Key Management in Comunicazioni sicure di gruppo

- **Ing. Francesco Giurlanda**

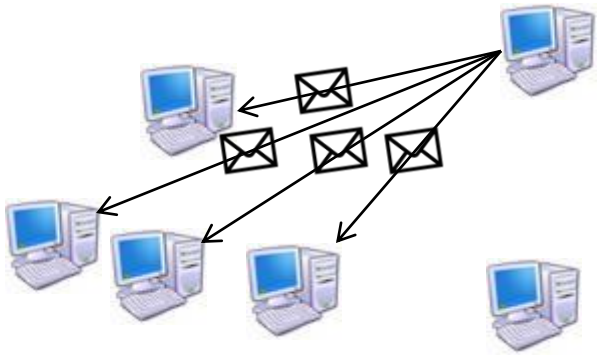
Comunicazioni di gruppo

Nell'ultimo decennio sono comparse molte applicazioni che si basano sulle comunicazioni di gruppo. Pay-TV, video conferenze, reti di sensori…

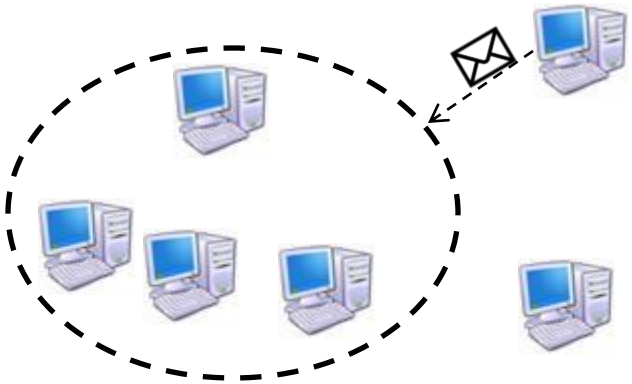
Comunicazioni di gruppo

IP multicast è una tecnica di trasmissione da una singola sorgente dati verso un gruppo di elementi interessati, senza dover duplicare i messaggi.

- **Pro**: efficiente e scalabile.
- **Contro**: controllo degli accessi.



IP Unicast



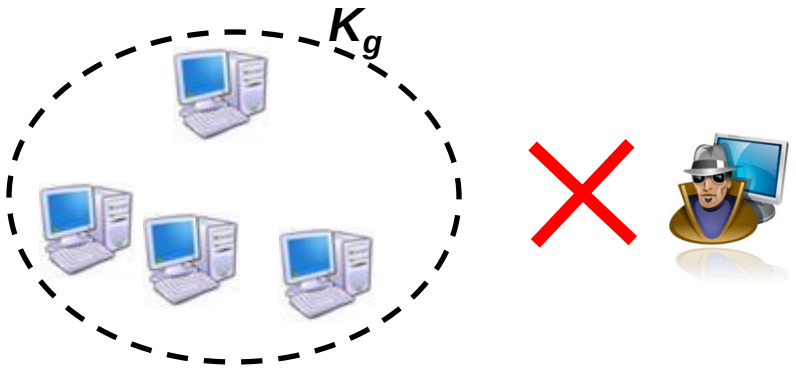
IP Multicast

Comunicazioni di gruppo

La crittografia permette di proteggere i messaggi scambiati tra i membri di un gruppo.

I membri di un gruppo condividono un segreto, **chiave di gruppo K_g**

Solo chi conosce la chiave di gruppo può comunicare con gli altri membri del gruppo.



Problema: Come gestire la chiave di gruppo?

Gestione della chiave di gruppo

Il gestore della chiave di gruppo o **key manager (KM)** deve:

- Provvedere all'**identificazione** e all'**autenticazione** di un membro.
- Applicare politiche di **access control** sulle richieste di nuovi membri.
- **Generazione, distribuzione** e **installazione** della chiave di gruppo.

Gestione della chiave di gruppo

La distribuzione della chiave di gruppo deve avvenire in modo

- **Sicuro:** solo gli appartenenti al gruppo devono conoscere K_g
- **Efficiente:** gruppi di grandi dimensioni potrebbero richiedere un tempo elevato per la distribuzione della chiave di gruppo

Gestione della chiave di gruppo

La chiave di gruppo va rinnovata **periodicamente**

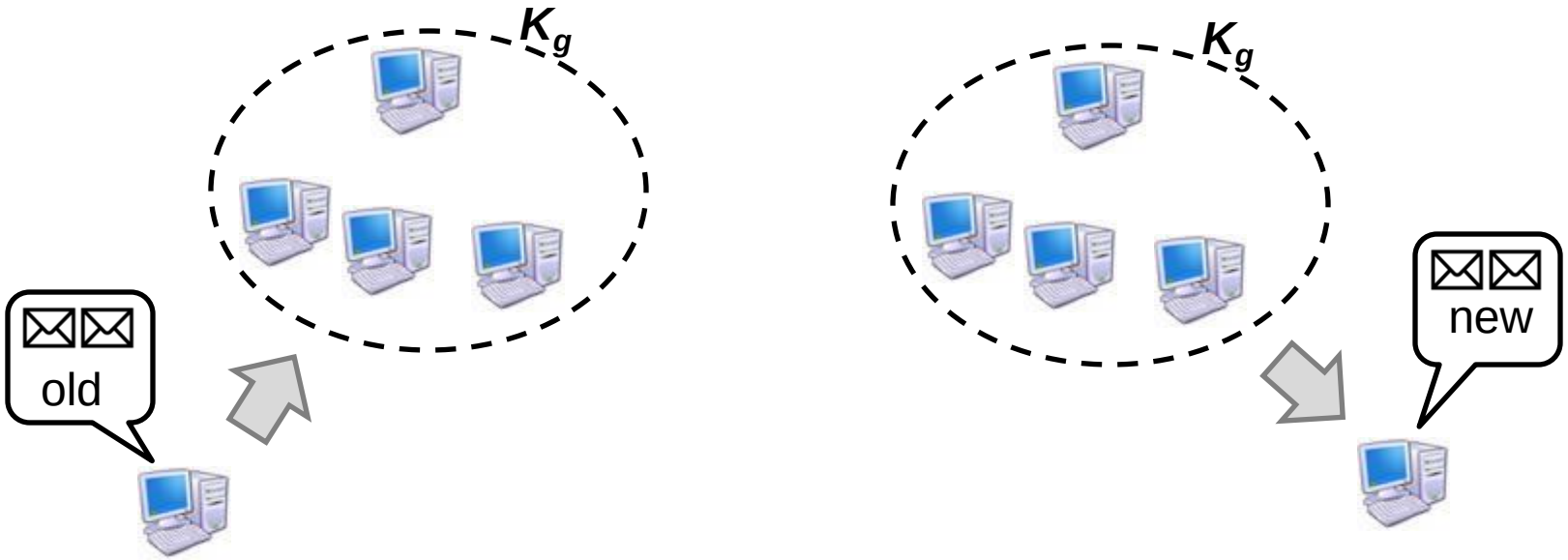
- la probabilità che un segreto condiviso venga svelato aumenta con il numero di individui a conoscenza del segreto

L'operazione di rinnovo della chiave di gruppo si chiama **rekeying.**

Gestione della chiave di gruppo

Il rekeying va effettuato se un nuovo membro si unisce al gruppo. Garantisce la **backward secrecy.**

Il rekeying va effettuato se un membro abbandona il gruppo. Garantisce la **forward secrecy.**



SKDC, una semplice soluzione

La prima soluzione al problema risale al 1997 e si basa su **Simple Key Distribution Center (SKDC)**.

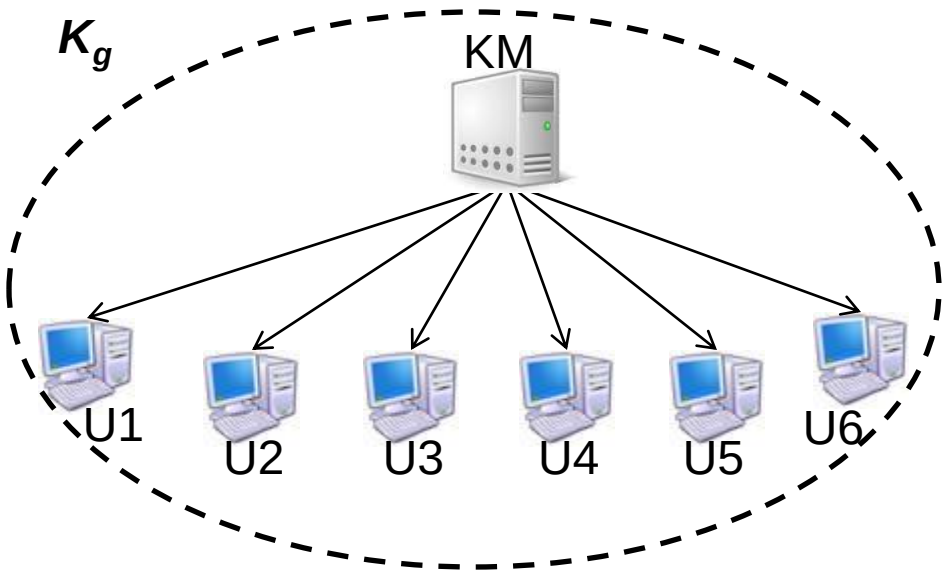
Un **key manager (KM)** si occupa della distribuzione della chiave di gruppo.

Per stabilire un canale sicuro, ciascun utente condivide inizialmente una **pairwise key** con KM che chiameremo K_i

SKDC, una semplice soluzione

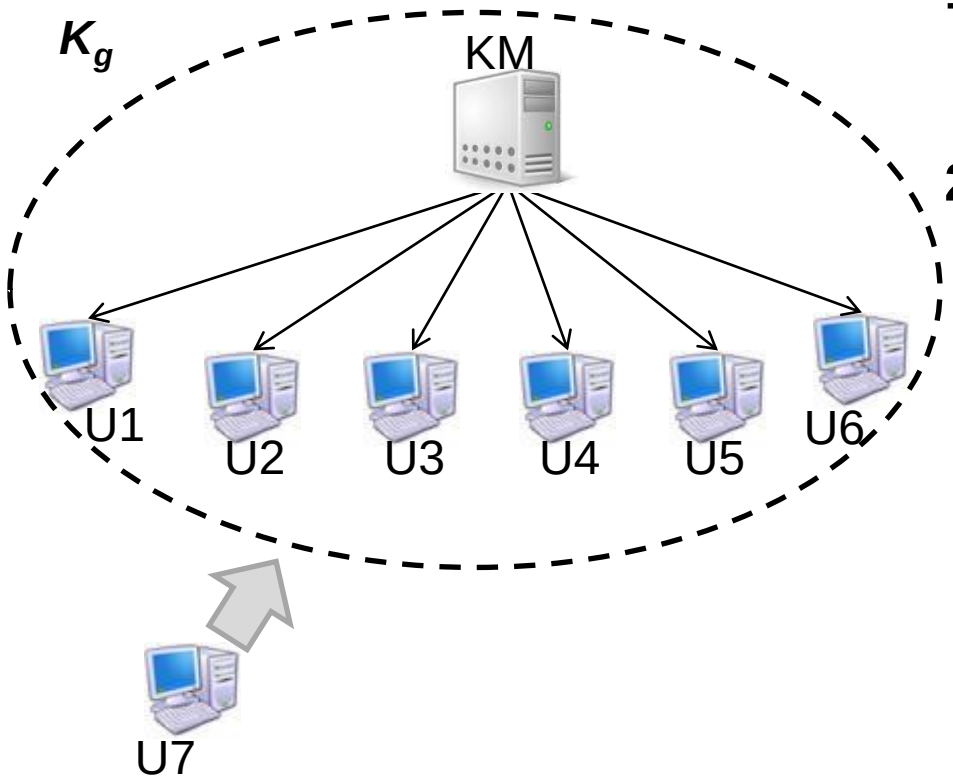
Distribuzione della chiave di gruppo:

$$KM \rightarrow U_i : E(K_i)\{K_g\}$$



SKDC, una semplice soluzione

Operazione di **join**: ingresso di un nuovo membro nel gruppo

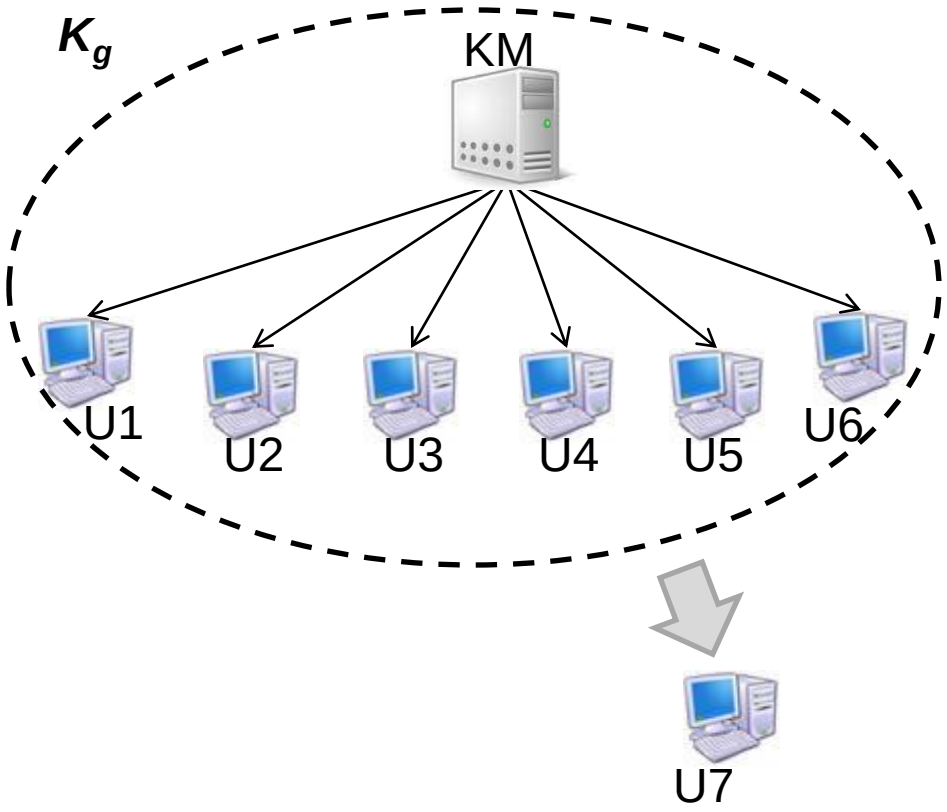


1) $KM \rightarrow All: E(K_g)\{\overline{K}_g\}$

2) $KM \rightarrow U_7: E(K_7)\{\overline{K}_g\}$

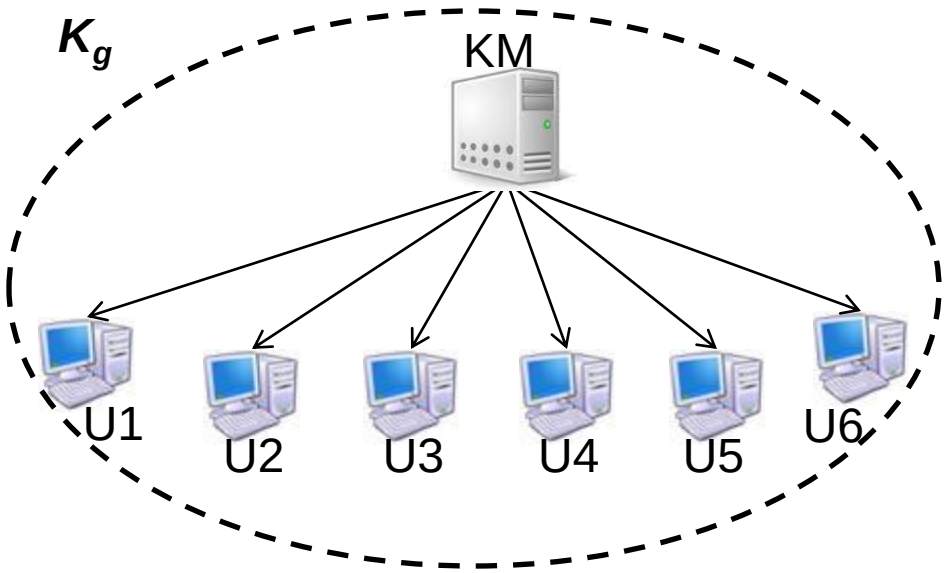
SKDC, una semplice soluzione

Operazione di **leave**: un membro abbandona il gruppo



$$KM \rightarrow U_i : E(K_i)\{\overline{K}_g\}$$

SKDC, una semplice soluzione



Numero di messaggi per le operazioni di rekeying ***O(n)***.

Pro: semplice.

Contro: inefficiente,
n = 1.000.000
key = 56 bits
dovrò inviare ***n* repliche** della chiave per un totale di informazione pari a
7 MB.

Protocolli di rekeying

Dal 1997 sono stati proposti molti approcci al problema del rekeying raggruppabili in tre grandi famiglie:

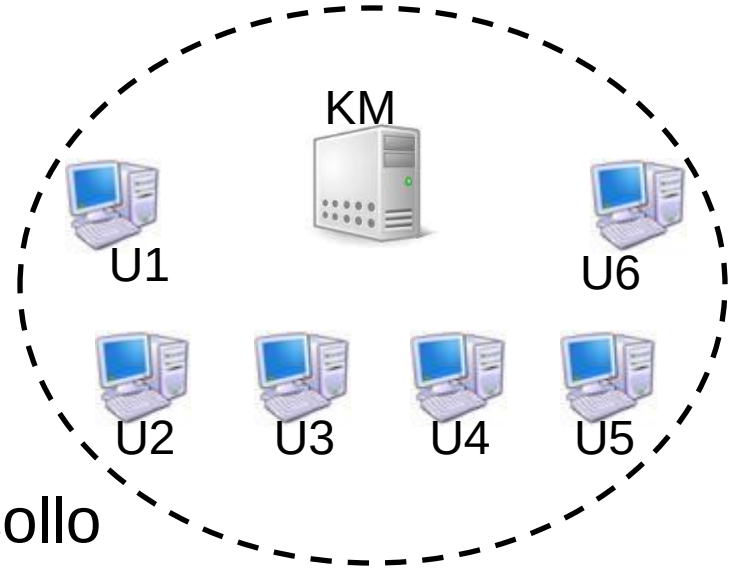
- **Centralizzati:** una singola entità controlla l'intero gruppo.
- **Decentralizzati:** più entità controllano l'intero gruppo, dividendolo in sottogruppi e cercando di ridurre l'accentramento delle operazioni.
- **Distribuiti:** non esiste un KM e i membri stessi di un gruppo contribuiscono alla generazione e distribuzione della chiave di gruppo.

Protocolli Centralizzati

Pro: Semplice da gestire

Contro:

- Poco tollerante ai guasti.
- Se la sicurezza di KM è compromessa, lo è anche quella dell'intero sistema.
- KM può rappresentare un collo di bottiglia per le prestazioni della rete



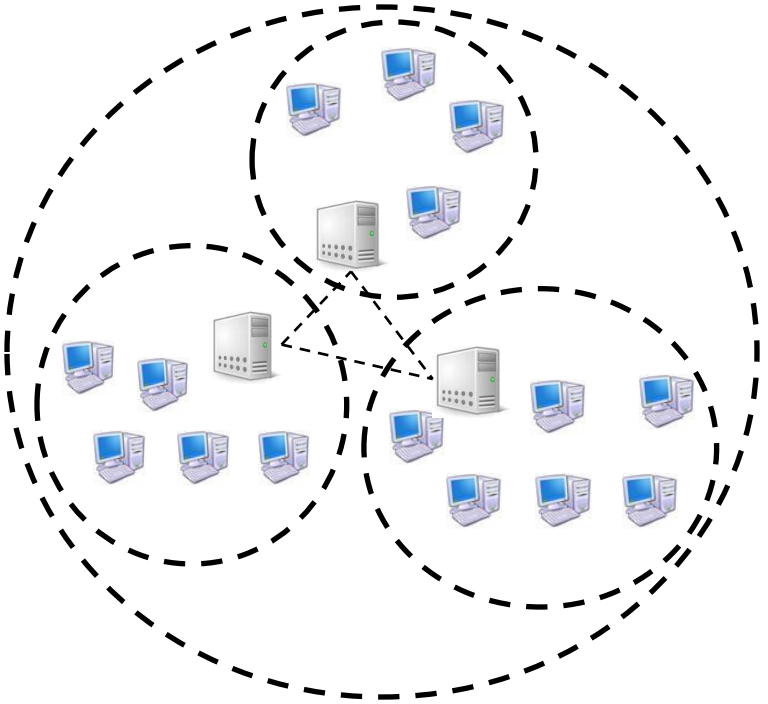
Protocolli Decentralizzati

Pro:

- Maggiore tolleranza ai guasti
- Carico di gestione viene diviso su più KMs
- Riduce i problemi dovuti all'accentramento

Contro:

- Maggiore complessità di configurazione del sistema
- Complesse interazioni tra i sottogruppi



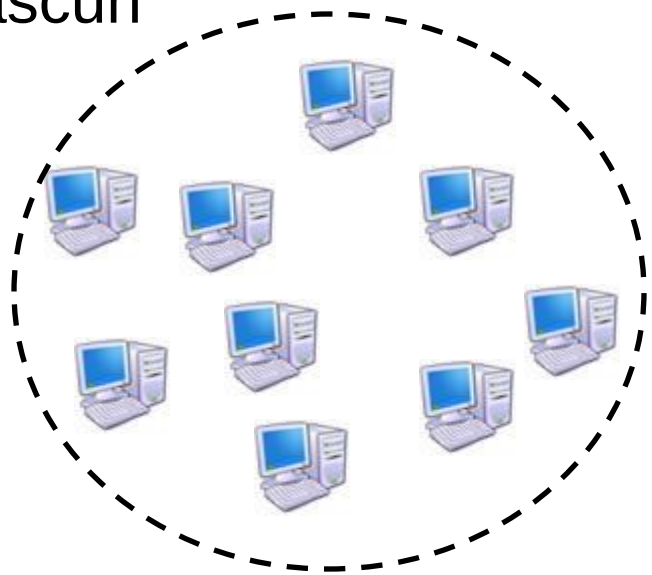
Protocolli Distribuiti

Pro:

- Completa tolleranza ai guasti, mancanza di KM
- Elimina i problemi dovuti all'accentramento

Contro:

- Carico computazionale per ciascun utenti cresce linearmente con il numero di utenti
- Ogni membro deve essere a conoscenza della lista di appartenenza al gruppo

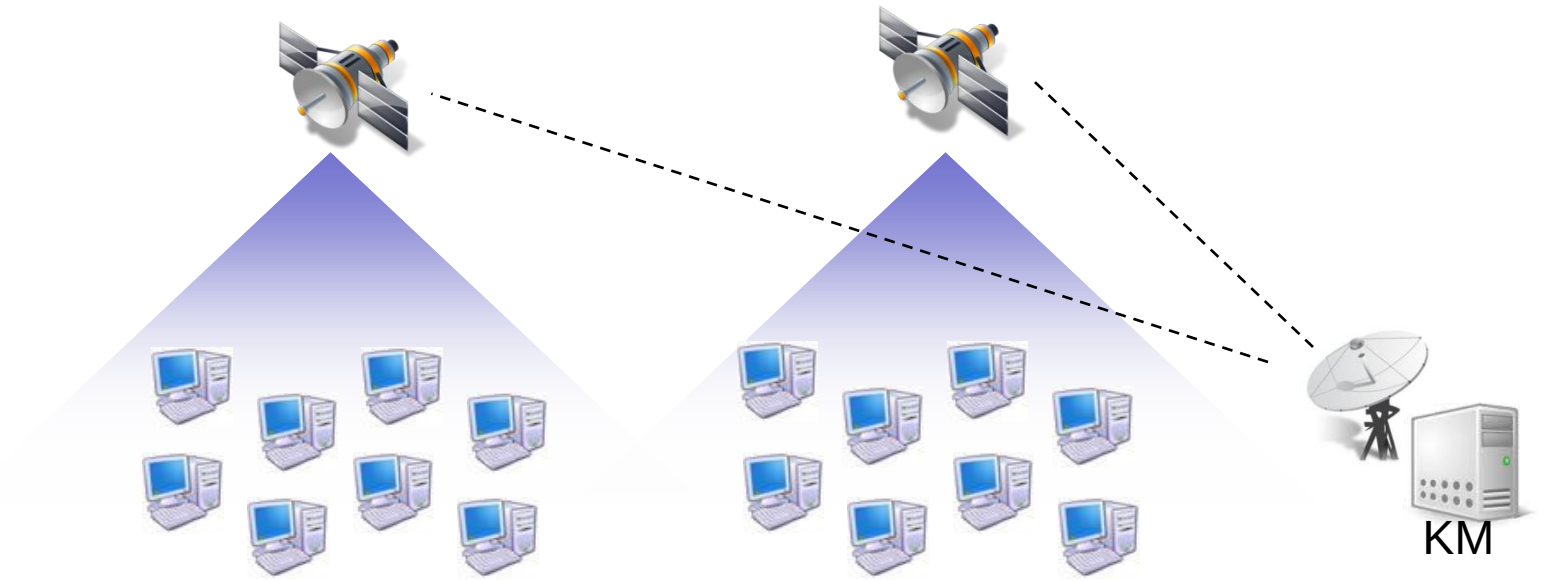


Logical Key Hierarchy (LKH)

LKH è un protocollo **centralizzato** per la gestione della chiave di gruppo.

Ideato da Wang et al. [1997] e Wallner et al. [1998].

Trova applicazione in reti che formano gruppi con un elevato numero di utenti ($\approx 10^6$) come le reti satellitari.



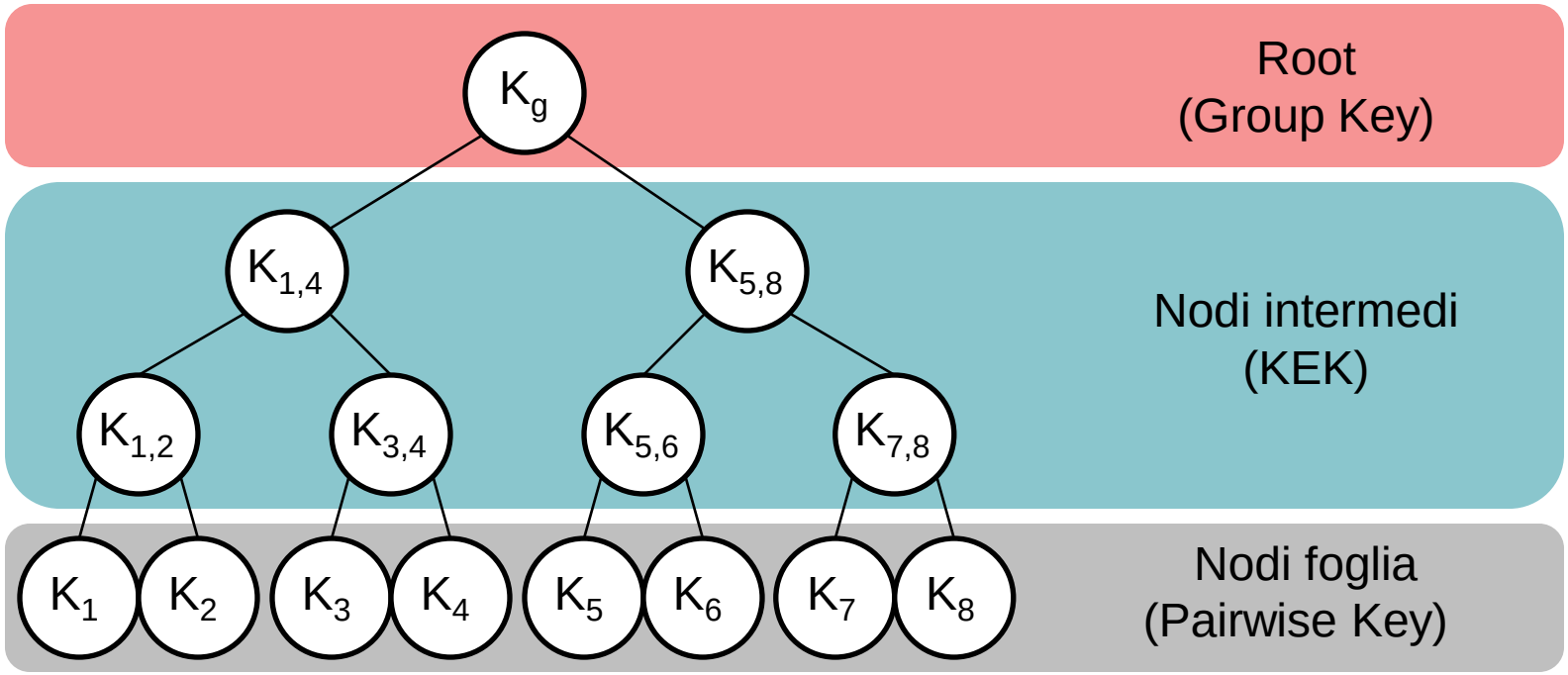
Logical Key Hierarchy (LKH)

Il KM utilizza una **struttura logica ad albero** per ottimizzare la procedura di rekeying.

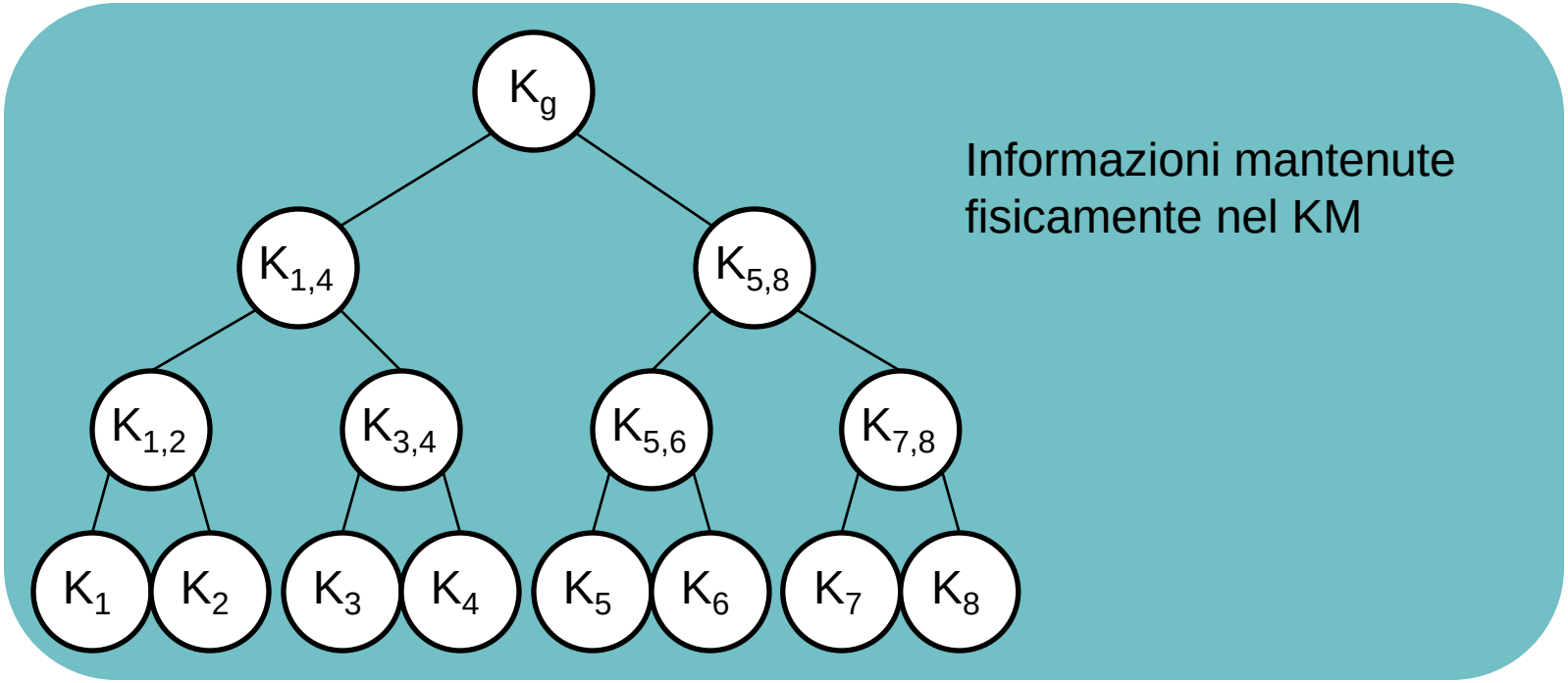
Oltre alla chiave di gruppo K_g e alle chiavi condivise con gli utenti K_i, LKH utilizza delle **key encryption keys (KEK)** per rendere più efficiente l'operazione di rekeying.

Le KEK sono utilizzate solo durante la procedura di rekeying.

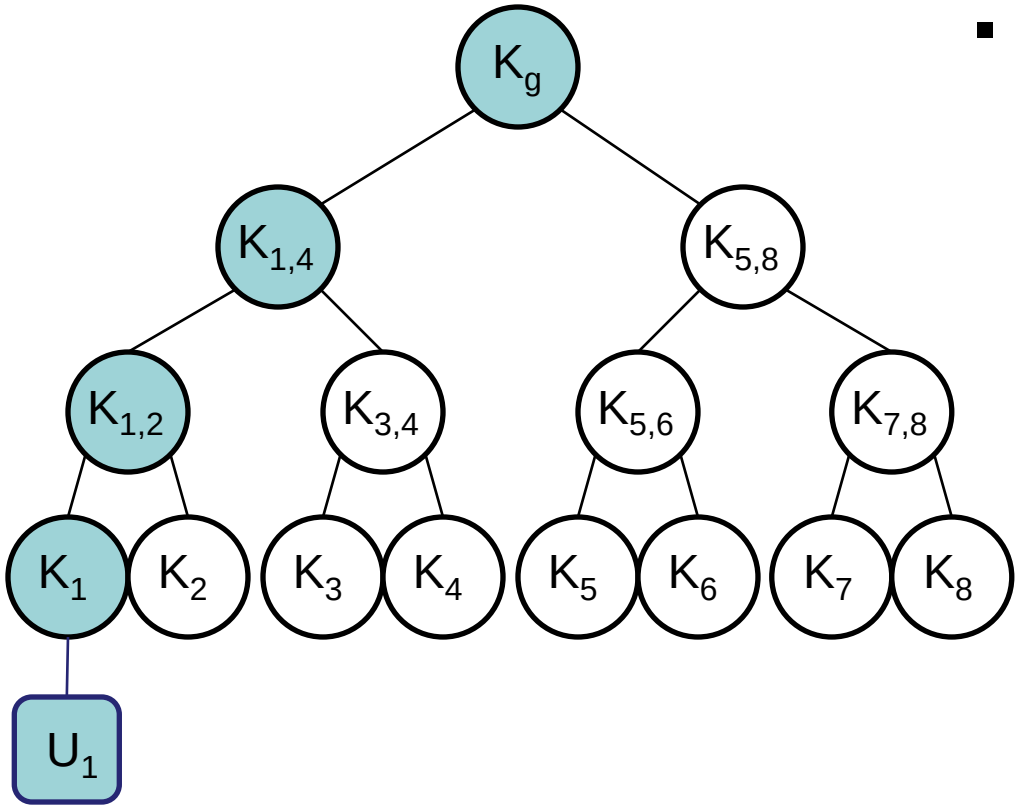
Logical Key Hierarchy (LKH)



Logical Key Hierarchy (LKH)

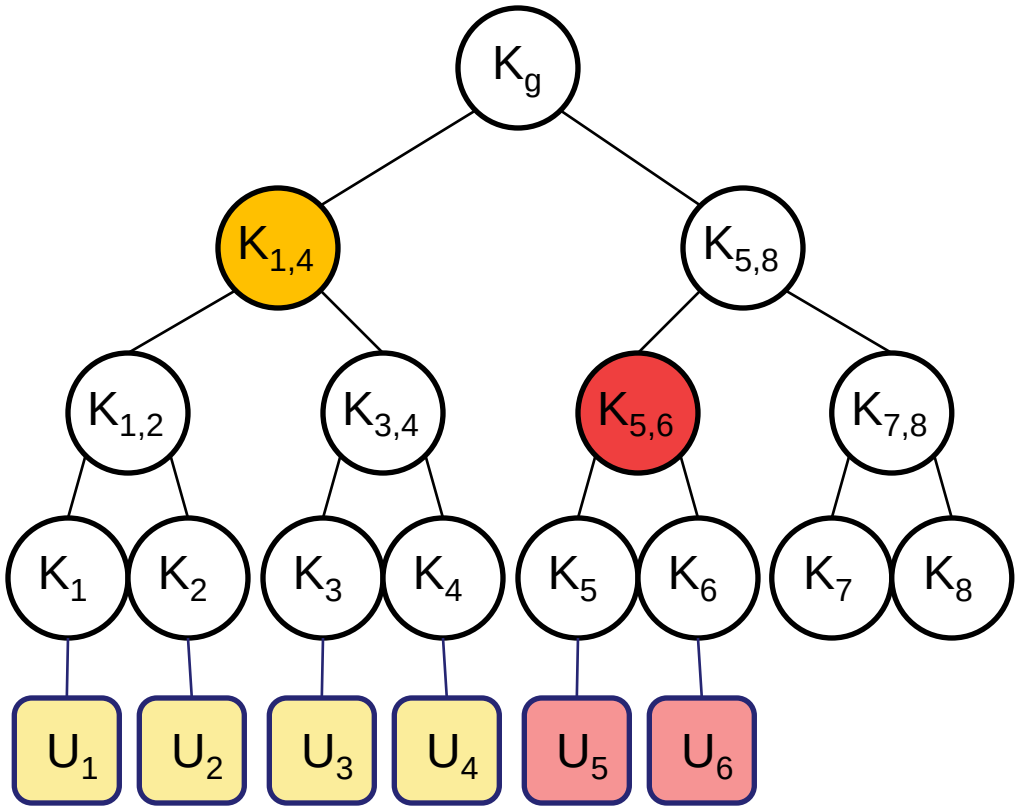


Logical Key Hierarchy (LKH)



- A ciascun utente è associata una foglia dell'albero
- L'utente conserva le chiavi lungo il percorso dalla root al proprio nodo foglia

Logical Key Hierarchy (LKH)

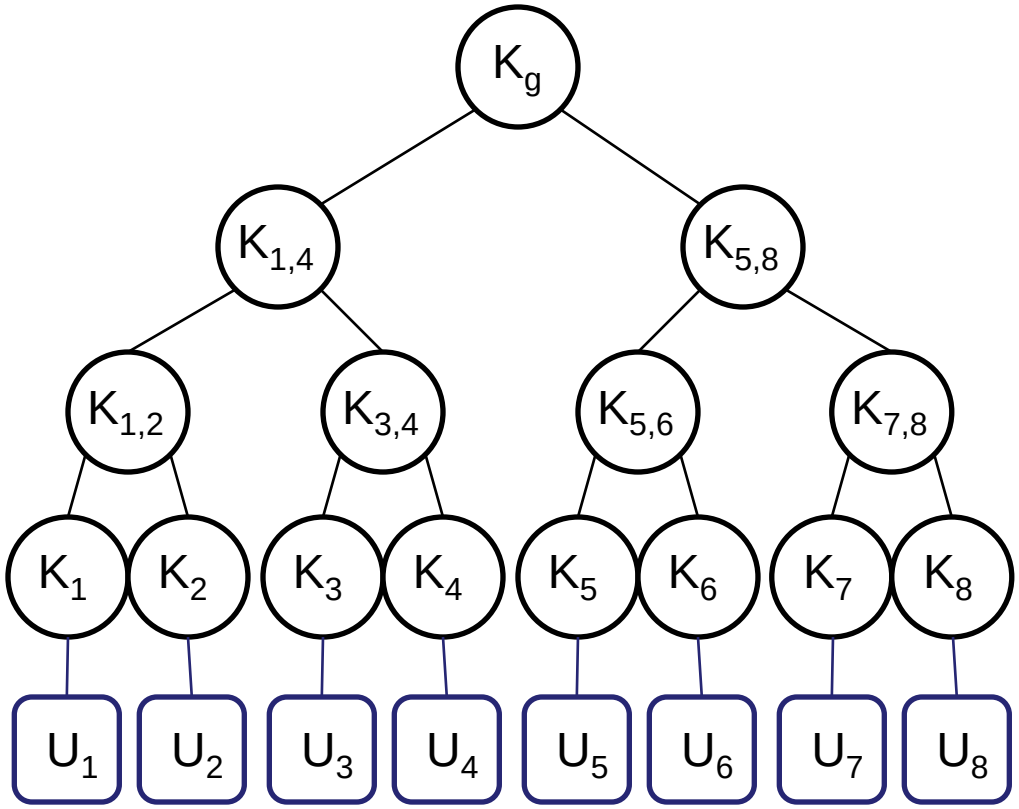


Il ruolo delle KEK è partizionare il gruppo in sottogruppi.

Una KEK è un segreto condiviso tra tutti i nodi foglia da essa derivanti.

La chiave di gruppo è condivisa tra tutti i membri del gruppo in quanto radice (root) dell'albero

Logical Key Hierarchy (LKH)

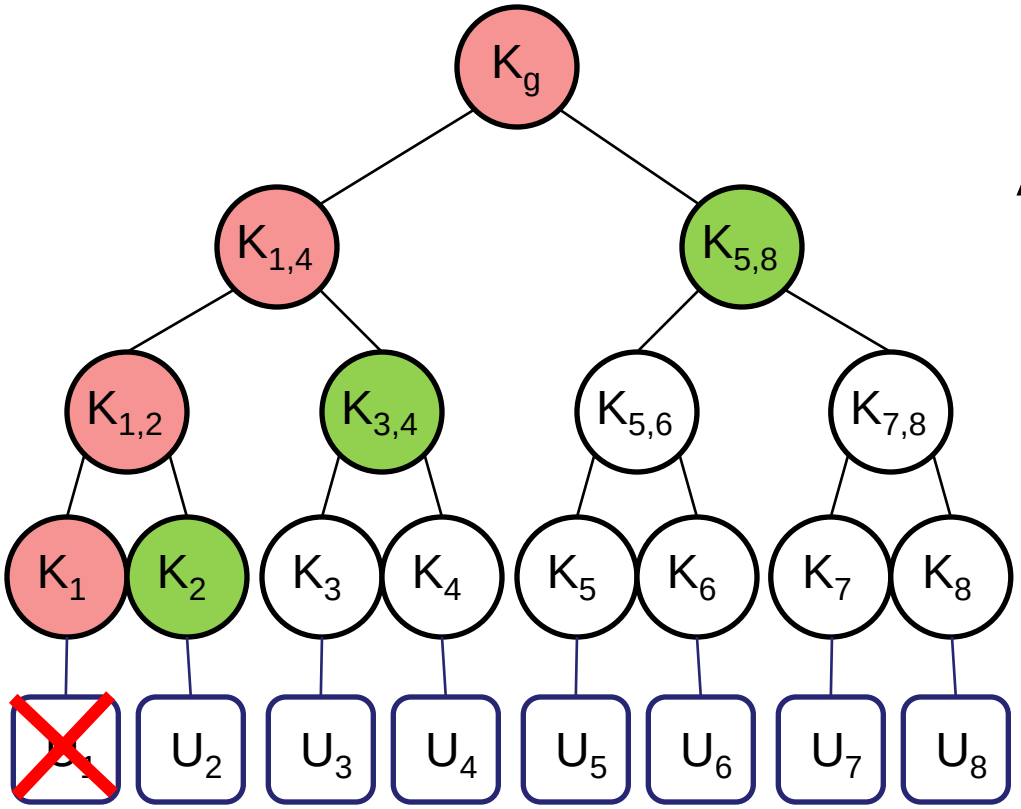


Gruppo con *n* membri.

Numero di chiavi da gestire:

- KM: **$2n-1$**
- Utente: **$\log(n)_2 + 1$**

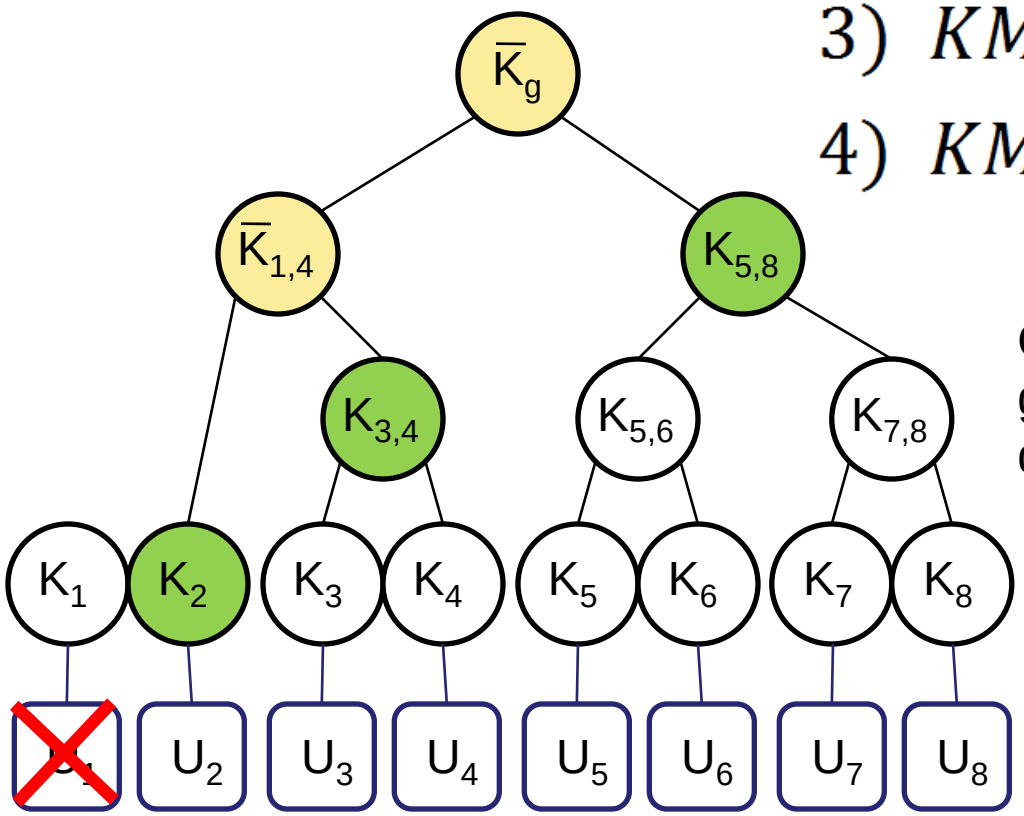
Funzionamento del protocollo



Supponendo che U_1 esca dal gruppo, la chiave di gruppo deve essere rinnovata (backward secrecy).

Anche le KEK di U_1 devono essere rinnovate.

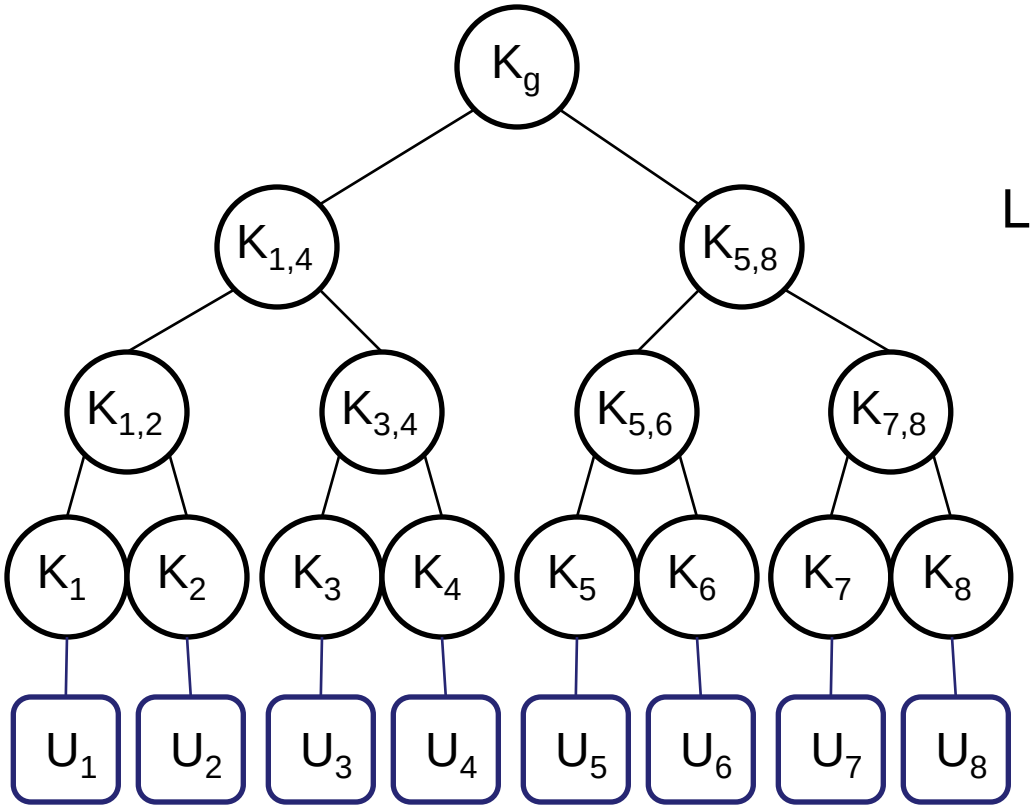
Funzionamento del protocollo



1) $KM \rightarrow U_2: E(K_2)\{\overline{K}_{1,4}\}$
2) $KM \rightarrow All: E(K_{3,4})\{\overline{K}_{1,4}\}$
3) $KM \rightarrow All: E(\overline{K}_{1,4})\{\overline{K}_g\}$
4) $KM \rightarrow All: E(K_{5,8})\{\overline{K}_g\}$

Ottengo il rinnovo della chiave di gruppo con 4 messaggi al posto di 7 messaggi.

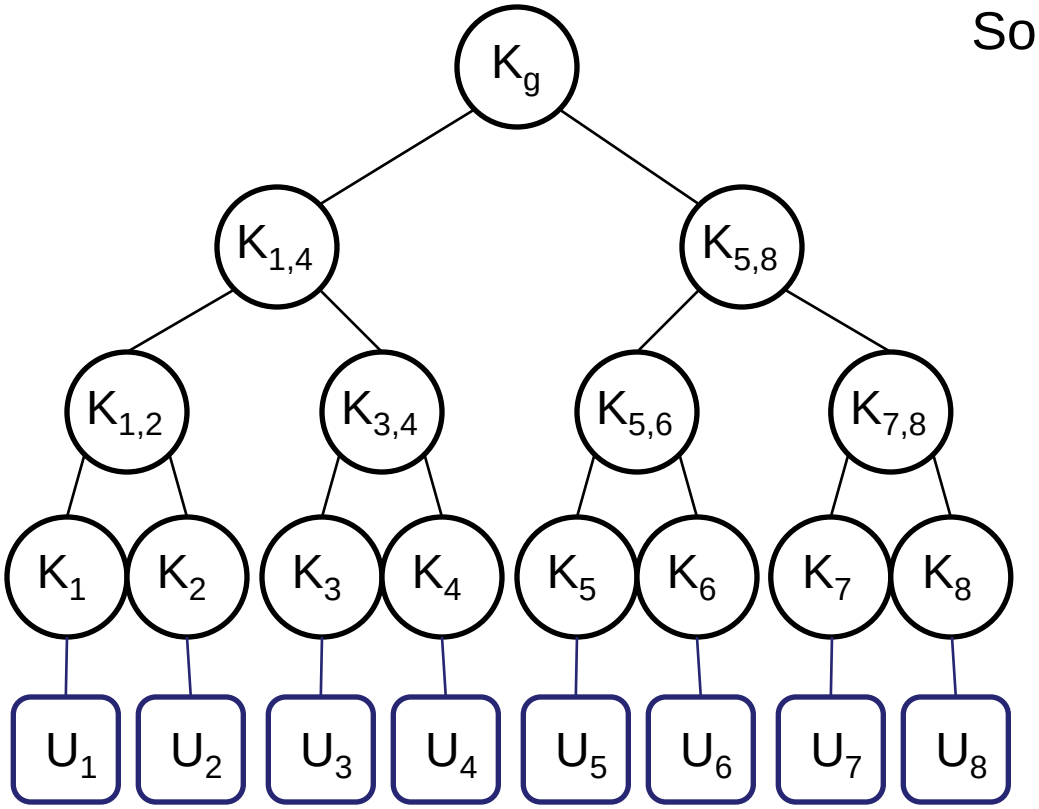
Funzionamento del protocollo



LKH si comporta allo stesso modo nel caso di un nuovo membro entrante nel gruppo.

LKH effettua un'operazione di rekeying inviando un numero di chiavi **$2\log(n)+1$**

Funzionamento del protocollo



Es:

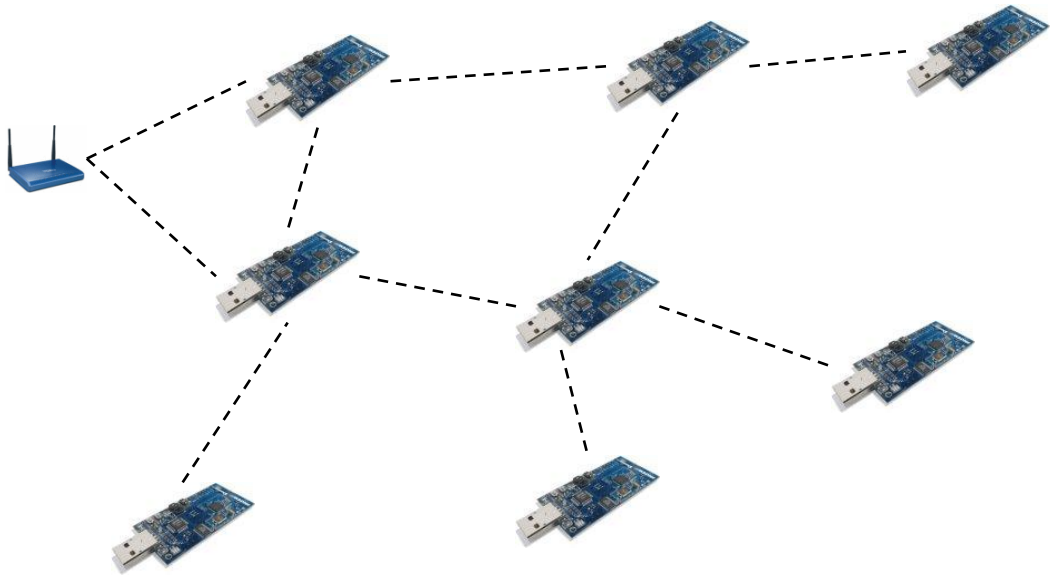
$n = 1048576 = 2^{20}$

key = 56 bit

Sono sufficienti **41** messaggi per un totale di **256 B** di informazione

Secure and Simple Rekeying Protocol (S^2RP)

- Protocollo centralizzato ideato da Dini e Savino [2006]
- Si basa su LKH e Hash-chain.
- Garantisce l'autenticazione delle chiavi.
- Indicato per reti di sensori dove le capacità computazionali sono ridotte.



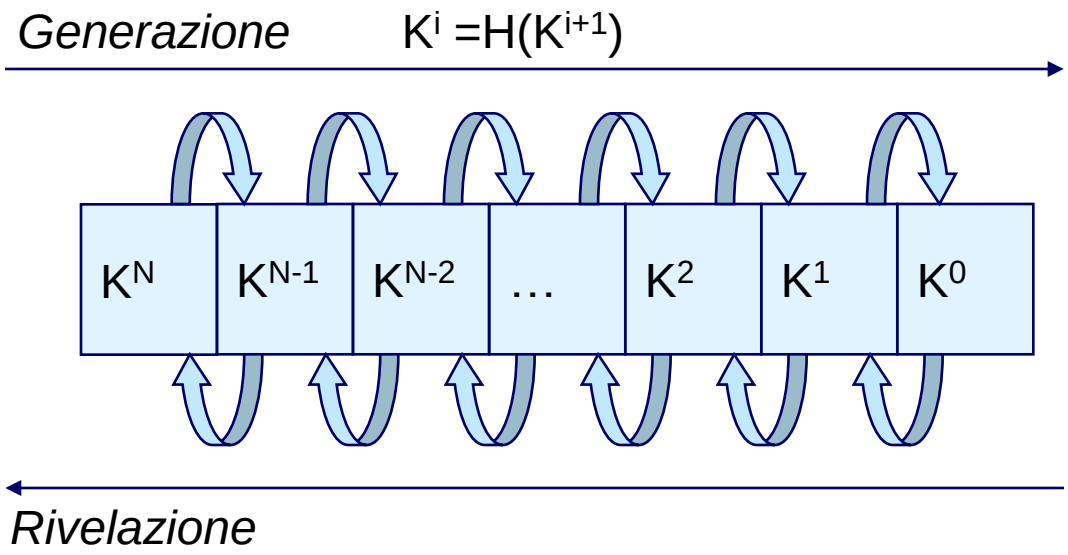
Hash-chain

L'autenticazione basata su Hash-chain.
Lamport [1981]

H è una OWHF con le seguenti caratteristiche:

- ***Semplice da calcolare***
- ***Preimage resistence****: dato un output y è computazionalmente impossibile calcolare il valore m tale che H(m) = y .*
- ***2° preimage resistence****: dato un input m è computazionalmente impossibile calcolare un secondo valore m' tale che H(m') = H(m).*

Hash-chain

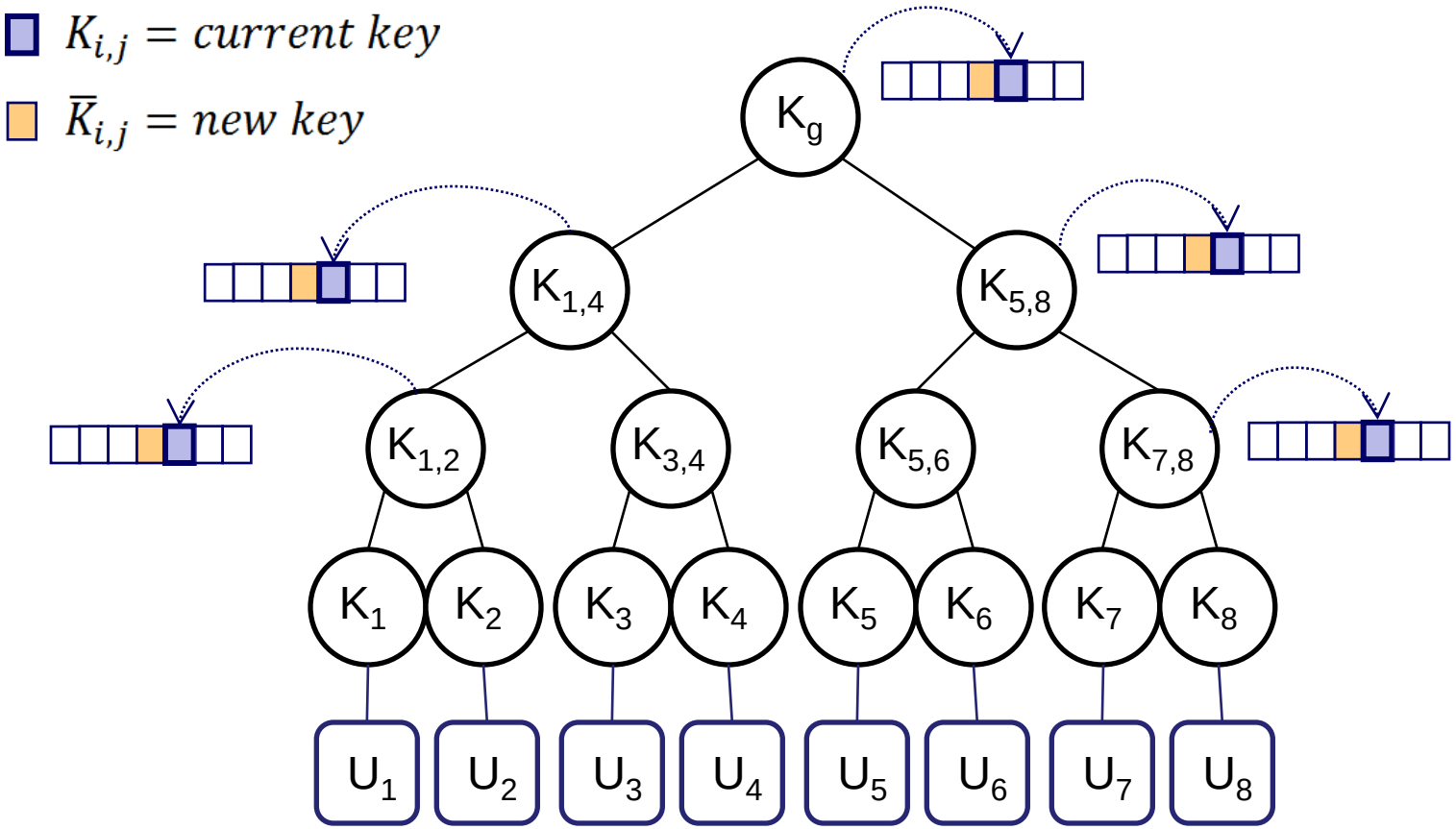


$$K = current\ key$$

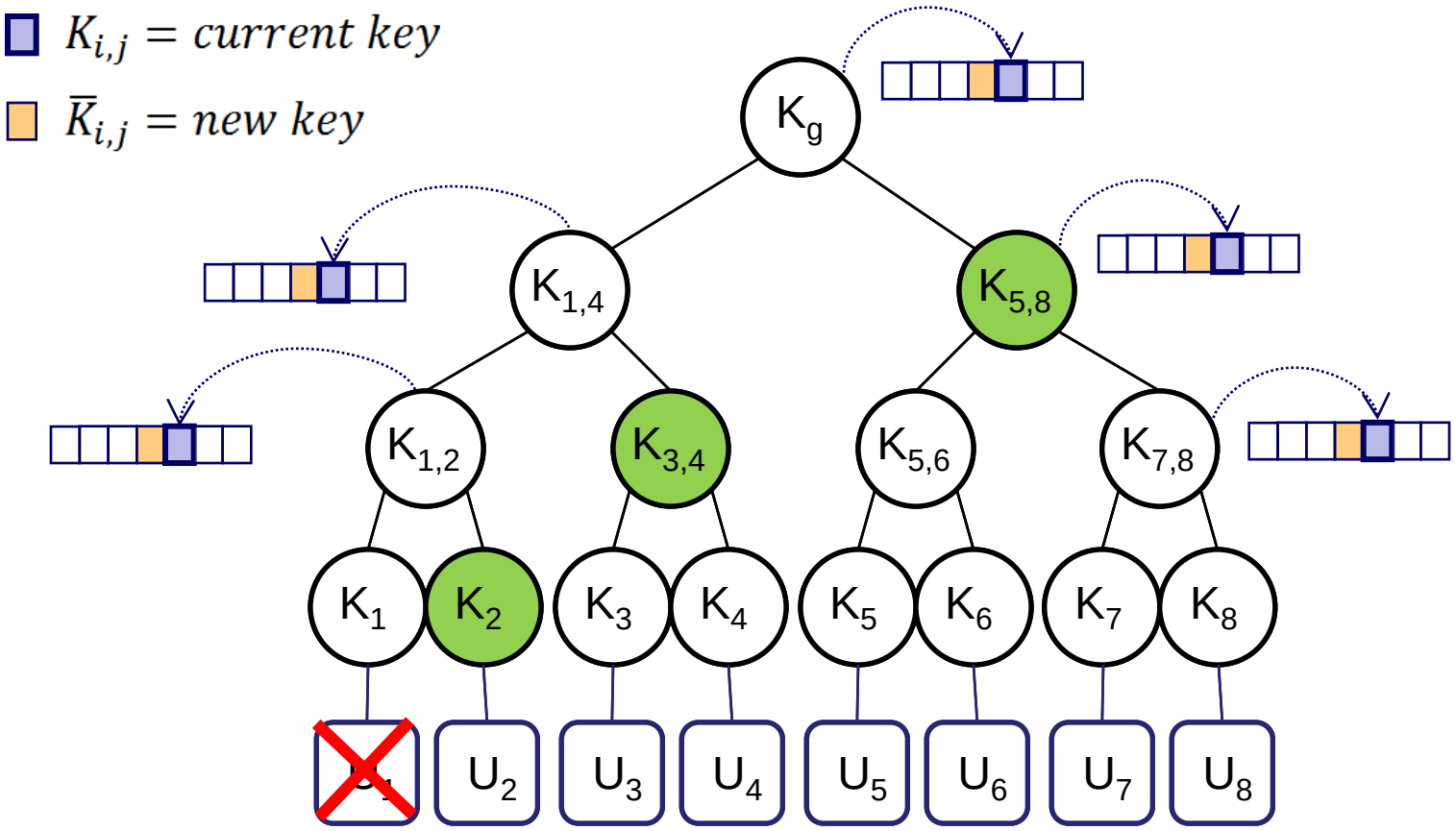
$$\overline{K} = new\ key$$

$$H(\overline{K}) = K$$

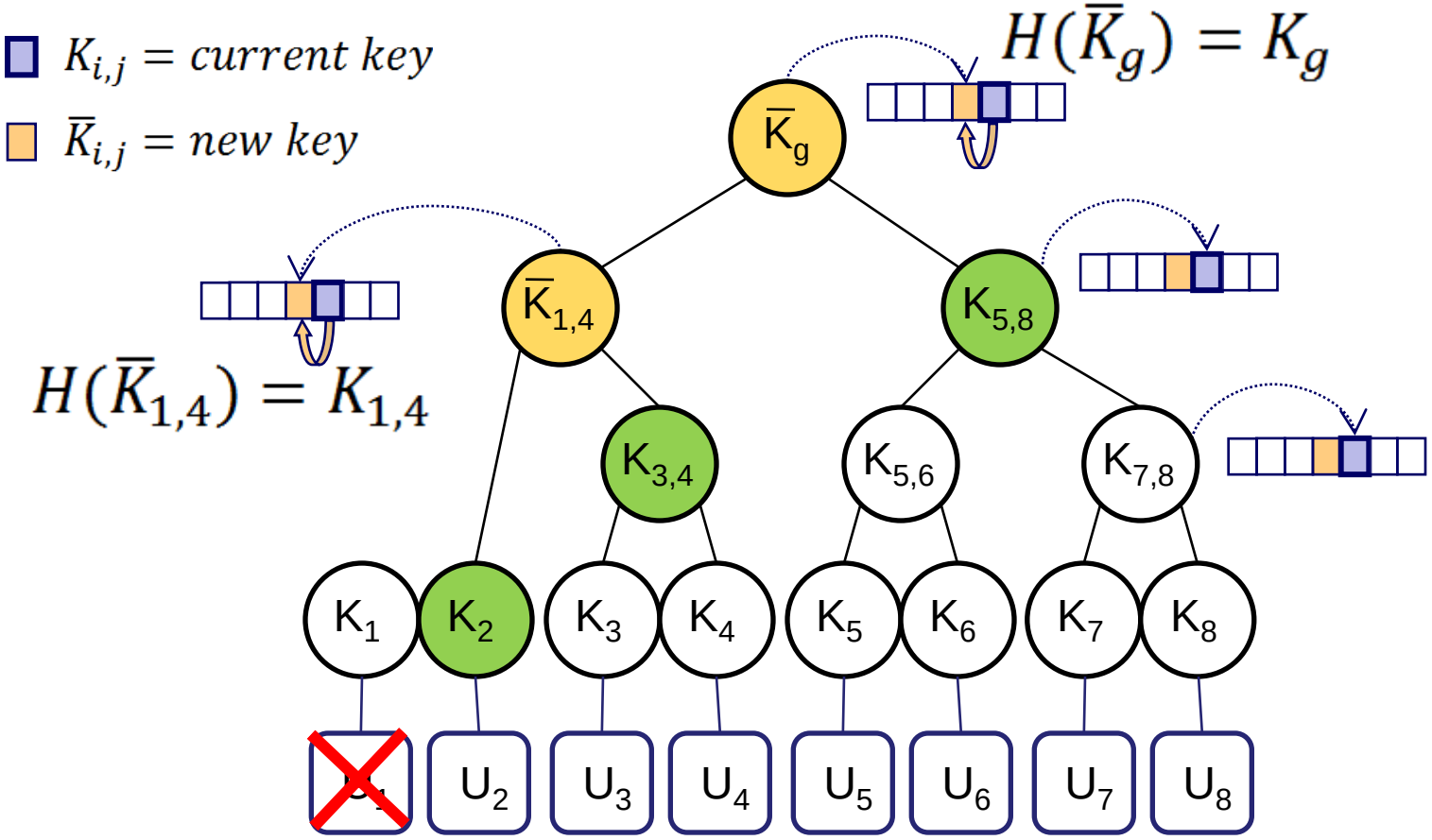
S²RP Funzionamento



S²RP Funzionamento



S²RP Funzionamento



S²RP Valutazione

- Eredita i vantaggi di LKH (efficienza, scalabilità)
- Autenticazione delle chiavi
- Computazionalmente efficiente (funsioni hash)
- Assicura Backward/Forward secrecy
- Collusion resistence

Protocolli Centralizzati

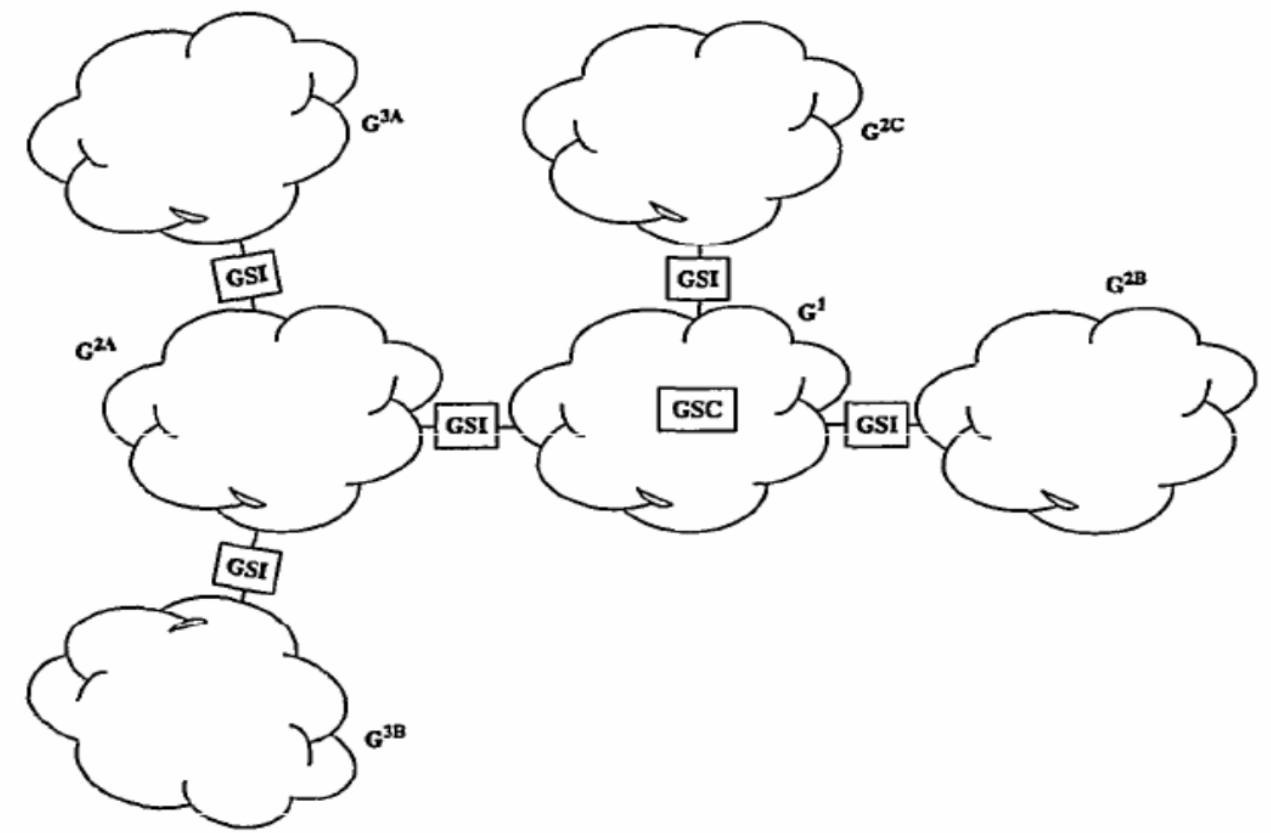
- LKH+, Waldvogel et al. [1999]
- One-way Function Tree (OFT), McGrew e Sherman [1998]
- One-way Function Chain Tree (OFC), Canetti et al. [1999]

Iolus

- Protocollo decentralizzato realizzato da Mittra [1997]
- Il gruppo è diviso in **sottogruppi** ciascuno gestito da un **Group Security Intermediary (GSI)**.
- I **GSI** sono raggruppati in un gruppo di livello superiore gestito da un **Group Security Controller (GSC)**.
- GSC e GSI sono chiamati **Group Security Agent (GSA)**.

Iolus

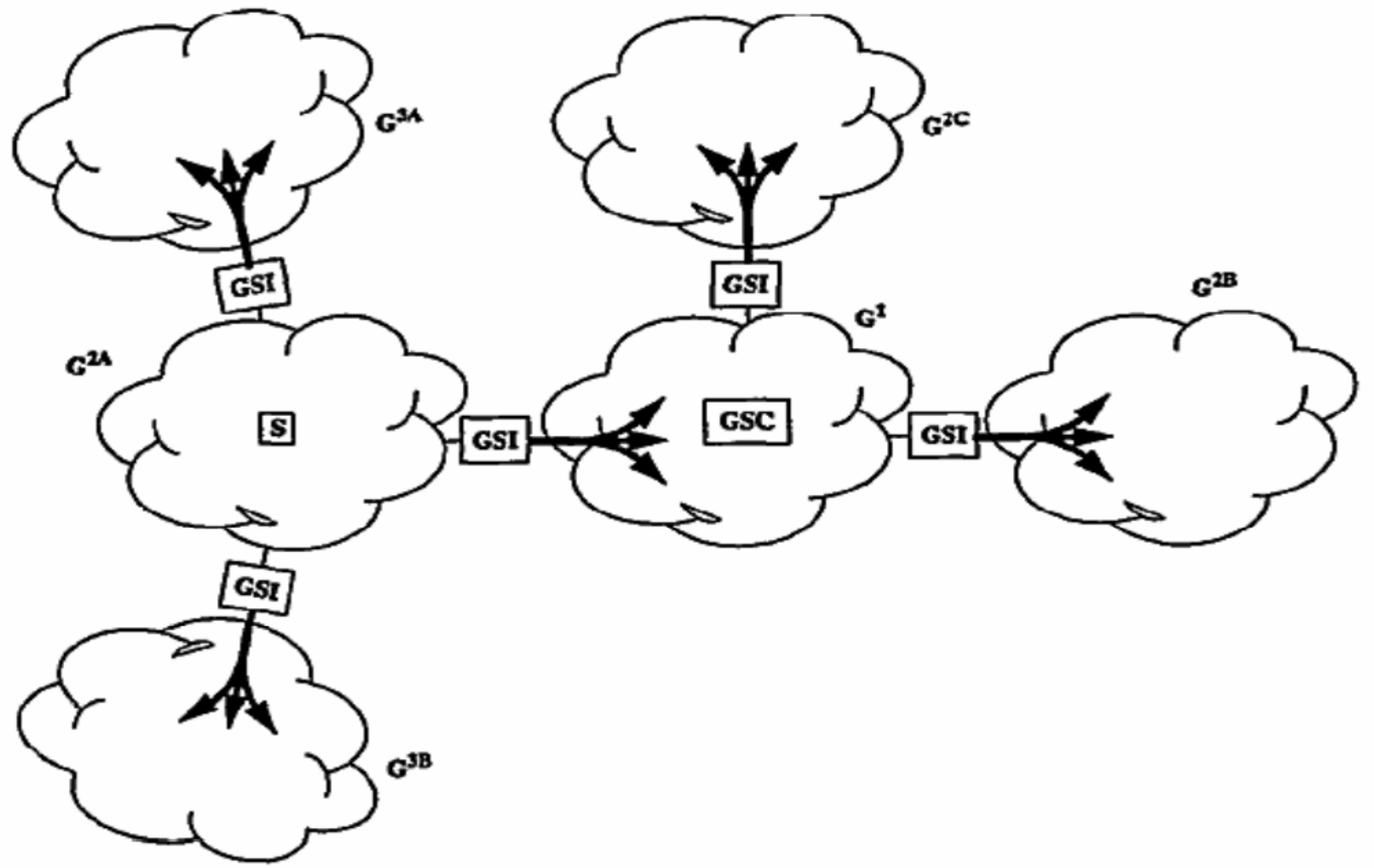
- Iolus usa **chiavi indipendenti (K_{sg})** per ogni sottogruppo. Il rinnovo della chiave viene gestito **localmente**.



GSC & GSI

- GSC:
  - mantiene una visione completa della topologia del gruppo.
  - si occupa della sicurezza tra GSI.
- GSI:
  - Controlla un sottogruppo.
  - Coppie di GSI formano dei bridge tra sottogruppi, permettendo lo scambio di dati tra sottogruppi.
  - Ciascun sottogruppo è gestito con un approccio simile a SKDC

Propagazione Dati



Iolus Performance

Pro:

- Utilizza chiavi indipendenti
- Rekeying locale, riduce gli effetti del problema conosciuto come *1-affects-n*
- Tolleranza ai guasti: il guasto di un GSI non compromette il funzionamento dei rimanenti sottogruppi

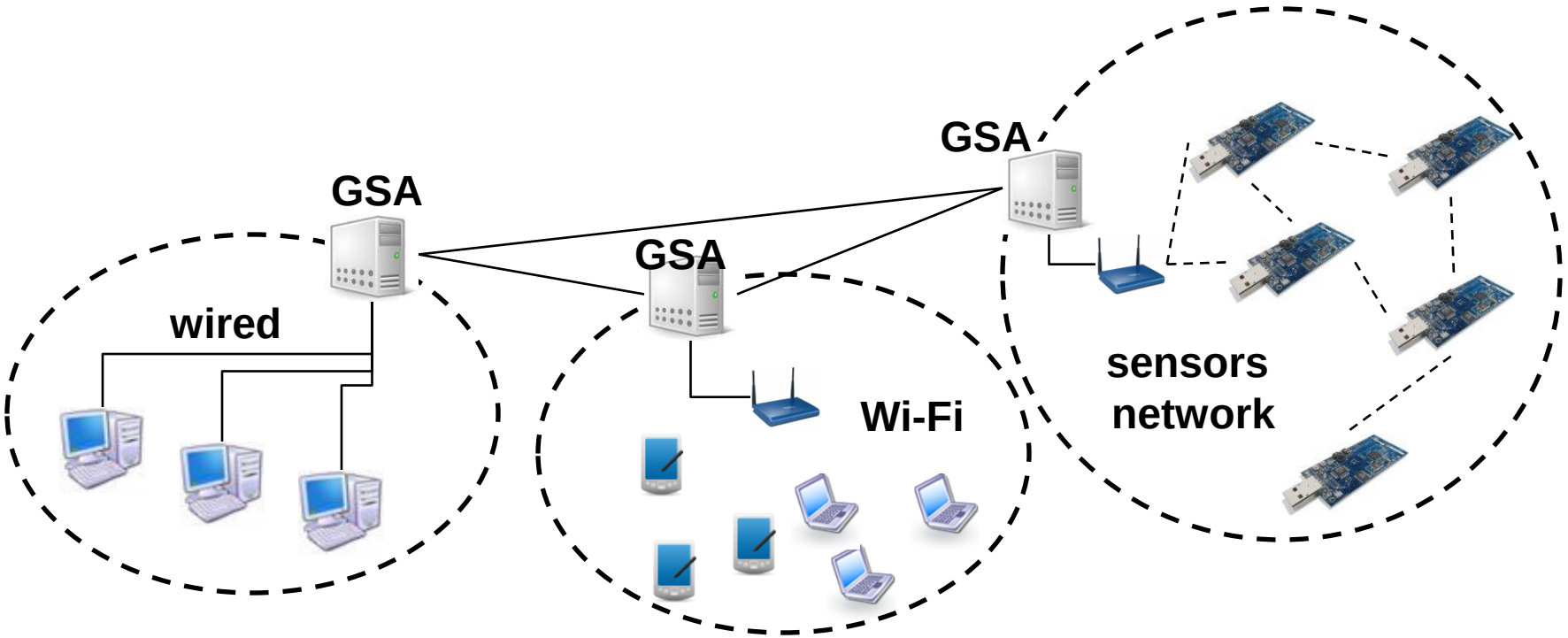
Contro:

- Complessa configurazione del sistema
- Ritardi introdotti nella propagazione dei dati, dovuti alle operazioni di traduzione da sottogruppo a sottogruppo

Applicazioni di Iolus

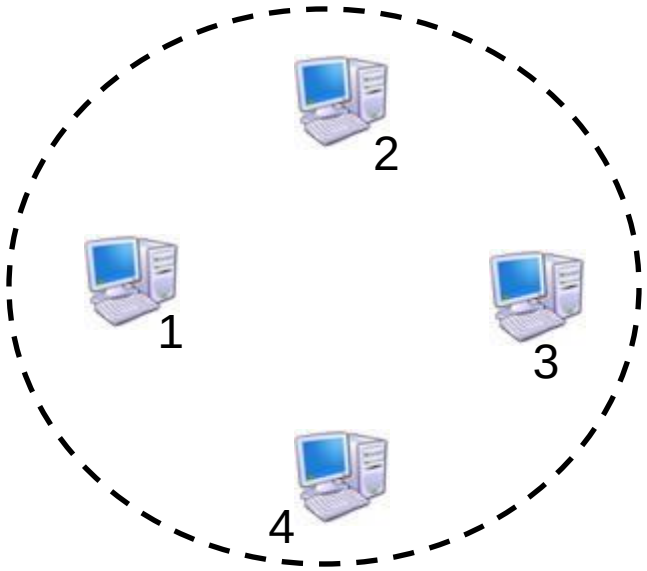
Iolus viene utilizzato in reti eterogenee interconnesse.

I sottogruppi coincidono con le varie infrastrutture di rete.



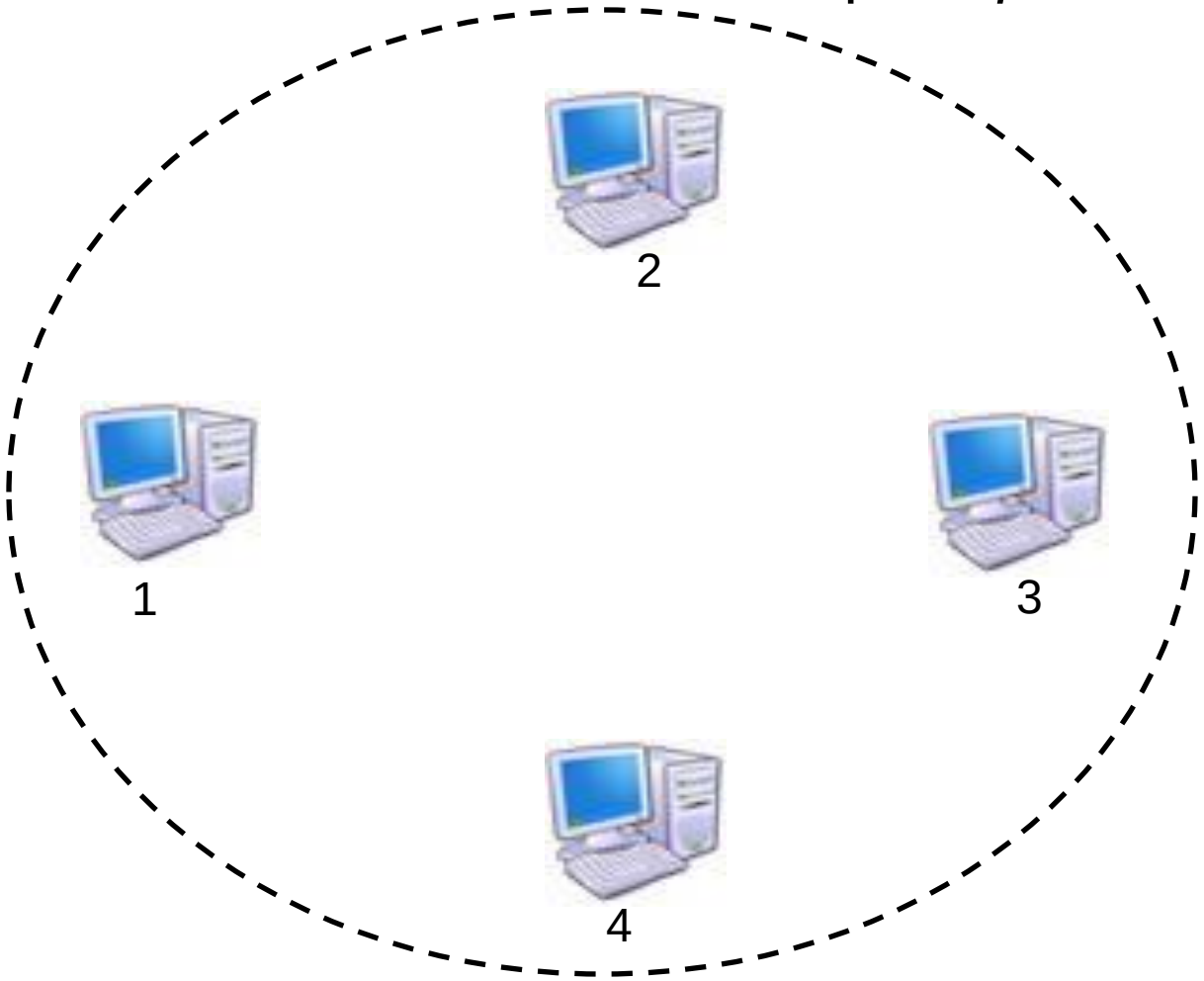
Group Diffie-Hellman Key Exchange

- Protocollo **distribuito** proposto da Steiner et al. [1996]
- Estensione del protocollo DH.
- Un gruppo di *n* elementi concordano una chiave condivisa



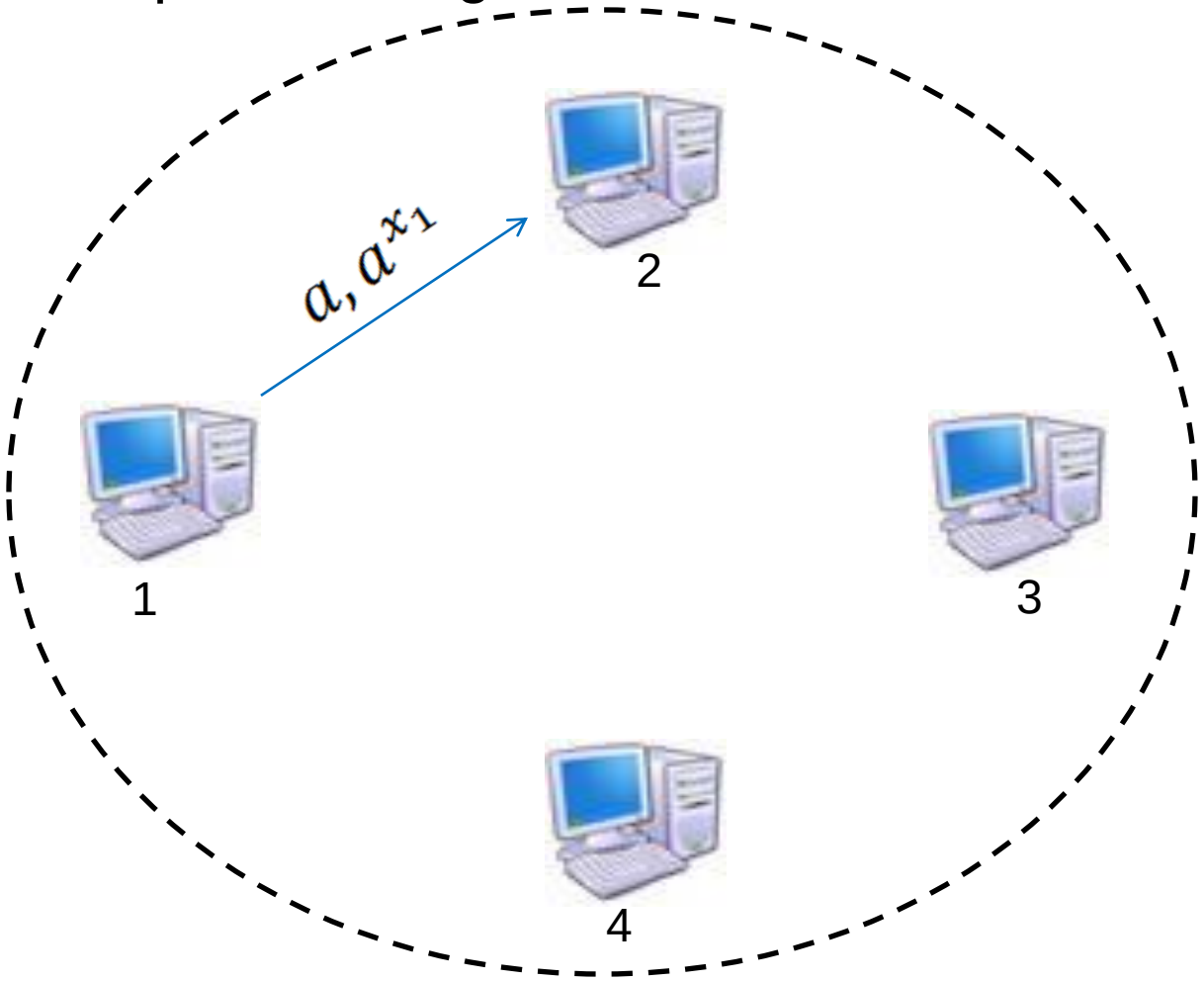
Descrizione protocollo (GDH)

- I membri concordano due numeri primi *p* e *a*



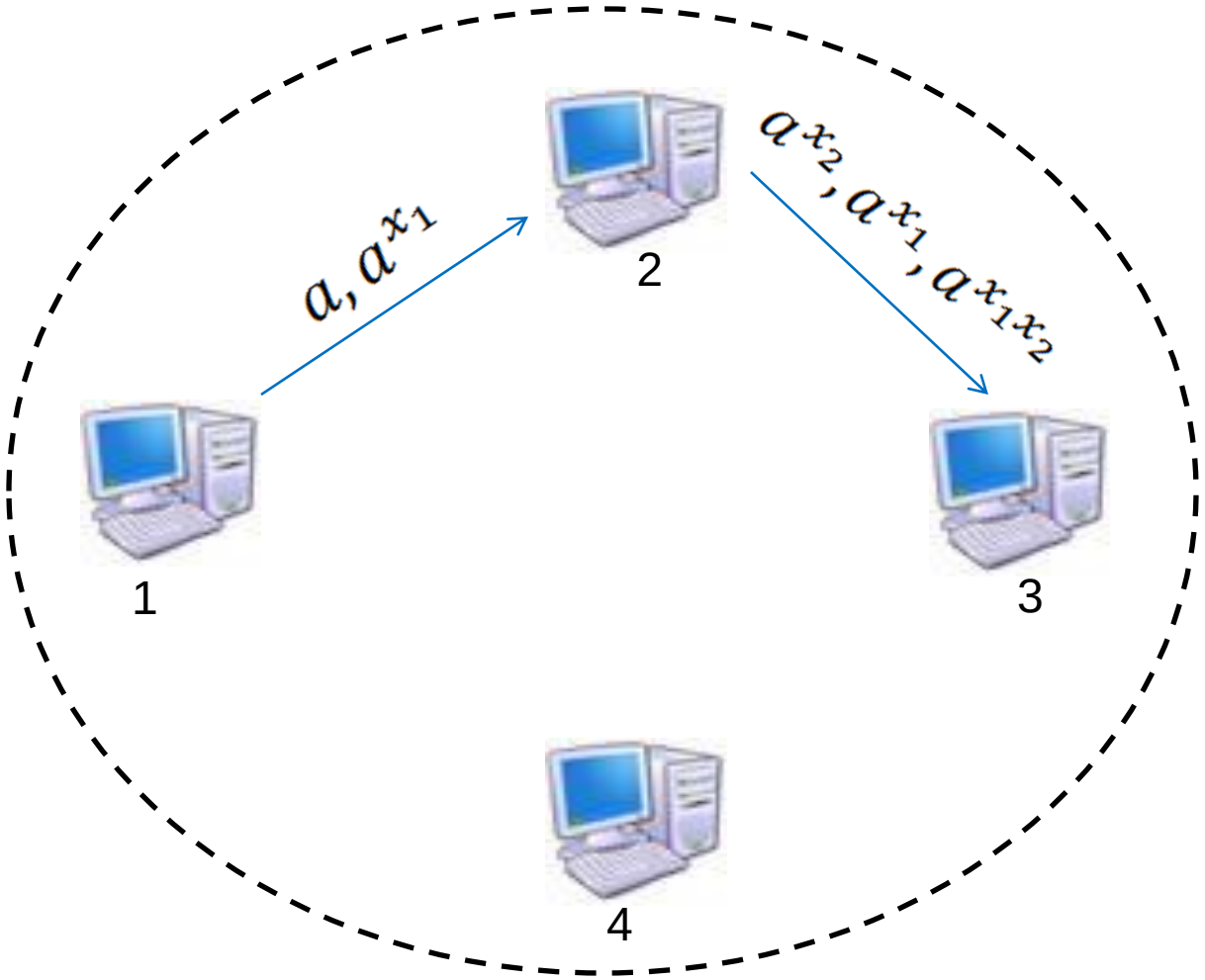
Descrizione protocollo (GDH)

U1 inizia la procedura generando un numero casuale x_1

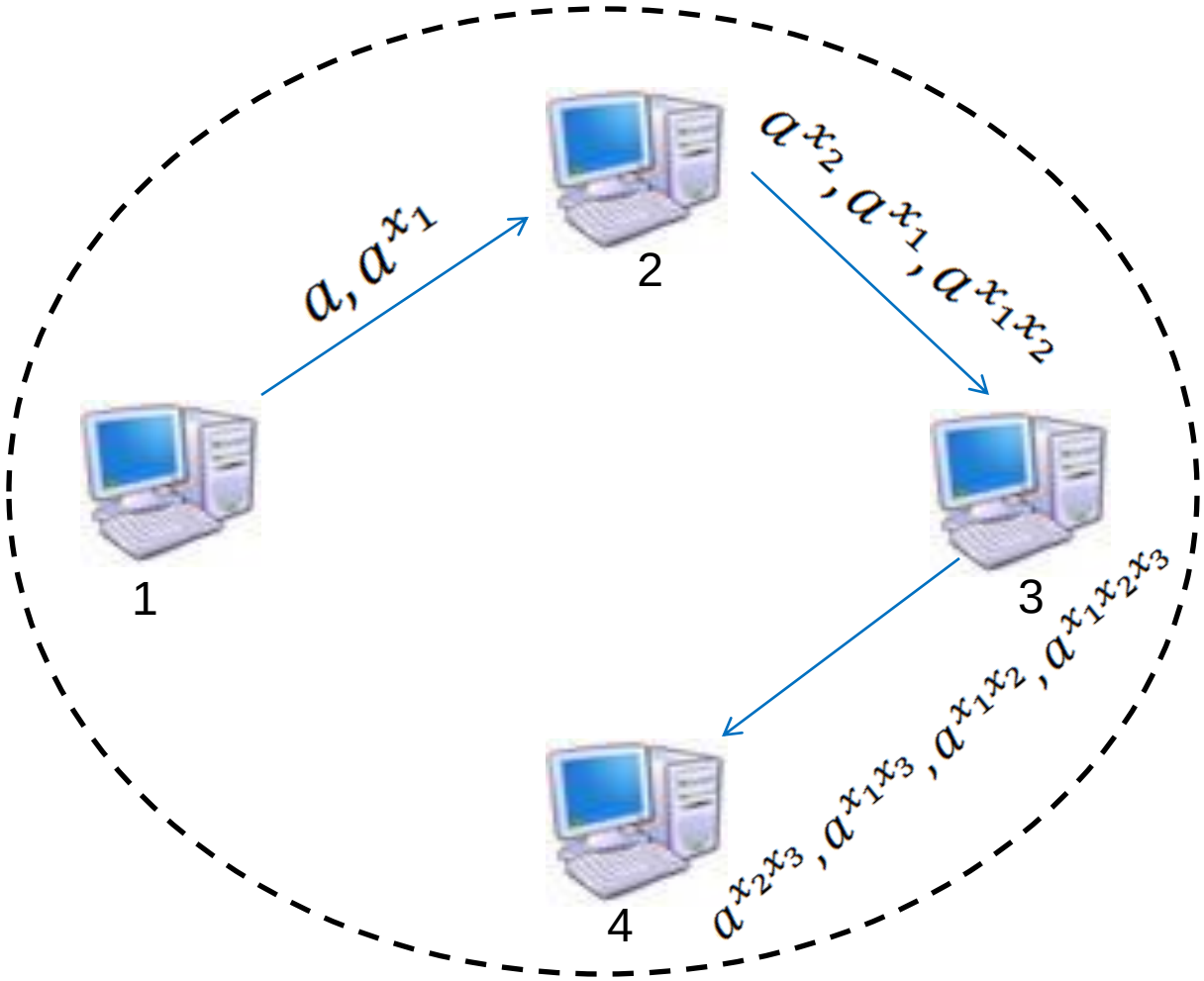


Descrizione protocollo (GDH)

Ciascun Ui contribuisce con un numero casuale x_i

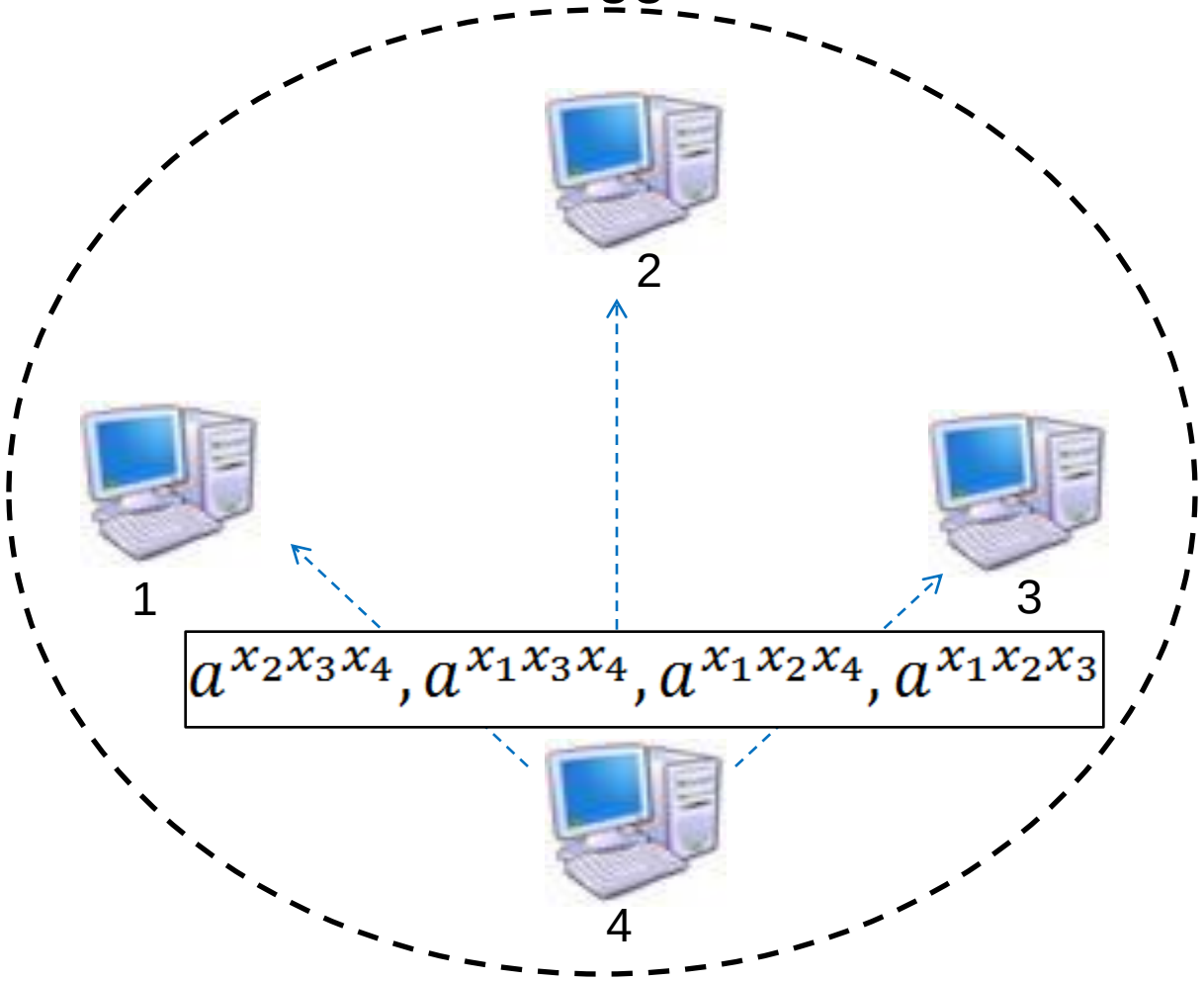


Descrizione protocollo (GDH)



Descrizione protocollo (GDH)

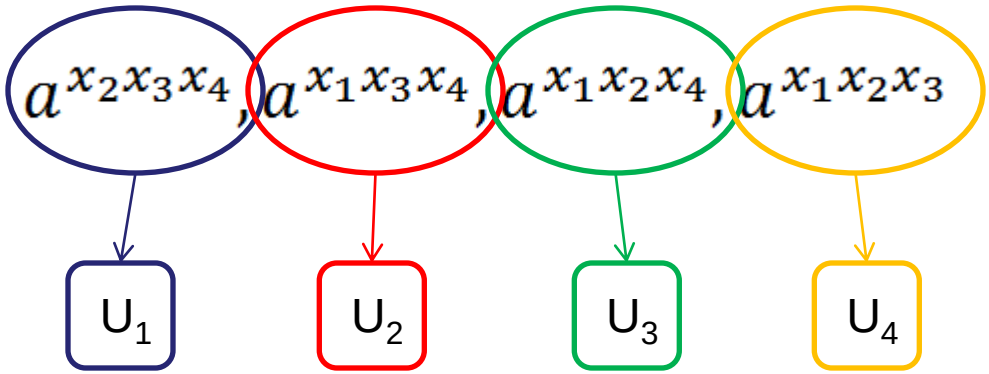
Infine U4 invia il suo messaggio in multicast



Descrizione protocollo (GDH)

Dall'ultimo messaggio qualsiasi membro potrà calcolare la chiave di gruppo che sarà:

$$K_g = (a^{x_1 x_2 x_3 x_4}) mod\ p$$



GDH Performance

Pro:

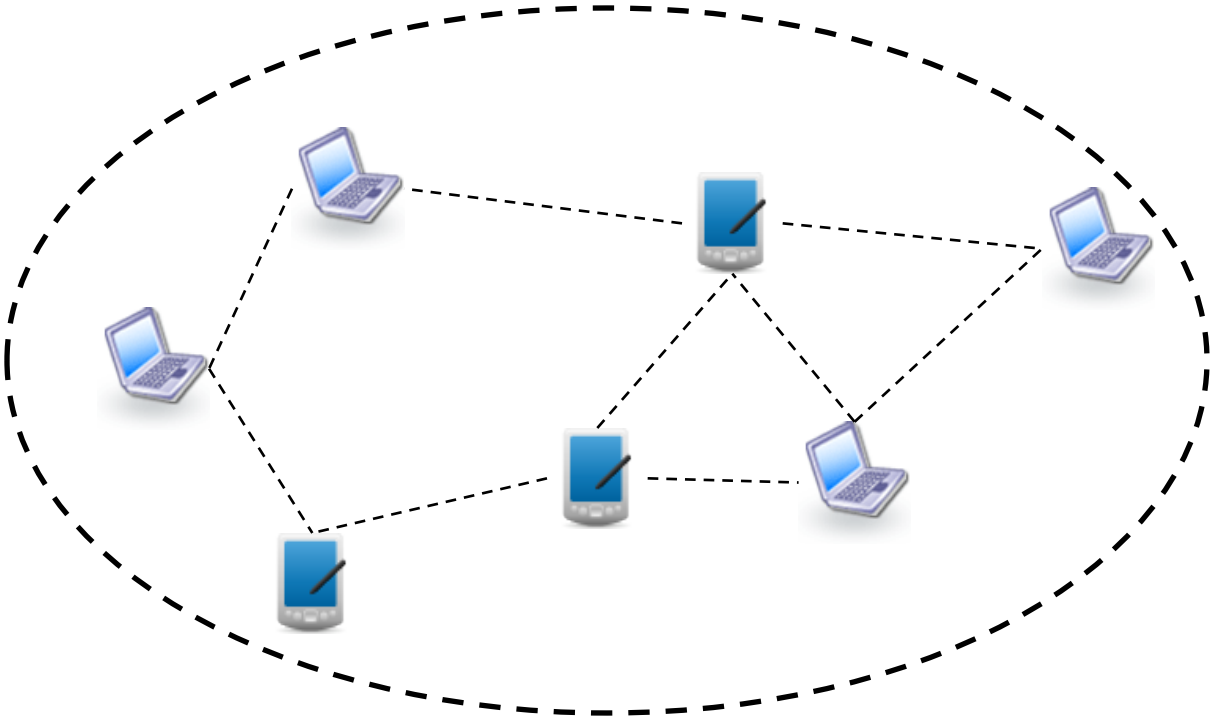
- non richiede KM
- è tollerante ai guasti

Contro:

- In un gruppo con *n* elementi, GDH impiega ***n* rounds** e ***n* messaggi** per generale la chiave di gruppo.
- Le operazioni di join/leave si traducono in una nuova creazione di un gruppo con/senza un membro.
- La dimensione dei messaggi aumenta con l'aumentare di *n.*
- Per i precedenti motivi il protocollo è **poco scalabile.**

Applicazioni

Solitamente gli algoritmi distribuiti si utilizzano in reti di dispositivi mobili, dove non è presente un'entità centrale di coordinamento.



Protocolli distribuiti

- Distributed Logiacl Key Hierarchy (D-LKH), Rodeh et al. [2000]
- Distributed One-way Function Tree (D-OFT), Dondeti et al. [1999]
- Diffie-Hellman Logical Key Hierarchy(DH-LKH), Perrig [1999] e Kim et al. [2000]

Conclusioni

La continua nascita di nuovi servizi basati sulle comunicazioni multicast/broadcast è stato l'interruttore che ha acceso l'interesse verso meccanismi che rendessero sicure queste comunicazioni.

Sono state ideate molte soluzioni per venire in contro alle diverse esigenze delle applicazioni, che differiscono per architettura, efficienza, scalabilità …

Q&A