Réseaux de Capteurs Sans Fils

Y. Challal

Contributeurs

- **Hatem Bettahar**
- **Boushra Maala**
- **Abdelraouf Ouadjaout**
- **Noureddine Lasla**
- **Mouloud Bagaa**
- **Ben Hamida Fatima Zohra**

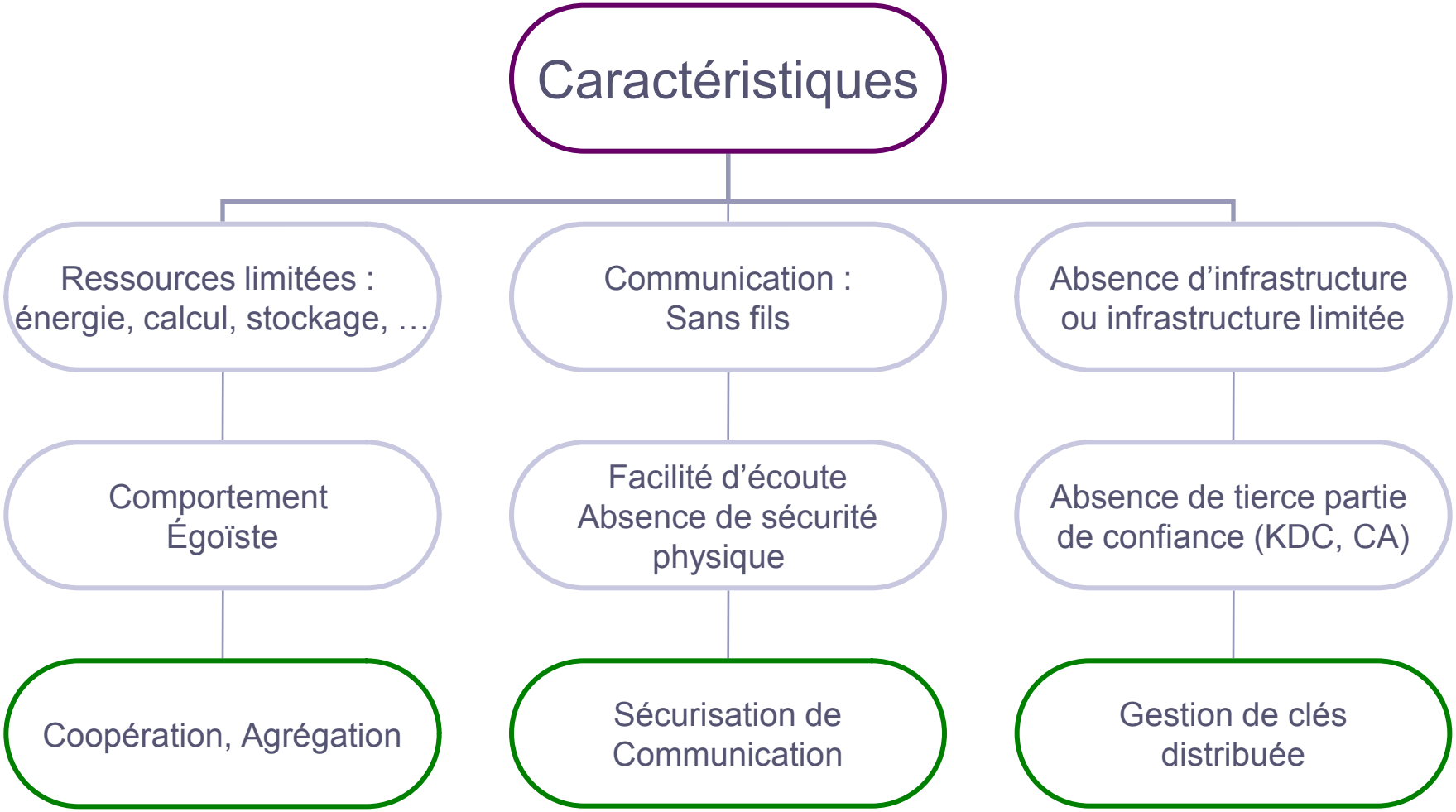
- **Chenyang Lu (Virginia)**
- **Kemal Akkaya and Mohamed Younis**
- **C. Intanagonwiwat, R. Govindan, D. Estrin, et. al., presented by Romit Roy Choudhury (Illinois)**
- **Martin Haenggi, « Wireless Sensor Networks »**

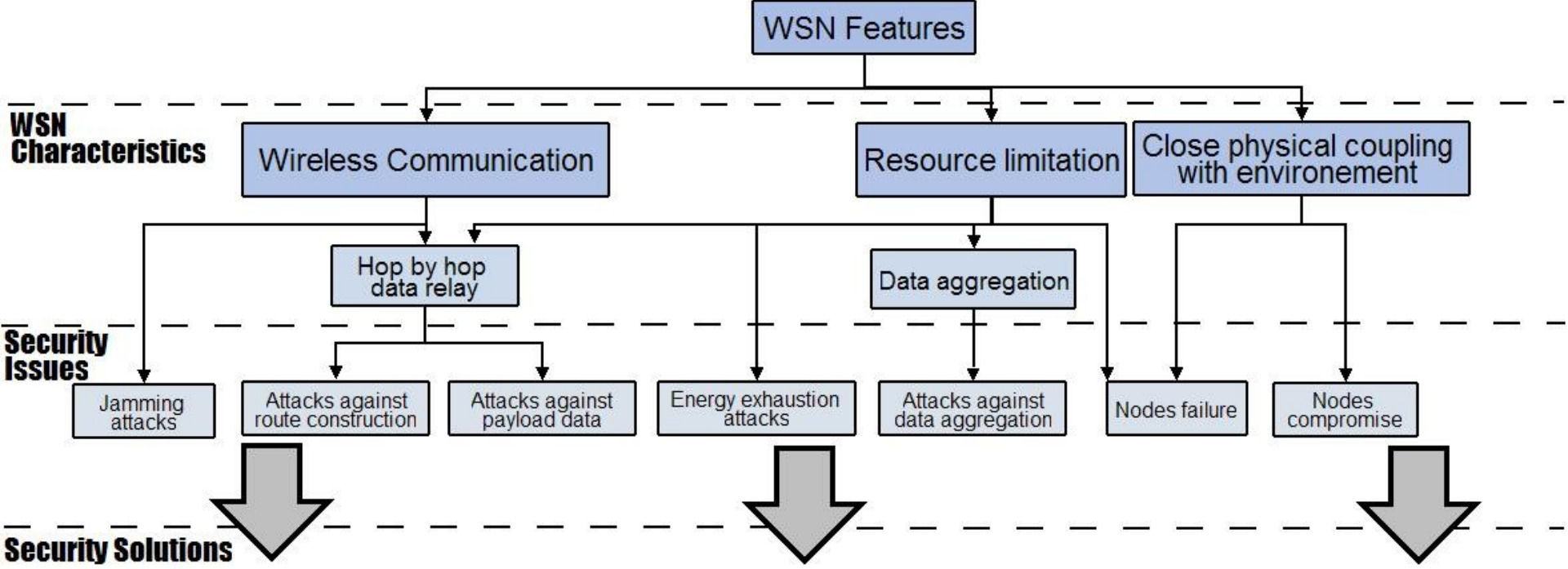
Sécurité dans les RCSF

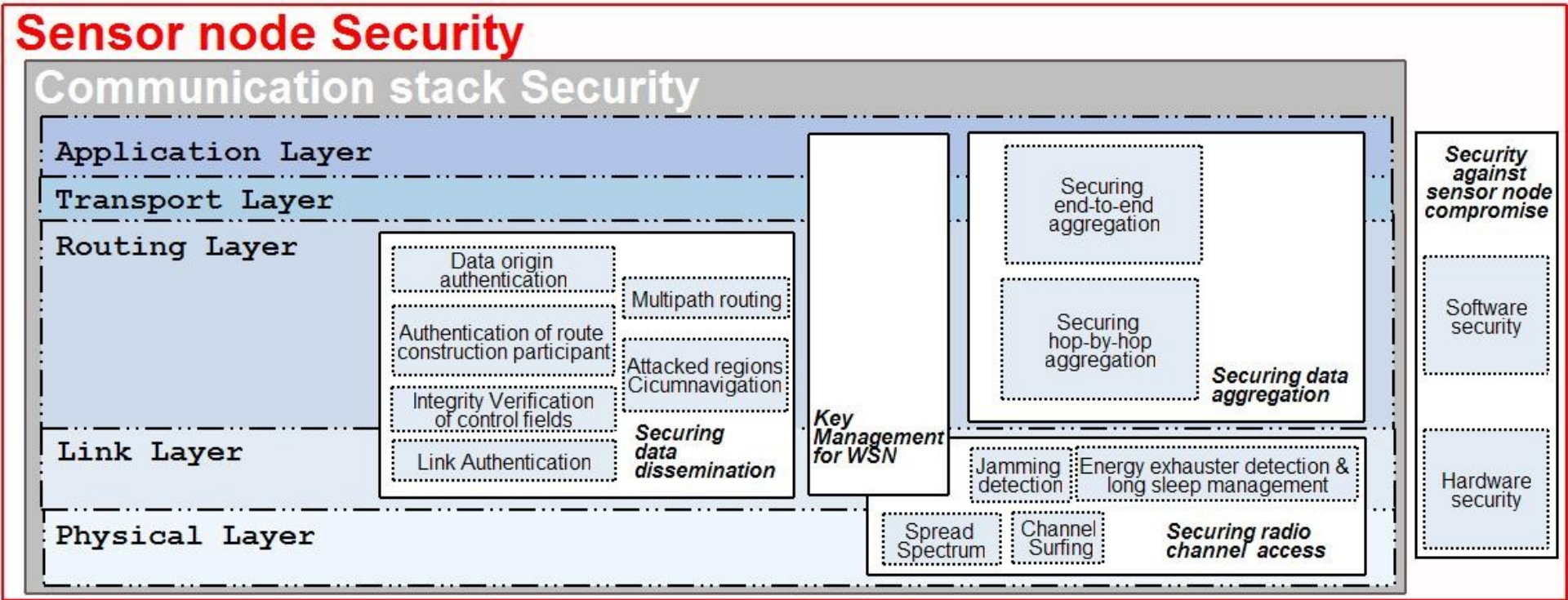
Sécurité des échanges dans RCSF

- **Pourquoi étudier la sécurité ?**
  - RCSF sont utilisés pour des applications vitales et cruciales
  - Fragilité intrinsèque des RCSF
- **Sécurité et fiabilité sont des challenges pour réussir les applications des RCSF**

Besoins en sécurité




WSN Features
WSN Characteristics
Wireless Communication
Resource limitation
Close physical coupling with environement
Hop by hop data relay
Data aggregation
Security Issues
Jamming attacks
Attacks against route construction
Attacks against payload data
Energy exhaustion attacks
Attacks against data aggregation
Nodes failure
Nodes compromise
Security Solutions


Sensor node Security
Communication stack Security
Application Layer
Transport Layer
Routing Layer
Link Layer
Physical Layer
Data origin authentication
Multipath routing
Authentication of route construction participant
Attacked regions Cicumnavigation
Integrity Verification of control fields
Securing data dissemination
Link Authentication
Key Management for WSN
Securing end-to-end aggregation
Securing hop-by-hop aggregation
Securing data aggregation
Jamming detection
Energy exhauster detection & long sleep management
Spread Spectrum
Channel Surfing
Securing radio channel access
Security against sensor node compromise
Software security
Hardware security

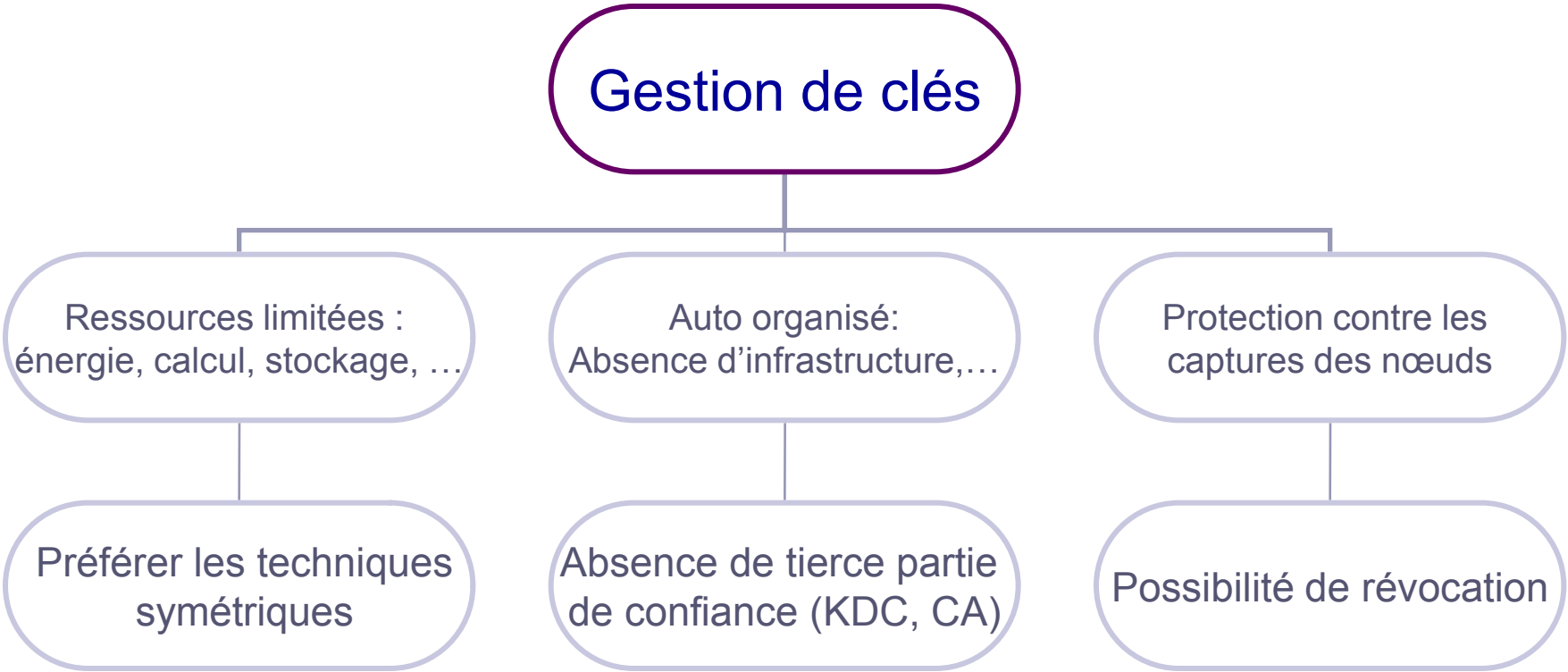
GESTION DE CLÉS DANS RCSF

Pourquoi la gestion de clés dans les RCSF

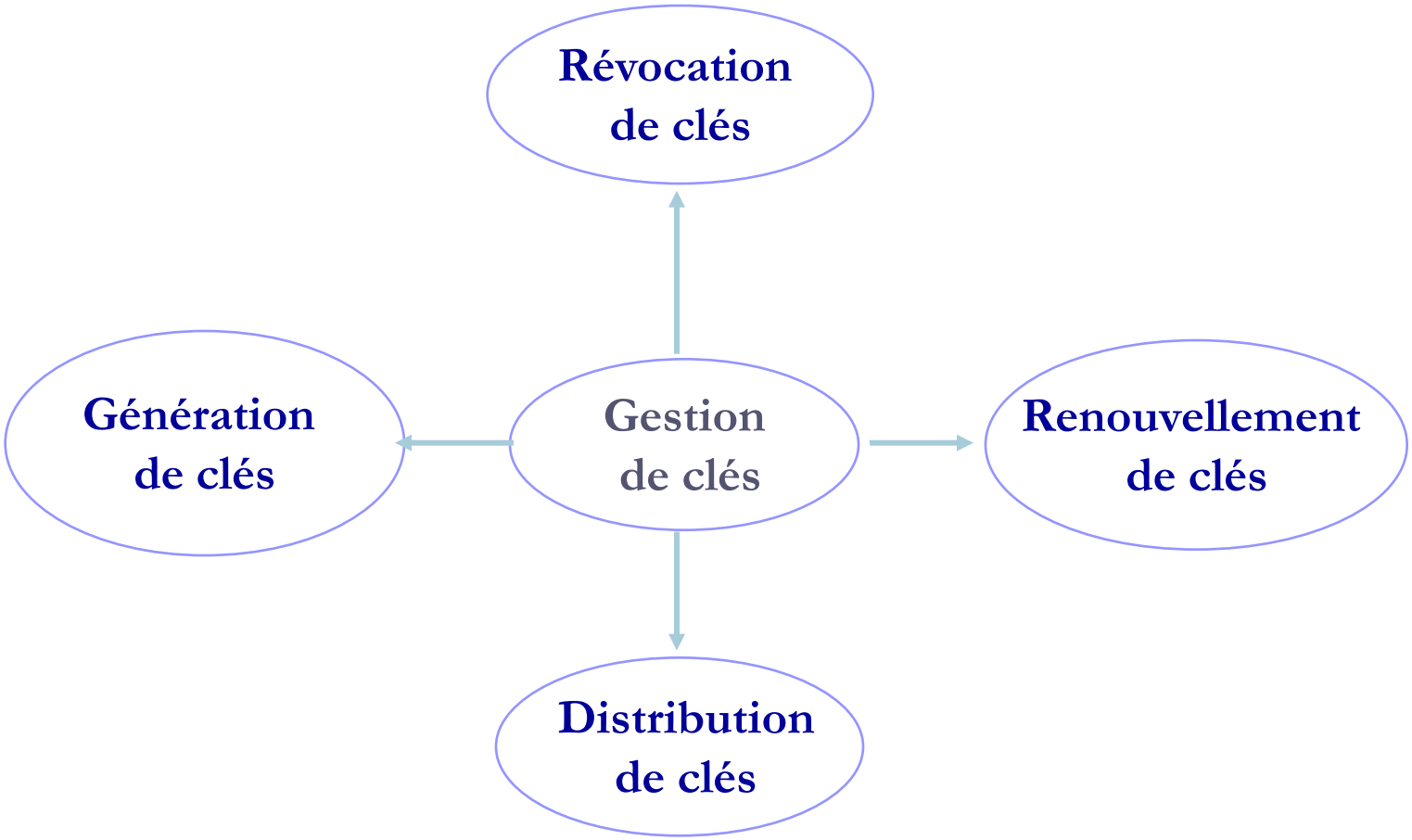
- **Après le déploiement, les capteurs ont besoin d'établir des clés cryptographiques avec leurs voisins pour assurer des services de sécurité**

- **Applications**
  - Sécuriser l'agrégation
  - Sécuriser le routage
  - Coopération (authentification)
  - …

Gestion de clés



Gestion de clés

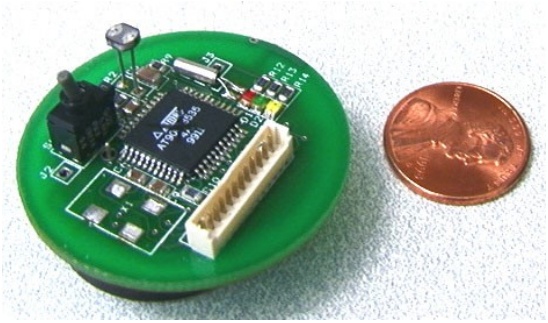


Gestion de clés (1/2)

- **Pourquoi la gestion de clés?**
  - Etablissement de clés entre tous les nœuds capteurs pour l'échange sécurisé des données
  - Fonctionner dans des environnements non défini à priori
  - Un nœud non autorisé ne peut pas effectuer des communications avec les nœuds du réseau

Gestion de clés (2/2)

- **A cause des limitations de ressources**
  - ✓ Petite mémoire
  - ✓ Faible énergie



Pré déploiement de clés

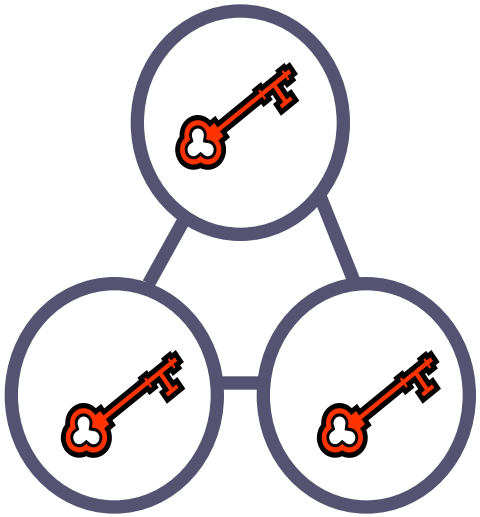
Quelques solutions naïve (1/2)

- Une seule clé partagée par le réseau

Tous les capteurs utilisent une seule et même clé pour la communication

☺ **Utilisation efficace de mémoire car le capteur a besoin de sauvegarder uniquement une seule clé**

☹ **Le compromis d'un seul nœud compromet le réseau entier**

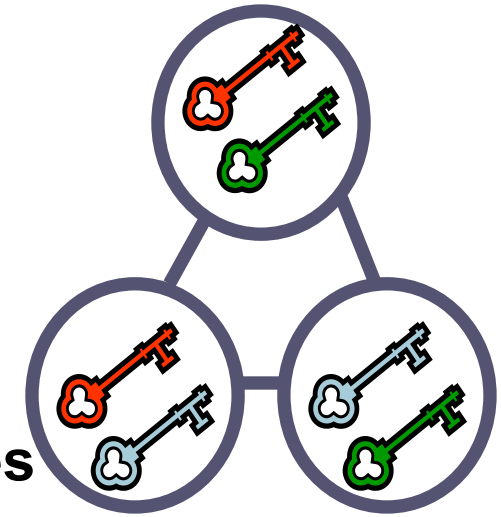


Quelques solutions naïve (2/2)

- Des clés partagées par paire de nœuds

Chaque nœud aurait une clé correspondant à chaque autre nœud dans le réseau.

Par conséquent si le réseau est de taille N, un capteur devra stocker (N-1) clés dans sa mémoire

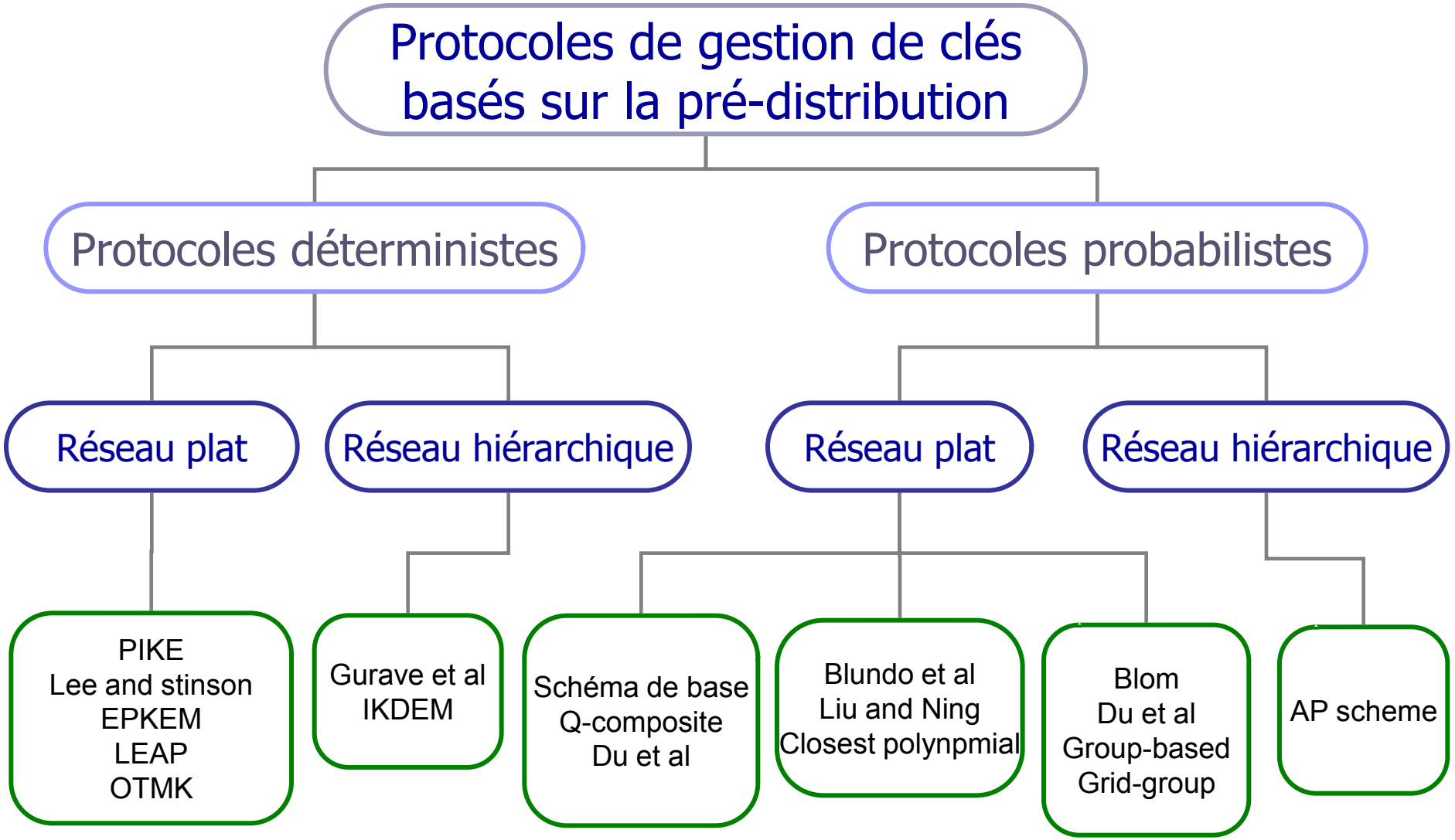


- ☺ **Convenable pour des petits réseaux et limité à la capacité mémoire du nœud capteur**
- ☹ **Non scalable, l'ajout et la suppression des nœuds pose problème**

Compromis

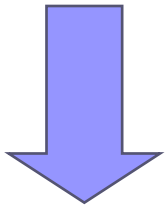
- **Comment peut-on avoir un équilibre entre ces deux solutions ?**

Taxonomie


Protocoles de gestion de clés basés sur la pré-distribution
Protocoles déterministes
Protocoles probabilistes
Réseau plat
Réseau hiérarchique
Réseau plat
Réseau hiérarchique
PIKE
Lee and stinson
EPKEM
LEAP
OTMK
Gurave et al
IKDEM
Schéma de base
Q-composite
Du et al
Blundo et al
Liu and Ning
Closest polynpmial
Blom
Du et al
Group-based
Grid-group
AP scheme

Protocoles déterministes

- **Assurent que chaque nœud est capable d'établir une clé avec ses voisins**
- **Une seule clé est déployée dans chaque capteur avant le déploiement**
- **Cette clé sera utilisée pour établir des clés par-paires entre des nœuds voisins**



- LEAP [Zhu et al. 2003]

LEAP [Zhu, 2003]

- **Protocole basé sur une clé initiale transitoire KIN**
  - Les auteurs de LEAP se basent sur l'hypothèse:
    - ✓ Pour compromettre un nœud , l'adversaire nécessite un temps minimal Tmin
      - C'est le temps de brancher un câble série.
      - Et le temps de copier le contenu de la mémoire du nœud compromis.
  - LEAP exploite ce temps (de confiance) pour permettre à deux nœuds voisins d'établir, d'une manière sécurisée, une clé symétrique entre eux.
  - Après Tmin la clé KIN est supprimée de la mémoire du nœud

LEAP

- **Chargement de la clé initiale**
  - Le contrôleur (SB) génère une clé initiale K_{IN} et charge chaque noeud avec cette clé
  - Chaque noeud u dérive une clé principale $K_u = f_{K_{IN}}(u)$.
- **Découverte des voisins**
  - Diffusion HELLO + lancement Timer Tmin
  - Réception ACK de tous les voisins, authentifiés avec un MAC

$$u \rightarrow * \ : u$$
$$v \rightarrow u \ : v, MAC(K_v, u|v)$$

- **Etablissement de la clé commune**

$$K_{uv} = f_{K_v}(u).$$

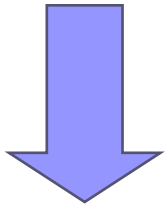
- **Effacement de clés**
  - Lorsque le Timer est expiré après Tmin, le noeud u efface Kin et toutes les clés principales (Kv) de ses voisins.

Discussion

- **Les protocoles déterministes assurent une connectivité totale (100%)**
- **Si un adversaire compromet un nœud après T_{min}:**
  - Seules les clés du nœud compromis sont découvertes
- **Si K_{IN} est compromise (avant T_{min}):**
  - Toutes les clés sont découvertes dans **LEAP**

Protocoles probabilistes

- Des chaînes de clés sont aléatoirement choisies parmi un ensemble de clés et distribuées aux réseaux de capteurs



  - Random Key Chain Pre-distribution [Eschenauer and Gligor 2002]
  - Random Pair-wise Key Scheme [Chan et al. 2003]

Schéma de Eschenauer-Gligor

- **Ce schéma est basé sur la clé probablement partagée entre les nœuds d'un graphe aléatoire**

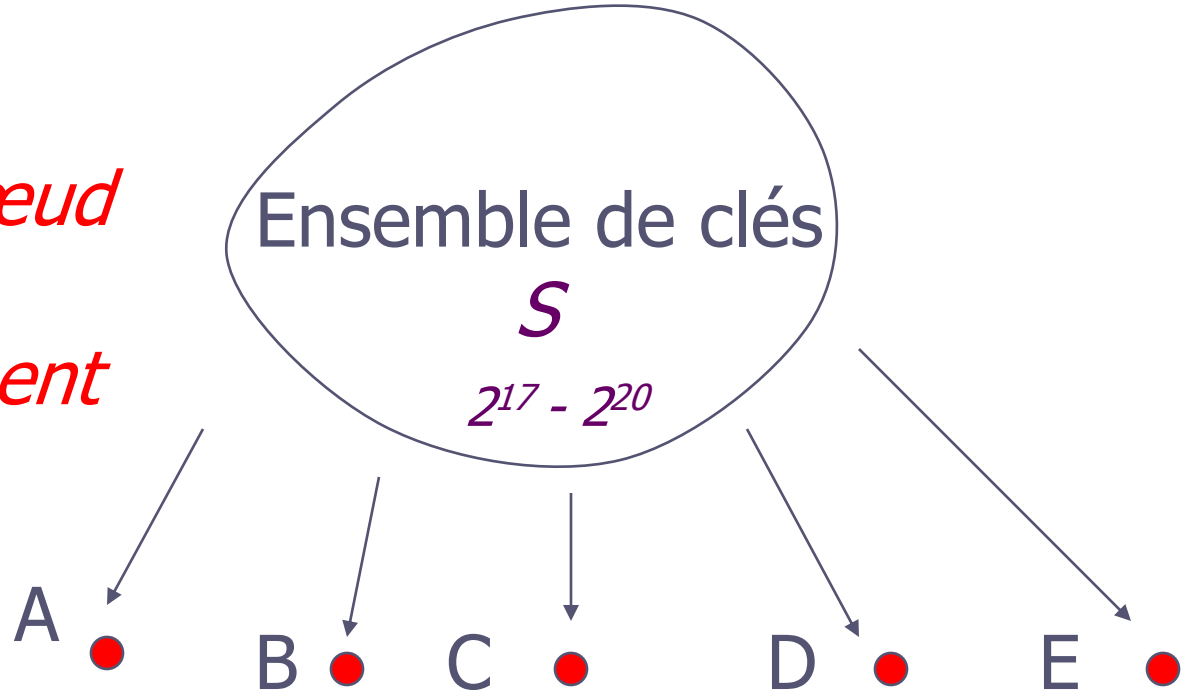
Il implique trois phases:

- Pré-distribution de clés
- Découverte de la clé partagée
- Etablissement de chemin de clé

Il prévoit également la révocation des clés

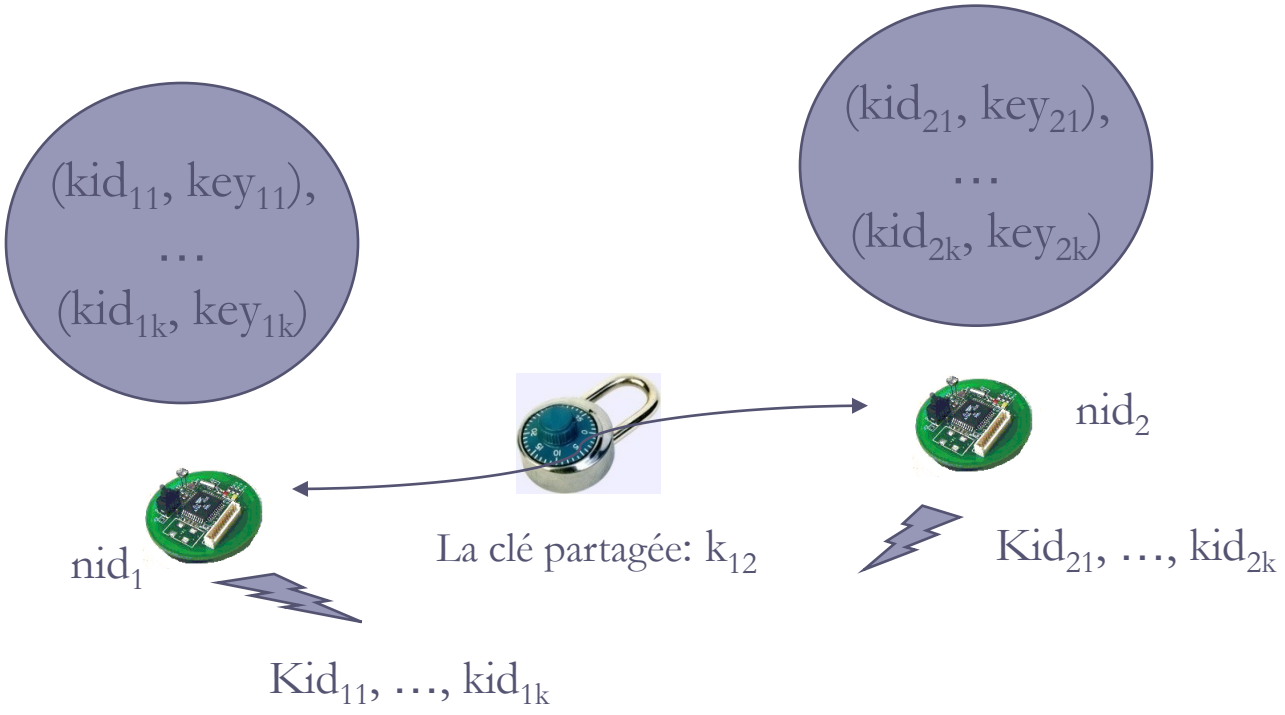
Schéma de Eschenauer-Gligor

Chaque nœud choisit aléatoirement m clés

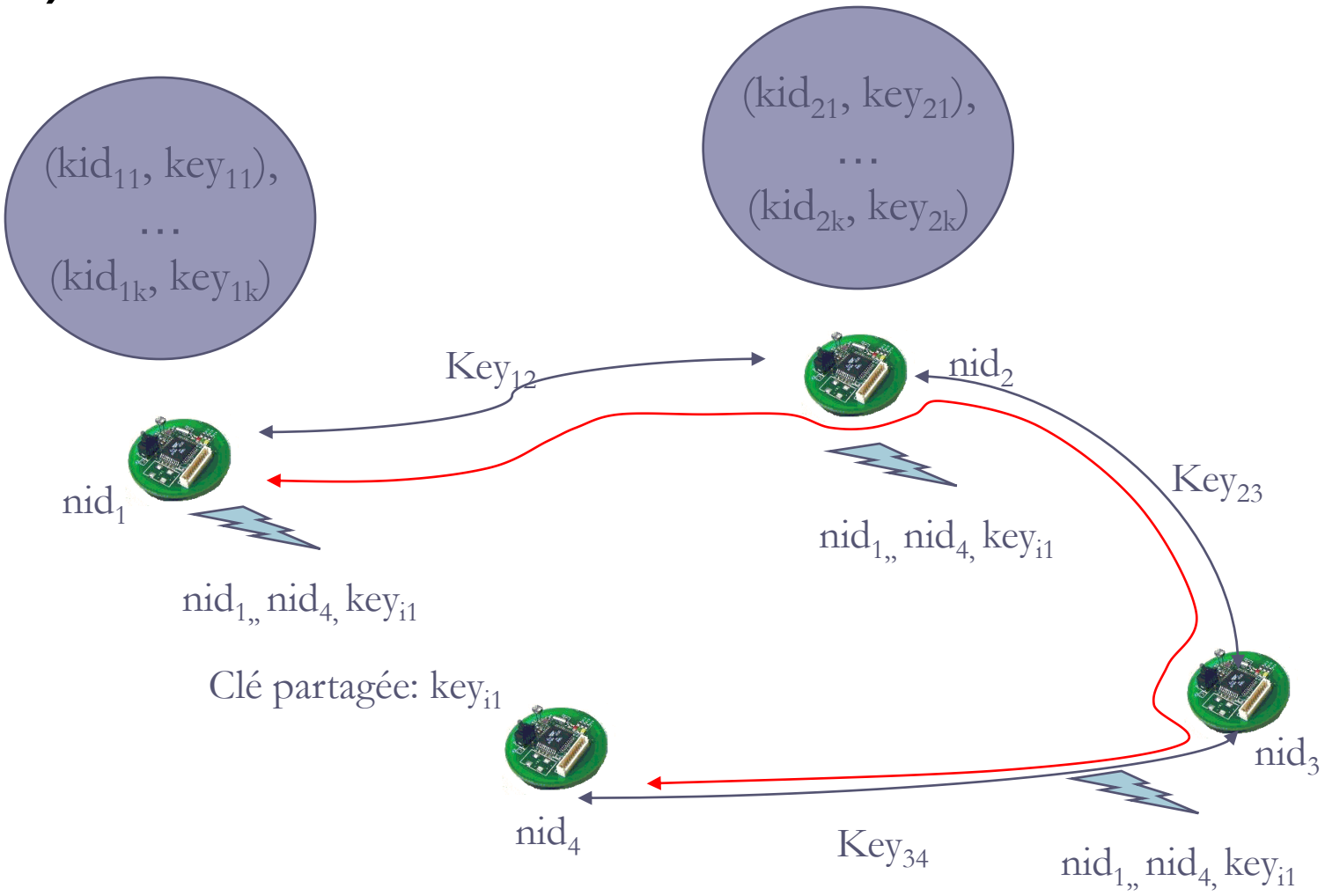


- Lorsque |S| = 10,000, m=75
  Pr (deux nœuds ont une clé commune) = 0.50

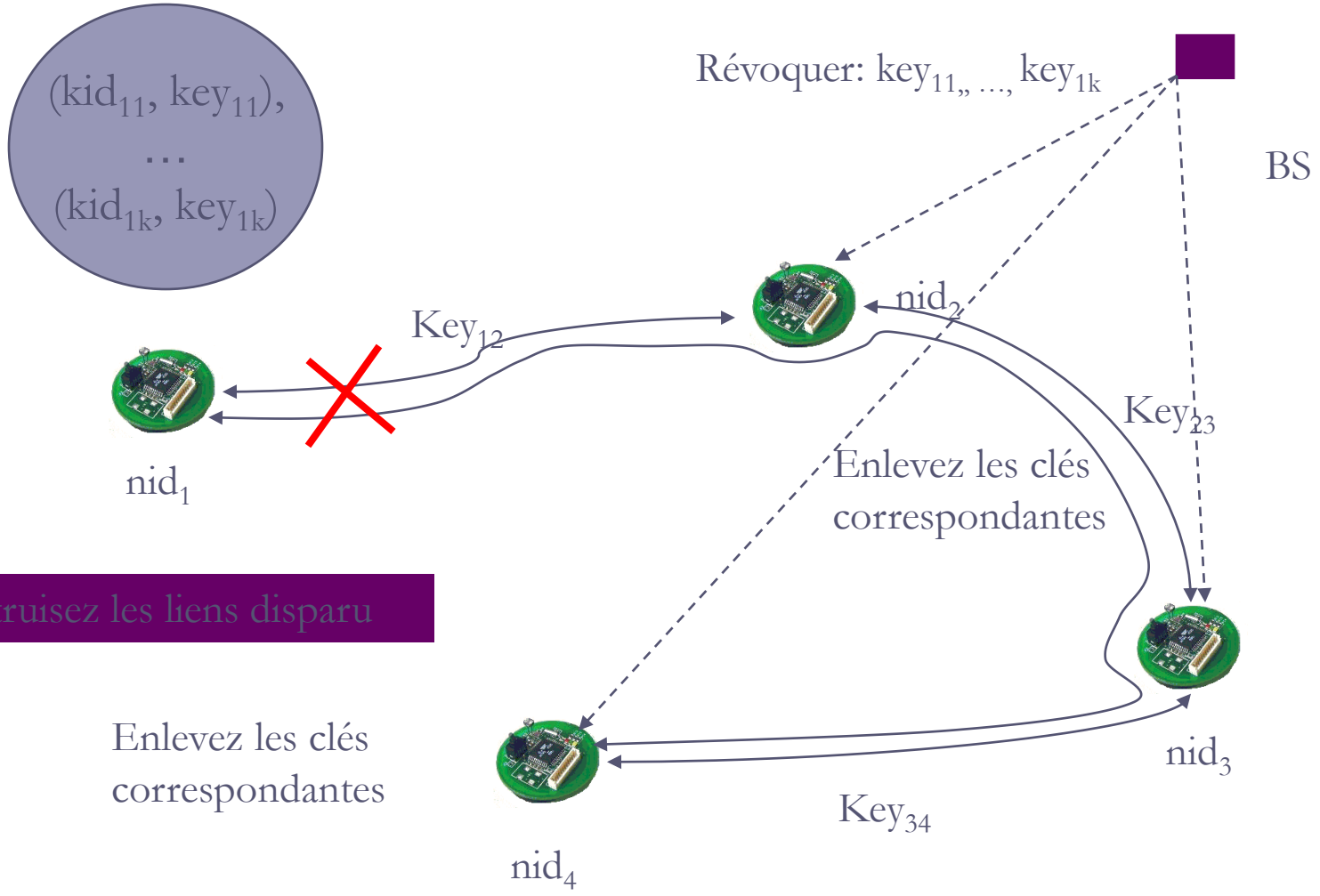
(2) Découverte de la clé partagée



(3) Etablissement de chemin de clé

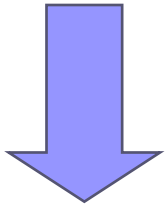


(4) Révocation de clés



Résilience à la capture de nœud

- **Plus robuste que l'approche d'une seule clé partagé par le réseau**
- **Au cas où un nœud serait capturé m<<n clés sont compromises**



- Ceci signifie que l'attaquant a une faible probabilité d'attaquer avec succès n'importe quel autre lien du réseau

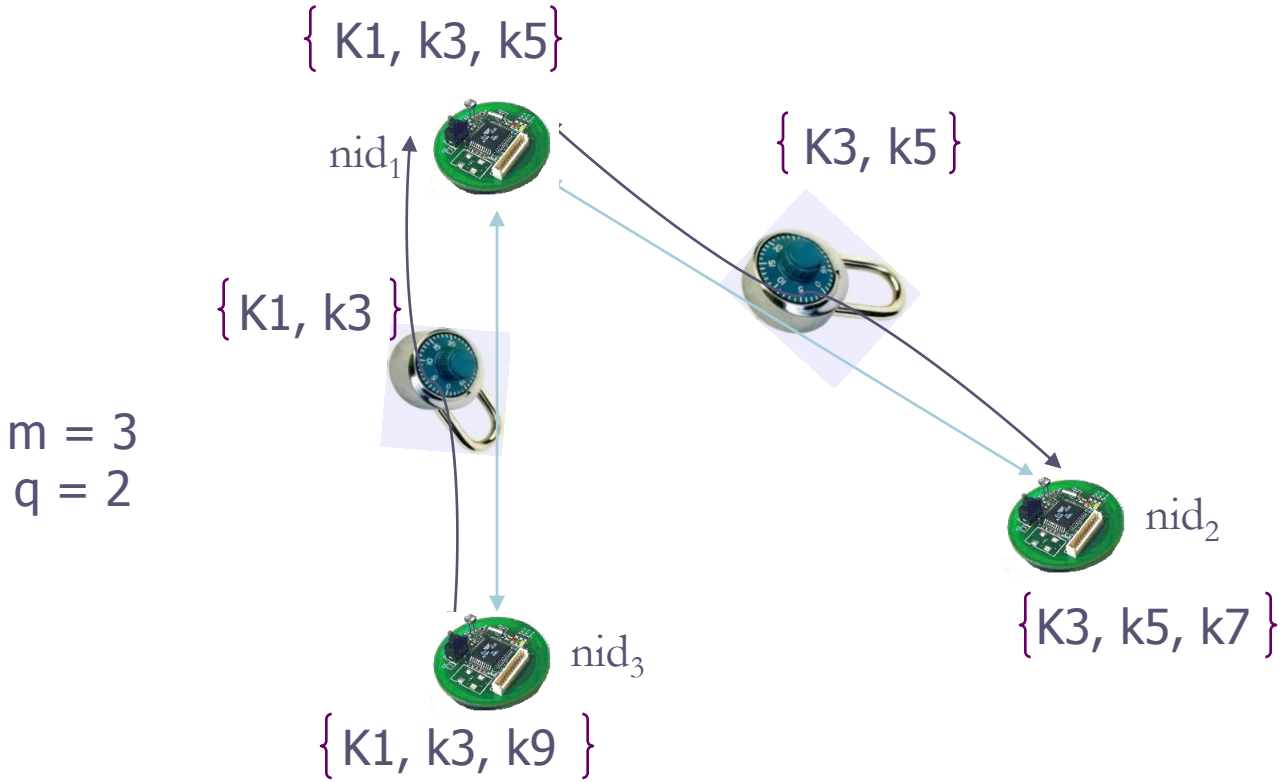
Schéma de Chan et al

- **Ce schéma est identique à celui de Eschenaur et Gligor sauf qu'à la place d'une clé partagée étant exigée pour la communication, une paire de nœud doit partager q clés avec q > 1 pour établir un lien sécurisé.**

- La nouvelle clé utilisée pour la communication entre deux nœuds quelconques qui partage q' clés (q'≥q) :

$$K = hash (k1 \,||\, k2 \,||\, \cdots \,||\, kq')$$

q-composite Clés (exemple)



Discussion

- **Le défi de l'échange probabiliste dans le schéma de pré-distribution de clés est de trouver un bon compromis entre la taille de P et la valeur m du nombre de clés par nœud pour une meilleure connectivité et une meilleure résilience**

  - Une grande |P| → réduit la connectivité
  - Une petite |P| → diminue la résilience
  - Un grand réseau → |P| et m >> → + Stockage
  - + Nb de nœuds compromis >> + la fraction des paires de clés affectées >>

SÉCURITÉ DU ROUTAGE

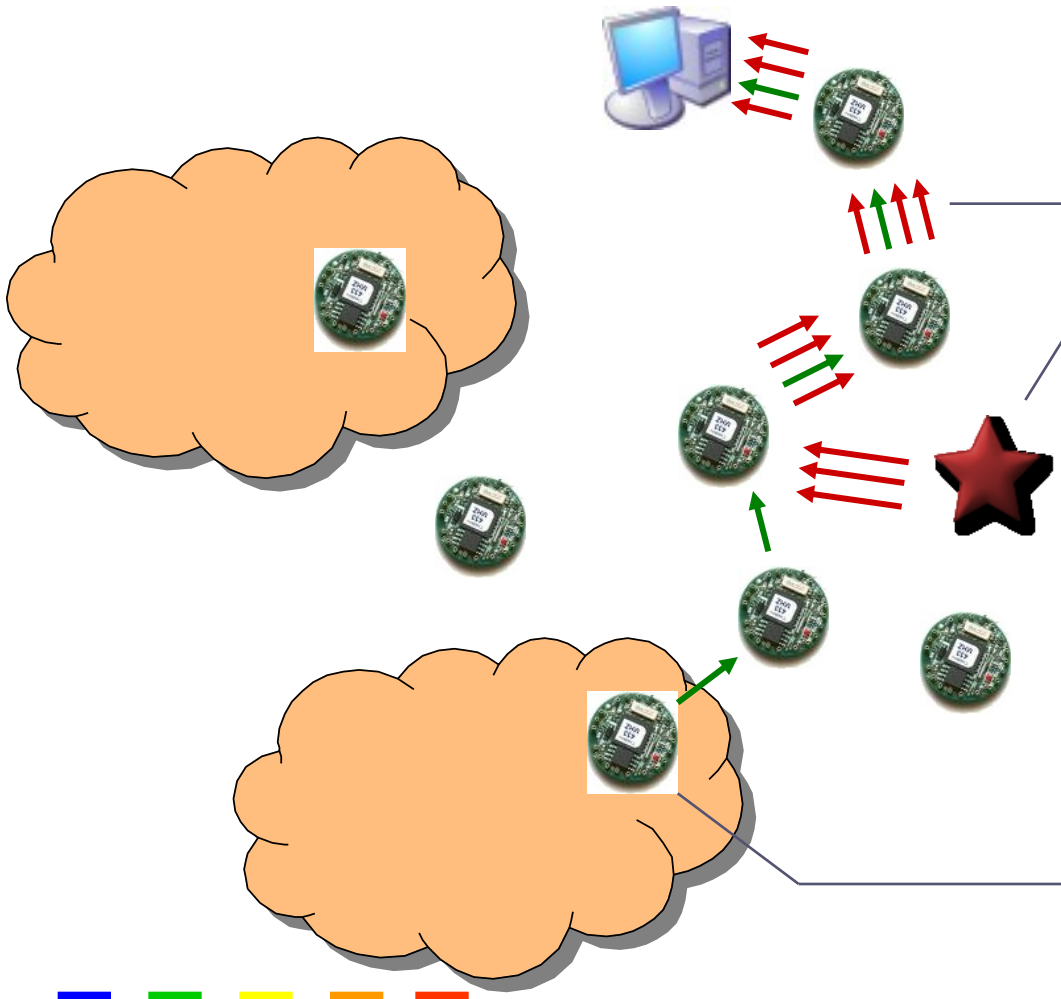
Vulnérabilités du routage

- **La plus part des protocoles supposent un environnement « honnête »**
  - Pas d'adversaire
- **Un adversaire peut donc bloquer la fonction de routage**

Types d'attaques

- **Déni de Service**
  - Energy Exauhstion
  - Blackhole
  - Routing Table Poisoning
- **Eavesdropping**
- **Toolbox**
  - Sinkhole, wormhole, …

Exemple 1 : PDoS



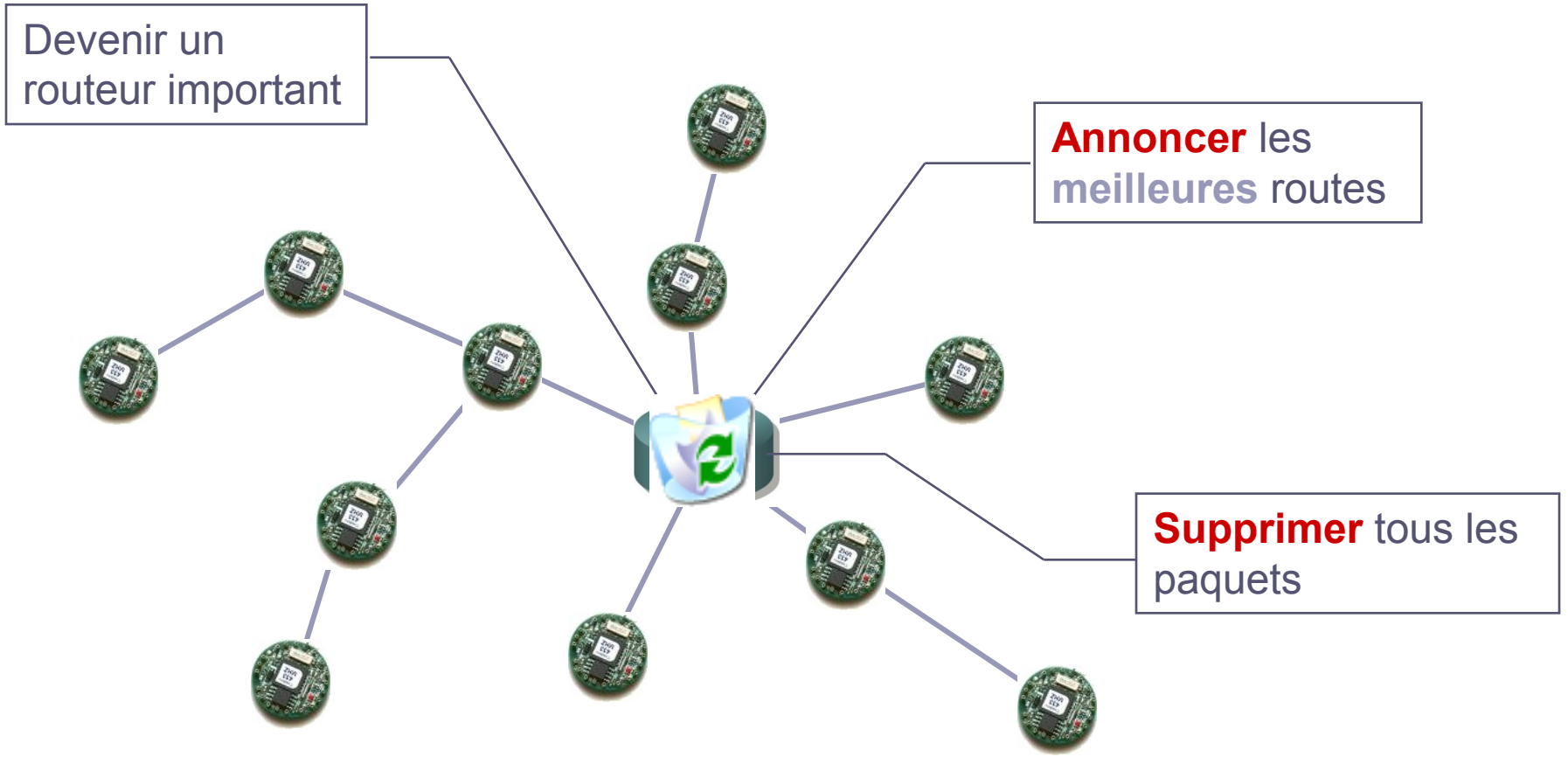
Communications normales

L'intrus émet beaucoup de paquets

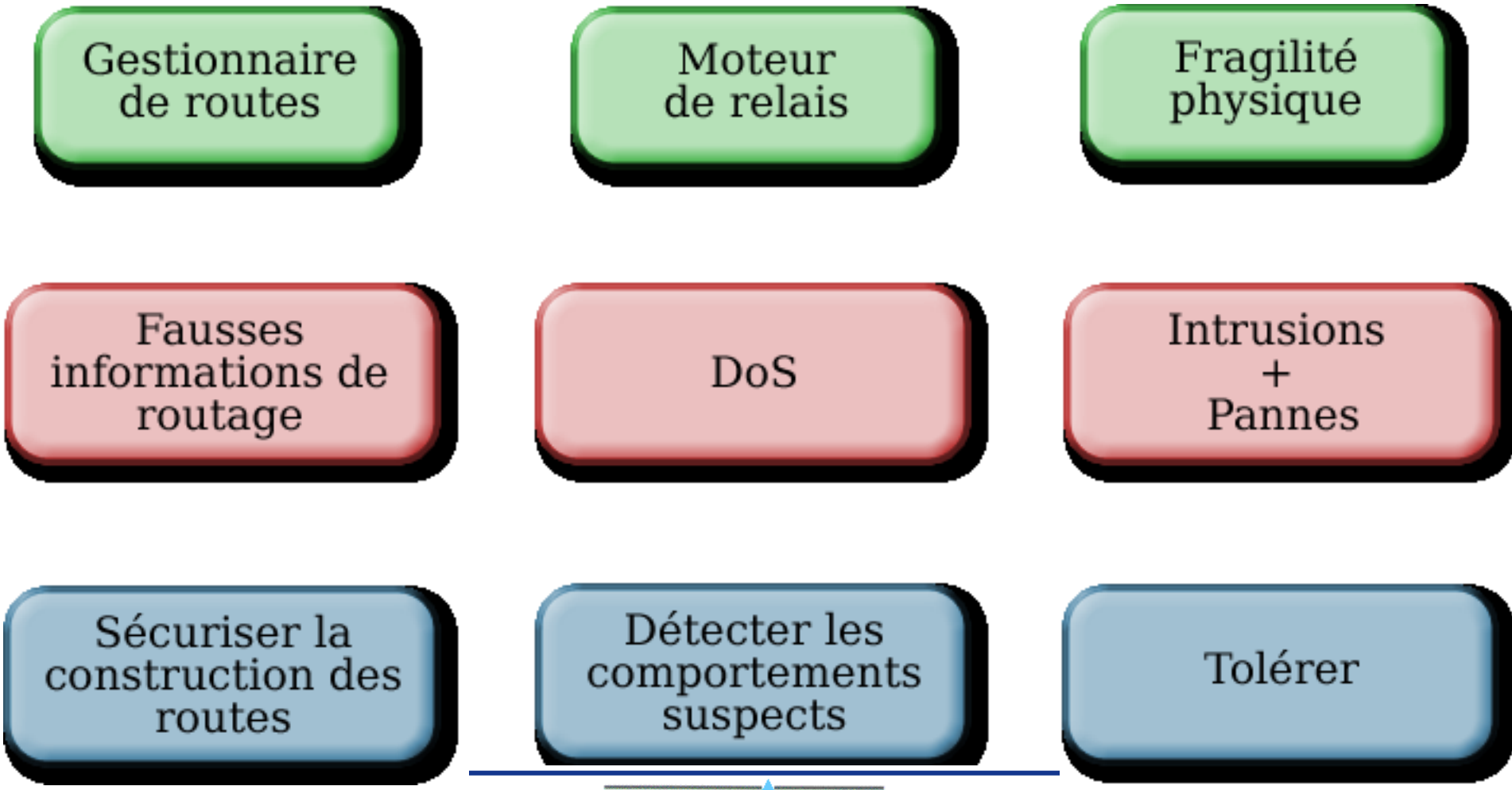
Les paquets épuiseront l'énergie des capteurs et/ou brouilleront le signal

Zone d'agrégation

Exemple 2 : Blackhole



Solutions


Gestionnaire de routes
Moteur de relais
Fragilité physique
Fausses informations de routage
DoS
Intrusions + Pannes
Sécuriser la construction des routes
Détecter les comportements suspects
Tolérer

Sécuriser la construction des routes

- **Comment construire les chemins ?**
- **Approche centralisée**
  - Routes calculées par la SB
  - **INSENS**
- **Approche distribuée**
  - Chaque capteur construit sa table
  - **SecRout**

INSENS [Deng, 2006]

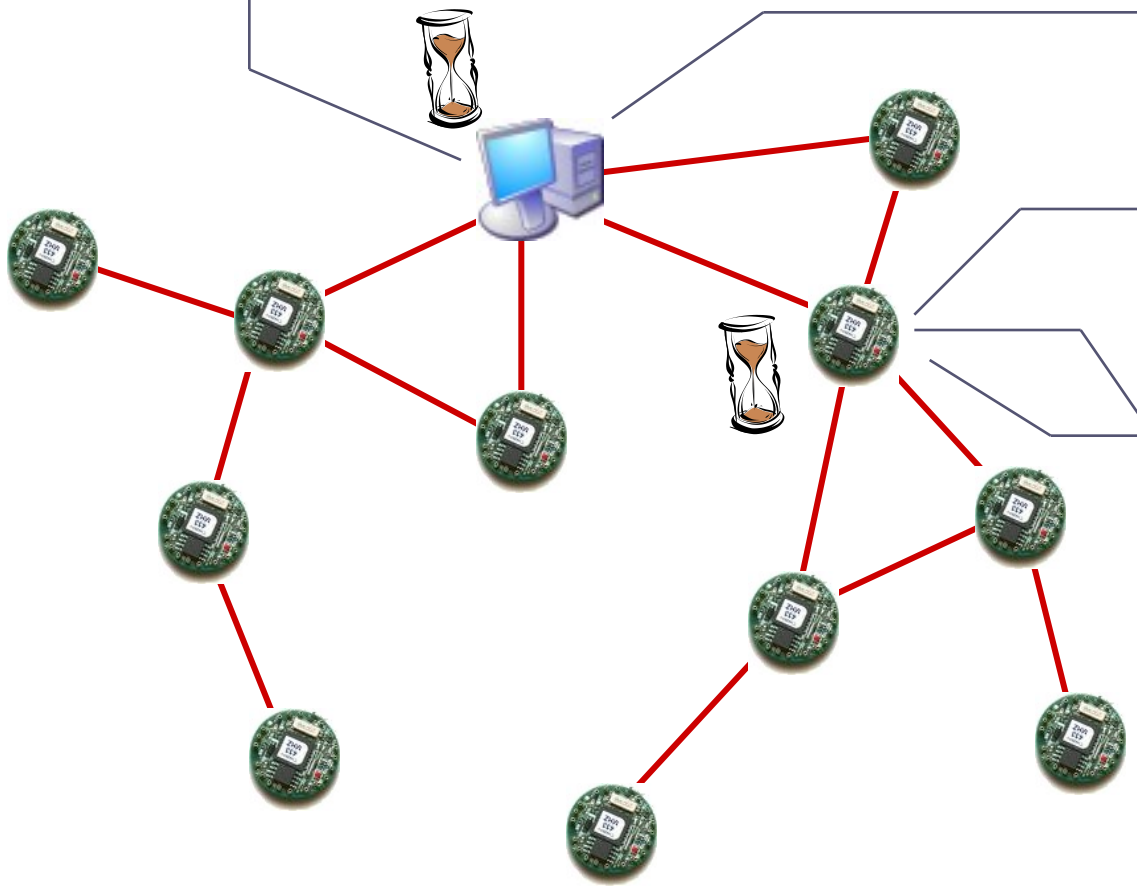
- **Assure l'authentification**
- **SB partage une clé avec chaque capteur**
- **SB récupère les états de lien de chaque capteur**
  - Chaque information contient une **preuve**
- **Compare les preuves**
- **Établie une cartographie correcte**
- **Construit 2 routes indépendantes pour chaque capteur**
- **Transmet les tables de routage**

Fonctionnement

Deux routes pour chaque capteur

Construction du graphe **correct**

Requête authentifiée



Émission d'un MAC + construction **arbre**

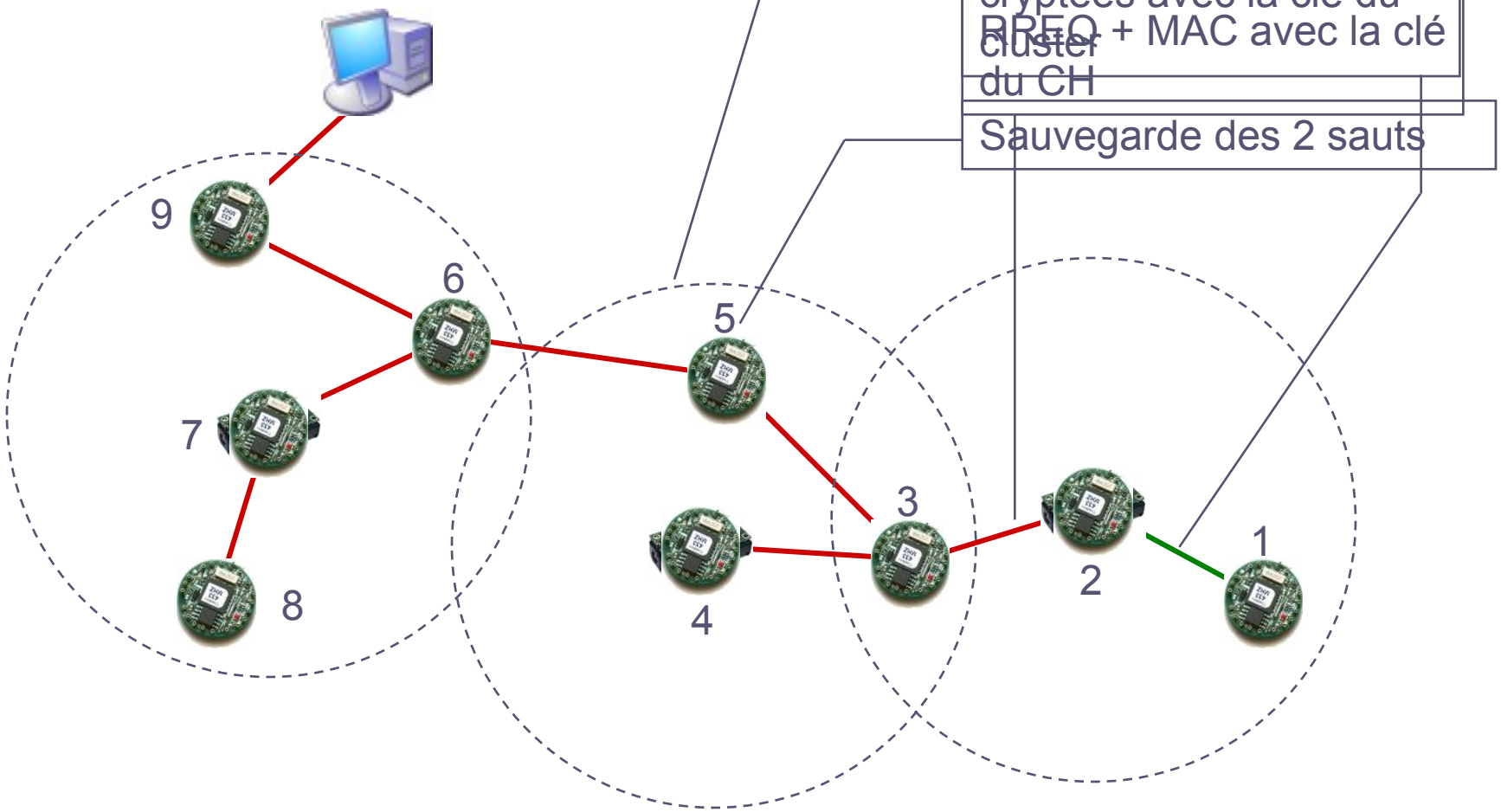
Récolte des MAC des voisins

Émission du feedback via le **père**

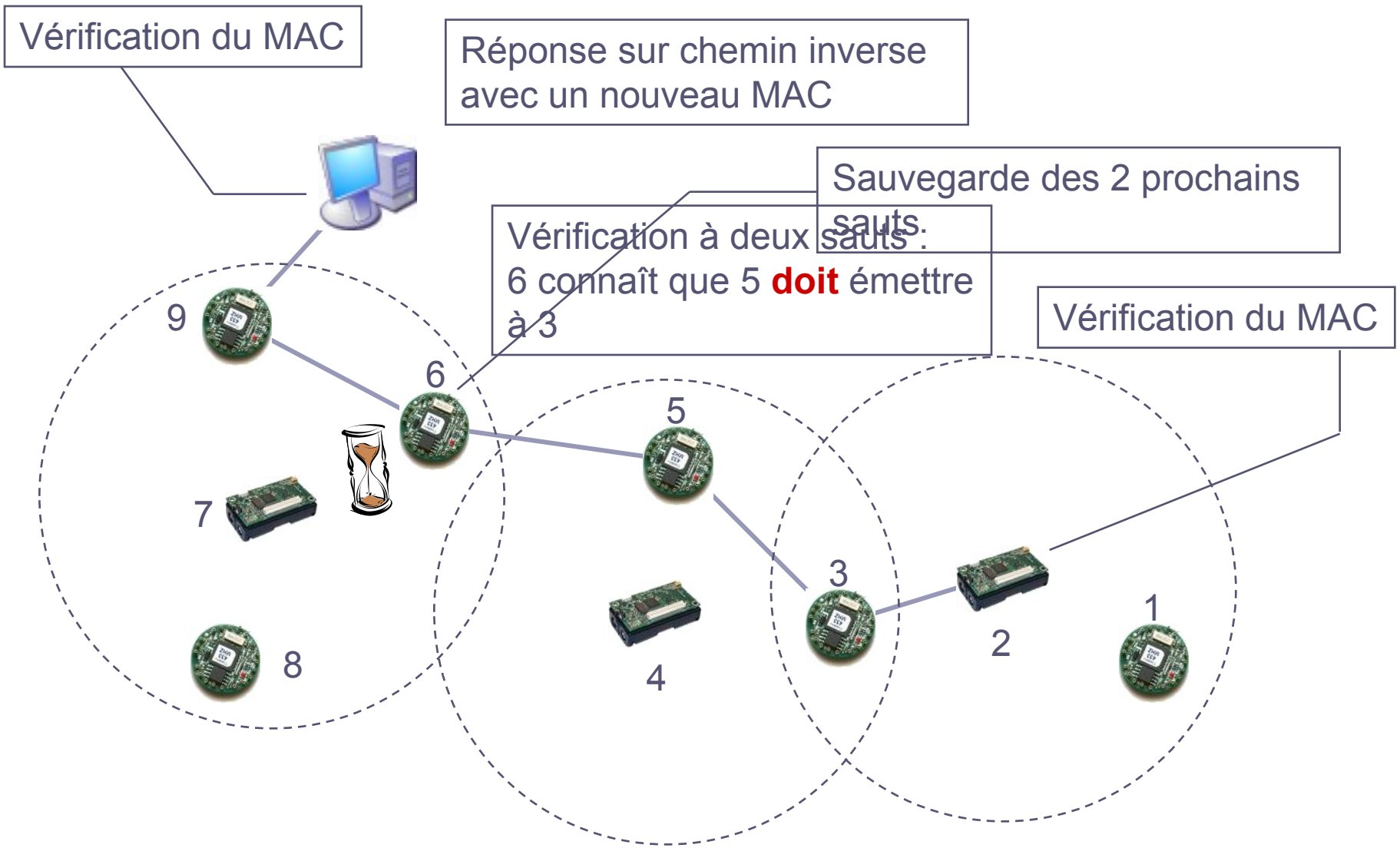
SecRout [Yin, 2006]

- **Assure l'authentification et la confidentialité**
- **Construire les routes d'une manière distribuée**
- **Deux niveaux**
  - Membres communiquent au cluster head (CH) les données
  - CH établit une route vers SB
- **Emploie une vérification à deux sauts**

Fonctionnement

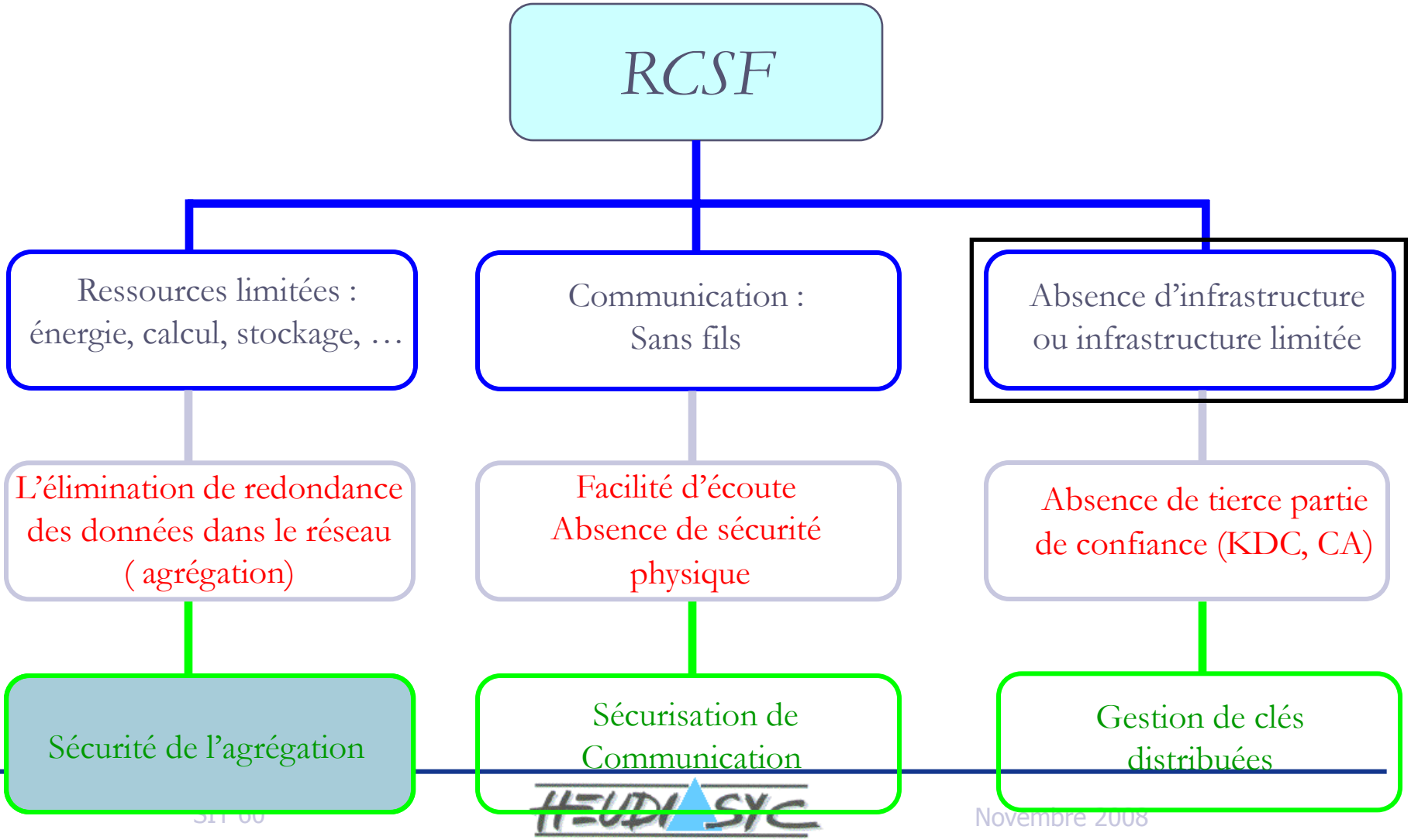


Fonctionnement



SÉCURITÉ DE L'AGRÉGATION

Besoins en sécurité



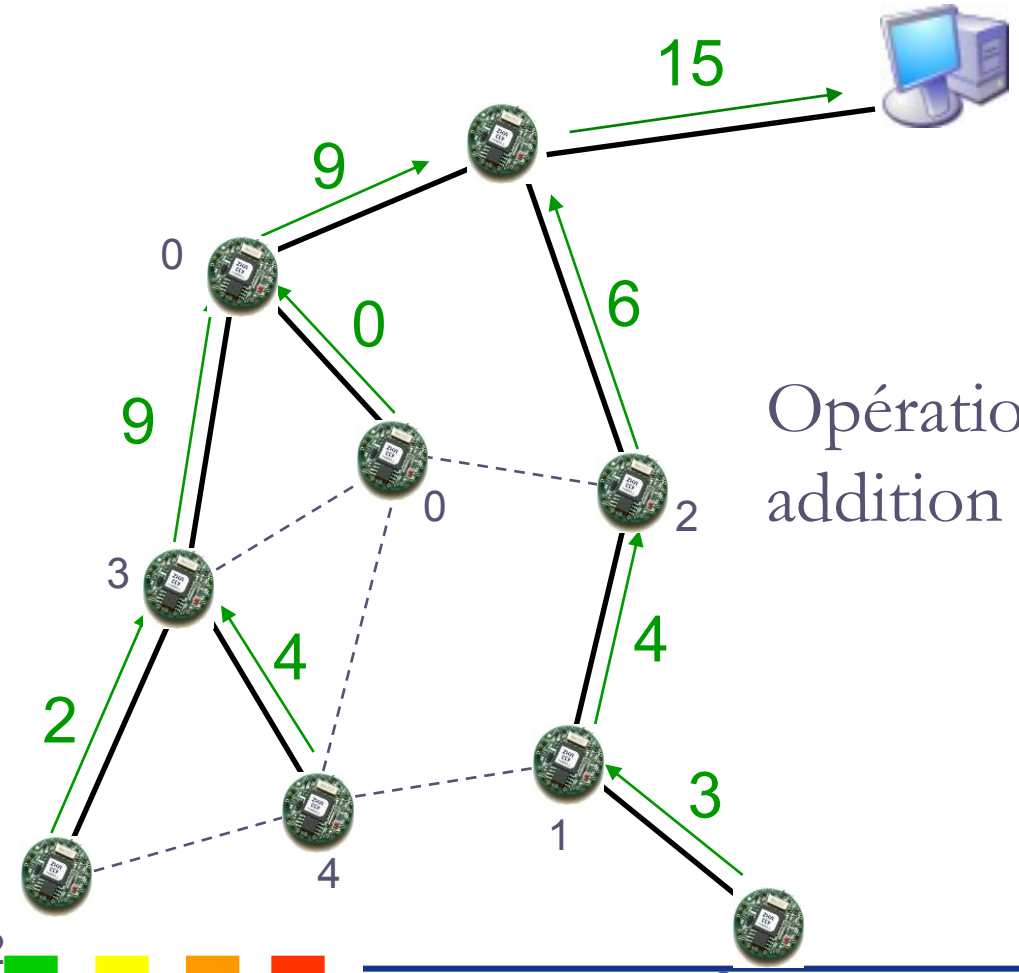
Problématique

- **Sauvegarder le gain de l'agrégation en terme énergétique**
  - Utilisation de cryptage.
  - Plus de messages de contrôle sont émis.
- **Les techniques traditionnelles de sécurité ne sont pas adéquates pour l'agrégation.**
- **L'utilisation de cryptage à clé publique n'est pas adéquat pour les RCSFs**
  - Consomme beaucoup d'énergie.
  - Temps d'exécution est important
-

Cohabitation entre sécurité et agrégation

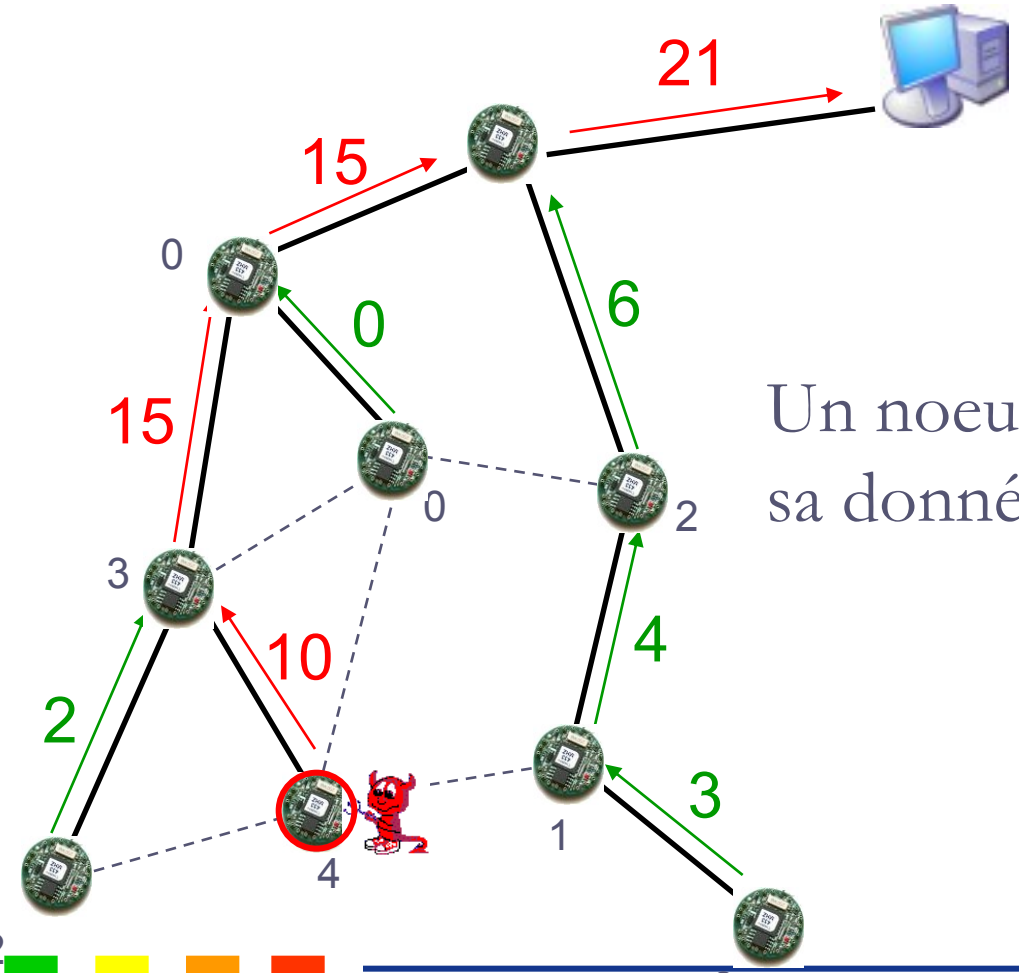
- **L'agrégation de données est vulnérable par rapport aux attaquants qui injectent de fausses données ou falsifient les valeurs agrégées.**
  - Chaque nœud peut manipuler les données des autres nœuds.
  - Un seul nœud compromis peut falsifier toutes les données d'une zone spécifique.

La sécurité de l'agrégation



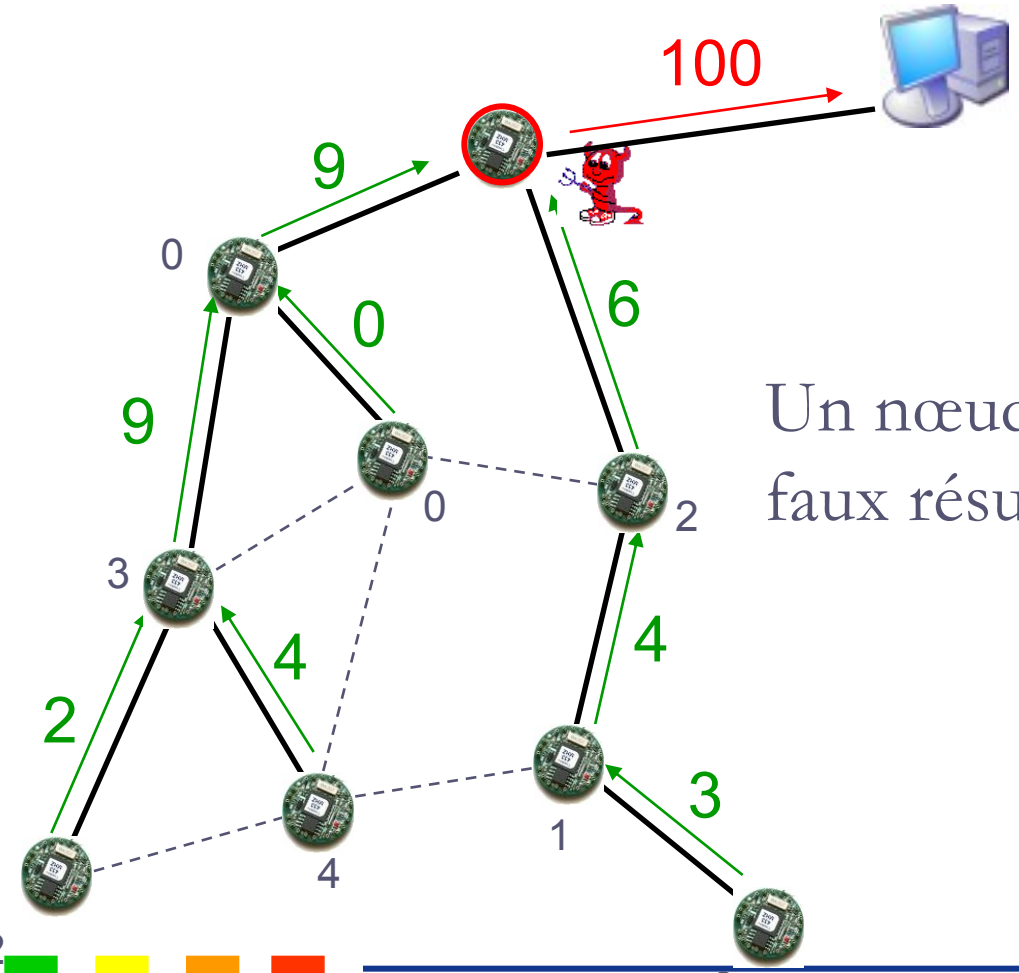
Opération d'agrégation= addition

La sécurité de l'agrégation



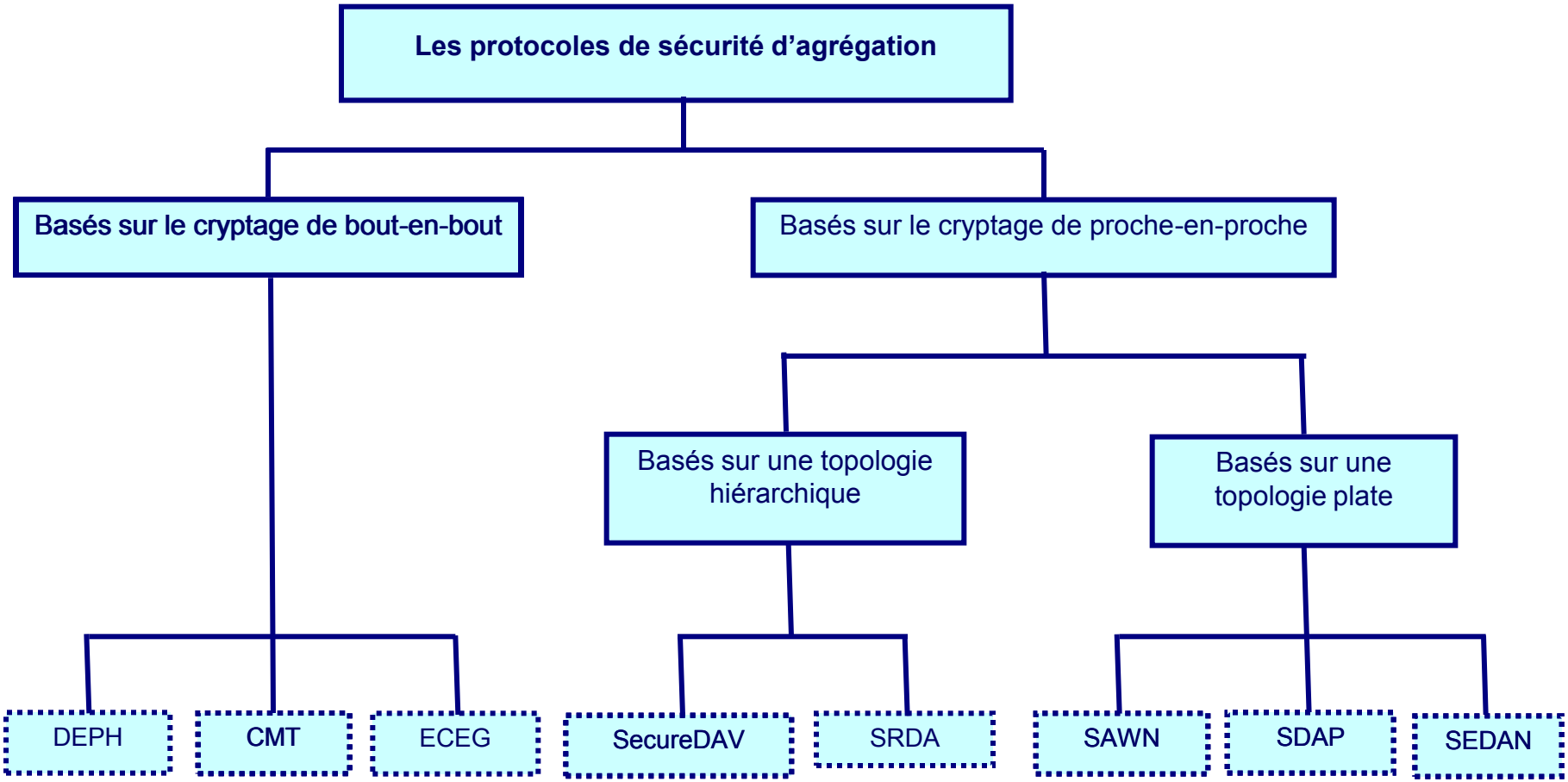
Un noeud malicieux envoi sa donnée erronée

La sécurité de l'agrégation



Un nœud malicieux envoi un faux résultat d'agrégation

Les protocoles de sécurité d'agrégation



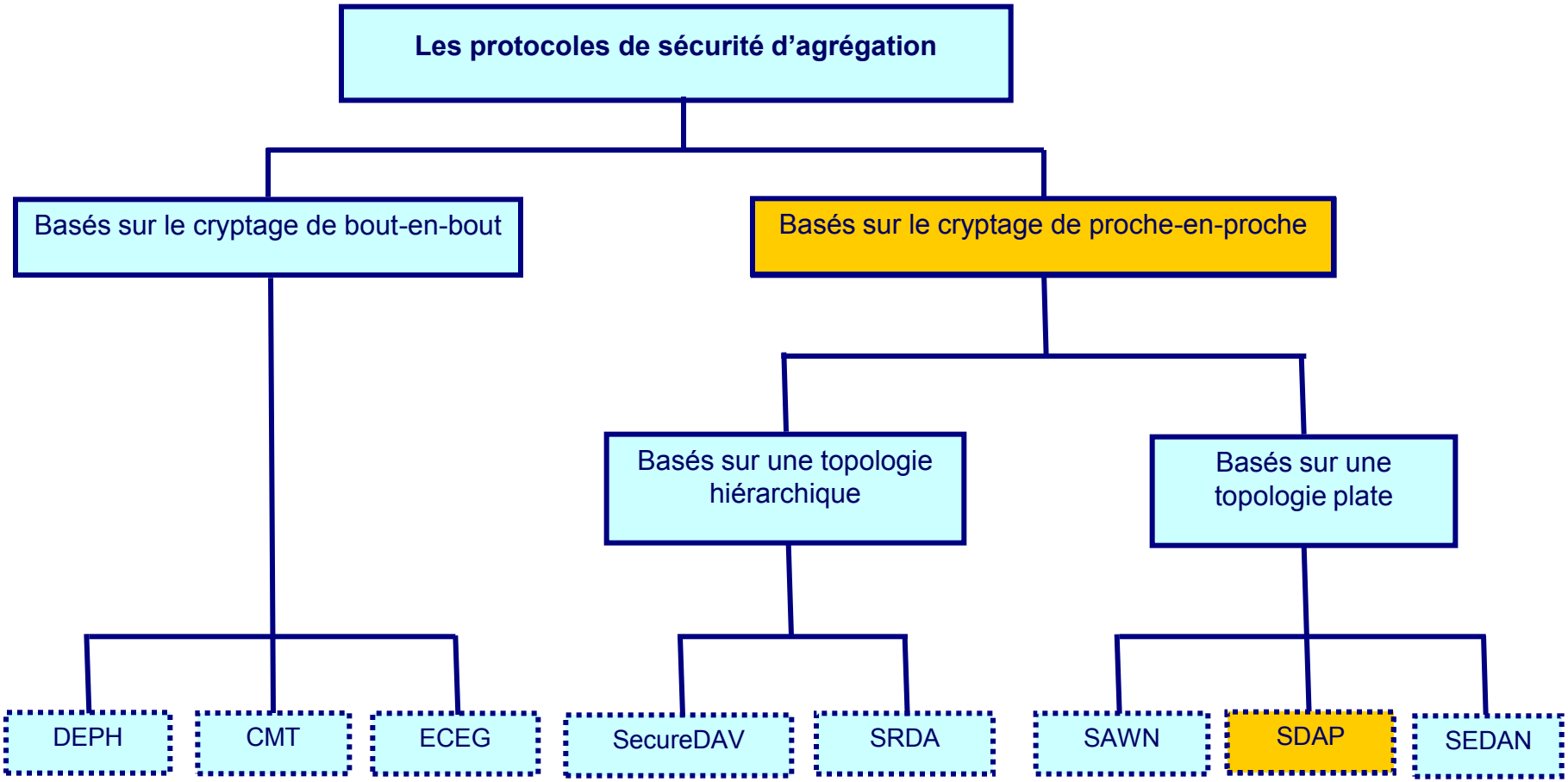
Les protocoles du cryptage bout-en-bout

- Utilisent le cryptage homomorphisme :
  - ✓ E(x+y) = E(x) + E(y).
- Caractérisés par :
  - ✓ Les nœuds intermédiaires ne peuvent pas manipuler les données claires.
    - Les nœuds feuilles envoient leurs données cryptées par des clés partagés entre ces nœuds et la station de base.
    - Les nœuds relais font l'agrégation seulement sur les données cryptées.
  - ✓ À la fin la station de base décrypte le paquet finale (la valeur de l'agrégation final crypté par la somme des clés des nœuds du réseaux).

Les protocoles du cryptage bout-en-bout

- Les inconvénients de ce type de protocoles :
  - ✓ L'injection d'une fausse donnée par un nœud compromis dans le réseau, implique le rejet de toutes les données envoyées à la station de base.
  - ✓ Ce type de protocoles ne peuvent pas localiser les nœuds compromis.

Les protocoles de sécurité d'agrégation

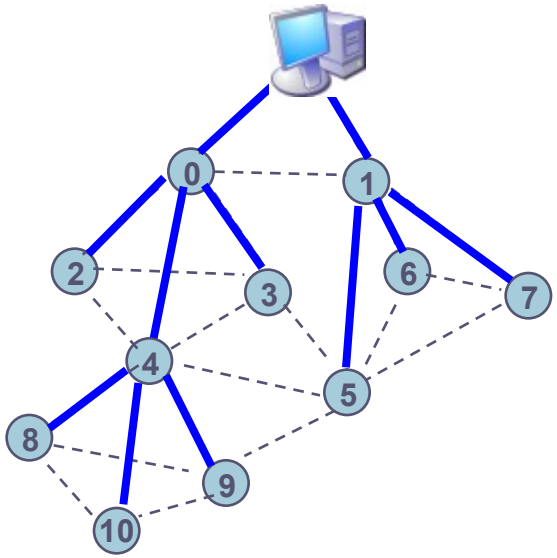


Le protocole SDAP

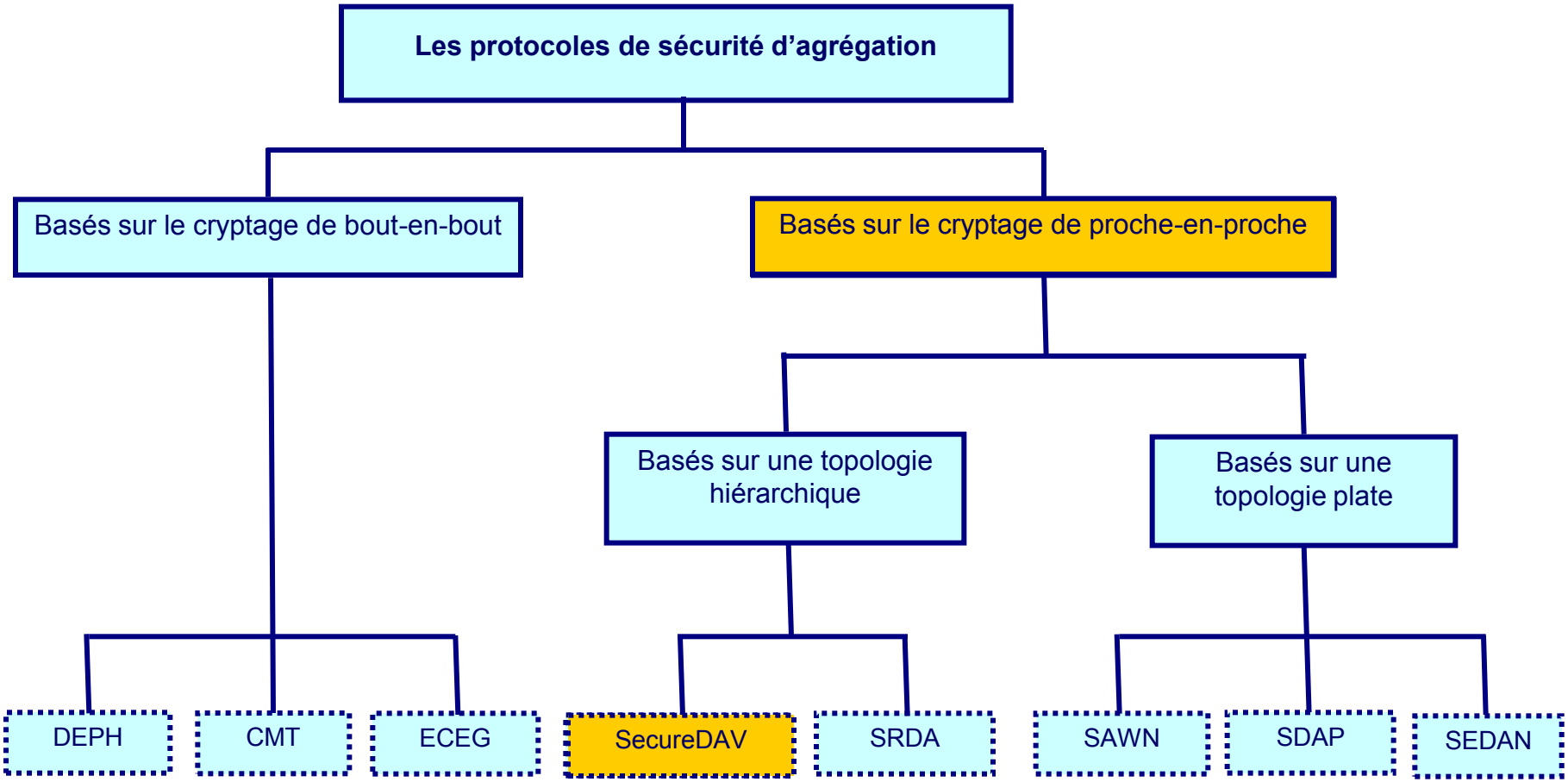
- Caractériser par :
  - ✓ L'agrégation de données est effectuée dans des cliques.
    - Chaque nœud (des fils et leur père) dans la clique écoute les messages diffusées dans cette clique.
  - ✓ Chaque nœud dans la clique calcule la valeur de l'agrégation.
    - Les fils vérifient le comportement de leur parent.
  - ✓ Les nœuds qui n'appartenant à aucune clique (éparpillés) envoient leurs données cryptés vers la station de base.

Le protocole SDAP

- La complexité d'implémentation de ces cliques
- La maintenance de telle topologie consomme beaucoup d'énergie, ainsi prend beaucoup de temps
- Le nombre de nœuds éparpillés est important dans telle architecture jusqu'à 50 % :
  - ✓ Les nœuds éparpillés : sont les nœuds qui n'appartiennent à aucune clique

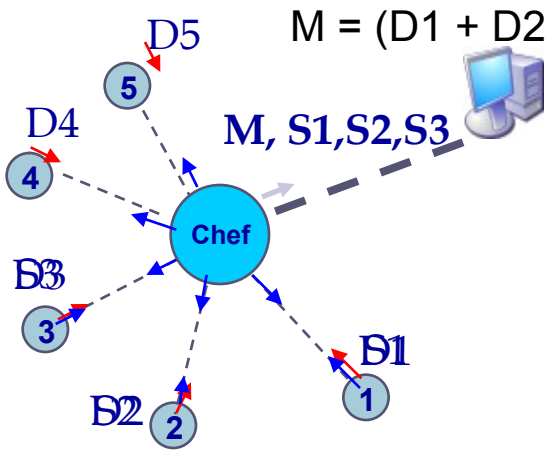


Les protocoles de sécurité d'agrégation



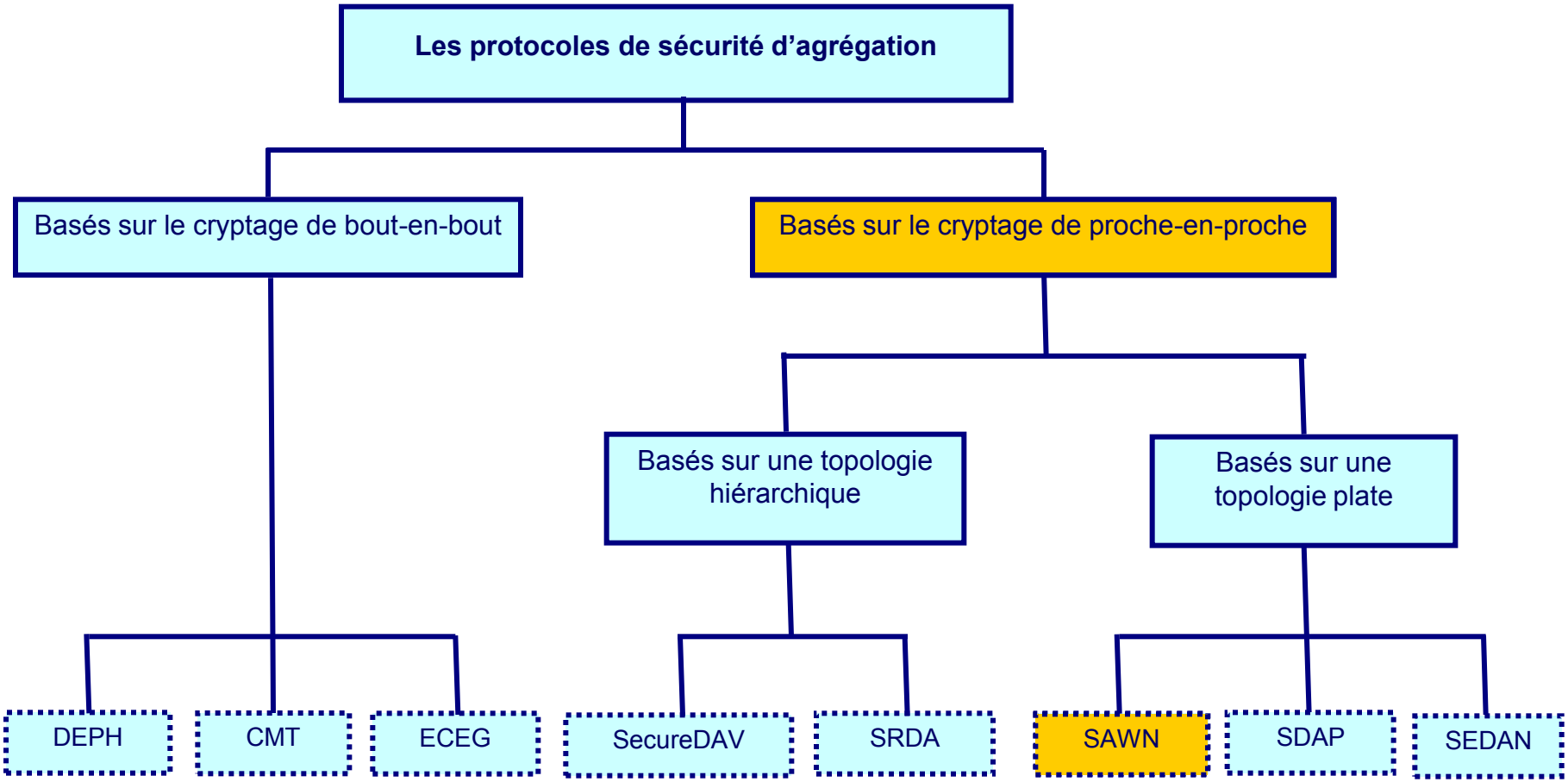
SecureDAV

- Chaque nœud {1, 2, 3, 4 et 5} envoie sa donnée au chef du cluster de manière sécurisée.
- Le chef du cluster:
  - ✓ Calcule la moyenne + D3 + D4 + D5) / 5.
  - ✓ Diffuse la valeur M au membres du cluster.
- Chaque membre :
  - ✓ Compare la différence entre sa lecture et M.
  - ✓ Signe la valeur M (signature partielle)(S1).
  - ✓ Envoie la signature partielle S1 au chef du cluster.



- Le chef du cluster:
  - ✓ Combine les différentes signatures reçues.
  - ✓ Envoie de manière sécurisée la valeur M et la signature combinée à la station de base.
- La station de base peut vérifier la validité de M par les signatures S1, S2, S3.

Les protocoles de sécurité d'agrégation

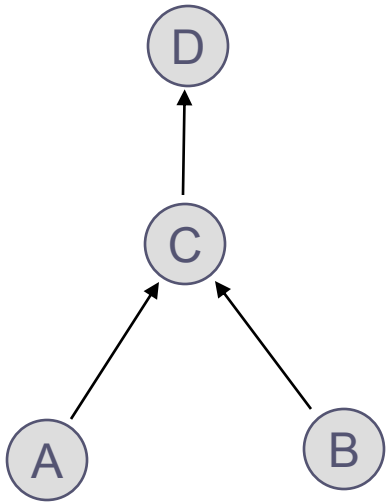


SAWN

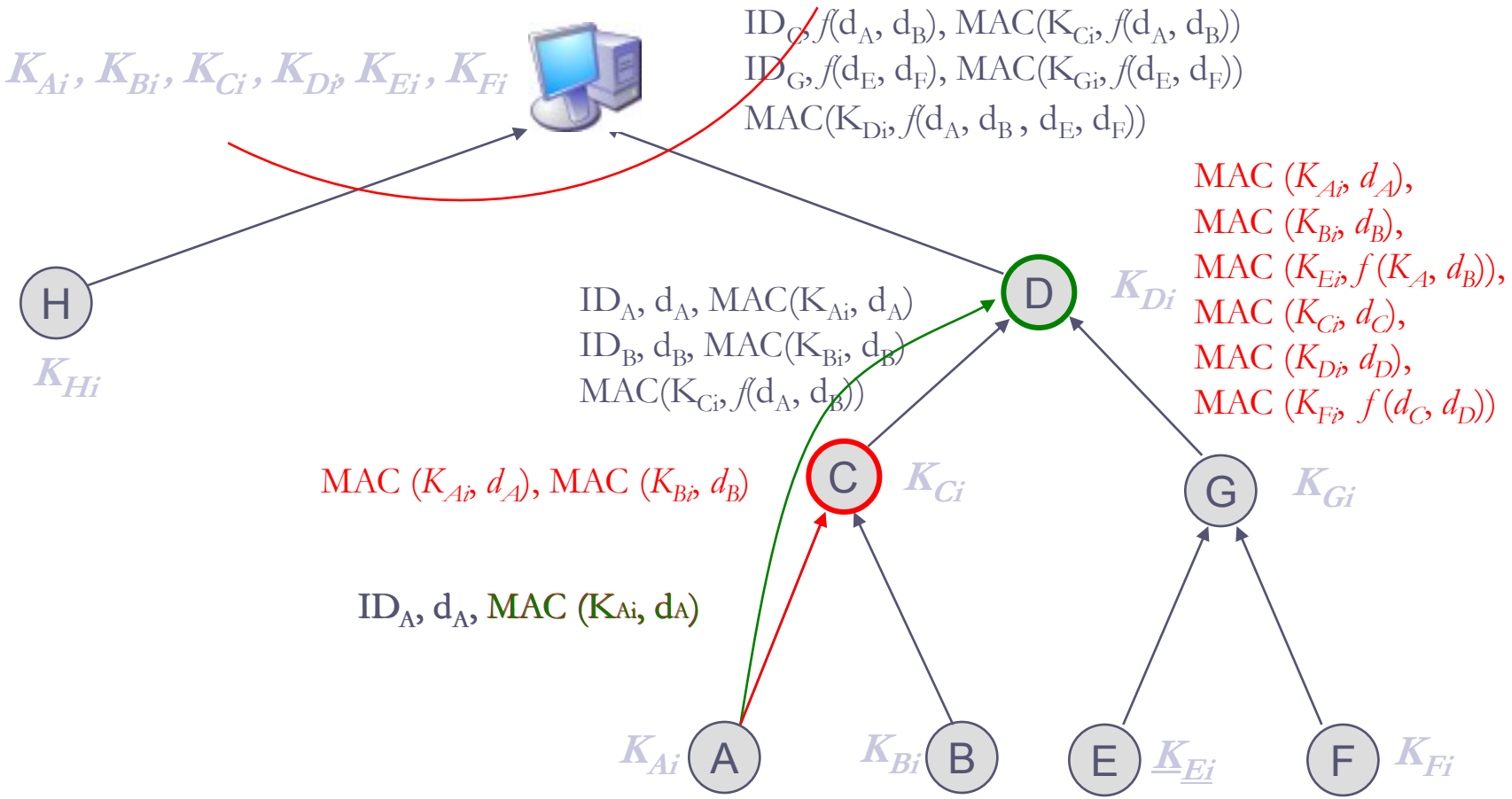
- Basé sur le mécanisme de vérification de deux sauts.
- L'hypothèse principale :
  - ✓ Deux noeuds consécutif ne sont pas compromis
- La vérification est retardée jusqu'à la réception de toutes les valeurs d'agrégations par la station de base
  - ✓ Elle diffuse les clés des noeuds vers tout le réseau
  - ✓ Permet à chaque nœud de vérifier l'intégrité des données de ses petits fils et la valeur d'agrégation de chaque fils

SAWN

- Vérification saut par saut (hop by hop).
- Concept de vérification à deux sauts
  - ✓ Vérifier si un nœud a bien calculer l'agrégation de ses fils.
- A envoie sa donnée à C :
  - ✓ Il inclut une preuve qui permet à D de vérifier si C n'a pas modifié la donnée A.
- C calcule l'agrégation et l'émet à D
  - ✓ Il émet aussi les preuves de A et B.
- D peut comparer entre l'agrégation de C et les preuves de A et B.



SAWN: Secure Aggregation for Wireless Networks



Intrusion-Fault Tolerance in Wireless Sensor Networks

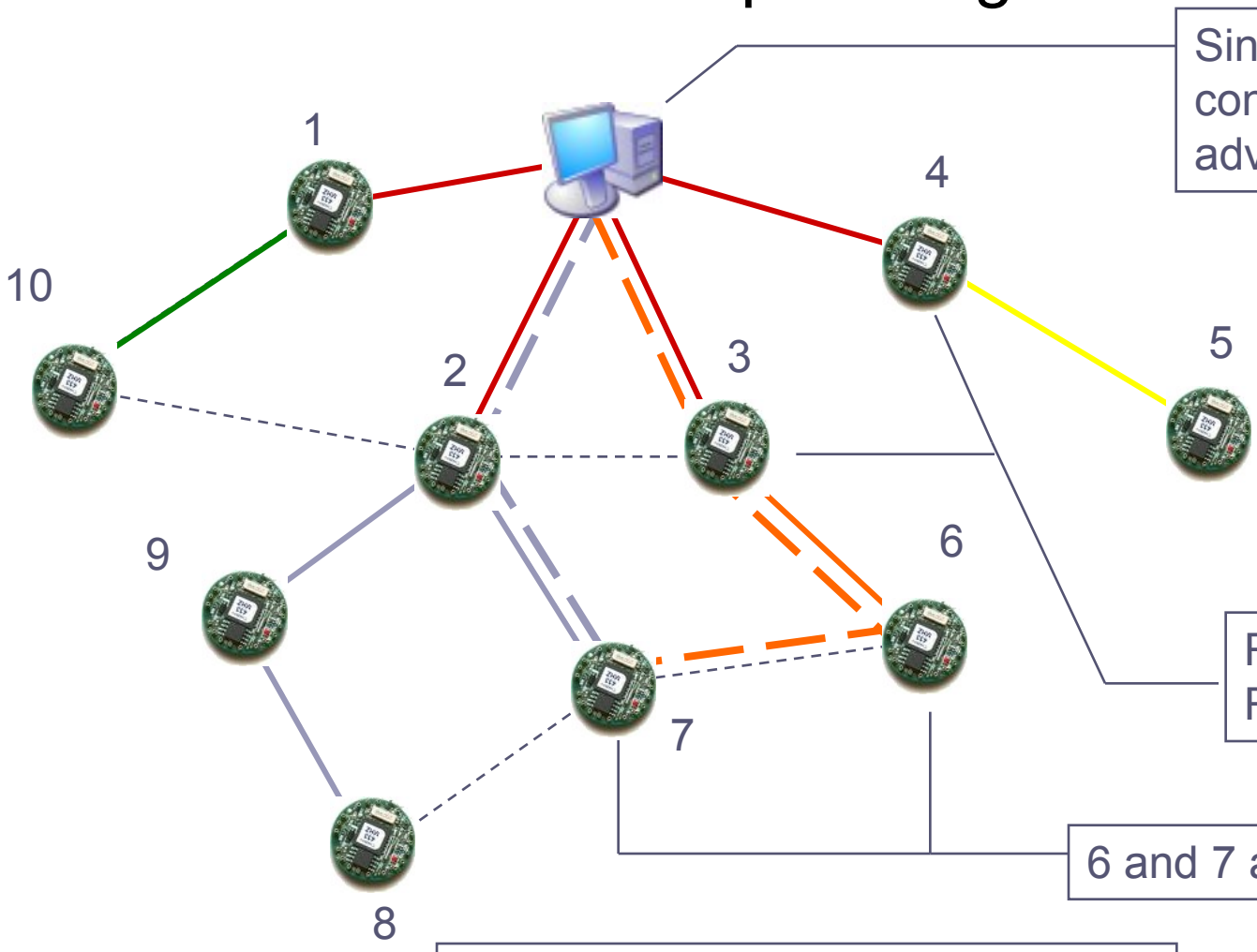
Abdelraouf Ouadjaout

with Y. Challal, N. Lasla and M. Baga

Our goals

- **High fault tolerance**
  - Node disjoint paths
- **Scalable and efficient**
  - **In-network verification**
  - Totally distributed **without referring to the BS**
    - ✓ Construction
    - ✓ Security checks
  - One message per sensor
- **Immediate detection**

Multipath engine



Sink starts route construction by advertising a RREQ

Root nodes **tag** the RREQ with their IDs

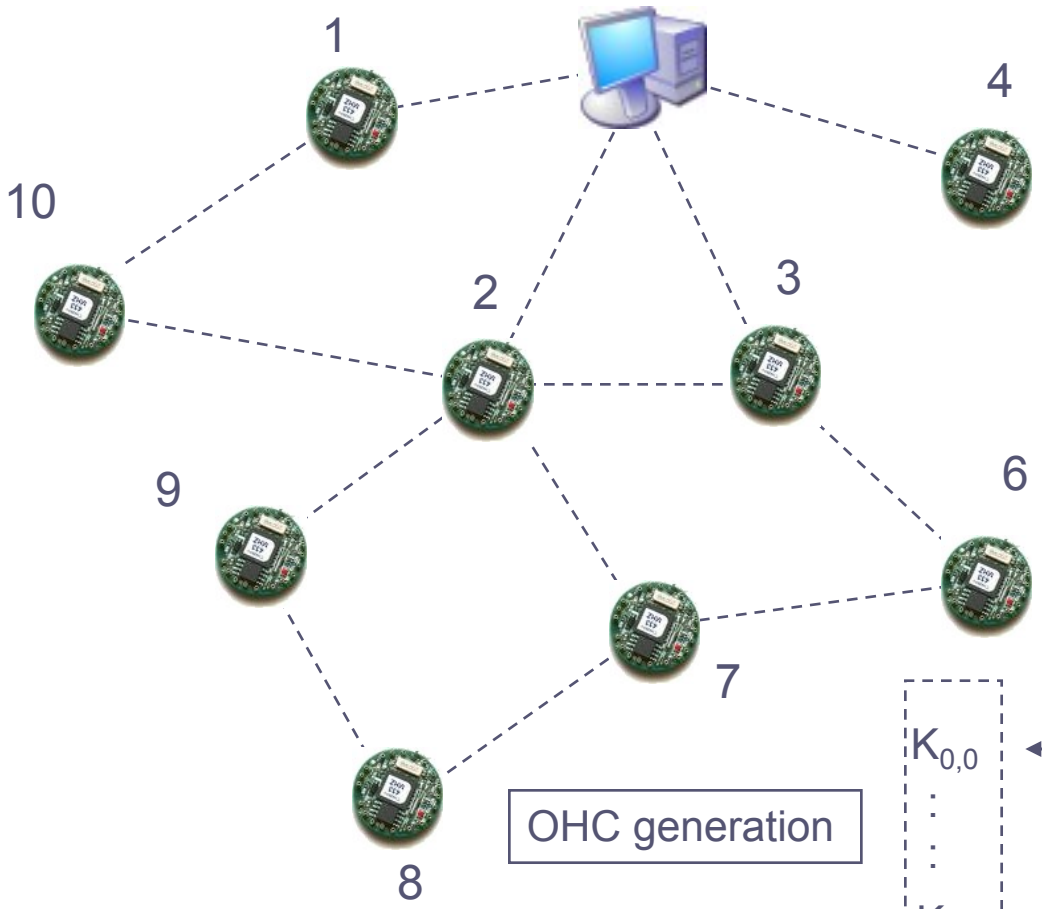
6 and 7 are **cousins neighbors**

They share two new **disjoint** paths

Secure Multipath Engine

- **Three security problems emerge**
  - "Plaintext" tagging
    - ✓ Sink hole attacks
  - Round initialization
    - ✓ BS spoofing
  - Parent ID
    - ✓ Hello flooding

Secure Branch Tagging



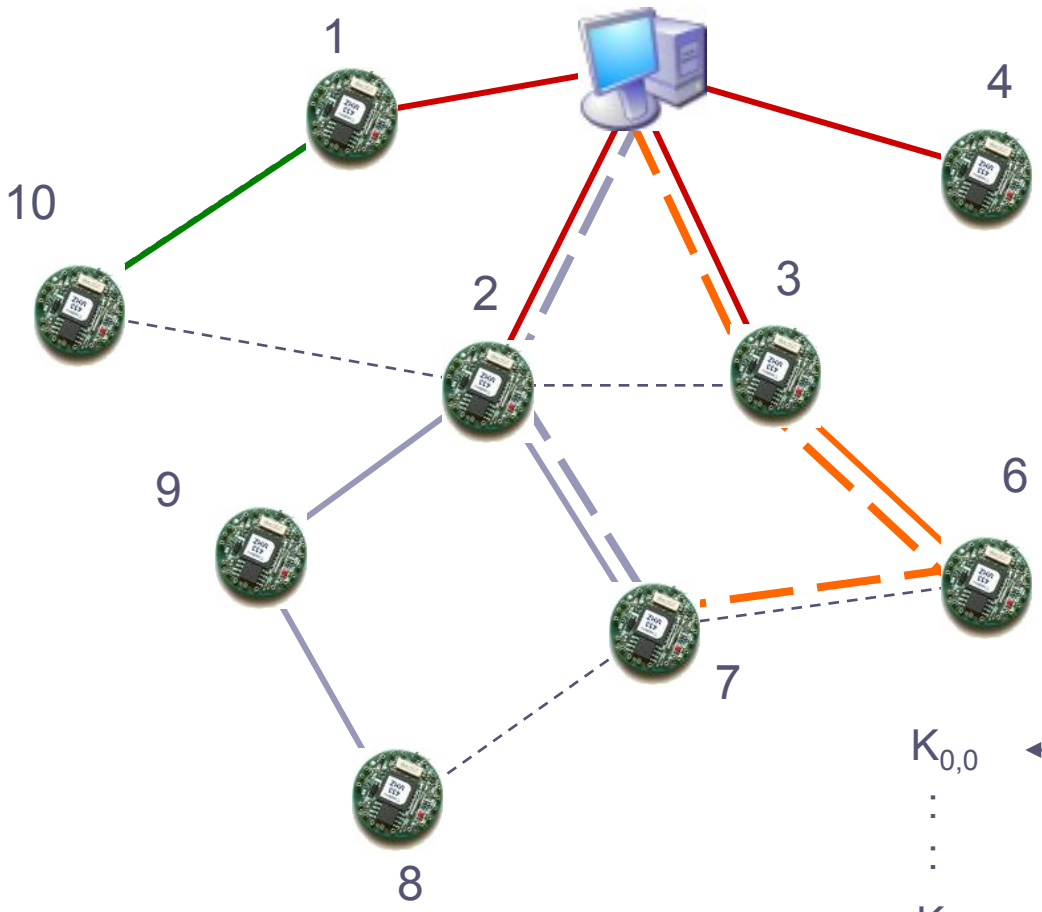
Sensors are preloaded with the first unused value of each chain

Each sensor *i* maintains for each chain *j* a **chain verifier** $CV_{i,j}$ and its **position** $P_{i,j}$ within the corresponding chain

OHC generation

$$\begin{array}{c} K_{0,0} \\ \vdots \\ K_{0,m} \end{array} \xleftarrow{F} K_{1,0} \xleftarrow{F} \cdots \xleftarrow{F} K_{n-1,0} \xleftarrow{F} K_{n,0}$$
$$\begin{array}{c} \\ \\ \end{array} \xleftarrow{F} K_{1,m} \xleftarrow{F} \cdots \xleftarrow{F} K_{n-1,m} \xleftarrow{F} K_{n,m}$$

Secure Branch Tagging (2)

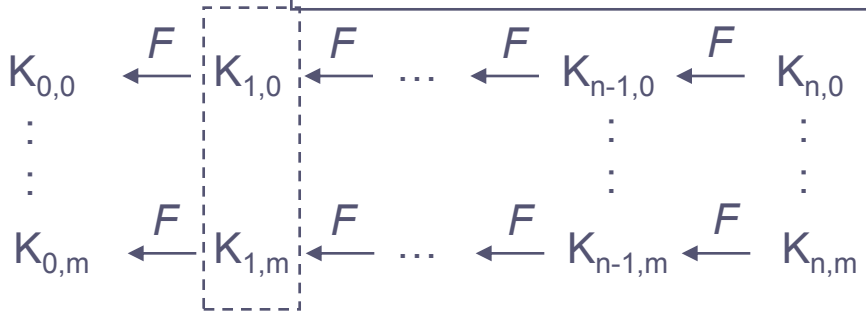


Tag distribution: each root receives its valid tag for the current round

Tree construction: relay of a RREQ message: n, p and V

Sensors verify:

$$\begin{cases} p > P_{i,n} \\ CV_{i,n} = F^{p-P_{i,n}}(V) \\ p - P_{i,n} < D \end{cases}$$



Round initialization

- **Sink node is the only initiator of route construction**
- **A special hash chain is reserved to round authentication**
- **At each round, the sink reveals in the RREQ the next unused value of the chain**

Local broadcasts

- **Each sensor should authenticate its RREQ messages**
  - Use of a local hash chain
  - Reachable neighborhood
- **When a sensor is deployed, it reveals its first value of the chain encrypted with a broadcast key**
- **At each emission of a RREQ, the sensor reveals the next value**

Conclusion

- **SEIF provides**
  - High tolerance
    - ✓ Node disjoint
  - Efficiency
    - ✓ Only symmetric crypto
    - ✓ 3 hashs per message
  - Scalability
    - ✓ One message per sensor