I·C·S
The Institute for Cyber Security

C·SPECC
Center for Security and Privacy Enhanced Cloud Computing

On the Feasibility of RBAC to ABAC Policy Mining: A Formal Analysis

Shuvra Chakraborty[1], Ravi Sandhu[1] and Ram Krishnan[2]

[1]Dept. of Computer Science, [2]Dept. of Electrical and Computer Engineering
[1,2]Institute for Cyber Security
University of Texas at San Antonio, TX 78249, USA

7th International Conference on Secure Knowledge Management in Artificial Intelligence Era (SKM '19)
Goa, India, December 21-22, 2019

Problem Motivation

- RBAC (Role-Based Access Control) is widely used but has notable limitations (e.g., role explosion)
- Using ABAC (Attribute-Based Access Control), access control policies can be written in more flexible and higher level way
- Automated migration of an existing RBAC system to ABAC system (defined as ABAC policy mining problem) cuts the cost and human efforts needed
- Stoller et. al. use explicit unique IDs in attribute set to resolve ABAC policy mining problem which is somehow conflicting with basic principle of ABAC

❖ *We introduce ABAC RuleSet Existence problem: questions the feasibility of ABAC policy mining problem in RBAC context*
  ❖ *If not feasible without ID, infeasibility correction technique is applied*
  ❖ *Eliminates use of explicit ID in ABAC policy mining*

Problem Background

- Policy mining (e.g., ABAC policy mining, Role mining problem, etc.)
  - ❖ helps to reduce the cost of migrating from an existing access control system to another
- ABAC policy mining
  - ❖ finding an equivalent ABAC system while an existing access control system and supporting data (e.g., attribute names, value assignment) are given (Introduced by Stoller et. al. in 2014)
  - ❖ works are available (migrate from ACL, RBAC, log data, sparse log, etc.)
- Role mining problem
  - ❖ finding set of roles / user-role assignment / role-permission assignment when optimization criteria and/or supporting data are provided
  - ❖ heavily explored (Survey of Role Mining by Mitra et. al. in 2016)

Terminologies (1/4)

1. Access control

2. An Access control system must have a checkAccess function which evaluates an access request (user, object, operation) to true/false

3. Two access control systems are equivalent iff i) set of users (U), objects (O), and operations (OP) are identical ii) for any access request, $checkAccess_{system1}$ and $checkAccess_{system2}$ evaluates the same

4. Our study includes 3 types of Access Control System
   a. Enumerated Authorization System (EAS)
   b. RBAC System
   c. ABAC System

Terminologies (2/4)

- EAS is a tuple <U, O, OP, AUTH, $checkAccess_{EAS}$>
- U, O, and OP are finite sets of users, objects and operations, respectively
- AUTH $\subseteq$ UxOxOP
- *Example 1:*
- U = {John, Lina, Ray, Tom}, OP = {read, write}, O = {Obj1, Obj2}

AUTH	Explanation
(John, Obj1, write) (John, Obj2, write) (John, Obj1, read) (Lina, Obj2, write) (Tom, Obj1, read) (Ray, Obj1, read)	e.g., John is allowed to do read operation on Obj1 but not allowed to do write operation on Obj1

***Our previous work: Feasibility of EAS to ABAC policy mining

Terminologies (3/4)

- RBAC system is a tuple <U, O, OP, Roles, RPA, RUA, RH, $checkAccess_{RBAC}$>
  - RPA : Role Permission Assignment
  - RUA: Role User Assignment
  - Permission is an object-operation pair
  - RH is the role hierarchy relation

Example 2:

- U = {John, Lina, Ray, Tom}, OP = {read, write}, O = {Obj1, Obj2} [same as Example 1]
- Roles = {R1, R2, R3}
- RPA(R1) = {(Obj1, write)}, RPA(R2) = {(Obj2, write)}, RPA(R3) = {(Obj1, read)}
- RUA(R1) = {John}, RUA(R2) = {Lina}, RPA(R3) = {Ray, Tom}
- RH={(R1,R2), (R1, R3)} [R1 is a senior role than R2, R3]

***EAS and RBAC system defined in example 1 and 2 are equivalent

Terminologies (4/4)

- ABAC system is a tuple <U, O, OP, UA, OA, UAValue, OAValue, RangeSet, RuleSet, $checkAccess_{ABAC}$ >
- *Example 3:*
- U, O, OP are same as Example 1
- UA ={Position, Dept.}, OA = {Type}

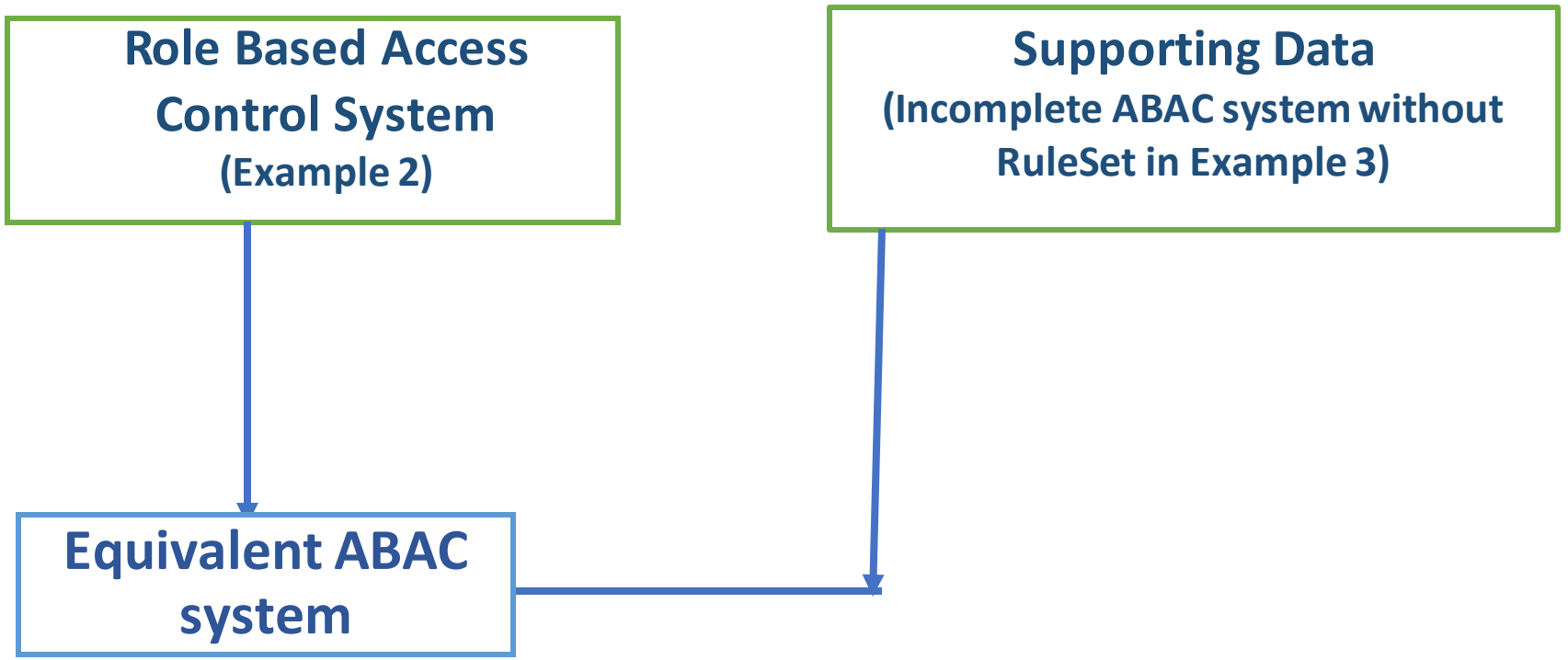
UAValue		
User (U)	**Position**	**Dept.**
John	Officer	CS
Lina	Student	CS
Ray	Officer	CS
Tom	Officer	CS

RangeSet	
Position	{Officer, Student, Faculty}
Dept.	{CS, EE}
Type	{File, Printer, Scanner}

OAValue	
Object (O)	**Type**
Obj1	File
Obj2	Printer

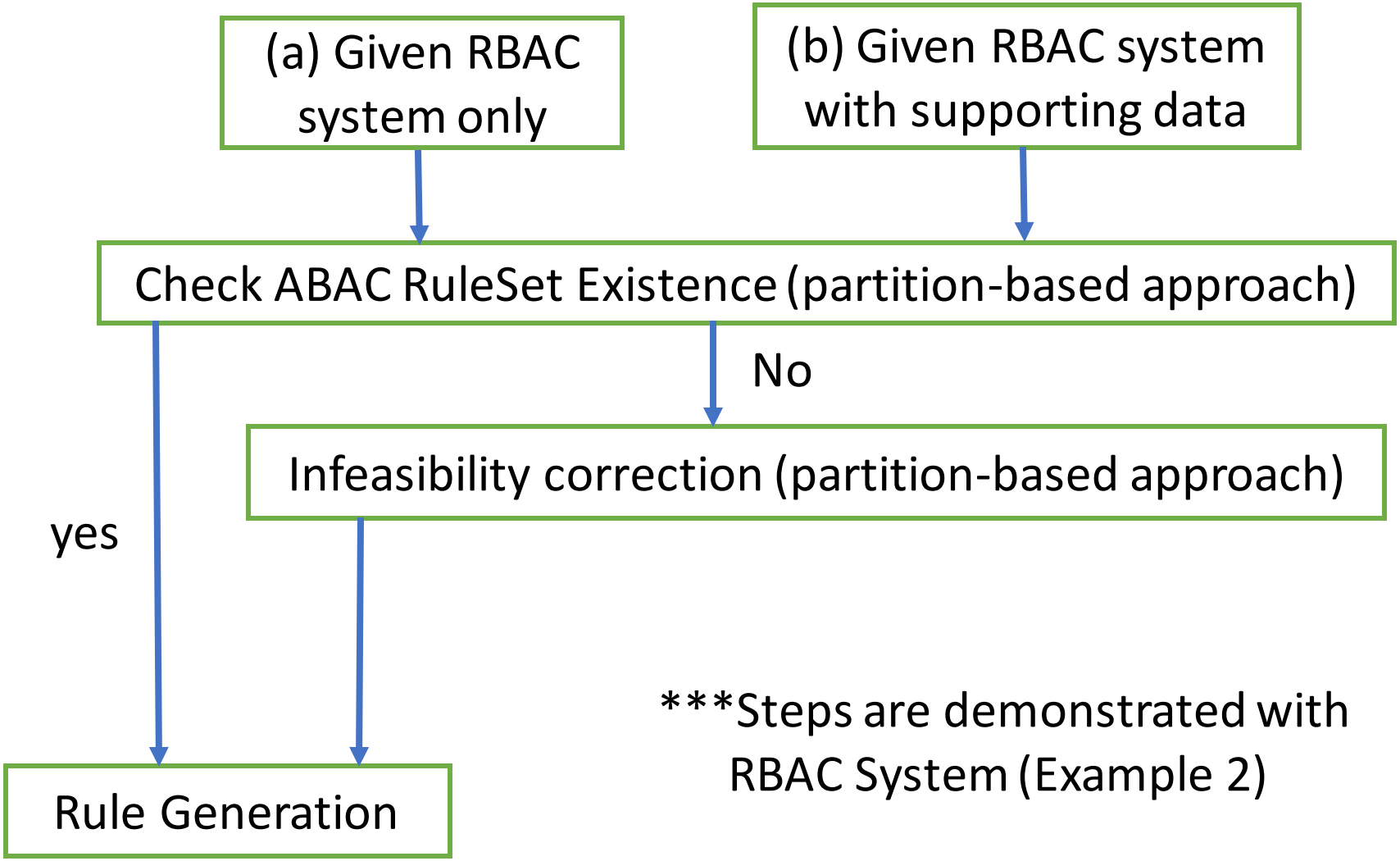
- RuleSet contains of one separate rule for each operation, {$Rule_{read}$, $Rule_{write}$}

ABAC RuleSet Existence



Does an equivalent ABAC system exist for the given RBAC system and supporting data?
Find the RuleSet -> *With ID, always possible, *No IDs → Not possible
e.g., cannot separate John from Ray and Tom in Example 3

Contribution: at a glance



(a) RBAC only

Step 1. Generate role-based attribute set

- For a user u, role-based user attribute denotes the set of roles possessed by u
- For a object-operation pair (obj, op), role-based object attribute denotes the set of roles where each role contains permission (obj, op)

UAValue	
User(U)	**uroleAtt**
John	{R1, R2, R3}
Lina	{R2}
Ray	{R3}
Tom	{R3}

OAValue		
Object(O)	**oroleAtt$_{write}$**	**oroleAtt$_{read}$**
Obj1	{R1}	{R1, R3}
Obj2	{R1, R2}	{}

Next step: partition set is generated on set UXO based on similarity in attribute value assignment

Concept: Conflict-free partition

1
Ray, Obj1
Tom, Obj1
John, Obj1

2
John, Obj2
Ray, Obj2
Tom, Obj2

Partition Set

3
Lina, Obj2

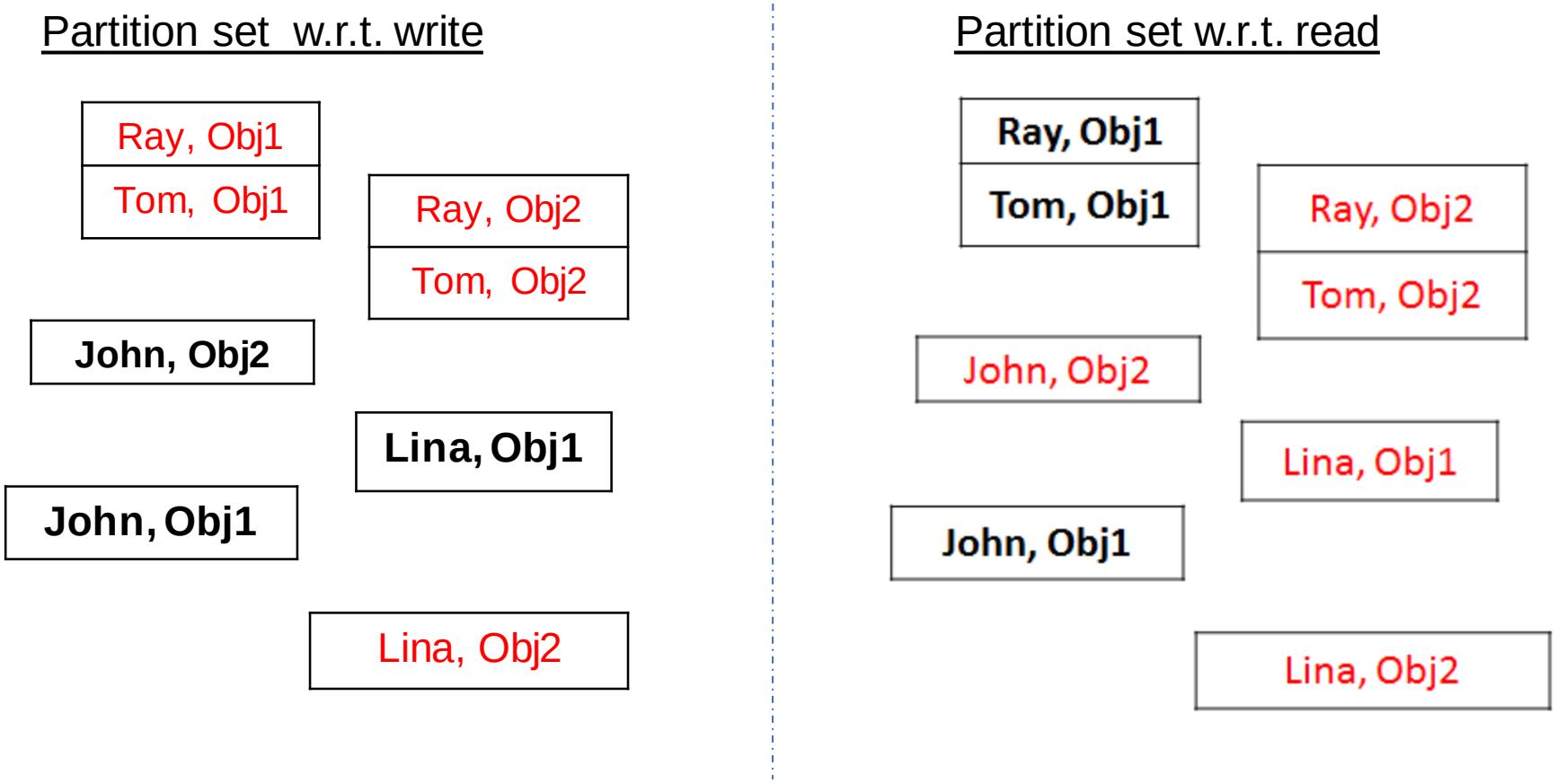
4
Lina, Obj1

Partition set w.r.t. op∈OP
Bold Black: Allowed
Red: Not allowed

1: Conflict
2, 3: conflict-free but not included in rule
4: conflict-free and included in rule

***A partition set is conflict-free w.r.t. an operation iff all partitions are conflict-free for that operation.

Step 2



Partition set is conflict-free w.r.t. read and write → YES

Step 3

- Given an operation op, if partition set is conflict-free and each partition is uniquely identified by the set of (attribute name, value) pair then RuleSet can be generated [Proved]
- A conjunction of (attribute name, value) pair is made for each conflict-free bold black partition and OR'ed to $Rule_{op}$

e.g., $Rule_{read} \equiv$ **<(uroleAtt(u) = {R3} Λ oroleAtt$_{write}$ (o)={R1} Λ oroleAtt$_{read}$ (o) = {R1, R3}) V (uroleAtt(u) = {R1, R2, R3} Λ oroleAtt$_{write}$(o)= {R1} Λ oroleAtt$_{read}$ (o)= {R1, R3})>**

*****$Rule_{write}$ can be constructed same way**
***RuleSet = {$Rule_{write}$, $Rule_{read}$}**
*****Equivalent ABAC system generation is always possible!**

(b) With supporting data

Role Based Access Control System

Equivalent ABAC system

Supporting Data

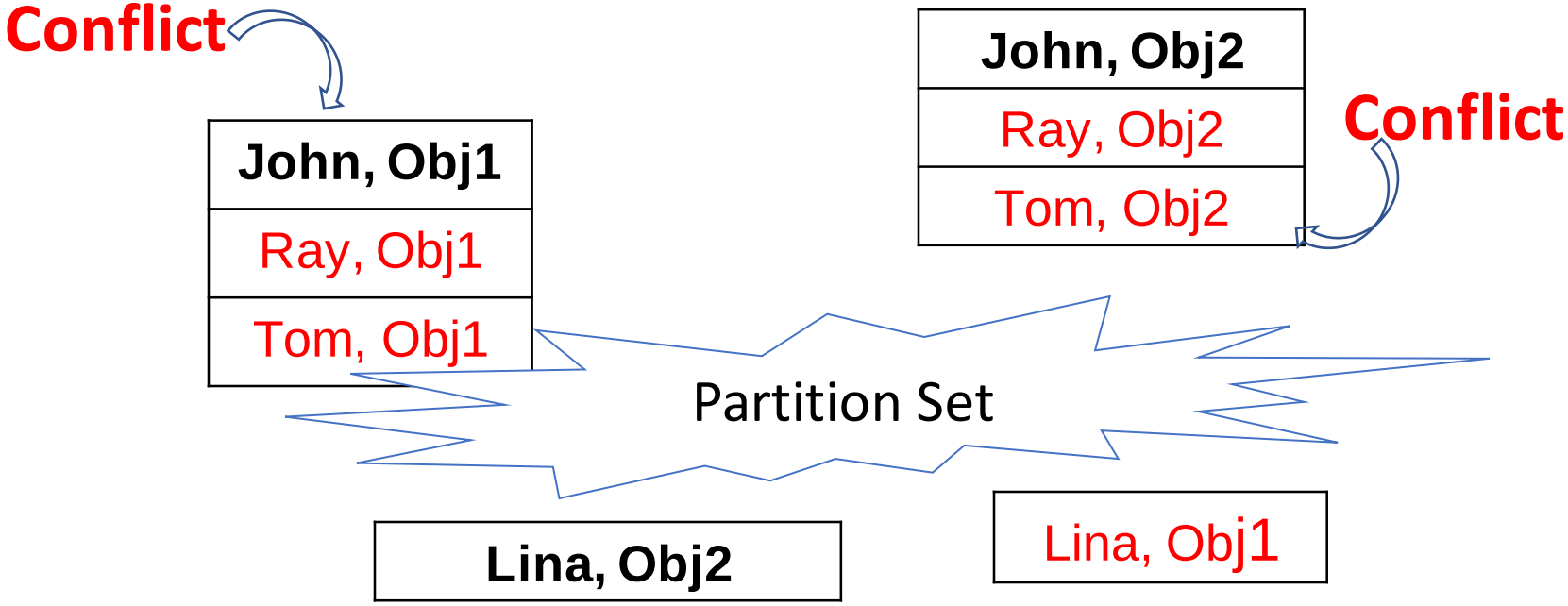
UAValue		
User (U)	**Position**	**Dept.**
John	Officer	CS
Lina	Student	CS
Ray	Officer	CS
Tom	Officer	CS

OAValue	
Object (O)	**Type**
Obj1	File
Obj2	Printer

RangeSet	
Position	{Officer, Student, Faculty}
Dept.	{CS, EE}
Type	{File, Printer, Scanner}

Step 1: Generate partition set based on similarity in attribute value assignment. Partition set might have conflicts!

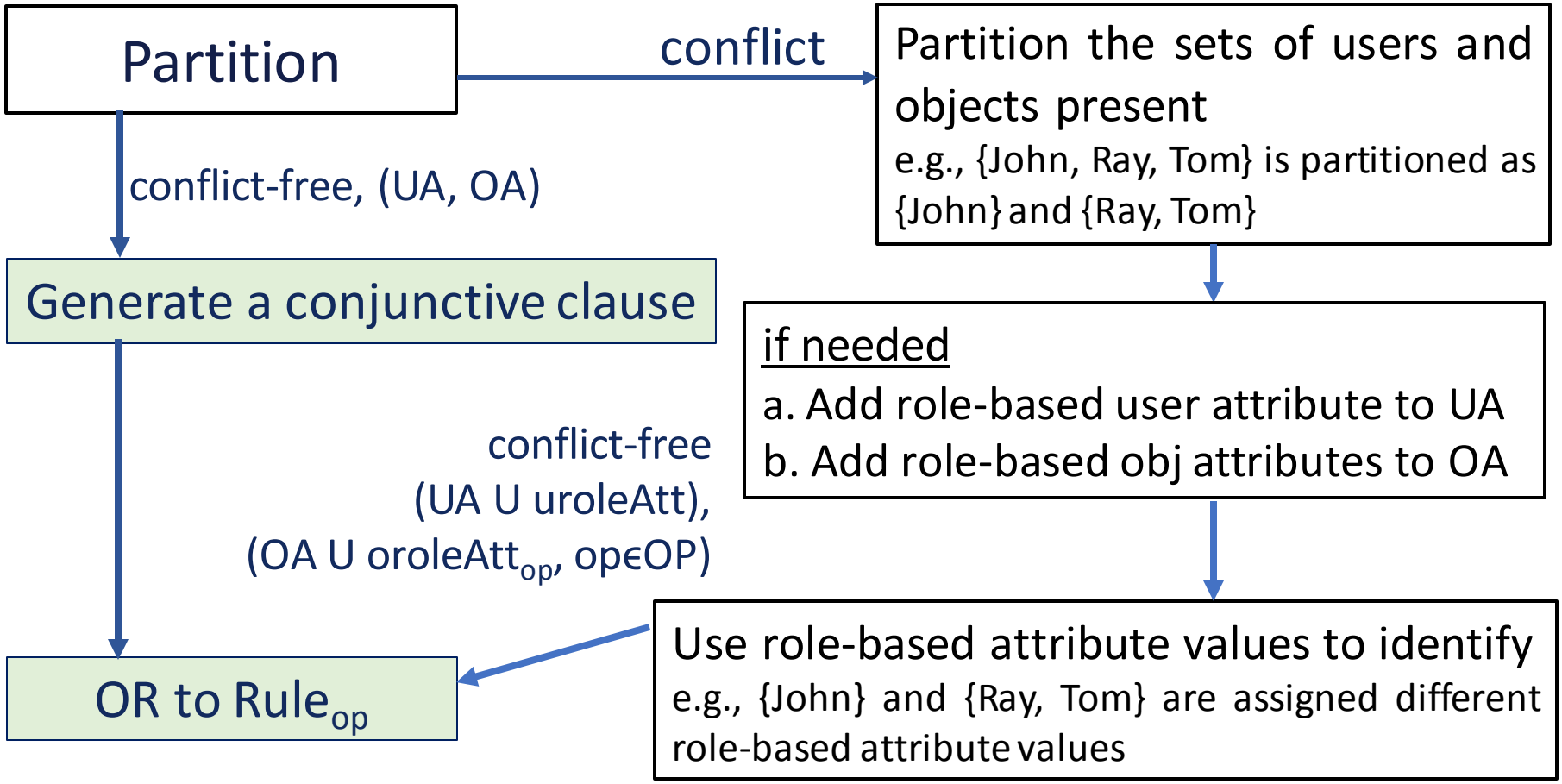
Step 1



*Partition set has conflict w.r.t. write → YES

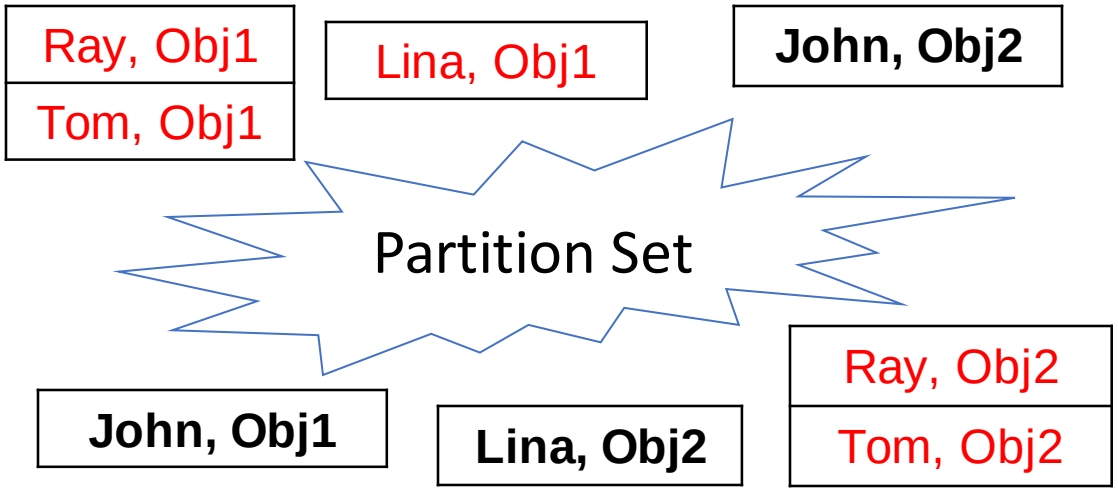
Next step: Apply infeasibility correction

Step 2 and 3



Infeasibility correction: exact solution can be achieved many ways

Partition set: corrected



UAValue	
User(U)	uroleAtt
John	{R1, R2, R3}
Lina	{R2}
Ray	{R3}
Tom	{R3}

$Rule_{write} \equiv <$(Position(u) = officer ∧ Dept(u) = CS ∧ uroleAtt(u)={R1, R2, R3} ∧ Type(o) = File) ∨ (Position(u) = officer ∧ Dept(u) = CS ∧ uroleAtt(u)={R1, R2, R3} ∧ Type(o) = Printer) ∨ (Position(u) = student ∧ Dept(u) = CS ∧ Type(o) = Printer)>

***RuleSet = {$Rule_{write}$, $Rule_{read}$}**

OAValue		
Object (O)	$oroleAtt_{write}$	$oroleAtt_{read}$
Obj1	{R1}	{R1, R3}
Obj2	{R1, R2}	{}

Conclusion

- Formalized notion of feasibility on RBAC to ABAC policy mining: first time
- The overall asymptotic complexity of ABAC RuleSet Existence problem is $O(|OP| \times (|U| \times |O|))$
- The overall asymptotic complexity of ABAC RuleSet Infeasibility Correction in RBAC context is $O(|OP| \times (|U| \times |O|)^3)$

Challenges

- Can you ensure partition split always equals 2?
- More compact set of rule generation
- Negative rules?
- Exact solution:
  - ❖Reduce number of split partitions
  - ❖Change number of attributes required
  - ❖Changing existing attribute set, possible?
- Approximate Solution
  - ❖Change RBAC system/authorization
  - ❖Change existing attribute value assignment


Questions