Efficient Key Management for Content Access Control in Hierarchies

Dr Hani Ragab

h.ragab@kent.ac.uk

School of Computing

University of Kent

Outline

- Content Access Control in Hierarchies
- Key Management for CACH
- The Resource Precedence Graph
- Discussion

CONTENT ACCESS CONTROL IN HIERARCHIES (CACH)

Content Access Control in Hierarchies

- Occurs in any context with
  - A set of subjects (users, processes, …)
  - A set of resources (servers, files, …)
- Examples include
  - Communications in hierarchical groups
  - Access control to databases
  - Multi-layered media streaming
  - Electronic newspapers
  - Services on mobile phones

CACH: Model

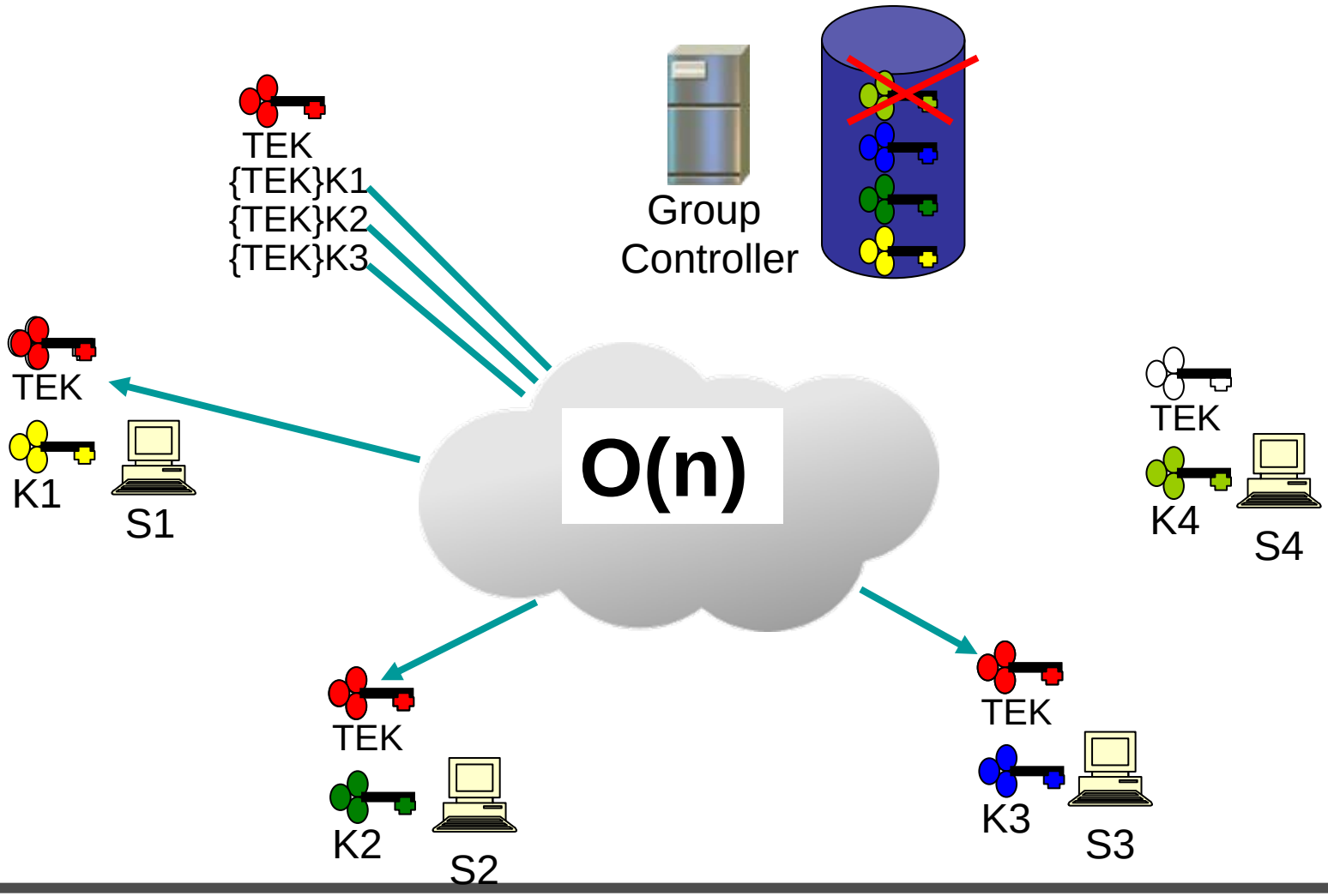
- Encrypt different resources using different keys
  - Resource Encryption Key (REK)
- Each subject needs to know the keys of his resources
- Subjects can change their access rights: REK of corresponding resource must be renewed
  - When the subject has no more access to it: Forward Secrecy
  - When the subject has just been granted access to it: Backward Secrecy

KEY MANAGEMENT FOR CACH

CACH: Key Management

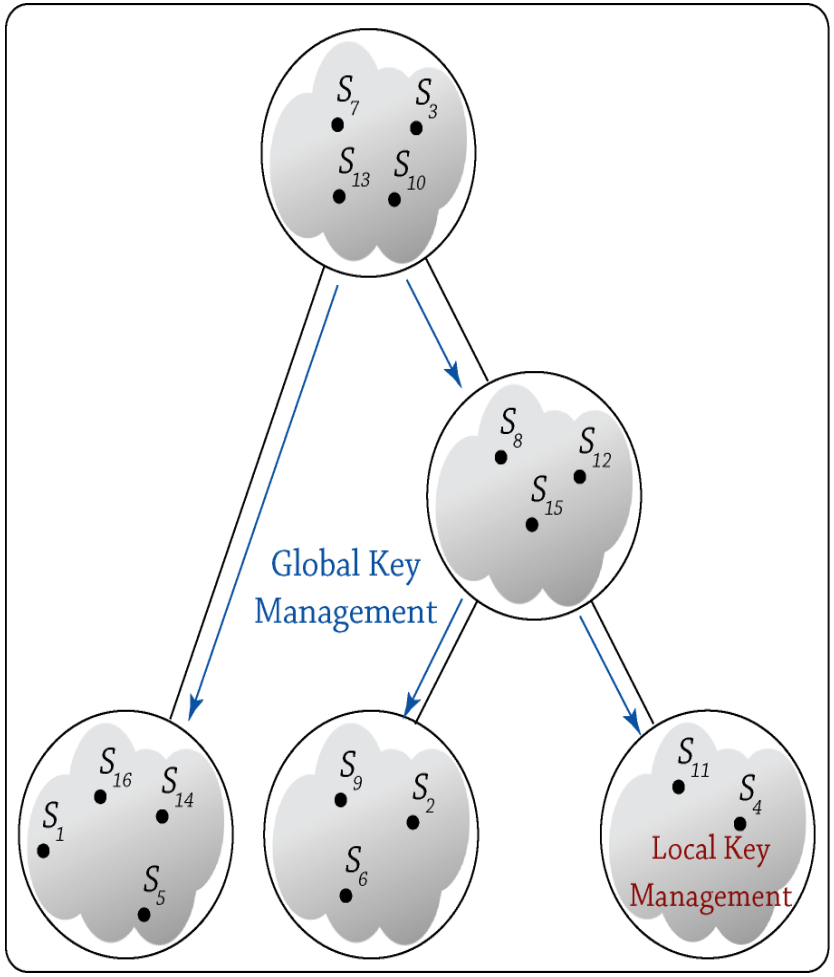
- A key management scheme is required to update REK after each access right change
  - Scalable schemes are required: large hierarchies, frequent changes
  - Prevent the “1 affects n” phenomenon
- Key management schemes regroup subjects into security classes/service groups

CACH: Key Management, 1 affects n

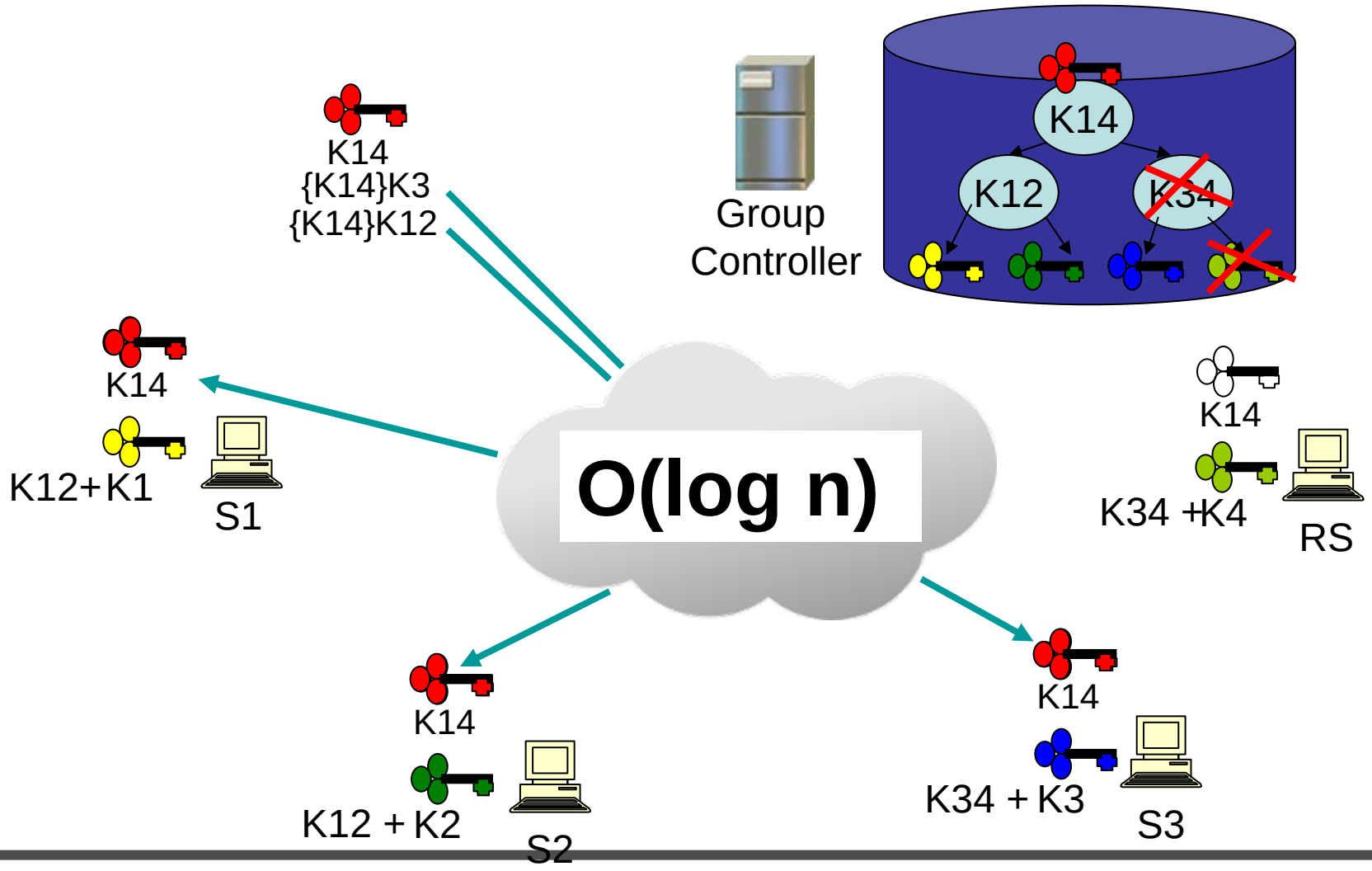


CACH: Key Management

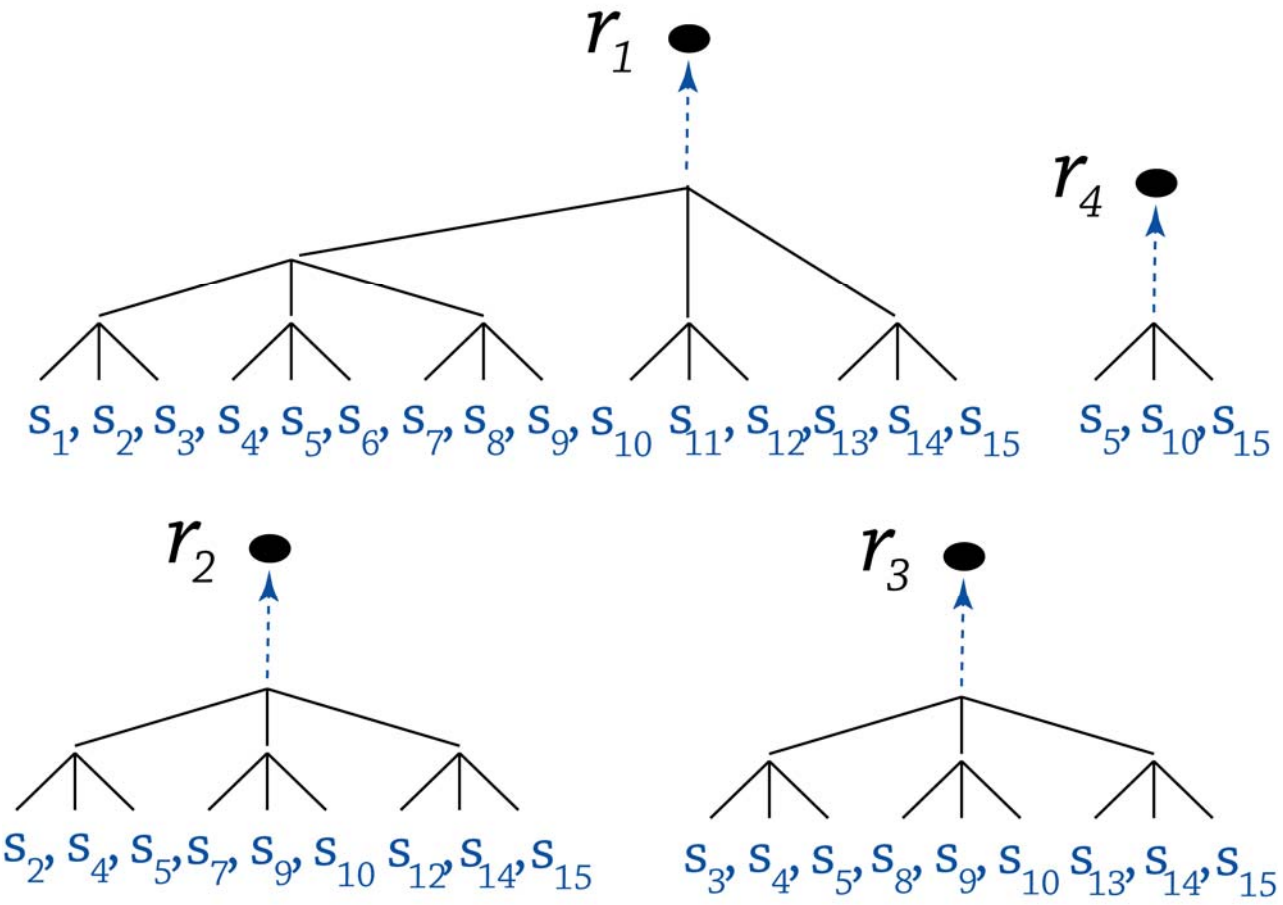
1. Global key management
   - Generate and distribute REK to subjects groups
2. Local key management:
   - Objective: distribute REK to subjects
   - Uses Key Encryption Keys
   - Can use traditional flat-groups key management schemes: LKH, Iolus



CACH: Local Key Management



CACH: Global Key Management



CACH: Key Management

- Problem: each subject needs to know REK of all the resources to which it has access
- Solution: two approaches in the literature
  - Dependent Keys: uses complex cryptographic /mathematical techniques to link keys, time and resource consuming for key generation
  - Independent Keys: the subject stores a set of REK. Main issue: **reduce** the number of REK to store by structuring the subjects

RESOURCE PRECEDENCE GRAPH (RPG)

Resource Precedence Graph

- Precedence
  - r2 |- (precedes) r1 iff any subject that has access to r2, has access to r2 too
- Examples
  - r2 |- r1
  - r3 |- r1
  - r4 |- r1, r2
  - r2 |- r3? r3 |- r2?

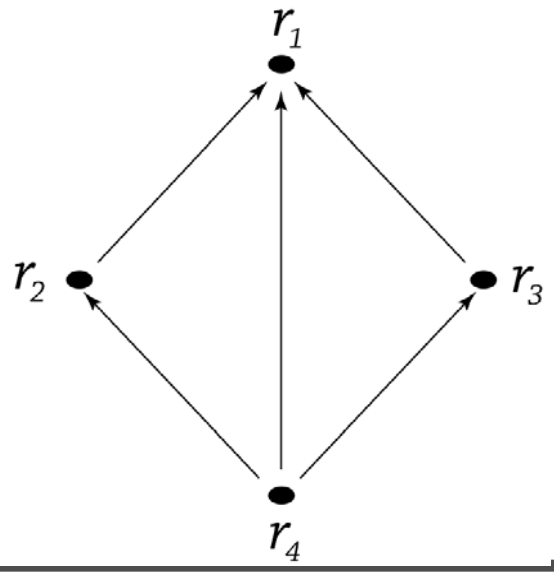
$$\mathcal{AR}' =$$

	r_1	r_2	r_3	r_4
s_1	1	0	0	0
s_2	1	1	0	0
s_3	1	0	1	0
s_4	1	1	1	0
s_5	1	1	1	1
s_6	1	0	0	0
s_7	1	1	0	0
s_8	1	0	1	0
s_9	1	1	1	0
s_{10}	1	1	1	1
s_{11}	1	0	0	0
s_{12}	1	1	0	0
s_{13}	1	0	1	0
s_{14}	1	1	1	0
s_{15}	1	1	1	1

RPG: Precedence Matrix, Core

- Calculating the precedence matrix
  - Suppose that r_{j1} precedes r_{j2}
  - If a subject s_i verifies $ar_{i,j1} = 1$ and $ar_{i,j2}=0$, remove the precedence relation
- The core of the RPG is built using this matrix

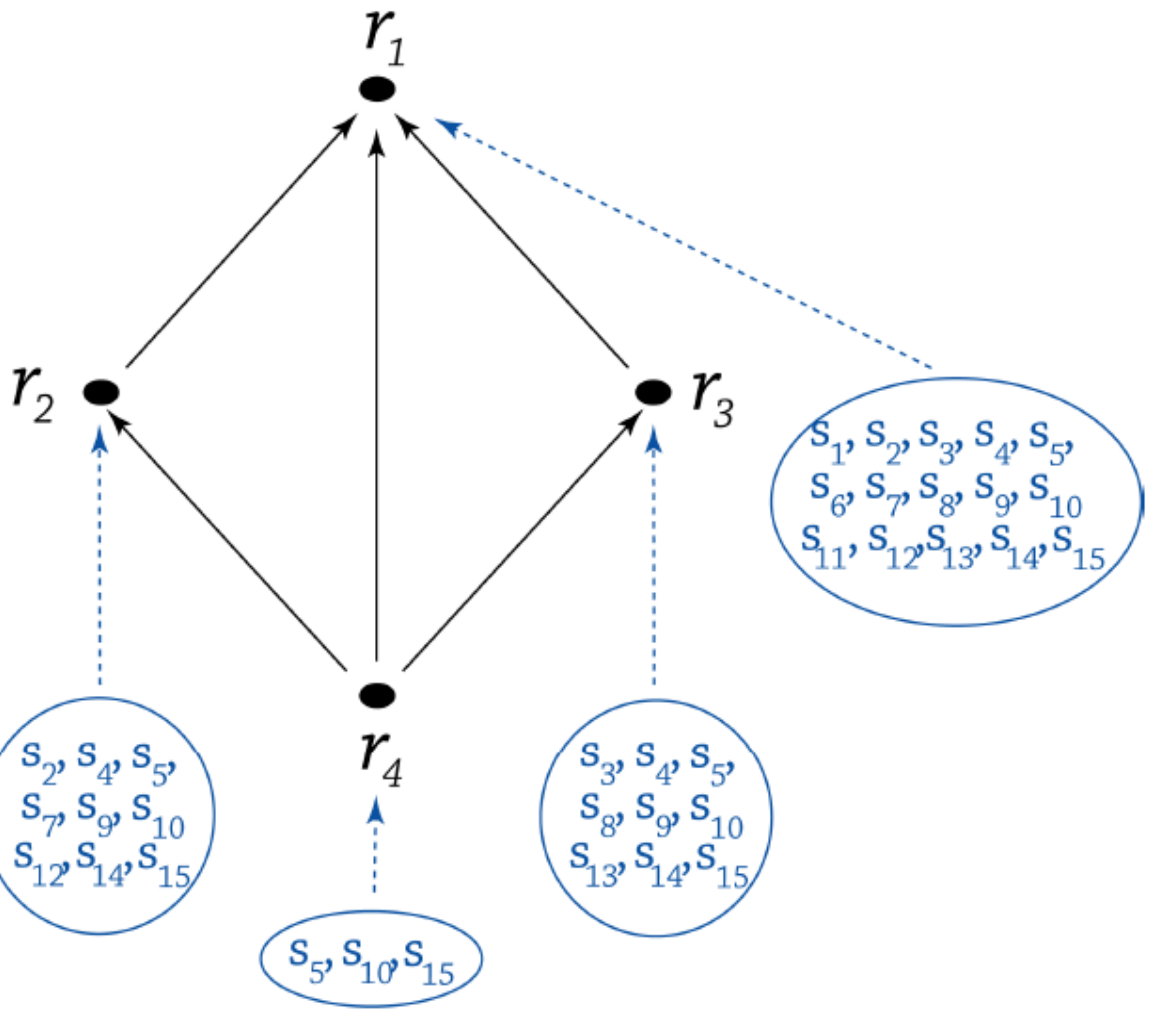
$$\mathcal{P}' = \begin{array}{c|cccc} & r_1 & r_2 & r_3 & r_4 \\ r_1 & 0 & 0 & 0 & 0 \\ r_2 & 1 & 0 & 0 & 0 \\ r_3 & 1 & 0 & 0 & 0 \\ r_4 & 1 & 1 & 1 & 0 \end{array}$$



Building the RPG: Initialization

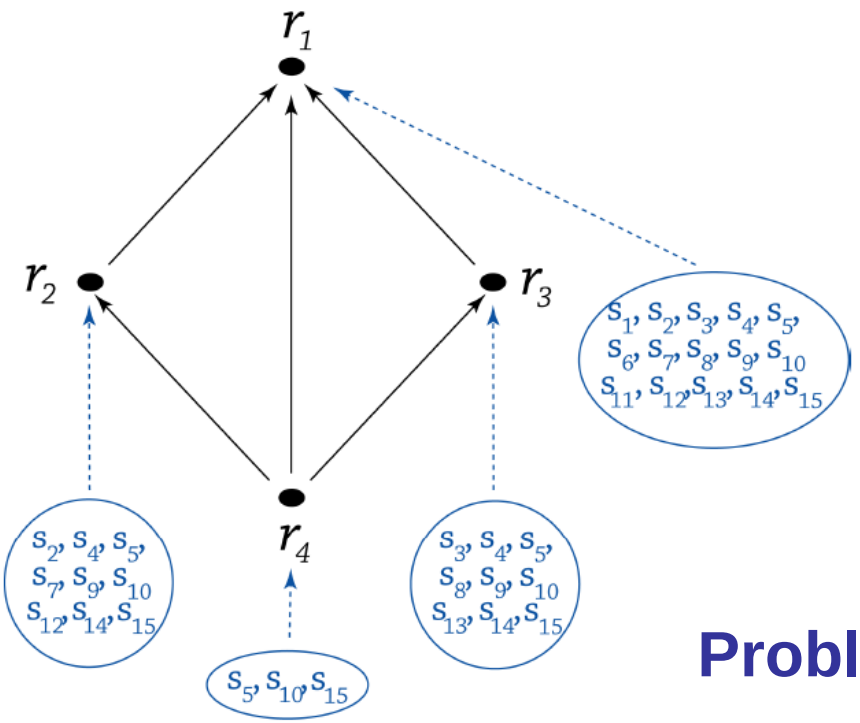
$\mathcal{AR}' =$

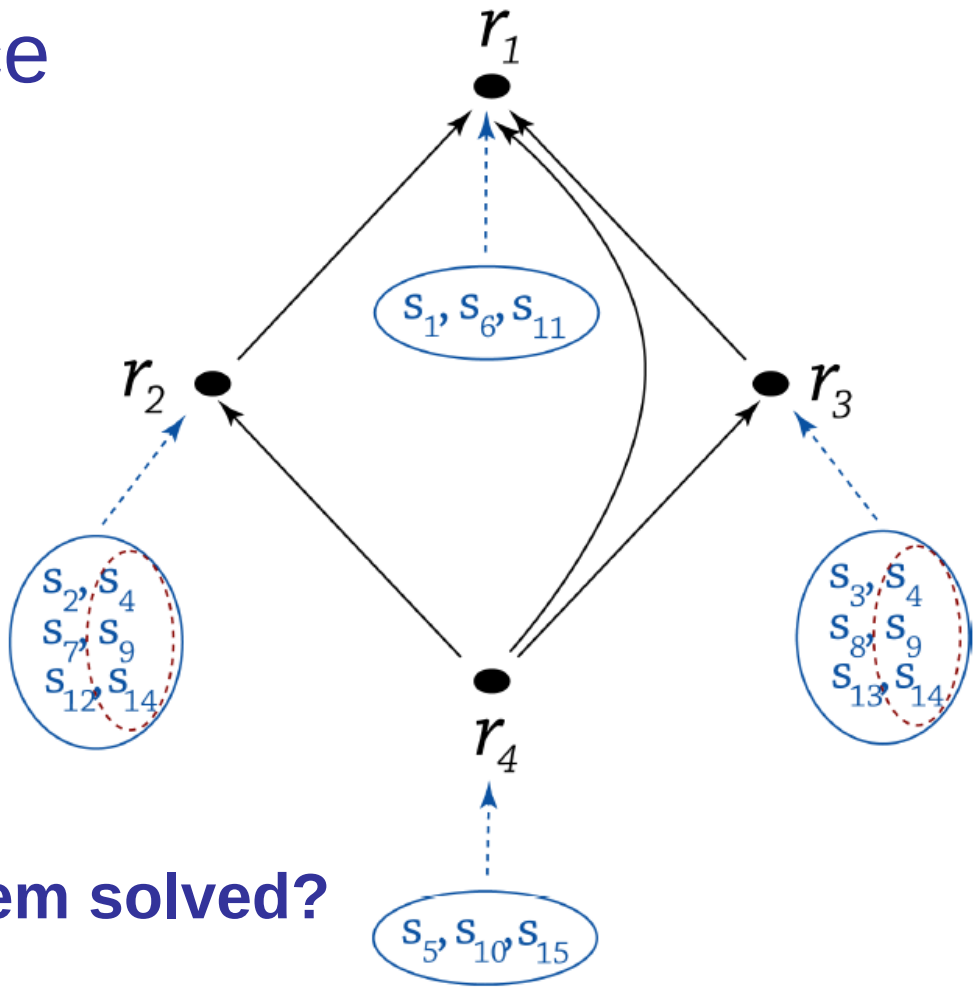
	r_1	r_2	r_3	r_4
s_1	1	0	0	0
s_2	1	1	0	0
s_3	1	0	1	0
s_4	1	1	1	0
s_5	1	1	1	1
s_6	1	0	0	0
s_7	1	1	0	0
s_8	1	0	1	0
s_9	1	1	1	0
s_{10}	1	1	1	1
s_{11}	1	0	0	0
s_{12}	1	1	0	0
s_{13}	1	0	1	0
s_{14}	1	1	1	0
s_{15}	1	1	1	1



Building the RPG: Phase 1

- Remove precedence redundancies

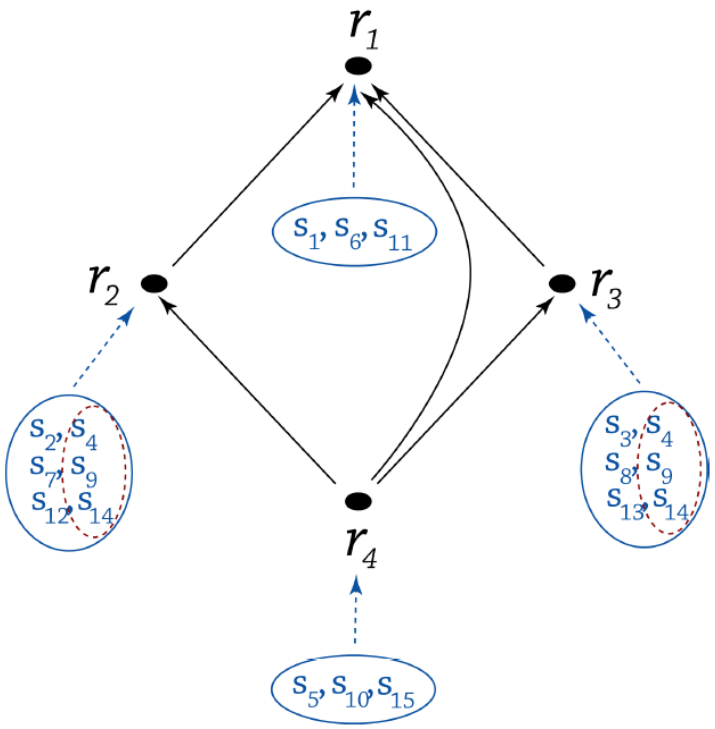


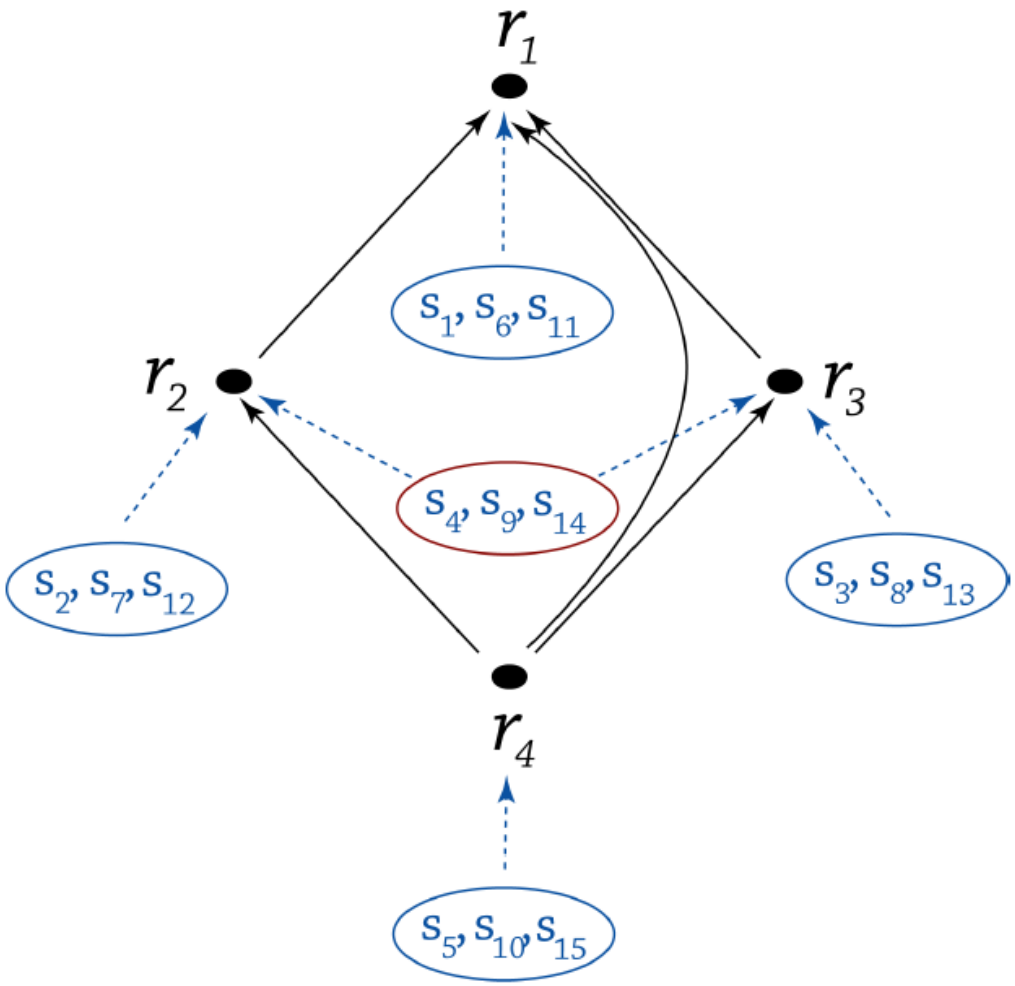


Problem solved?

Building the RPG: Phase 2

- Remove parallel redundancies





RPG: Key Management

- The resulting subgroups of subjects are service groups
  - Subjects have the same access rights
- Assume that subject s leaves SG_i to SG_j:
  - Global key management: all REK in $(REK(SG_i)\text{-}REK(SG_j) \cup (REK(SG_j)\text{-}REK(SG_i))$ must be renewed
  - Local key management: any local key management scheme can be used to update the REK

Discussion

- We evaluate key management schemes using two criteria:
  - Storage: the number of keys stored by subjects;
  - Bandwidth: the number of messages that are sent to update subjects' keys after a key renewal
- Each of these criterion has a local and a global component

Discussion

- Local storage: $local\_storage(RPG) = f(|SG(s)|)$
- Global storage: $global\_storage(RPG) = |Res(s)|$
- Local bandwidth:

$$\sum_{u \in SG'(s)} g(|SG'(s)|) + \sum_{u \in SG(s)} g(|SG(s)|)$$

- Global bandwidth:

$$|(Keys'(s)\text{-}Keys(s)) \cup (Keys(s)\text{-}Keys'(s))|$$

Discussion

- Comparison to other schemes (global)

Approach	Bandwidth	Storage
Dependent : Indirect	$O(N_{SG})$	$O(\log(N_{SG}))$
Dependent : Direct	$O(N_{SG})$	$O(\log(N_{SG}))$
Independant	$O(\log(N_{SG}))$	$O(\log(N_{SG}))$
RPG	**$O(\log(N_R))$**	**$O(N_R)$**

- **H. Ragab Hassan, A. Bouabdallah, H. Bettahar, and Y. Challal**, "*Key management for content access control in a hierarchy*", 51(11), 3197-3219, **Computer Networks**, August 2007.

Future Works

- A formal proof of optimality of RPG for both criteria

- Investigating the existence of optimal dependent key management scheme
  - Define optimality criteria
  - Propose a common abstract model

QUESTION?