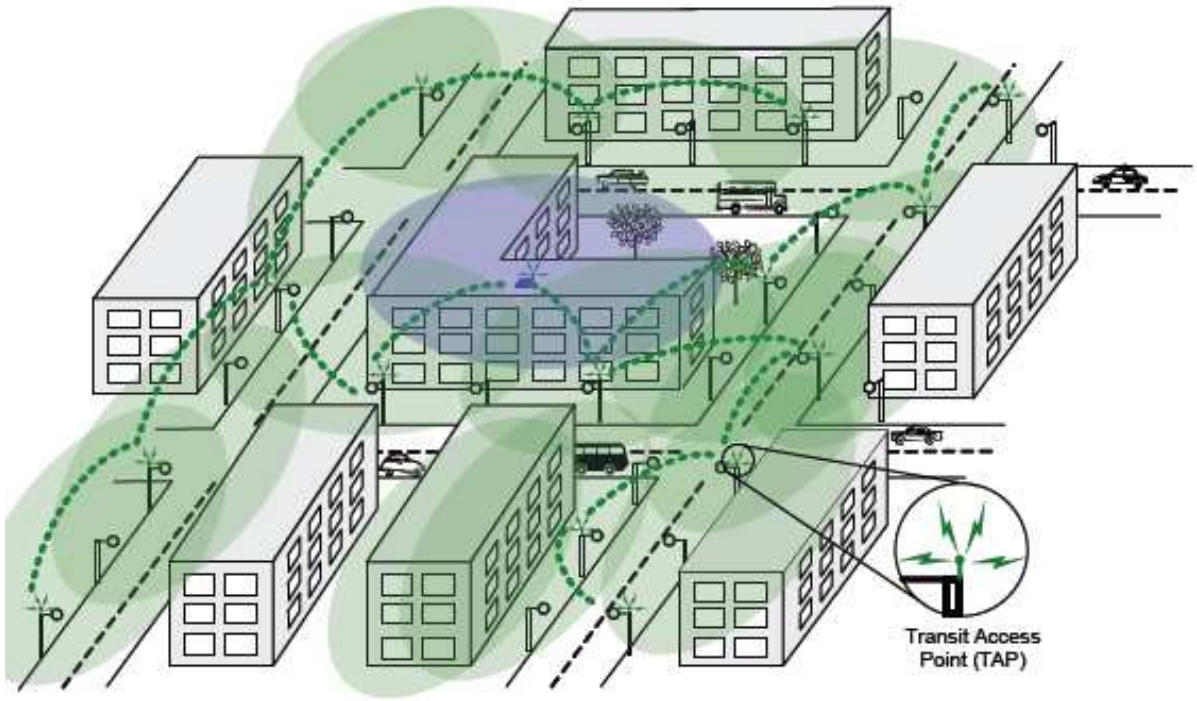
Exemple d’utilisations : capteurs

- Réseau de capteurs : spécificités
  - Applications : surveillance, reconnaissance, acquisition de données,…
  - Réseau dense, déploiement aléatoire, communcations spécifiques, énergie limitée



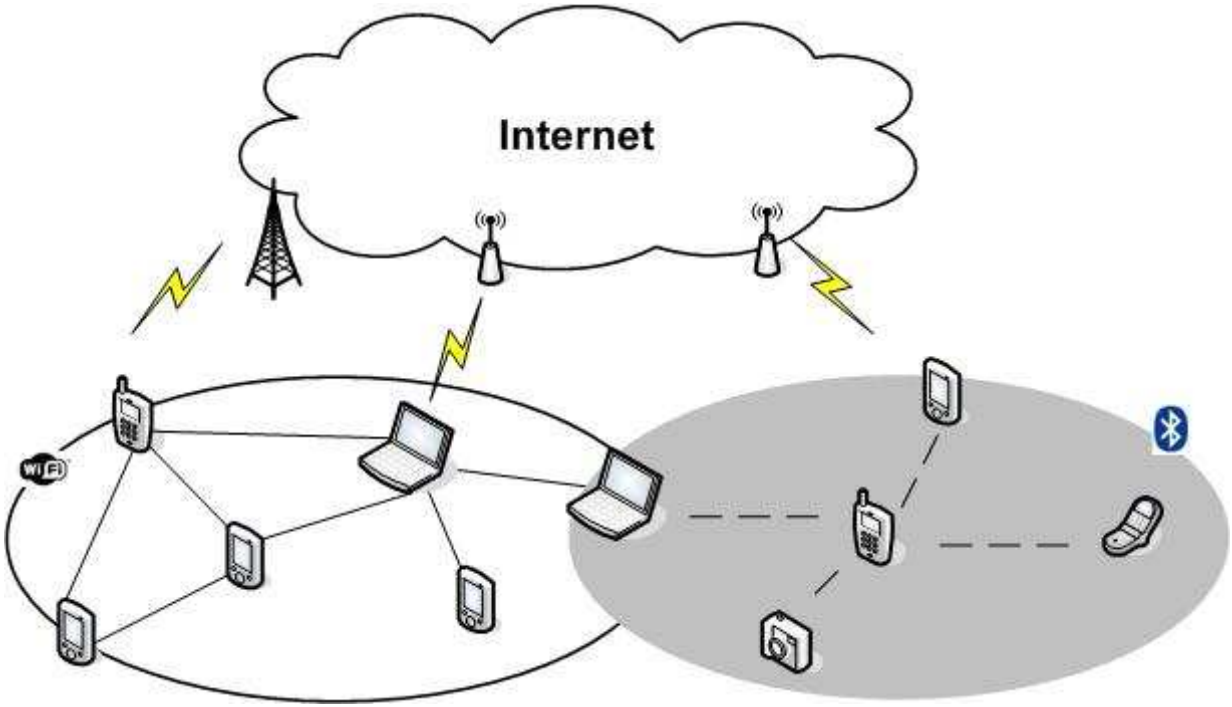
Nouvelles applications (1/2)

- Mesh networks :réseaux ad hoc statiques



Nouvelles applications (2/2)

- Réseaux hybrides :



Communications dans les réseaux ad hoc

Deux aspects (1/2)

- Accès au médium radio :
  - Contrôle via 802.11 et la couche MAC
  - http://perso.ens-lyon.fr/isabelle.guerin-lassous/Enseignement/MAC.pdf
  - DCFWMAC (DFC – 802.11b)
  - Réseau de capteurs : S-MAC (gestion sommeil)

  - => garantir une qualité de service

Deux aspects (2/2)

- Routage
  - Gestion des communications multi-sauts
- Spécificités des réseaux ad hoc :
  - Mobilité des stations (perte de liens,…)
  - Critère différents : consommation d'énergie, stabilité des routes,…
- => défi du groupe MANET

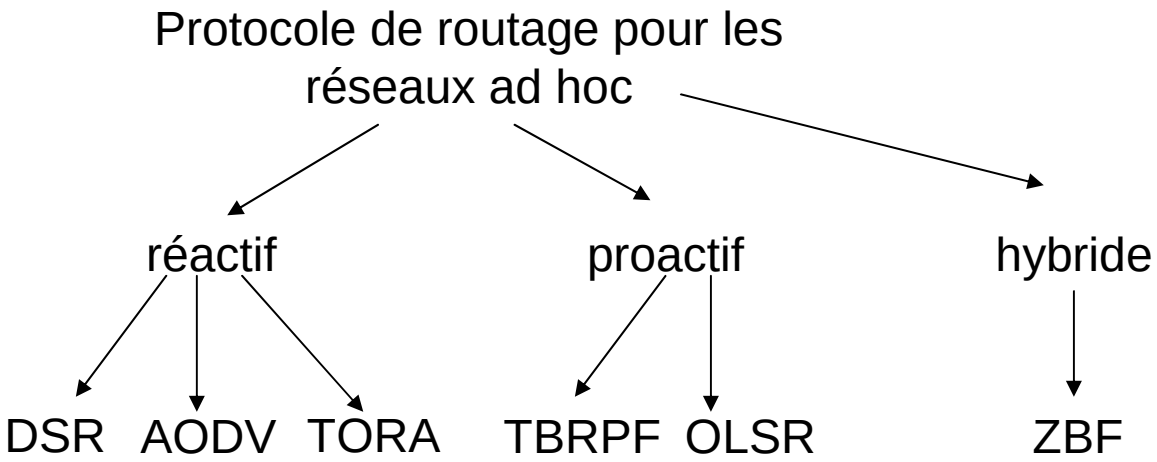
Buts du routage

- Surcoût de contrôle minimal (réduire l'utilisation de la bande passante, des batteries,…)
- Surcoût de traitement minimal
- Maintenance dynamique de la topologie
- « sans boucle »

Protocoles de routage en ad hoc

- Classification :
  - Réactif : construction de la route à la demande
  - Proactif : routes maintenues périodiquement
  - Hybride : proactif en local et réactif en extérieur
  - Hiérarchique : fondée sur une structure spécifique avec des entités ayant des rôles particuliers
  - Géographique : fondé sur des informations de positionnement

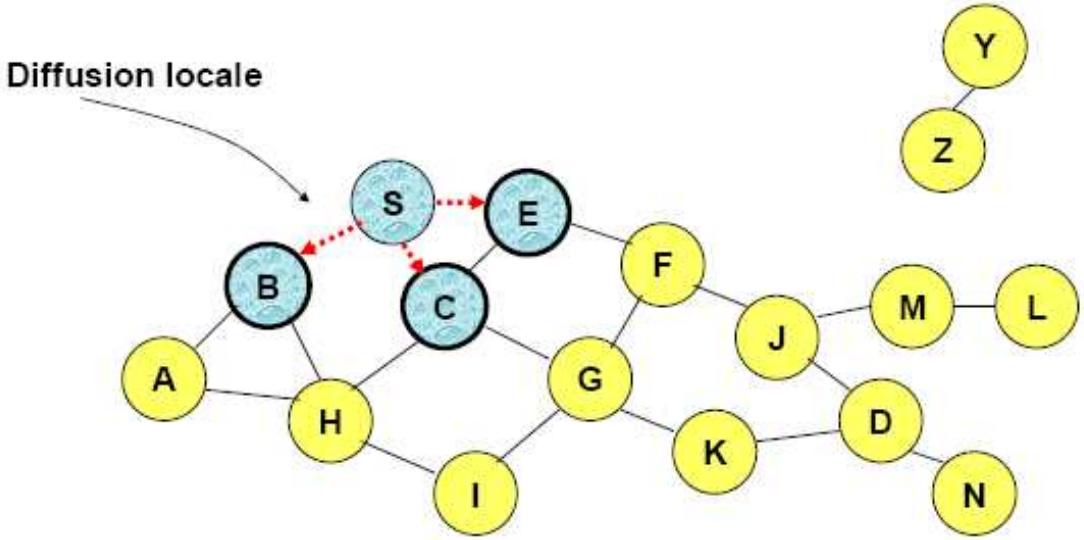
Exemples


Protocole de routage pour les réseaux ad hoc
réactif
proactif
hybride
DSR
AODV
TORA
TBRPF
OLSR
ZBF

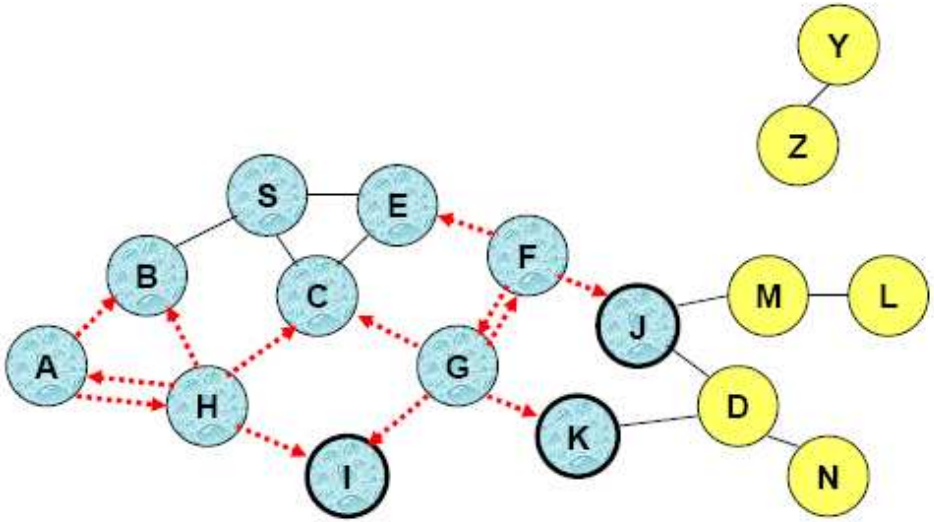
Réactif : fondée sur l'inondation

- Simple : la source diffuse le paquet à tous ses voisins.
- Chaque mobile retransmet ce paquet si pas déjà fait
- Utilisation de numéro de séquence pour éviter les boucles

Principes de l'inondation (1/4)

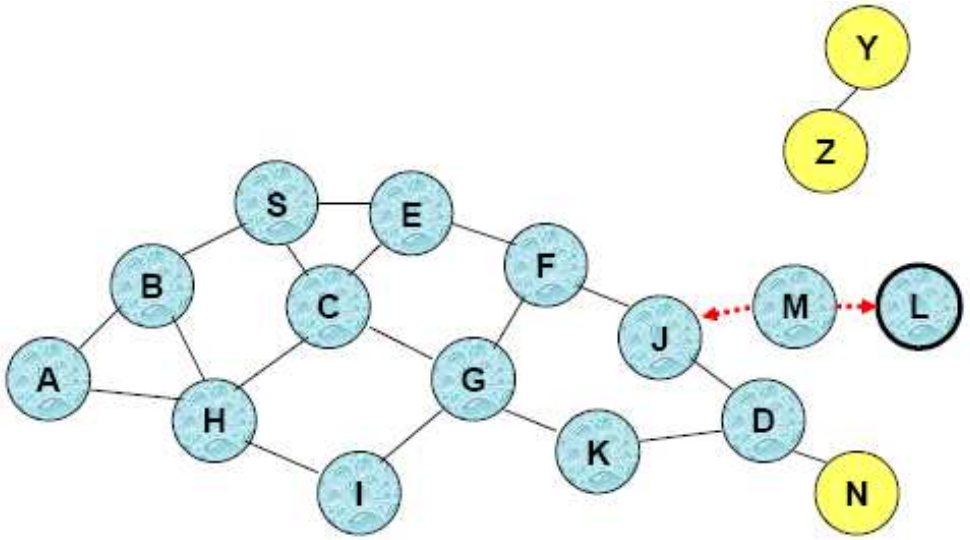


Principes de l'inondation (2/4)



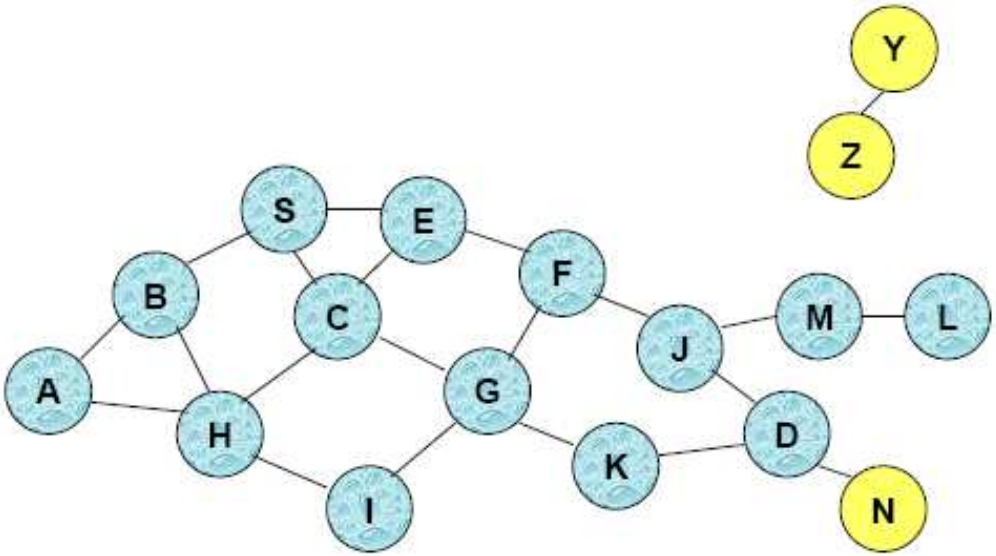
• C reçoit des paquets de G et de H, mais ne les retransmet pas car il l'a déjà fait

Principes de l'inondation (3/4)



- D ne retransmet pas mais l'inondation continue

Principes de l'inondation (4/4)



• Dans le pire cas, tous les mobiles reçoivent le paquet et donc le retransmettent (sauf D)

Conclusion inondation

- Avantages :
  - Simple
  - Efficace si peu de transmission de données et topoligie changeante
  - Robuste (plusieurs chemins)
- Désavantages :
  - Surcoût peu être important
  - Diffusion locale rarement fiable

Protocoles réactifs

- Fondés sur l'inondation d'un paquet de contrôle pour construire une route
- Les routes sont construites quand on en a besoin
- Pas de table de routage

- Construction
  - Si pas de route disponible => envoi d'un paquet de recherche
  - Si route dispo ou destination => envoi paquet de réponse

DSR : Dynamic Source Routing

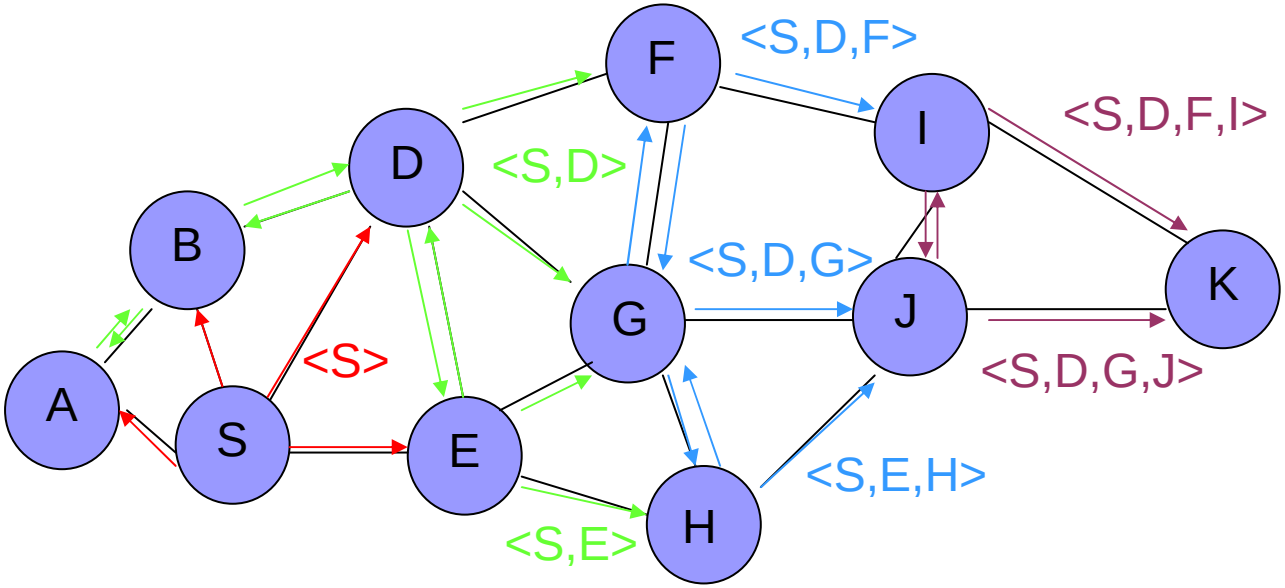
- Protocole réactif (à la demande)
  - Fondé sur le principe du routage à la source
    - Les noeuds n'ont pas besoin de tables de routage
    - Maintien juste d'un cache dynamique de routes (autorisant plusieurs entrées pour une même destination)
  - Permet à chaque émetteur de sélectionner et de contrôler les routes utilisés
    - Plus de 200 nœuds
    - Forte mobilité des noeuds

DSR (2/6)

- Deux opérations :
  - Découvertes de routes
    - Routerequest (RREQ) et Routereply (RREP)
  - Maintenance de routes
    - Routeerror (RERR)

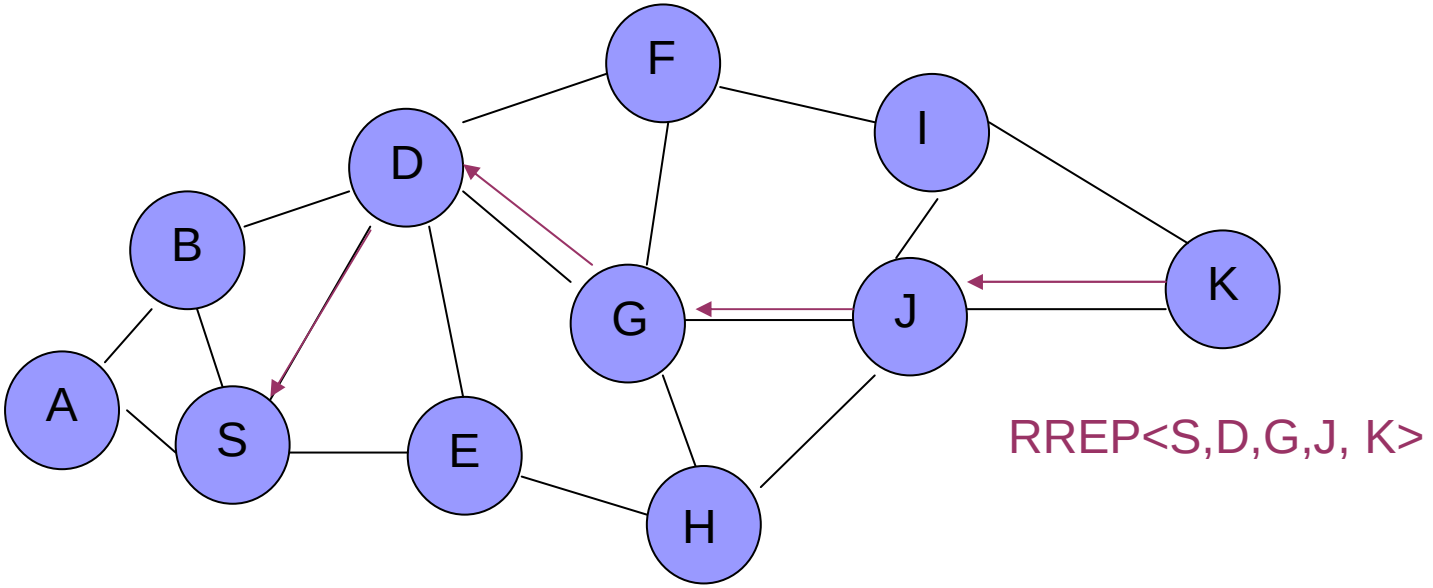
DSR (3/6)

- Le nœud source S veut parler au noeud destination K : RREQ



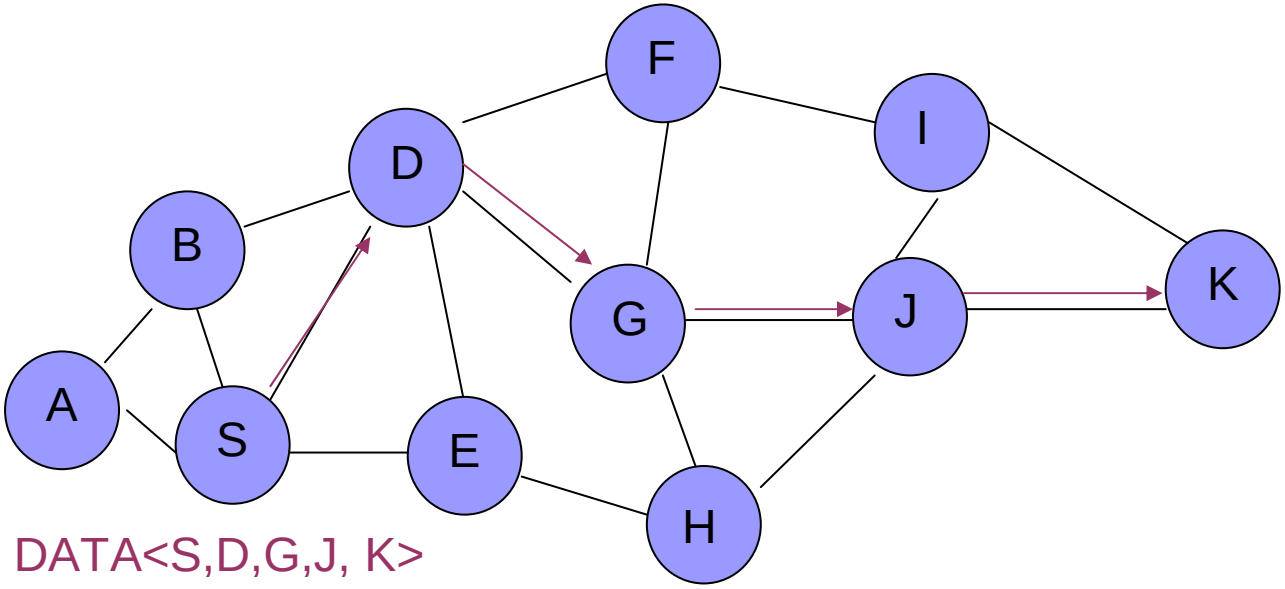
DSR (4/6)

- Sélection de la meilleure route (lien bidirectionnel)
- Envoi d'un RREP de K à S par chemin inverse



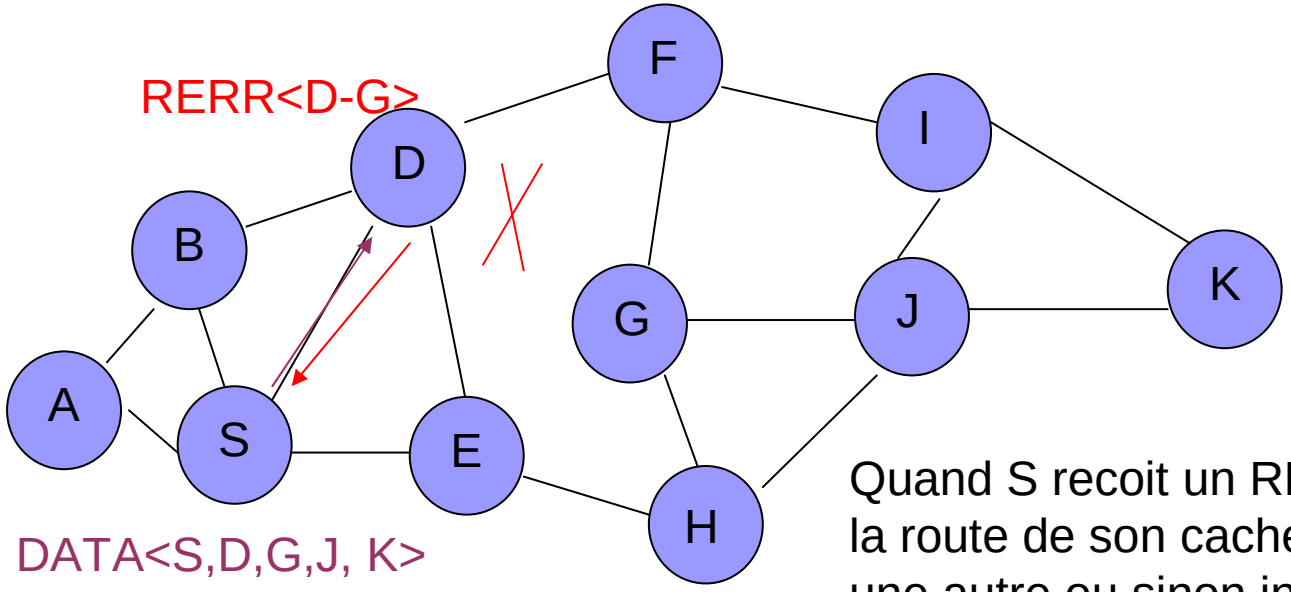
DSR (5/6)

- Data Delivery :



DSR (6/6)

- Route Error :



Quand S recoit un RERR, il retire la route de son cache, en essaye une autre ou sinon initialise une nouvelle ROUTE DISCOVERY.

Conclusion DSR (1/2)

- Routage par source
  - Chemin donné dans paquets de contrôle et données
  - Taille entête fonction longueur route
- Liens unidirectionnels
  - Recherche de route Dest./Source + ajout route construite
- Cache de routes
  - Multi-chemins possible
  - Initialement, pas de durée de vie
  - Utilisation d'info. diffusées

Conclusion DSR (2/2)

- Performances
  - Pas de maintenance inutile
  - Caches limitent recherche de routes
  - Routes possibles nombreuses
  - Pb : coût de l'entête, de l'inondation, délai d'envoi du premier paquet
- Autres exemples traités après : AODV

Protocole proactif

- Les routes sont maintenues en permanence
  - Construction périodique d'une table de routage
  - Chaque nœud, a tout instant, maintient une route dans sa table vers tout nœud du réseau

Exemple : OLSR

- Optimized Link State Routing
- Protocole proposé par INRIA/ Projet Hipercom

- But : limiter le trafic de contrôle par
  - Détection de voisins
  - Relais multipoints
  - Pour limiter l'inondation et la sous-structure utilisée

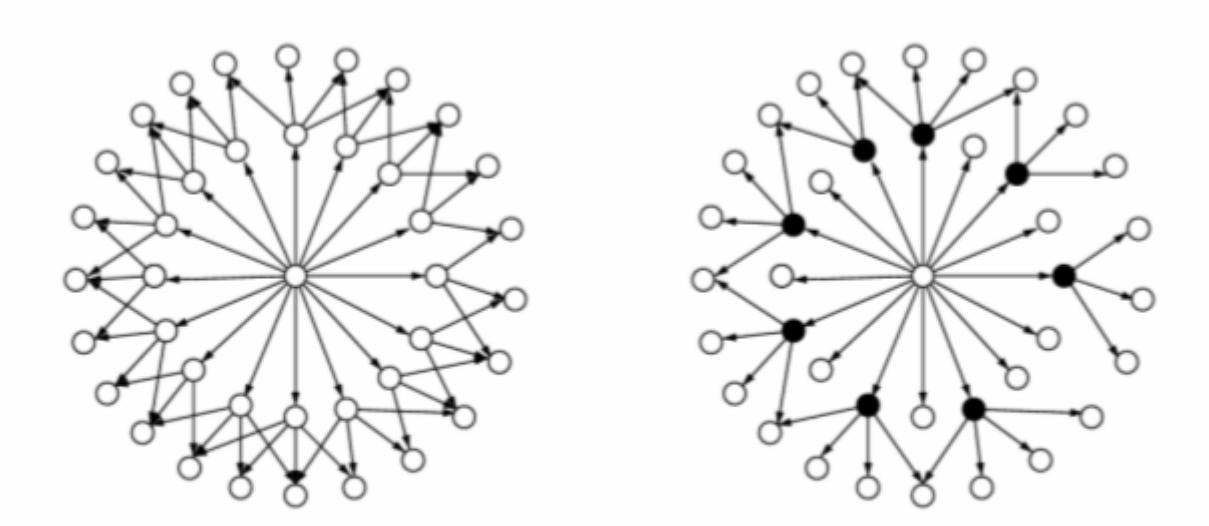
Paquets Hello et TC

- Paquets « Hello » (à 1 saut)
  - Découverte du voisinage et maintien de cette relation
  - Chaque équipement émet un « Hello » périodique (2 secondes)
  - Un broadcast détermine qui sont les voisins et leur lien (asymétrique, symétrique)
- Topology Control (TC) : relayé
  - Mise à jour de la topologie

Relais Multi-point (1/3)

- Permet de réduire le nombre de retransmissions dans une inondation
- Restreindre la topologie diffusée dans le réseau
- Condition pour devenir MPR : pouvoir atteindre tous les noeuds à une distance de 2 sauts, avec un lien symétrique MPR communiqués à tout le réseau par des messages TC (Topology Control) périodiques
  - À la réception des TC, mise à jour des tables de routage

Relais Multi-point (2/3)



Inondation pure

Inondation par MPRs

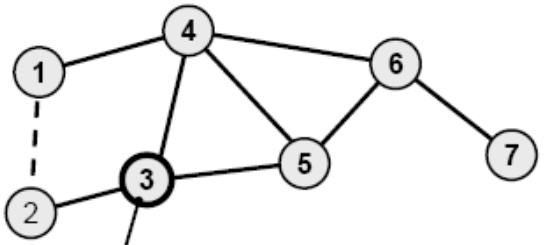
Copyright Daniele raffot

Relais Multi-point (3/3)

- Minimiser le nb de MPR = pb NP complet
- Heuristique
  - Voisins indispensables
  - Voisins avec le plus de voisins à deux sauts
- Ens des sélecteurs MPR : ens. des voisins qui m'ont choisi comme MPR
- Diffusion de cette info : par les MPR et via les MPR

Maintenance

- Construction d'une table de routage sur chaque nœud
  - Connaissance partielle du réseau
  - Algo du plus court chemin (Dijkstra)



Dest	Next	Hops
1	4	2
2	2	1
4	4	1
5	5	1
6	4 (5)	2
7	4 (5)	3

OLSR : Performances

- Les routes sont immédiatement disponibles
- Peu de surcoût de contrôle
- Seuls les MPRs vont transmettre les messages topologiques (découverte du réseau)
- Adapté aux réseaux avec faible mobilité

Conclusion partielle

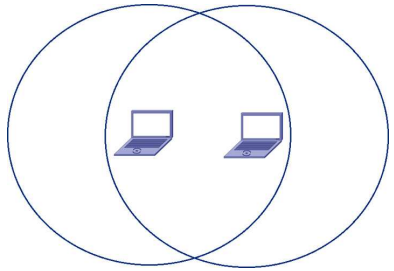
- Etat des RFCs en 2004
  - Réactif : AODV, DSR
  - Proactif : OLSR, TBRPF

Qui dit nouveau réseau dit…
nouvelles attaques !

Les nouvelles causes…

- L'infrastructure des réseaux ad hoc :
  - => pas d'infrstructure, chaque nœud est routeur !
- Topologie dynamique
  - => les nœuds bougent et peuvent plus facilement dupliquer leur identité ou leur adresse
- Communications non filaires :
  - Peut de moyens de défense contre le bruit ou les interférences…
- La relation liant des voisins est souvent par défaut de confiance
  - Suppose que tous les nœuds soient honnêtes…
- => Bref : beaucoup de nouveaux risques !

Les nouvelles attaques !



- Attaques passives :
  - Écoute passive : analyse de trafic, récupération de l'information
  - Menace contre la confidentialité et l'anonymat

- Attaques actives : cas dédiés
  - Usurpation : Sybille attack, réplication, impersonification,…
  - Modification des données : destruction du message, retardement de la transmission,…
  - Déni de service : tentative de debordement des tables de routages des noeuds servant de relais, tentative de gaspillage de l'energie de nœuds (sleep deprivation torture attack),...

Attaques actives : plusieurs attaques possibles

- *Impersonification* : modification de l'identité de l'émetteur ou du récepteur
- *Altération des données* (modification du contenu)
- *Destruction du message*
- *Retardement* de la transmission
- *Répudiation* du message = l'émetteur nie avoir envoyé le message

- Cryptographie : permet de lutter contre toutes ces attaques
  - Garantie la confidentialité, l'intégrité, l'authenticité (authentification et identification) et la signature

Menaces : spécificité des réseaux ad hoc

- Problème essentiel :
  - Déni de Service
  - Ecoute : situation ou l'attaquant se contentera d'écouter le trafic sans le modifier
  - Détournement de trafic (le plus fort)
    - L'attaquant force une communication entre deux noeuds à transiter par lui.
    - Il peut ensuite en faire ce qu'il veut (la bloquer, l'altérer et la faire suivre, ...)
    - Le routage est distribué sur tous les nœuds donc le détournement de trafic est simple

Comparaison filaire/sans fil (1/3)

- Déni de Service : environnement radio
  - Brouillage du canal radio pour empêcher toute communication.
  - Tentative de debordement des tables de routages des noeuds servant de relais.
  - Non-cooperation d'un noeud au bon fonctionnement du reseau dans le but de préserver son energie par exemple.
  - Tentative de gaspillage de l'energie de noeuds ayant une autonomie de batterie faible.
    - L'attaque consiste a faire en sorte que le noeud soit obligé de rester en etat d'activite et ainsi de lui faire consommertoute son energie (sleep deprivation torture attack).

Comparaison filaire/sans fil (2/3)

- Ecoute
  - Filaire :
    - Ecoute simple en environnement Hub (plus complexe avec des switchs)
    - Nécessité d'accès physique au réseau (à mitiger avec les réseaux WIFI)
  - Ad hoc :
    - Réseau sans fil donc écoute simple
    - Mais écoute limitée au réseau environnant
  - Conclusion :
    - Situation similaire entre filaire et ad hoc
    - => menace sur la confidentialité et l'anonymat

Comparaison filaire/sans fil (3/3)

- Détournement de trafic
  - Filaire :
    - Nécessité d'une position de routeur
    - Attaque dicile à mettre en œuvre (possible cependant dans certains cas)
  - Ad hoc :
    - Le noeud est déjà en position de routeur
    - Il ne reste plus qu'à s'assurer que les échanges à intercepter passe bien par nous
  - Conclusion :
    - La principale difficulté du filaire (être routeur) disparaît en ad hoc.

Comment être routeur dans un réseau ad hoc ?

- Sans tricher :
  - On se trouve sur le bon chemin de routage, on est dé jà routeur
  - Position géographique : on se déplace pour être sur le bon chemin de routage

Comment être routeur dans un réseau ad hoc ?

- En trichant :
  - Attaque du Siphon : attirer un maximum de traffic
    - mentir sur le nombre et l'identité de ses voisins => création d'une sorte de trou noir au milieu du réseau
  - Attaque du trou de ver :
    - L'attaquant construit un tunnel entre lui et son complice => fait croire aux victimes que le meilleur chemin passe par là
  - L'usurpation de l'identite d'un noeud en leurrant les mecanismes de contrôle d'accès => attaques actives
    - Attaque black Hole : lors du mécanise de découverte de route, il annonce un chemin de côut minimal par « route reply ». Le nœud abusé met à jour sa table de routage avec la fausse route
    - => les paquets sont captés et absorbés par le nœud malicieux.

Solution à ces menaces : la cryptographie !

- La Cryptographie garantie :
  - la confidentialité : assurer que les données ne sont dévoilées qu'aux personnes autorisées
  - l'intégrité : assurer que les données ne sont pas altérées
  - l'authenticité :
    - Authentification : prouver l'origine d'une donnée
    - Identification : prouver qu'une personne est qui elle prétend être
  - La signature : rend impossible le fait de renier un document.

Donc

- Réseaux ad hoc : plusieurs problèmes
  - Sécurité des données
  - Identification, authentification
  - Sécurité du routage
- Deux approches :
  - Modèles de gestion de clés, de gestion des identités et de sécurité
  - Sécurisation des protocoles de routage
- Besoin de cryptographie dédiée
  - Car les modèles centralisés ne fonctionnent plus

Sécurité dans les réseaux ad hoc

Introduction

- Réseaux ad hoc : deux problèmes
  - Sécurité du routage
  - Sécurité des données
- Quatre approches :
  - Modèles de gestion de clés et de sécurité
  - Sécurisation des protocoles de routage
  - Systèmes de détection d'intrusion
  - Modèles de confiance et de réputation

Cryptographie dédiée :

- Modèles précédents ne fonctionnent pas :
  - Certification centralisée
  - PKI
  - Vient de la gestion centralisée…
- Modèle de gestion de clés :
  - Cryptographie à seuil
  - Certification distribuée
  - SPKI (présentation étudiant)

Cryptographie à seuil

- Principe :
  - étant donné N utilisateurs et un seuil k
  - On cherche à diviser un secret s en N morceaux $s_1,\ldots,s_N$ tel que :
    - Tout groupe de taille au moins k puisse conjointement reconstruire le secret : la connaissance de n'importe quels k morceaux de s permet de retrouver le secret
    - Tout groupe de taille au plus k-1ne peut reconstruire le secret. la connaissance de n'importe quels k-1 morceaux (ou moins) de s laisse le secret complétement inconnu
  - Typiquement : k = ½ N ou 1/3 N (dépend des besoins)

Exemple : le Schéma de Shamir (1979)

- Soit s dans F_q le secret à partager par N utilisateurs
- Soit $x_1,\ldots,x_N$ N valeurs non nulles de F_q
- On choisit aléatoirement k-1 éléments $a_1,\ldots,a_{k-1}$ dans F_q et on construit le polynôme :

$$y=f(x) = s + \sum_{i=1}^{k-1} a_i x^i \mod q \qquad (f(0)=s)$$

- Chaque personne recoit :

$$s_i = (x_i,y_i) \mod q$$

Reconstruire le secret

- Reconstruire le secret s à partir d'un sous ensemble $\{s_1,s_2,\ldots,s_k\}$ est équivalent à résoudre un système d'équations linéaires. (cela fonctionne pour tout sous ensemble de k éléments au moins) :

$$y_i = f(x_i) = s+a_1x_i^1 +\ldots+ a_{k-1}x_i^{k-1}, \forall\ i$$

- Résolution : formule d'interpolation de Lagrange :

$$l_m(x) = \prod_{j=1,\, j\neq m}^{k} \frac{x-i_j}{i_m-i_j} \pmod q$$

$$S = \sum_{i=1}^{k} s_i l_i(0) \pmod q$$

Comment utiliser cela ?

- Plusieurs applications possibles :
  - Signature à seuil
  - Certification distribuée

Signature à seuil

- Problème: N participants authentifie un message en générant une seule signature chaque sous groupe de taille au moins k peut recréer une signature mais pas k-1 participants.

- Solution: signature à seuil : personne ne connait la clé secrète seulement son secret personnel.

- Avantages :
  - Signature décentralisée avec un “donneur de confiance” au départ
  - Très intéractif dans la distribution du secret et dans la vérification
  - La taille de la signature augmente seulement linéairement en fonction du nombre de participants.

Exemple : signature RSA

- Utilisé par le projet COCA (université de Cornell et IBM Zurich (projet SINTRA)

- Quand a-t-on besoin de ces signatures :
  - **Cas Optimal :** n'importe qui peut vérifier la validité de la signature en utilisant la clé publique => toutes les propriétés cryptographiques sont garanties.
  - **Mauvais cas :** cette signature ne permet pas de vérifier :
    - Quel secret est incorrect => besoin de plus de preuves d'exactitude et de vérification de partage de secret
    - On peut quand même créer des signatures à partir des bons secrets partagés.

- On peut utiliser ces méthodes de façon proactive en calculant de nouveaux secrets à partir des anciens sans compromettre le service et la clé privée originelle

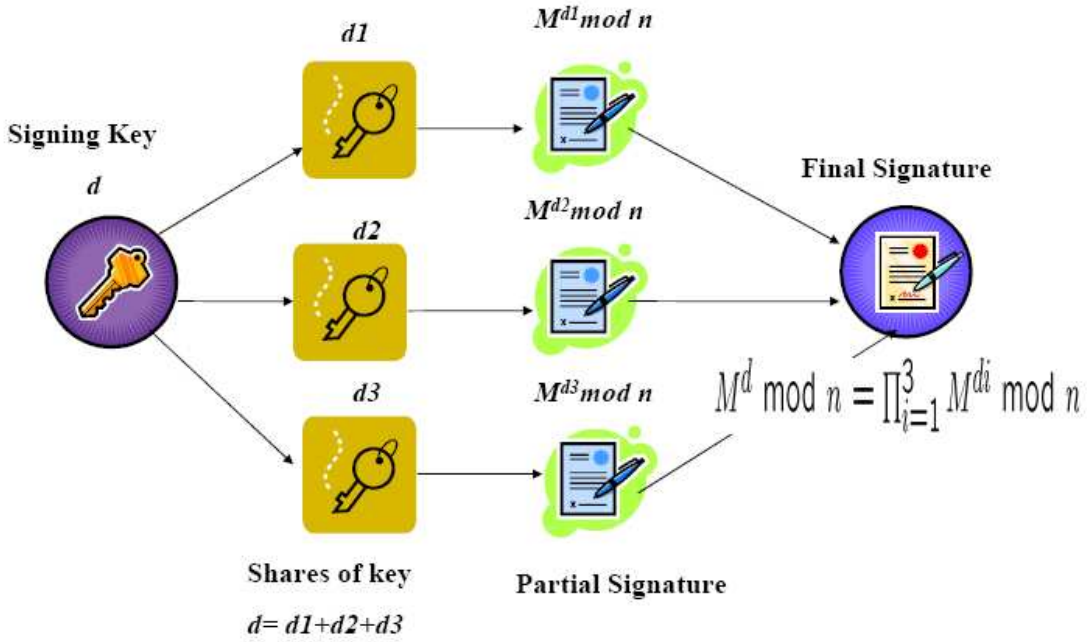
RSA à Seuil : génération

- Un gentil A “dealer de confiance” génère les valeurs publiques de RSA (n,e) et la clé privée correspondante (d)
- Partage ensuite la clé privée en N secrets sachant que k parmi N sont suffisant pour retrouver le secret.

- Pour cela : choisit aléatoirement un polynôme de degré k-1 (cf Shamir)
- Il calcule les secrets individuels s_i
- Il crée également la preuve de vérification de signature en créant les bonnes exponentiations modulaires

- Plus cher que RSA normal
- Attention aux nombres premiers choisis !

Méthode de base de partage d'une signature



- Ne permet pas de savoir qui triche si il y a triche => pas de vérification de partage du secret

Plus loin avec RSA à seuil...

- Chaque entité reçoit son secret partagé
  - Calcule sa signature personnelle et une preuve d'exactitude de cette signature à l'aide de l'algorithme de vérification
  - Envoie sa signature et sa preuve au combineur

- Le combineur :
  - Collecte les signatures individuelles
  - Vérifie grâce à la preuve d'exactitude qu'elles ont bien été généré à partir des secrets
  - Génère la signature de groupe

Proposition avec vérification de RSA à seuil (1/2)

- $\Delta = l!$ (l = nb secrets partagés)
- Les secrets d_i / $d_i = f(i)$ polynome de degre k et $f(0)=d$

- Coefficients de Lagrange dans ce cas :

$$\lambda_{i,j}^{S} = \Delta \frac{\prod_{j' \in S \setminus \{j\}} (i - j')}{\prod_{j' \in S \setminus \{j\}} (j - j')} \in Z$$

$$\Delta d = \sum_{i \in S} \lambda_{0,i}^{S} d_i$$

- Chacun envoie au combiner : $s_i = x^{d_i}$ avec $x = hash(m)$

Proposition avec vérification de RSA à seuil (2/2)

- Le combiner calcule alors :

$$s=\prod_{i \in \mathcal{S}} s_i^{\lambda_{0,i}^{S} d_i} = s^{\sum_{i \in S} \lambda_{0,i}^{S} d_i} = s^{\Delta d} \bmod N$$

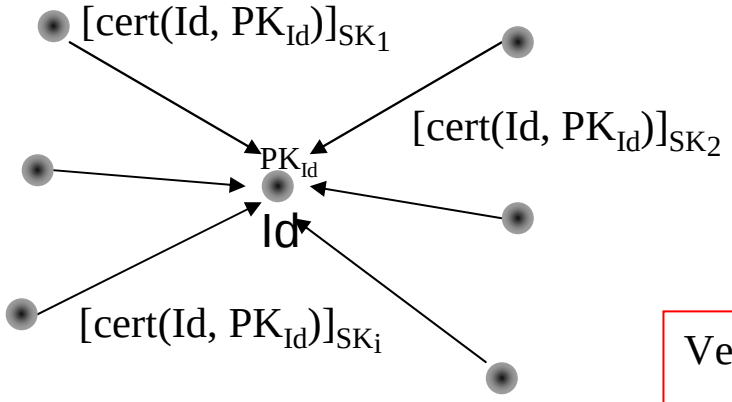
- Le combiner est dans ce cas capable de retrouver $s^d \bmod N$ par le calcul d'une racine n-ième
- Est capable de vérifier tous les bouts de signatures à l'aide de la preuve fournie en plus par chaque noeud
- Génère une signature de groupe exacte

Vérification :

- Tout le monde peut vérifier la validité de la signature à l'aide de la clé publique
- Pas plus de détails sur vérification => utilisation de l'interpolation de Lagrange modulaire.
- Coût vérification : comme RSA.

Certification « à seuil »

- Fondée sur la cryptographie à seuil
- Preuve d'une clé publique et d'une identité cert(Id, PK_{Id}) par k autres nœuds



CERT(Id,PK_{Id})$_{SK}$ {
[cert(Id, PK_{Id})]$_{SK1}$
[cert(Id, PK_{Id})]$_{SK2}$
…
[cert(Id, PK_{Id})]$_{Ski}$
…
}

Verification de CERT(Id, PK_{Id})$_{SK}$ par tous les noeuds Connaissant PK

Certifications à seuil

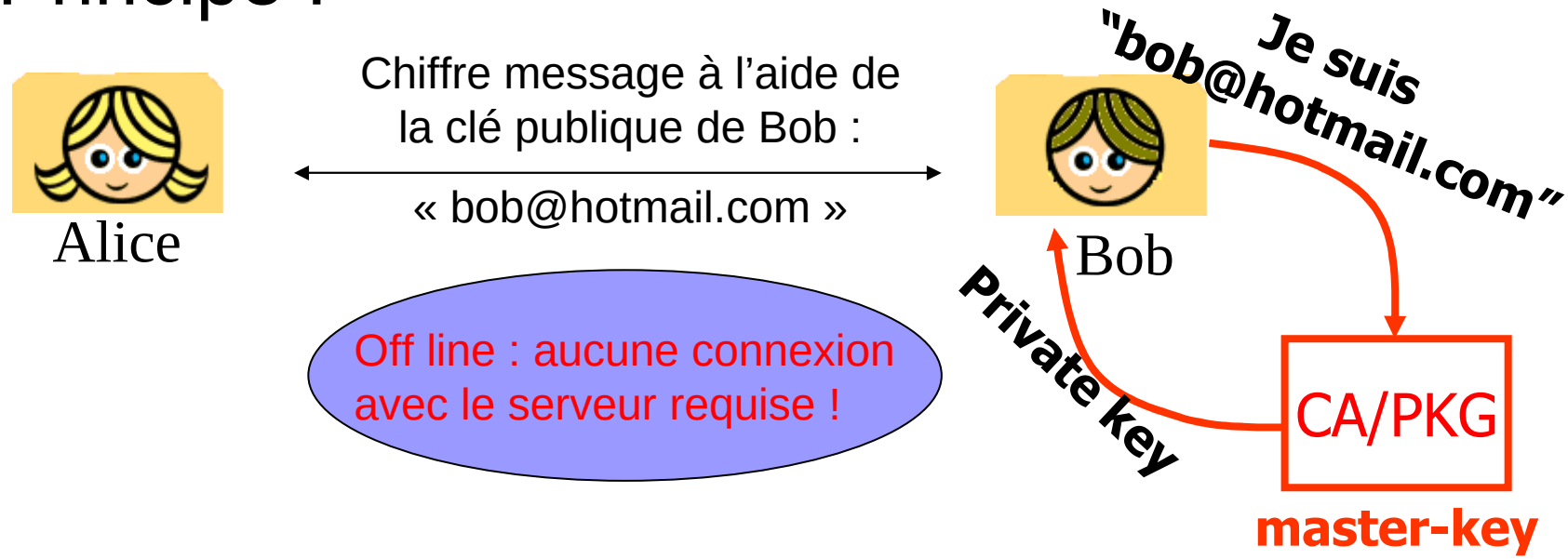
- Avantages :
  - Distribuées et auto-organisées
- Inconvénients :
  - La phase de distribution du secret doit se faire pendant une phase « fermée »
  - Problème de densité du réseau
  - N’empèche pas l’attaque de Sybille : création de plusieurs identités pour un même nœud.

Certification distribuée (suite)

- Résoud le problème de *Single Point of Failure* de la PKI
- Crypto à seuil est un système de partage de clé privée où un groupe doit coopérer pour utiliser la cryptographie
- Un schéma (n, t+1) autorise n parties à partager la capacité à faire de la cryptographie (signatures) à condition que au moins T+1 éléments le fassent ensemble
- Pour signer un certificat, chaque partie doit fournir son bout de signature en utilisant la clé privée partagée Si
- Tous les Si sont combinés dans le combineur qui peut générer la certification

Autre solution : la cryptographie fondée sur les identités

- [Shamir 84] et [Boneh-Franklin 01]
- Principe :



Principe : Indentity Based Encryption

- Composé de 4 algorithmes :
  - setup : genère les parametres globaux du systeme **params (publiques)** et la **master-key (connue uniquement de la PKG),** à partir d'un paramêtre de securité **k**.
    - Input : k (parametre de securité)
    - Output : params et master key
  - extract : la PKG utilise la **master-key** pour extraire la **clé privée** qui correspond à la **clé publique** donnée **(ID** $\in$ {0,1}***)**. (chaine de caractères identifiante)
    - Input : params, master key (Output de setup) et ID
    - Output : clé privée d
  - encrypt : signe les messages envoyés en utilisant la clé publique.
    - Input : params, ID, et un message M
    - Output : une empreinte C de M.
  - decrypt : déchiffre les messages reçus en utilisant la clé privée correspondante.
    - Input : params, ID, C et la clé privée d.
    - Output : M

- Règle : Pout tout M, Decrypt**(**params,ID,**C**,d**)=M avec C=**Encrypt**(**params,ID,**M)**

Exemple : [Boneh - Franklin]

- Fondée sur le « pairing » vieille fonction mathématique sur les courbes elliptiques qui vérifie :

$$Pair(aB, cD) = Pair(cB, aD)$$

- Utile en cryptographie sur les courbes elliptiques

[Boneh – Franklin 2001]

- Setup
  - La PKG génère un secret s, un aléa P
  - Donne à tout le monde P (param publique), sP (clé pub)

- Key Generation
  - Bob s'authentifie : Bob@hotmail.com, ID=HASH(Bob@hotmail.com) et demande sa clé privée.
  - La PKG lui renvoie sID

[Boneh – Franklin 2001]

- Encryption :
  - Alice hash Bob@hotmail.com -> ID
  - Chiffre le message avec k = Pair(rID, sP)
  - Envoie le message chiffré et rP

- Decrypt
  - Bob déchiffre avec k = Pair(sID, rP)
  - Le k de Bob et celui d'Alice sont identiques !

Avantages :

- Est « Cross-Domain »
- Pas besoin de connexions durables avec le serveur (pas de gestion de clés, pas de grosse gestion d'utilisateurs, pas d'augmentation de la taille selon mb utilisateurs,…)

- Révocation sans liste de révocation,…
- Petite signature (environ 160 bits)
- Echange de clés par identités
- Pas de certificats / pas de CA

- Pb : création d'ID buggués…

Comparaison IBE / PKI

PKI
- Protection maximale
- Bon pour signature et authentification
- Besoin de
  - Genèrer des clés pour les utilisateurs
  - Management pour la certification

Utile pour
- Authentification
- Signature
- À l'intérieur d'une structure

Identity-Based Encryption
- Bon pour chiffrement
  - Pas d'échange de clés
  - Révocation simple
  - Utile pour réseau ad hoc

Utile pour
- Chiffrement
- À l'intérieur ou extérieur d'une structure

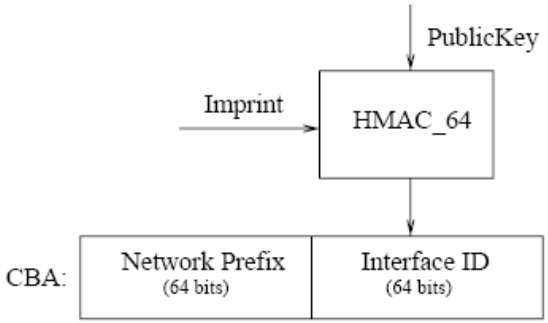
Proposition d'utilisation de IBE

- SPKI [Rivest] : un certificat possède 5 champs :
  - Issuer : qui donne l'autorisation
  - Sujet : qui acquière la permission
  - Délégation : le sujet peut il déléguer sa permission
  - Autorisation : spécifie la permission construite
  - Validité : temps de validité du certificat

- SPKI : key-oriented
  - Pas d'autorité de certification nécessaire
  - S_K(K' droit R, t) : K donne des droits à K' pour une période t (K et K' clés publiques de deux nœuds)

Statistically **U**nique **C**ryptographically **V**erifiable Ids : SUCV

- Projet ENS Lyon
- Utilisé quand deux peer veulent utiliser IPV6 pour créer un tunnel IPSec
- Utilisation dans ce cas d'addresses SUCV :
  - Fondé sur les crypto based address : concatenation du préfixe IPV6 de 64 bits || HASH(sa clé publique)



  - Puis après construction clé de session avec Diffie-Hellman

Mélanger tout cela :

- ID-based encryption
  - PK = HASH(ID)
  - SK calculé par PKG
- Utilisation de cryptographie à seuil pour distribuer la SK de la PKG
- Avantages : pas de certifications, pas de serveur centralisé
- Pbs : distribution des secrets initiaux

PKI, SPKI, certification distribuée,… en résumé

- PK Certificat = $(ID,PK)_{CA}$
  - Self-organized CA
  - Web of trust (PGP)
- Certificate-less
  - Crypto-based IDs: ID = h(PK)
  - ID-based Crypto: PK = f(ID)
- Pre distribution des clés
  - En choisissant aléatoirement des clés dans un espace donné

Sécurisation des protocoles de routage

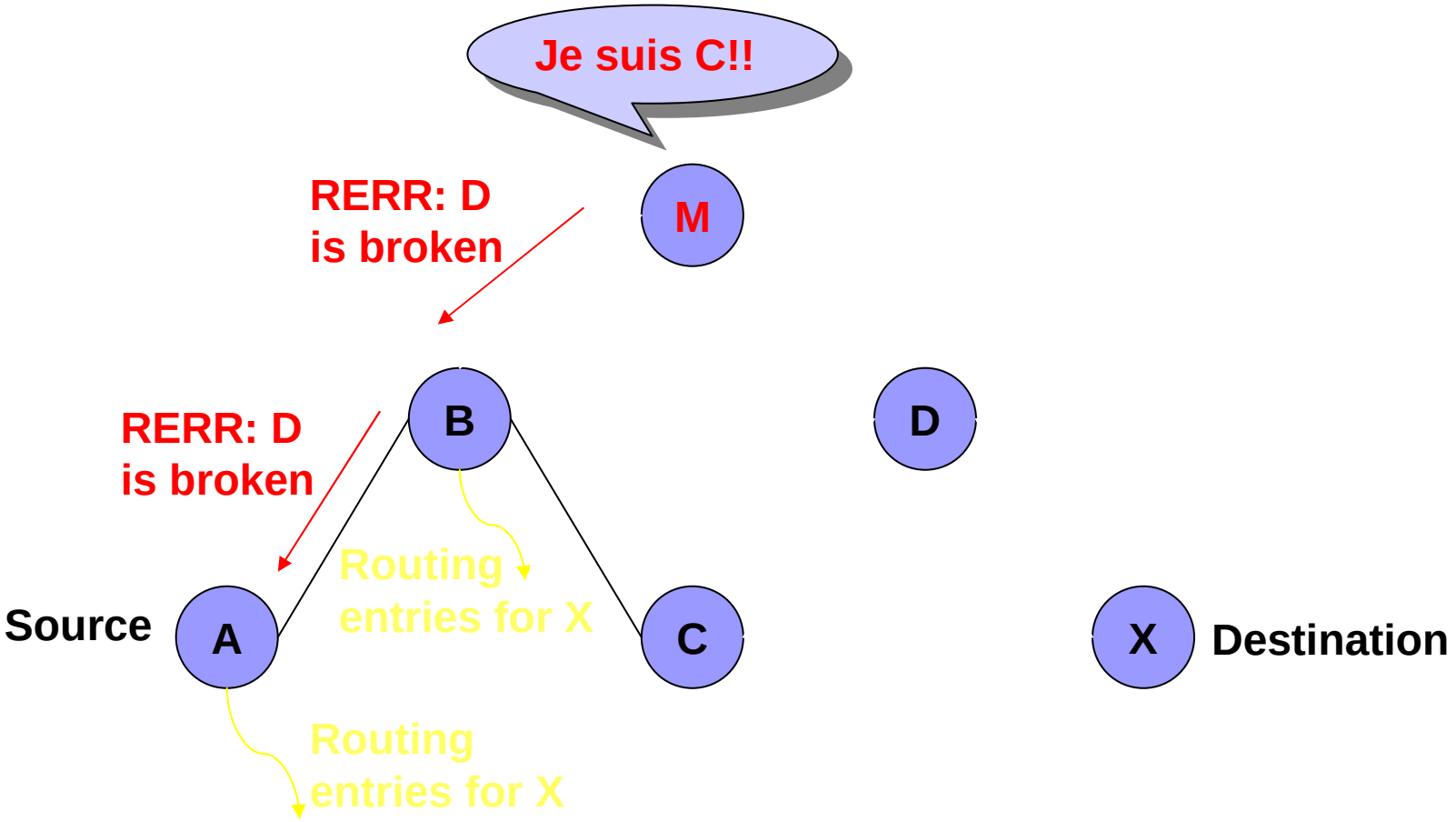
Introduction

- Plusieurs protocoles de routage :
  - DSR (réactif), AODV (réactif), OLSR (proactif),…
- Les attaques
- Proposition de solutions :
  - [Papadimitratos - Haas] : SRP pour Secure Routing Protocol sur DSR et ZRP
  - ARAN (AODV), ARIADNE (DSR), SEAD,TIK,...

DSR : rappel

- Protocole réactif (à la demande)
  - Fondé sur le principe du routage à la source
  - Permet à chaque émetteur de sélectionner et de contrôler les routes utilisés
- Messages :
  - route discovery (par l'envoie des routes requests)
  - route maintenance (par l'envoie des routes error)

Attaques utilisant la falsification



Falsification de routes dans DSR

Attaques utilisant la falsification

- Empoisonnement de Route Cache
  - Exploite la promiscuité
  - En broadcastant les paquets usurpés sur la route de X en les faisant passer par lui
  - Les noeuds voisins qui surprennent la transmission des paquets peuvent ajouter cette nouvelle route à leur ROUTE CACHE

Autres attaques

- L’attaque du trou de ver fonctionne aussi contre ce protocole
- Wormhole attack :
  - Deux attaquants A et B reliés par un connexion privée
  - A transfère chaque paquet reçu à travers le trou vers B qui broadcaste derrière (et réciproquement)
  - => Interruption potentielle du routage en court circuitant le flot normal des paquets routés

Proposition de sécurisation

- Pour DSR :
  - ARIANDE (DSR et TESLA)
  - S-DSR
  - SPR (ne voit pas, papier proposé)

- Aucune norme : à l’état de recherche !

Ariadne [Hu-Perrig-Johnson 03]

- Sécurisation du routage à la demande
- Résiste aux compromissions de noeuds
- Utilise TESLA [Canetti-Perrig...] : authentification de messages de routage avec :
  - Secrets partagés entre des paires de noeuds
  - Secrets partagés entre les noeuds communiquants combinés avec une authentification sur le broadcast
  - + signatures (autre version de TESLA)

ARIANDE : généralités

- Utilise les MACs pour l'authentification
- Prérequis :
  - Toute paire de noeuds S et D partage deux clés MAC K_{sd} and K_{ds} (une pour chaque direction)
  - Chaque noeud a sa propre TESLA key chain
- RREQ : 8 champs
  - <Request, Initiateur, Cible, Id, ti, HashChain(), listes de noeuds, liste de mac>
  - ti = Intervalle temps = la pire estimation entre la source et la destination
  - Un noeud intermédiaire vérifie si l'intervalle de temps fourni est valide

Ariadne : généralités

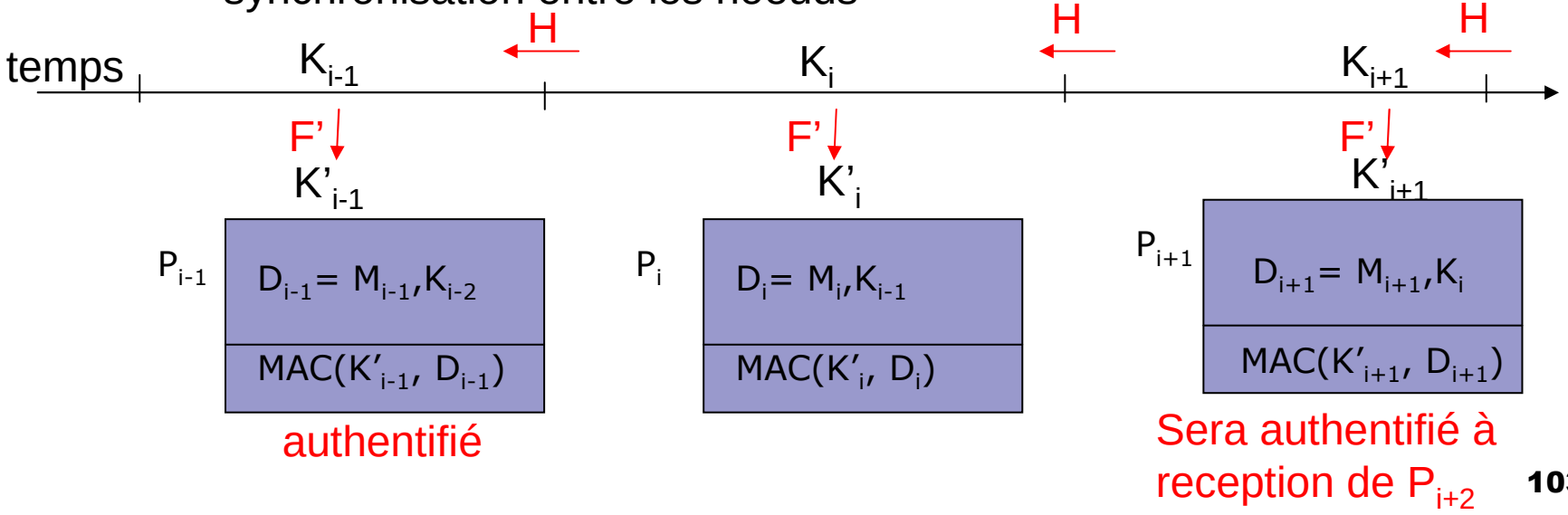
- Deux noeuds S et D partage deux clés MAC K_{sd} and K_{ds} (une pour chaque direction)
- Authenticité des ROUTEREQUESTS vient de
  - Message Authentication Code (MAC) (fonction de hachage ave clé)
  - Calculé à partir de K_{sd} avec un timestamp
- Authentification du ROUTEREPLY
  - Le noeud cible authentifie chaque noeud de la liste du ROUTE REQUEST
  - La réponse se fait à travers les mêmes noeuds
- Un hashage permet de savoir les noeuds qui ne sont plus présents dans la liste

Sécurisation de DSR : TESLA

- Protocole d'authentification de paquets
  - Authentification et intégrité des messages de routage par MAC
  - Choix d'un nb. al. K_n par le sender qui génère une hash-chaine de clés :

$$K_j = H^{n-j}(K_n)$$

  - Choix d'une deuxième fonction F' pour calculer les clés MAC
  - Les clés sont révélées au receveur
    - Dans l'ordre inverse de leur construction
    - En fonction d'un temps T dépendant du délai de transmission et de la synchronisation entre les noeuds



Ariadne : TESLA exemple de découverte de route

- RoutREQUEST : 8 champs
  - Request
  - Initiateur
  - Cible
  - Id
  - Intervalle temps
  - Hash chaine
  - Liste de noeuds
  - Liste de MAC (auth.)

ROUTE REQUEST
Route : $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$
$M = \langle \text{Request}, S, D, id, ti \rangle$

Clé de A

Adresse de A

$S : h_0 = \text{MAC}_{K_{SD}}(M)$
$S \rightarrow * : \langle M, h_0, (), () \rangle$

$A : h_1 = H(A, h_0)$
$M_A = \text{MAC}_{K_{A_{ti}}} \langle M, h_1, (A), () \rangle$
$A \rightarrow * : \langle M, h_1, (A), (M_A) \rangle$

$B : h_2 = H(B, h_1)$
$M_B = \text{MAC}_{K_{B_{ti}}} \langle M, h_1, (A, B), (M_A) \rangle$
$B \rightarrow * : \langle M, h_2, (A, B), (M_A, M_B) \rangle$

$C : h_3 = H(C, h_2)$
$M_C = \text{MAC}_{K_{C_{ti}}} \langle M, h_3, (A, B, C), (M_A, M_B) \rangle$
$C \rightarrow * : \langle M, h_3, (A, B, C), (M_A, M_B, M_C) \rangle$

Ariadne : Route Reply

- Quand le noeud cible reçoit RREQ, il vérifie la validité de RREQ
  - Détermine les clés spécifiées
  - Vérifie la chaine de Hash

    H[node_n, H[node_n-1, H[. . . , H [node_1, original hash chain] . . .]]]

- Si RREQ est valide Il retourne le RREP à la source

- Celui-ci vérifie le RREP par des tests => si tests passés le noeud S accepte le RREP sinon non.

Exemple

ROUTE REQUEST
Route : $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$
M = ⟨Request, S, D, id, ti⟩

$S : h_0 = \text{MAC}_{K_{SD}}(M)$
$S \rightarrow * : \langle M, h_0, (), () \rangle$

$A : h_1 = H(A, h_0)$
$M_A = \text{MAC}_{K_{A_{ti}}} \langle M, h_1, (A), () \rangle$
$A \rightarrow * : \langle M, h_1, (A), (M_A) \rangle$

$B : h_2 = H(B, h_1)$
$M_B = \text{MAC}_{K_{B_{ti}}} \langle M, h_1, (A, B), (M_A) \rangle$
$B \rightarrow * : \langle M, h_2, (A, B), (M_A, M_B) \rangle$

$C : h_3 = H(C, h_2)$
$M_C = \text{MAC}_{K_{C_{ti}}} \langle M, h_3, (A, B, C), (M_A, M_B) \rangle$
$C \rightarrow * : \langle M, h_3, (A, B, C), (M_A, M_B, M_C) \rangle$

D vérifie que les clés ne sont pas révélées
Que la chaine de Hash est correcte

ROUTE REPLY
M = ⟨Reply, D, S, ti , (A, B, C), (M_A, M_B, M_C) ⟩

$D : M_D = \text{MAC}_{KDS}(M)$
$D \rightarrow C : \langle M, M_D, () \rangle$

$C \rightarrow B : \langle M, M_D, (K_{Cti}) \rangle$

$B \rightarrow A : \langle M, M_D, (K_{Cti}, K_{Bti}) \rangle$

$A \rightarrow S : \langle M, M_D, (K_{Cti}, K_{Bti}, K_{Ati}) \rangle$

Ariadne: Route Discovery

- La sécurité provient des propriétés suivantes :
  - Le noeud destination peut authentifier le noeud source à l'aide de la clé partagée
  - Le noeud source peut authentifier chaque entrée du chemin du RREP
  - Aucun noeud intermédaire ne peut remplacer un autre noeud dans la liste de RREP
  - =>Toute altération de la liste de noeuds sera détectée

Ariadne: Route Discovery

- L’intervalle de temps est la pire estimation entre la source et la destination
- Un noeud intermédiaire vérifie si l’intervalle de temps fourni est valide

Ariadne: Maintenance de route

- RERR est renvoyé au sender en cas de problème de route
- RERR : <RERR, *sender*, *receveur*, *Ti*, *erreur MAC*, clé *TESLA récente*>
- Les noeuds intermédiaires
  - Forwardent le paquet et cherchent dans leur cache toutes les routes contenant <sender, receveur>
  - Vérifient la validité de l'intervalle de temps Ti
  - Si il est valide, authentifient RERR
  - Jusqu'à l'authentification, ils conservent RERR en mémoire jusqu'à ce que la clé soit dévoilée et suppriment dans ce cas toutes les mauvaises routes impliquées

Sécurité des RERRs

- L'authentification des RERRs empèche les noeuds non autorisés d'envoyer des erreurs
- Les noeuds intermédiaires ne peuvent pas que forwarder les paquets autorisés

- => Résiste aux compromissions de nœuds (pas de possibilité pour un noeud malicieux de s'authentifier)
- => Protection contre la modification, la fabrication, l'impersonification

- Par contre :
  - L'attaque par trou de vers fonctionne encore contre la version de base de Ariadne
  - Amélioration : utilisation de paquets "à laisse" permettant de détecter les trous de vers
  - Pour empécher les attaques par empoisonnement de Cache, la solution est de désactiver dans DSR la fonction d'"écoute de promiscuité"

Ariadne: ce qu'il faut offline

- Secrets partagés entre les pairs de noeuds

- Un mécanisme pour distribuer une clé publique TESLA pour chaque noeud (preuve sur K_{Ati})
- Les liens dans le réseau sont bidirectionnels

Ariadne : analyse de sécurité (1/4)

- Peut on modifier les routes sources dans les paquets ?

- La HASH chaine permet de vérifier que tous les noeuds sont dans la liste des noeuds :
  - => Toute altération de la liste de noeuds sera détecté

Ariadne : analyse de sécurité (2/4)

- Comment prévenir l'injection de routes non valides dans le réseau ?

- Le RERR contient des informations d'authentification à l'aide de clé TESLA
  - => pas de possibilités d'enlever des vrais routes
  - Tous les noeuds du chemin sont prévenus en cas de lien cassé

Ariadne : analyse de sécurité (3/4)

- Ariadne empèche les attaques par impersonnification
  - Car chaque paire de noeuds possède une clé partagée qui permet une authentification mutuelle
  - => pas de possibilité pour un noeud malicieux de s'authentifier
  - Protection contre
    - Modification
    - Fabrication
    - Impersonation

Ariadne : analyse de sécurité (4/4)

- Par contre :
  - L’attaque par trou de vers fonctionne encore contre la version de base de Ariande
  - Amélioration : utilisation de paquets “à laisse” permettant de détecter les trous de vers
  - Pour empécher les attaques par empoisonnement de Cache, la solution est de désactiver dans DSR la fonction d’”écoute de promiscuité”

Autre proposition : ARAN

- Proposé par des universités américaine (chicago, santa barbara,...)

ARAN : description rapide (1/5)

- Fondé sur du chiffrement à clé publique
- Requiert un serveur de certificats T :

  A recoit $cert_A = [IP_A, K_A, t, e]_{KT}$

  (t timestamp, e tps validité)
- Deux phases :
  - Authentification
  - transmission

ARAN : authentification (2/5)

- Assure l'existence de routes sûres entre une source et une destination
  - Chaque nœud intermédiaire stocke une paire de nœuds : (nœud_précédent, nœud_destination)
  - Chaque nœud signe le paquet de découverte de route
  - => nœud suivant s'assure de la validité du nœud précédent
  - Le nœud destination répond en envoyant son certificat qui sera utilisé durant la transmission

ARAN : transmission (3/5)

- Les paquets de découverte de routes RDP :

$$A \rightarrow brdcast:[RDP, Ip_{dest}, cert_A, N_A, t]_{KA}$$

$$B \rightarrow brdcast:[[RDP, Ip_{dest}, cert_A, N_A, t]_{KA}]_{KB}, cert_B$$

...

- Dest X recoit RDP et renvoie REP :

$$X \rightarrow D:[REP, Ip_A, cert_X, N_A, t]_{KX}$$

$$D \rightarrow C:[[REP, Ip_A, cert_X, N_A, t]_{KX}]_{KD}, cert_D$$

...

ARAN : maintenance de routes (4/5)

- Protocole à la demande
- Les messages d'erreurs sont signés par le nœud précédent l'erreur et le refait suivre dans l'autre sens afin d'enlever les routes cassées.
- Comme les messages sont authentifiés => pas de nœud malicieux qui peut générer des route errors
- Vérification des voisins du nœud « cassé »

ARAN : évaluation (5/5)

- Avantages
  - Sûr tant que le CA n'est pas corrompu
  - Confidentialité garantie : chiffrement des messages
  - La structure du réseau n'est pas exposée : chiffrement
  - Résiste à la plupart des attaques

- Inconvénients
  - Besoin de place mémoire supplémentaire : stockage des paires de nœuds pour la route
  - Perte de vitesse liée au chiffrement

Autres propositions de sécurisation de DSR

- S-DSR proposé par Huang – Li – Jia
- Utilise IPv6 et des propriétés particulières de IPv6
  - Proposition d'un protocole DAD (Duplicate Address Detection) fondé sur NDP (Neighbor Discovery Protocol) qui prend deux types de messages :
    - NS (Neighbor Solicitation)
    - NA (Neighbor Advertisement)

S-DSR (1/6)

- Utilise des adresses CGA (Cryptographically Generated Address)

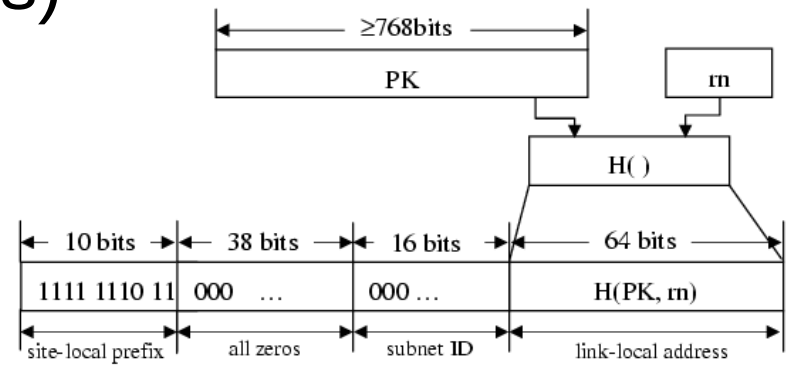


Figure 1: The CGA site-local IPv6 address.

- Incorpore une auto-configuration pour l'adresse
- et un auto enregistrement pour le DNS

S-DSR (2/6)

- On choisit une fonction de hashage H
- Un serveur DNS du réseau MANET possède une paire de clé publique/clé privée connue de tous les noeuds avant qu'ils n'entrent dans le réseau
- Chaque noeud fournit au DNS : (domain name, IP address)
- Signature au fur à mesure des différents noeuds sur la route avec vérification de l'adresse IP

S-DSR (3/6)

- Routage :
  - $(S_{IP},D_{IP},seq,SRR,[S_{IP},D_{IP},seq]_{SSK},S_{PK},S_{nd})$
  - Chaque noeud intermédiaire vérifie si S_{IP} correspond à $H(S_{Pk}, S_{rn})$.
  - Il vérifie la signature $[S_{IP},D_{IP},seq]_{SSK}$ avec S_{PK} et vérifie si le résultat correspond à $[S_{IP},D_{IP},seq]$ du message
  - Il vérifie chaque adresse IP de SRR.
    - Pour une adresse IP I_{IP}, les informations correspondantes sont $[S_{IP},seq]_{ISK}$, I_{IP}, I_{PK},I_{rn},
    - En regardant si I_{IP} correspond à $H(I_{PK},I_{rn})$ et si $[S_{IP},seq]_{ISK}$ peut être déchiffre par I_{Pk} et est égale $[S_{IP},seq]$.
    - Vérifie si *seq* est plus petit que le nombre fournit par S dans RREQ

S-DSR (4/6)

- Chaque noeud intermédiaire I ajoute $[S_{IP},seq]I_{SK},I_{IP},I_{PK},I_{rn}$ à la route sûr enregistrée dans SRR et rebroadcast le message.
- => RREQ valide
- Le noeud de destination D unicaste alors RREP
  - $(S_{IP},D_{IP},seq,RR,SR(D\text{-}S)$ $[S_{IP},seq,SR(D\text{-}S)]D_{SK},D_{PK},D_{rn})$ à S selon la route SR(D-S) dérivée de SRR

S-DSR (5/6)

- Le routage peut aussi faire intervenir directement le serveur DNS

S-DSR : analyse de sécurité (6/6)

- S-DSR resiste à :
  - Black hole attack
  - Route request (RREQ) message reply attack
  - Forged route request (RREQ) message attack
  - Forged address reply (AREP) message attack
  - Forged route error (RERR) message attack
  - Tampered control message attacks
  - DNS server impersonation attack

AODV

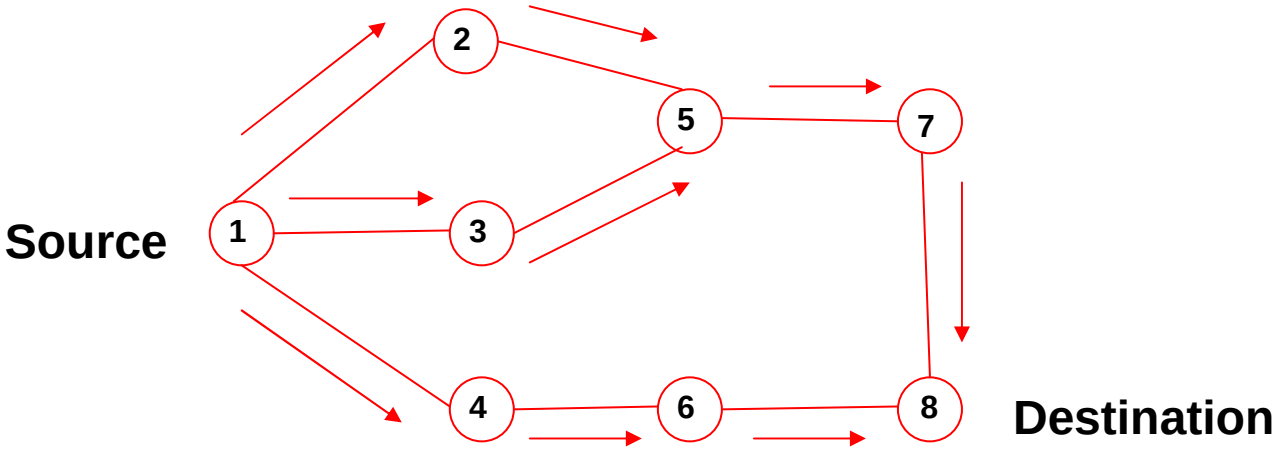
AODV : *Ad hoc On Demand Distance Vector*

- Créé par les concepteurs de DSDV
- Protocole réactif adapté aux réseaux qui affichent une topologie fortement dynamique
- Utilise des tables de routage traditionnelles, une entrée par destination : l'entrée stocke le prochain saut
  - Basé sur le routage de vecteur de distance
- Mécanisme de découverte de routes similaire à DSR
- Découverte de route par inondation
  - Calcul des tables de routage
  - Évaluation des surcouts

AODV

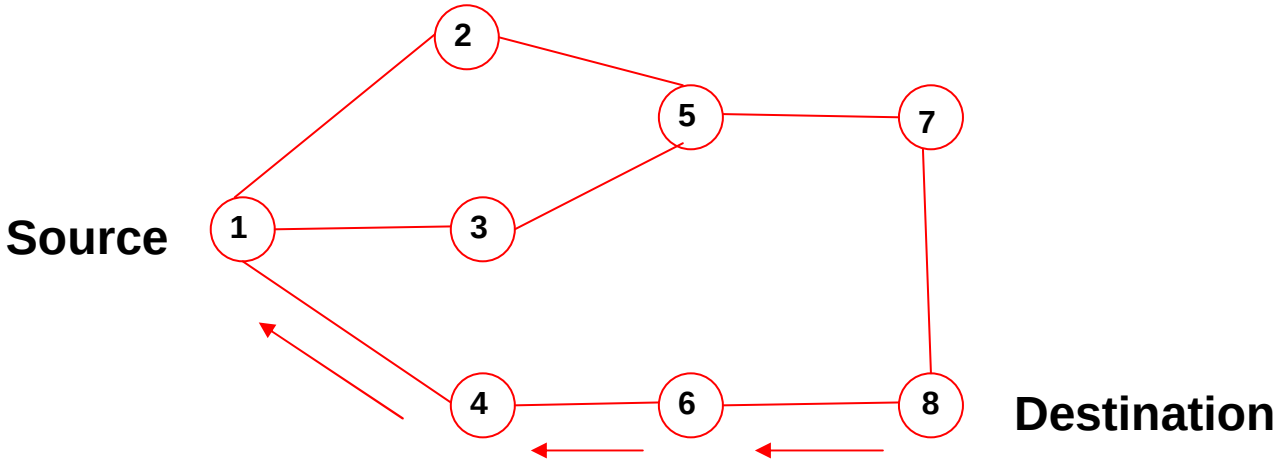
- Pas besoin d'échanges de signalisation entre voisins pour mettre à jour les tables de routage
  - À la demande d'un routage, une table est implantée dans chaque nœud
- Pour cela, la source doit se livrer à l'inondation d'une requête (idem DSR)
  - RREQ : Route REQuest
  - Tous les noeuds ménagent une entrée dans leur table locale pour l'orientation du flux
  - La destination répond à la source avec RREP (Route REPly)
- La topologie du réseau peut se modifier
  - Coupure d'un lien radio
  - Le noeud victime avertit la source par un message d'erreur
  - La source effectue une reprise : découverte d'une nouvelle route

AODV: découverte de routes



Propagation d'un ROUTEREQUEST par inondation

AODV: découverte de routes



Chemin pris par le route reply

AODV: découverte de routes

- Maintien des routes à jour
  - Utilise des séquences de nombres
  - Un noeud compare la destination du RREQ avec les routes qu'il a déjà dans sa table
  - Soit il répond avec sa propre route si l'entrée est récente soit il re-broadcaste à ses voisins
  - Les noeuds recevant le paquet mettent à jour leur information relative à la source et établissent des pointeurs de retour vers la source dans les tables de routage.
  - RREQ contient : l'IP de la source, le numéro de séquence courant et l'ID de broadcast, le numéro de séquence de la destination le plus récent connu de la source.
  - Un noeud recevant un RREQ émettra un paquet route reply (RREP) soit si il est la destination, soit si il possède une route vers la destination avec un numéro de séquence supérieur ou égal à celui repris dans le RREQ.
  - Si tel est le cas, il envoie (unicast) un paquet RREP vers la source. Sinon, il rebroadcast le RREQ. Les noeuds conservent chacun trace des IP sources et des ID de broadcast des RREQ.
  - Si ils recoivent un RREQ qu'ils ont déjà traité, ils l'écartent et ne le transmettent pas.

AODV: Maintenance de routes

- Une entrée d'une table de routage est dite "expirée" si elle n'a pas été utilisé récemment
- Un ensemble de noeuds prédécesseurs d'un noeud est maintenu pour chaque entrée de la table
- Ces noeuds sont notifiés par des RERR si une entrée a expiré
- Si un lien casse pendant qu'une route est active, le noeud propage à la source un message RERR

SAODV : Secure AODV

- Une version d'ARAN existe aussi pour AODV
  - Même système que précédemment
- Il existe une extension sûre d'AODV : SAODV
  - Université espagnole : Guerrero Zapata, Manel
  - Fournit sécurité de base : intégrité, authentification et non-répudiation
  - Chaque noeud a une paire de clé publique/privée
  - Certification Addresse/clé

SAODV: Hash chains et Signatures

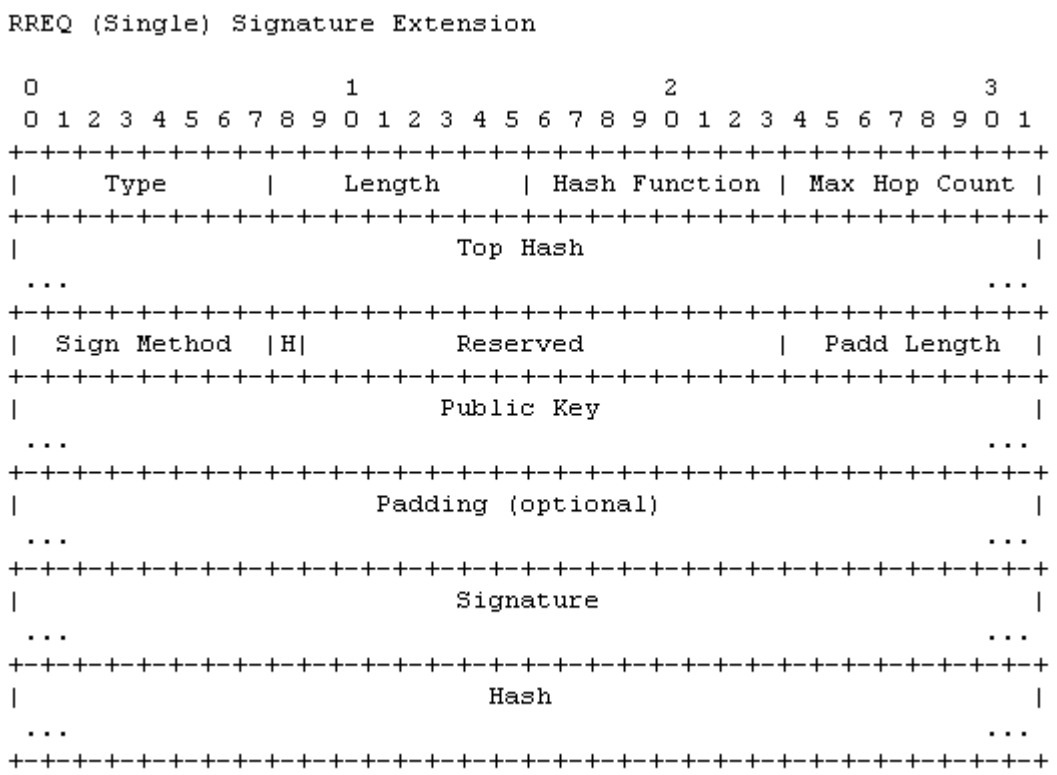
- Sert à protéger l'intégrité et la "non-mutation" des données des messages RREQ et RREP
- On signe tout dans le paquet sauf le compteur de sauts et la chaine de hash
- Quand un noeud reçoit un message de routage, il vérifie avant tout la signature

SAODV: Hash Chains et signature

- Utilisé pour authentifier les messages RREQ et RREP
- Chaîne de hachage pour authentifier le compteur de sauts
  - Assure que le compteur de saut n'a pas été modifié par un attaquant
- Extenstion de la signature pour empécher les attaques par impersonification
  - RREQ-SSE inclus des messages RREQ signés et une chaine de hachage qui dépend d'une valeur aléatoire choisie par le sender
  - Le noeud receveur répond par RREP-DSE (même format)
  - La signature protège les messages RERR

SAODV : les paquets (1/2)

- Type : 64, Length : longueur des paquets spécifiques
- Hash Function : fonction de hash utilisée
- Max Hop Count : le nb de saut max supporté par l'authentification
- Top Hash : hash du compteur de sauts
- Signature : algo utilisé
- H : Half Identifier flag. 1 = HID, 0 FID
- Public Key : clé public sender
- Signature : signature des champs précédents sauf le compteur de saut
- Hash : actuelle valeur du compteur de saut (chaine de hash)

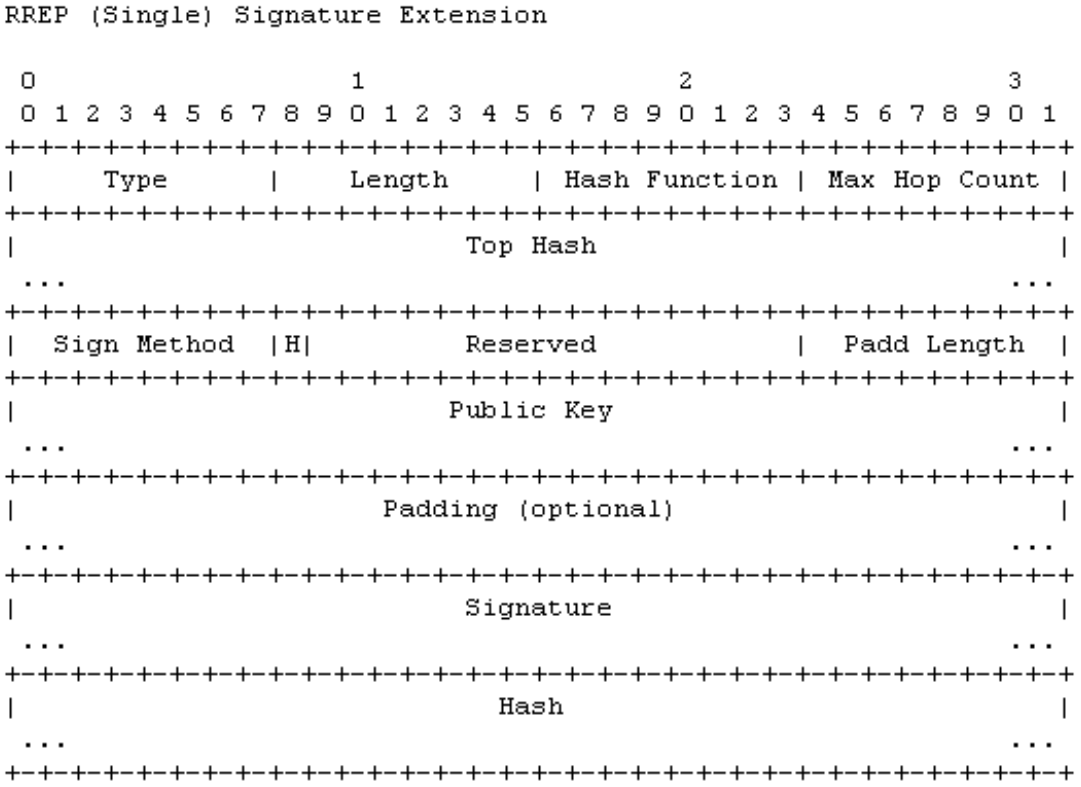


SAODV : les paquets (2/2)

- Type : 65, Length : longueur des paquets spécifiques
- Même chose

- => existe une version avec deux champs de signature pour valider le paquet RREQ.

RREP (Single) Signature Extension



SAODV : identités et clés

- En développement actuellement :
  - N’évite pas la certification par une PKG extérieure
  - L’adresse d’un nœud est un préfixe de 64 bits SAODV_HID (Half IDentifier) ou de 128 bit SAODV_FID (Identifier)
  - Ces deux valeurs sont générées comme des SUCV :
    - SAODV_HID = SHA1HMAC_64(PublicKey, PublicKey)
    - SAODV_FID = SHA1HMAC_128(PublicKey, PublicKey)

SAODV: Analyse de Sécurité (1/2)

- Preuve de validité par signature
- Empèche l'attaque par injection de routes errors
  - la signature numérique révèle que le noeud malicieux n'est pas sur la route
- => Détection de noeuds malicieux
- La chaine de Hash empéche la modification du compteur de saut

SAODV : analyse de sécurité (2/2)

- SAODV empèche les attaques par :
  - modification, fabrication et impersonification

- Cependant => ne résiste pas à l'attaque par trou de vers

OLSR : rappel

- Protocole proactif
  - Applique dans un contexte ad-hoc les règles de routage centrées sur l'état du lien
  - Ce type de routage devient une référence en matière de routage en IP filaire (avec OSPF)

- Chaque noeud connaît parfaitement la position des autres dans le réseau
- Choix du chemin le plus court ou le plus rapide = formalité
  - Algorithme de Dijkstra

- Pas d'inondation car elle générerait de la redondance
  - Les paquets parviennent aux noeuds 1 seule fois
  - Le noeud élit parmi ses proches un représentant

- Rôle de relais multipoint : Multi-Protocol Router (MPR)
- Condition pour devenir MPR : pouvoir atteindre tous les noeuds à une distance de 2 sauts, avec un lien symétrique MPR communiqués à tout le réseau par des messages TC (Topology Control) périodiques
  - À la réception des TC, mise à jour des tables de routage

OLSR : rappel

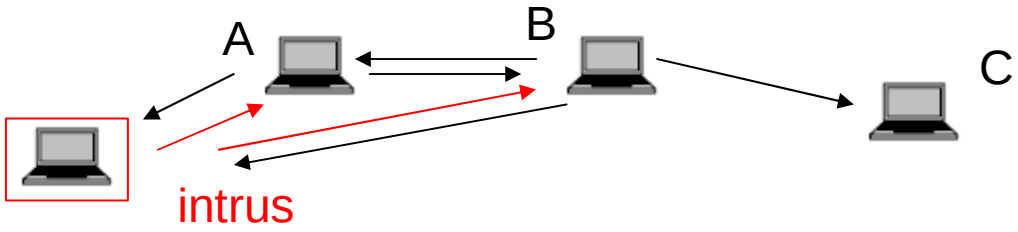
- OLSR est un protocole proactif basé sur l'état de lien
- Eviter l'inondation : MPRs (Multi point relay)
- Seuls les MPRs vont transmettre les messages topologiques (découverte du réseau)
- Adapté aux réseaux avec faible mobilité
- Modification de l'algorithme de routage
- Shortest Path First (Le plus court chemin)

OLSR : messages

- « Hello »
  - Découverte du voisinage et maintien de cette relation
  - Chaque équipement émet un « Hello » périodique (2 secondes). Un broadcast détermine qui sont les voisins et leur lien (asymétrique, symétrique)
- Topology Control (TC)
  - Mise à jour de la topologie

Exemple d'attaques contre OLSR

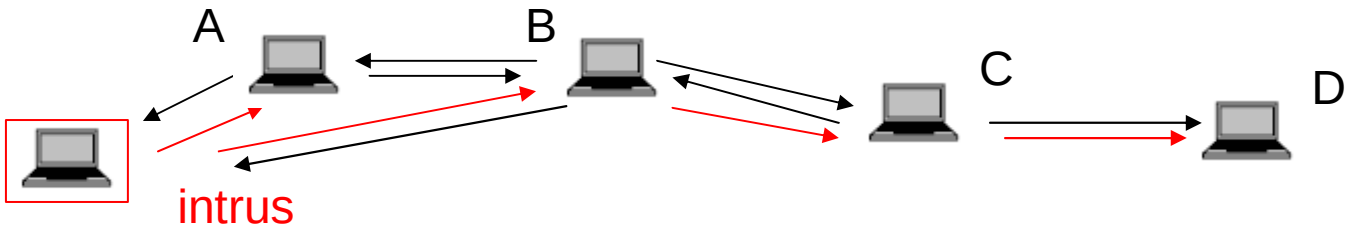
- Insertion de faux messages « hello »



- B est MPR de A. C est à 2 sauts de A
  - Envoi de messages Hello par B
  - Envoi de messages Hello par A
- Insertion d'un message Hello par l'Intrus annonçant à A, B, C et X un lien symétrique
- Conséquences :
  - Sélection de l'Intrus comme MPR par A et B
  - Le trafic de A vers C passera par l'Intrus

Exemple d'attaque contre OLSR

- Insertion de faux messages TC



- A possède une route vers D à 3 sauts en passant par B
  - Envoi d'un message TC par C : l'intrus identifie D à trois sauts
  - Envoi d'un message TC par l'Intrus annonçant que D fait partie de ses points
- Conséquences :
  - A arrête d'envoyer du trafic vers D en passant par B
  - A envoie son trafic vers D en passant par l'Intrus

=> D ne réagit pas aux fausses infos annoncées dans le TC qu'il reçoit

Autres attaques etc.

- Deux attaques essentielles :
  - Identity spoofing
  - Link spoofing
- Les attaques sur la modification de paquets OLSR fonctionne aussi !
- Exemple : modification du Message Sequence Number
  - Un intrus change cette valeur de x à x+i
  - Voyant cela, d'autres nœuds arrêtent de transférer les messages TC provenant de A ayant un Message Sequence Number < x+i

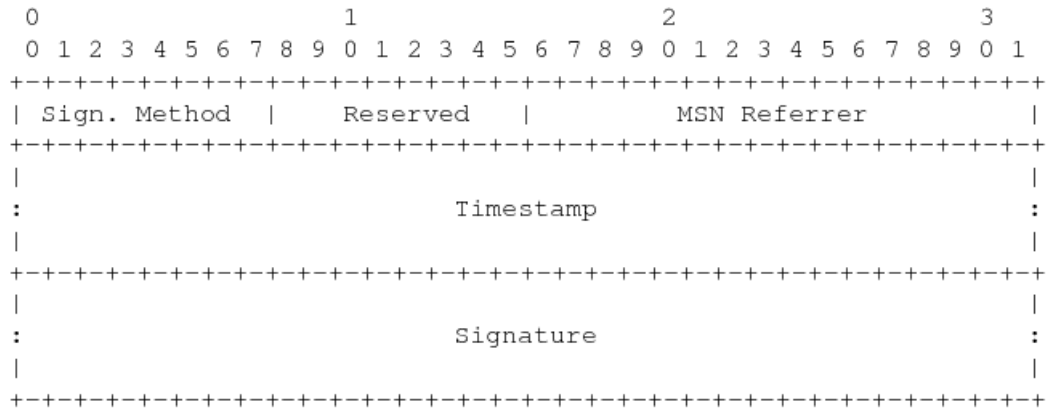
Proposition de sécurisation de OLSR

- Deux propositions françaises :
  - Hipercom (projet INRIA) : approche centralisée
  - Supélec Rennes : approche distribuée

S-OLSR (INRIA Paris)

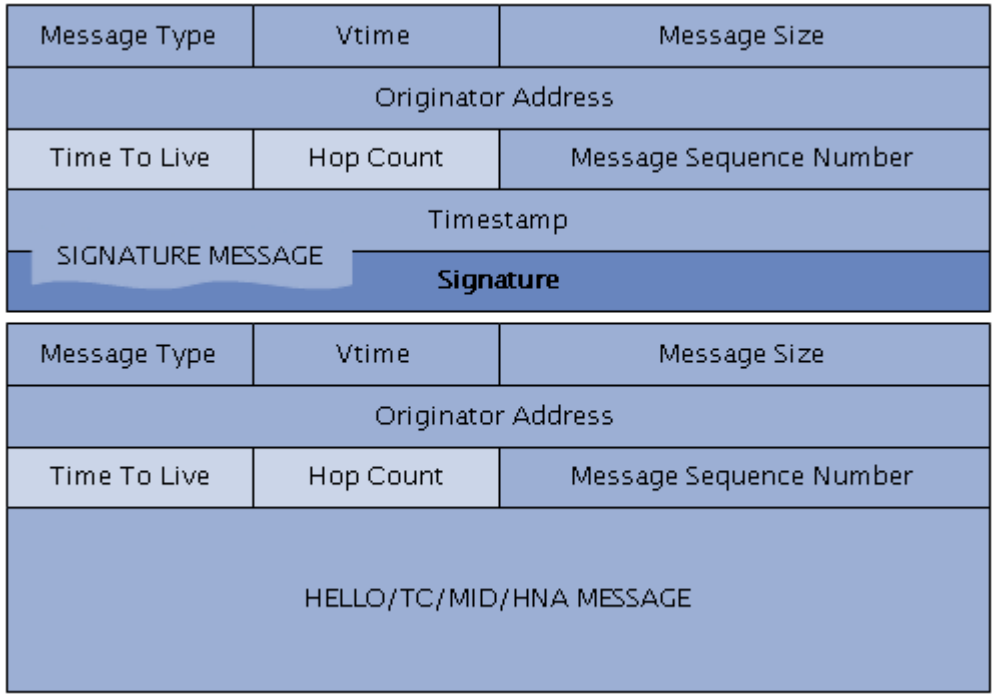
- Utilisation d'une signature type
  - Sign(nodeid, key, message)
  - Verif(originatorid, key,message, signature)
- Principe :
  - Signature de la source sur les messages de contrôle (HELLO et TC) transmise avec le message
  - Avec un timestamp

Format de la signature



S-OLSR

Format du paquet



S-OLSR

- Format de la signature :
  - Signature originator = sign(nodeid, key,<message contrôle, timestamp>)
  - Le nœud receveur
    - vérifie la signature et l'intégrité du message
    - Si oui => vérification du timestamp
    - Si OK => traiter le message de contrôle sinon écarter le message

S-OLSR

- Les timestamps :
  - Utilisés pour éviter les replay attacks et les modifications champ time to live.
  - Pour cela : preuve de freshness du timestamp
  - Timestamps possibles :
    - Timestamps en temps réel (horloge)
    - Timestamps logiques (augmente de 1 quand événement)
    - Protocole d'échange des timestamps (Needham –Shroeder)
    - Ici, remplace la validation par chaine de hachage

S-OLSR

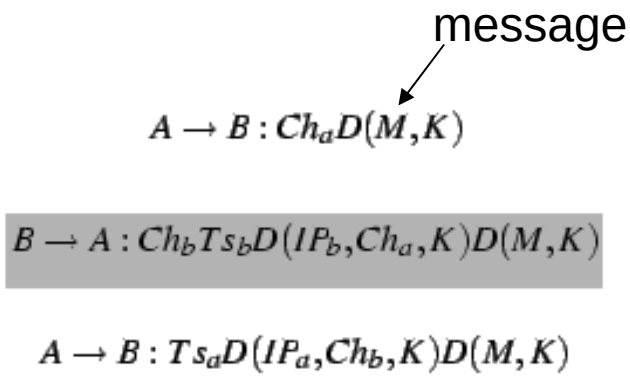
- Protocole d'échange de timestamps initial entre deux nœuds voisins (création initiale par braodcast) :

Cha, Chb : val. al. Ou horloge

A signe ce message avec clé partagée K

B renvoie son timestamp, son IP.

A fournie son timestamp et son IP.



- => fin échange timestamp les deux nœuds garde en mémoire la différence de temps qui les sépare

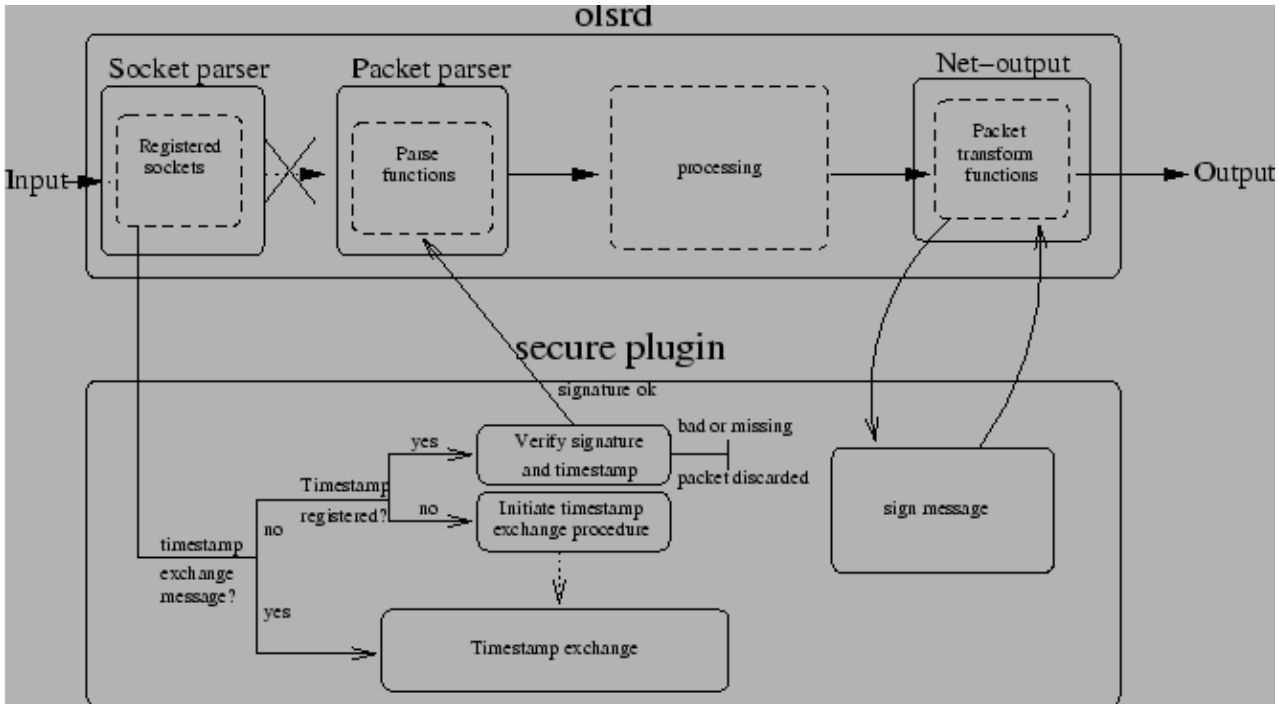
S-OLSR

- Généralisation de ce processus aux nœuds du réseau => maintien d'une table de timestamps pour chaque nœud

- Un timestamp dans un message signé est accepté si $|T_{sender} - T_{receveur}| <$ seuil connu

S-OLSR

- Structure générale de sécurité :



S-OLSR

- Gestion des clés : Simple proactive PKI
- Trois classes de nœuds :
  - Trusted nodes : un nœud a fait confiance à un nœud x si la clé de x est connu de a et validé par une autorité de certification du réseau
  - Untrusted nodes : clé inconnu ou non validé
  - Autorités de signature : nœud dont la clé publique est connu de tous les nœuds du réseau.
    - Elles autorisent les nouveaux nœuds à enregister leur clé de façon sécurisée => trusted nodes
    - Distribuent périodiquement des certificats contenant la liste des trust nodes

S-OLSR

- Chaque nœud doit faire enregister sa clé puis rafraichissement périodique des certificats
- Gestion et création des MPR
- Un nœud maintient des tables :
  - Pour les trust nodes
  - Pour les untrust nodes
  - Sélection des MPR parmi le voisinage sûr

S-OLSR

- Algorithme :
- Une autorité de certification distribue des clés à ces voisins
- Les voisins regardent leur propre voisinage et fournissent des clés aux nœuds à deux sauts
- Les MPRs sont choisis parmi les premiers nœuds
- Etc…

S-OLSR

- Parade contre les nœuds compromis
  - Maintenir le timestamp
  - Sélection de MPRs uniquement parmi les trust nodes
  - Etude de la position géographique des nœuds (SIGLOG)
  - Signatures multiples
  - Ou méthode de détection d'intrusion sur comportement malveillant (système d'accusation,…)
- Autre système proposé :
  - Simple reactive PKI avec un key request et un key reply à destination d'un nœud précis

Conclusion partielle

- Méthodes de sécurisation en ad hoc emploient nouvelles formes de cryptographie distribuée
- Pb : on ne peut jamais vraiment se passer d'une gestion centrale au moins au début
- La sécurité rajoute toujours un cout :
  - Augmentation de la taille des paquet
  - Augmentation du nombre de calculs nécessaires

Modèle de réputation et de confiance

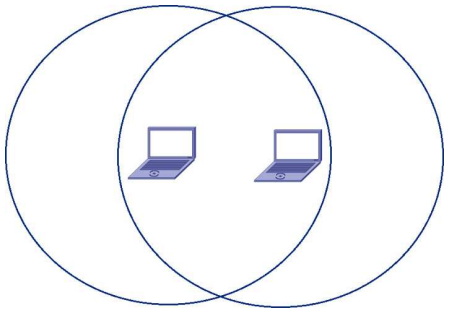
Introduction

- Cherche-t'on vraiment de la sécurité ?
  - On cherche à protéger le routage d'attaques
  - Une coopération à la base est nécessaire pour le bon fonctionnement des réseaux ad hoc et la fourniture de service
  - Environnement militaire, diplomatique : besoin de sécurité
  - Environnement urbain,… ?

De la sécurité à la confiance

- Besoin d'être sûr de quelques nœuds mais pas forcément de tous
- Dépend du réseau

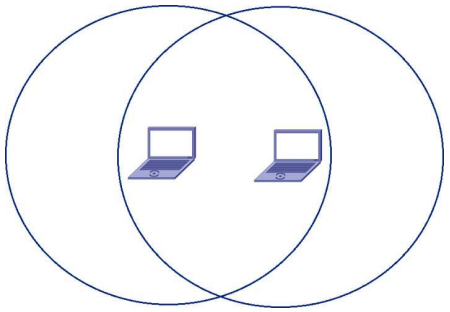
- Environnement fermé (entreprise,...)
  - La confiance existe à priori
  - L'authentification implique la fidélité
  - MAIS
    - Besoin d'infrastructure
    - Pénurie de ressources (capteurs,...)
    - Manque de connectivité



- Environnement ouvert
  - Pas de confiance à priori
  - L'authentification est difficilement garantie

- Nouveaux paradigmes :
  - Protocole d'établissement de la confiance fondé sur un historique, des jeux,…
  - Modèle de réputation

Réseaux ad hoc : modèle de confiance



- Pour établir :
  - L'identification des nœuds
  - La relation et le degré de confiance entre les nœuds (exemple : PGP : confiance transitive, deux niveaux)
- Habituellement :
  - Confiance = identités avec une gestion centralisé (type serveur)
- Ici :
  - Distribution de l'autorité de confiance déterminée par les caractéristiques du réseau ad hoc
    - Les nœuds ont-ils une relation préalable : même entreprise,…
    - Quel est leur capacité ? La taille du réseau ?...
  - Sinon : « réseau spontané »
    - Comment gérer la confiance ?

Premier Exemple : PGP

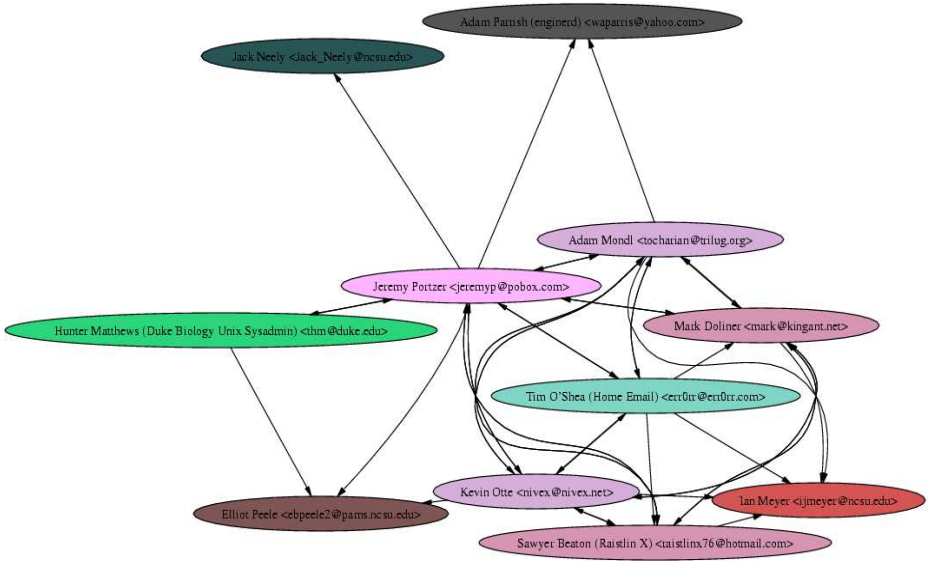
- PGP = Pretty good privacy sous licence GPL (jusqu'à la version 2.x)

- Cryptosystème hybride permettant l'échange de données chiffrées/authentifiées
  - Cryptographie symétrique pour chiffrer
  - Asymétrique pour signature et transport de clé de session

PGP et le « Web of trust »

- Authentification repose sur l'anneau de confiance : web of trust
  - L'authenticité d'une clé publique est prouvée de proche en proche
  - Adaptée au communautés pas à grande échelle !
  - Fondée sur la notion de COI « Community of interest »

- Exemple : le « Debian Keyring Web of Trust »



Principe : je signe avec ma clé secrète la clé publique des personnes dont je suis sûre

Distribution de paire de clés

- Généré par vous-même
- Ou serveur de clés et serveur référencent les clés
  - http://wwwkeys.pgp.net
  - Recherche sur une chaine de caractères, un nom,…
- Je donne ma clé publique à mes amis qui la signe si il me font confiance
  - Par exemple lors d'une « GnuPG Keysigning party »

PGP

- Pas d'autorité de certification
- Alice $\rightarrow$ Bob et Bob $\rightarrow$ Eve $\Rightarrow$ Alice $\rightarrow$ Eve
- Fusion de plusieurs certificats
  - Réputation cachée

- Pours : pas de gestion centralisée
- Contres :
  - Initialisation / stockage
  - Transitivité de la confiance (pb si un noeud malveillant au milieu)

Modèle de réputation

- Principe de notation des et sur les nœuds du réseau
  - Deux notes : une note de confiance en les nœuds eux-mêmes
  - Une note de confiance sur les avis sur d'autres nœuds

- Un noeud souhaite interagir avec un autre
  - Comment décider s'il le fait ou pas ?
    - Confiance personnelle
    - Confiances extérieures
    - Principe de notation : > ½ confiance sinon non

Modèle de réputation

- Définition d'une métrique particulière permettant de définir :
  - Des notes d'intéractions directes selon services, échanges fournis,…
  - Des poids des différents nœuds en qui ont à confiance
  - Le poids de leur avis sur d'autre nœuds (anciennes recommandations),…
  - Le poids de ces avis
  - Maitenance de tables de confiance

Modèle de réputation

- Problème essentiel :
  - Toujours besoin d'une authentification : comment faire ?
  - Un groupe de nœuds peut détruire la réputation d'un nœud isolé
  - Autre modèle : modèle de recommandation
  - Reste également les pbs de topologie du réseau

Liste de solutions

Token-based [Yang,Meng,Lu] } Threshold cryptography

Nuglets [Buttyan,Hubaux]
SPRITE [Zhong, Chen, Yang] } Micro-payment

CONFIDANT[Buchegger,Le Boudec]
CORE [Michiardi,Molva]
Beta-Reputation [Josang,Ismail] } Reputation-based

Bittorrent (P2P) } Built-in with Application

Enjeu de sécurité dans les réseaux ad hoc

- Authentication / Integrite / Condentialite / Disponibilite :
- Authentification : pierre angulaire d'un reseau ad hoc securisé.
  - Avant confidentialité et intégrité car avant communication avec la bonne entité

- Si l'authentication est mal gerée, un attaquant peut injecter des messages erronés.
  - Puis intégrité des messages et des nœuds.
  - Confidentialité des données echangées
- La disponibilité = difficile à gérer car contraintes :
  - Topologie dynamique
  - Ressources limitées sur certains noeuds de transit.
  - Communications facilement brouillées ou perturbées.

Conclusion

- Besoin de sécurité dans les réseaux ad hoc à plusieurs niveaux
  - Routage
  - Identification, authentification, contrôle d’accès,…
- Domaine en pleine expansion