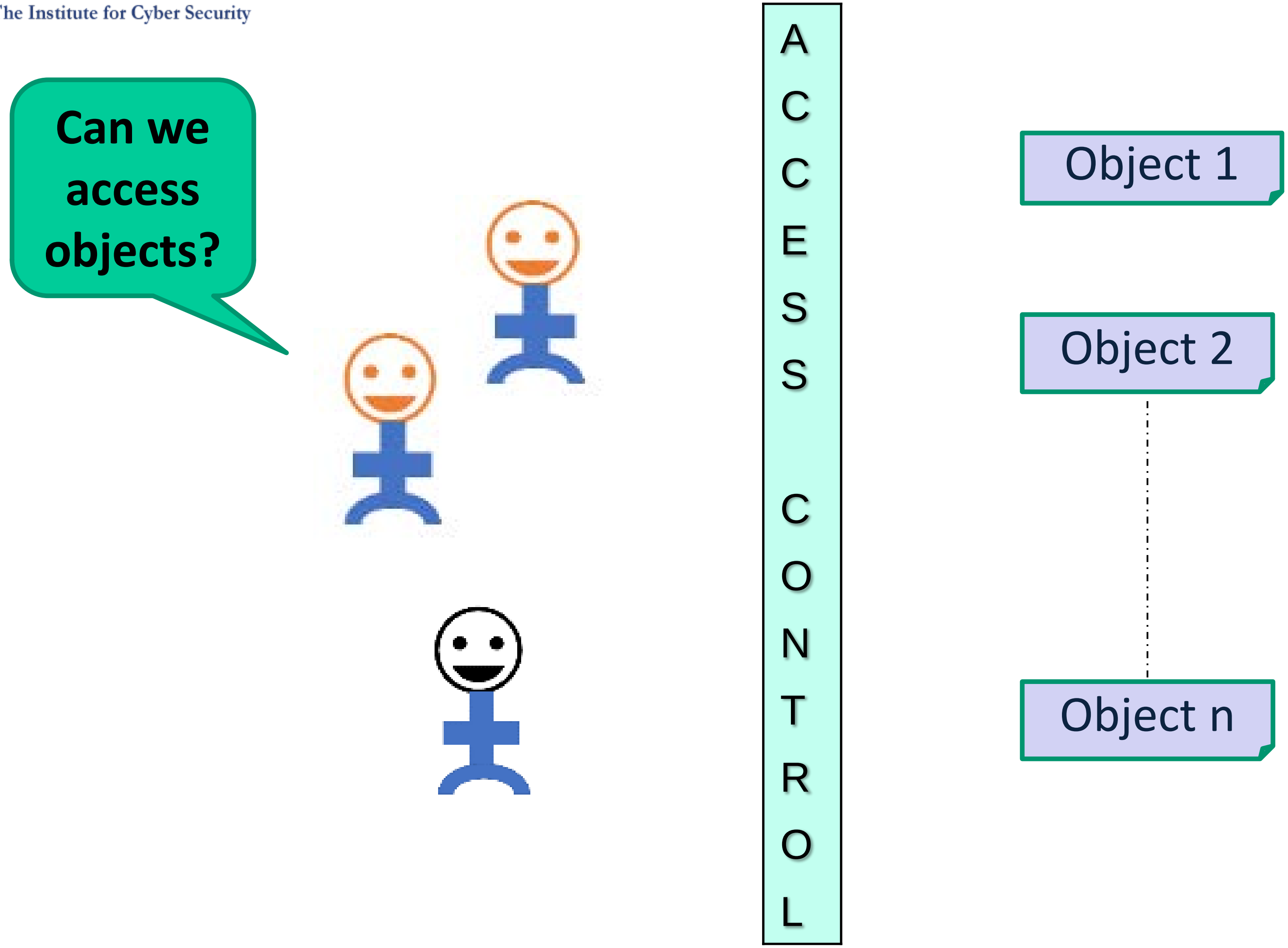
Access Control Policy Mining: Feasibility Analysis

L10-1
CS6393
Spring 2020

Introduction

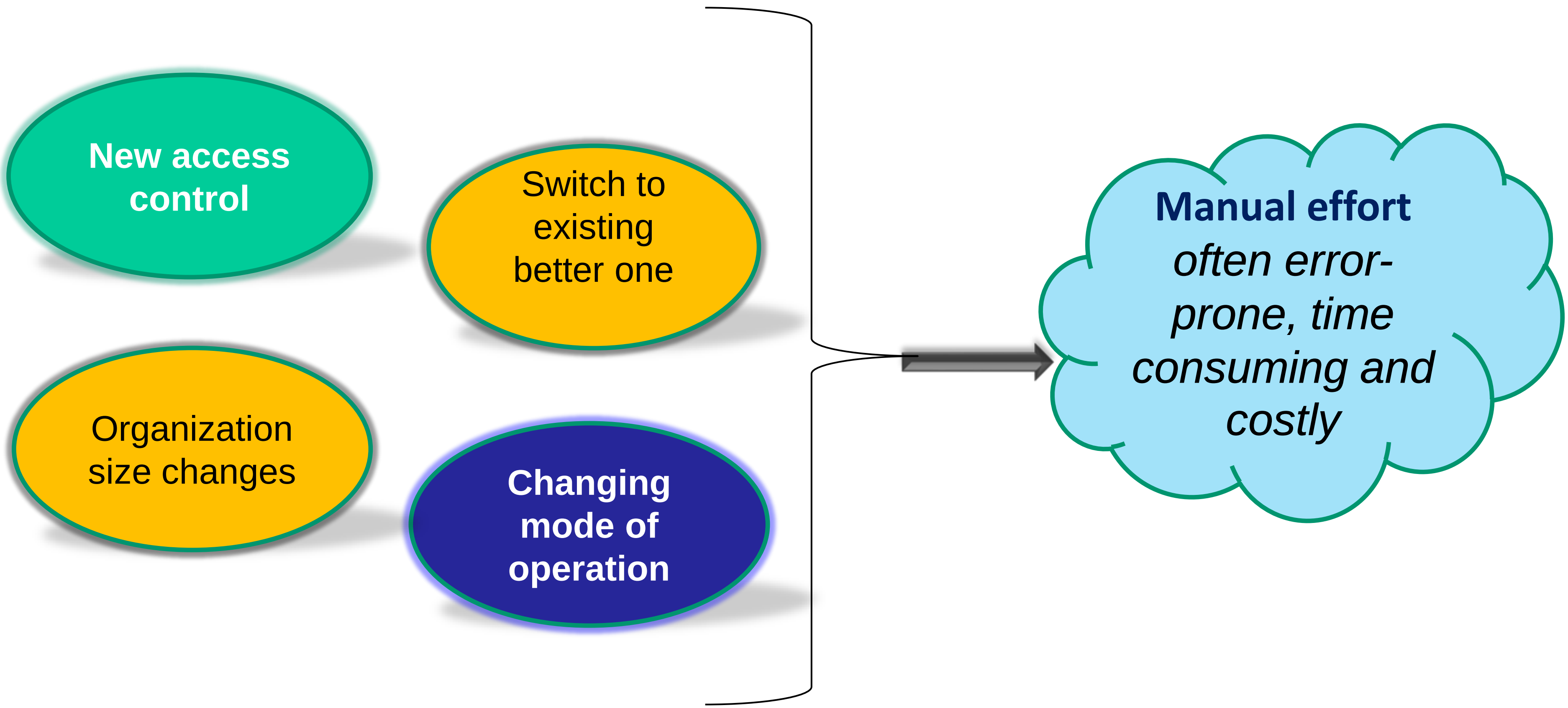


Legitimate users get legitimate access only

e.g., Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC)

Introduction

❖ **Problem: migration from an existing access control model to another one**



Is automation possible?

Introduction

Access Control List / Log / RBAC + Supporting attribute data → ABAC policy mining

Access Control List + Supporting Relationship data → ReBAC policy mining

Given an access control system + Supporting data → Another access control model

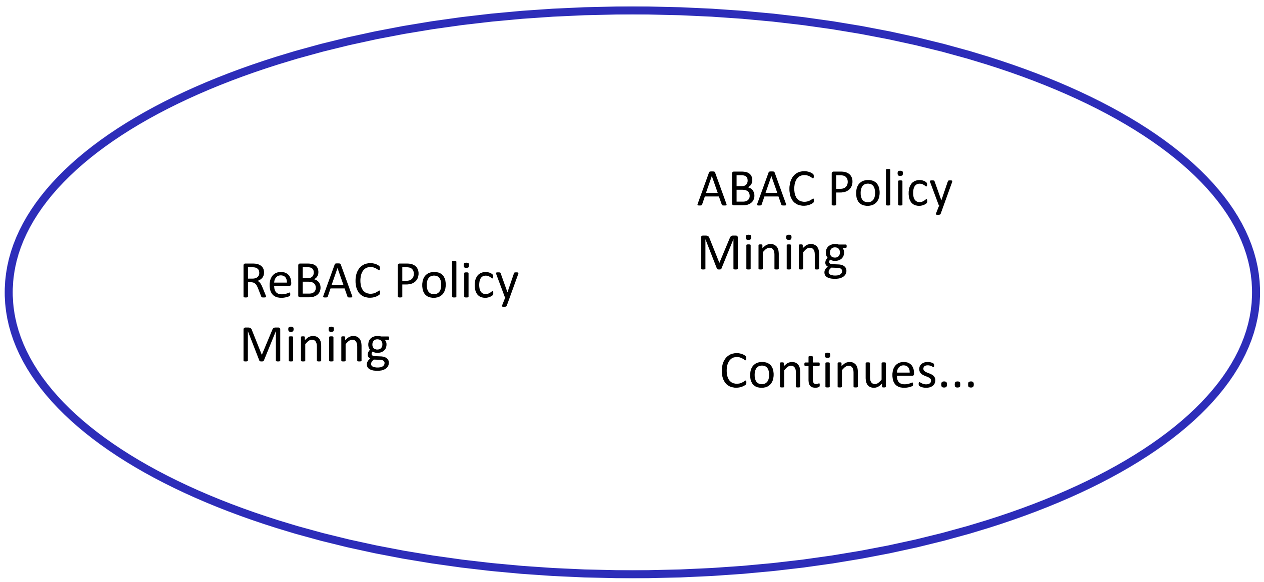
General term
Access control policy mining

Mining is partially automated so far…

*** Relationship-Based Access Control (ReBAC)

Problem Statement and Scope

❖ *The feasibility analysis of the access control mining problem studies whether the migration process is possible or not under the set of imposed criteria.*



Domain of feasibility analysis in Access Control Policy Mining

What is being done…

- ❖ Developing feasibility analysis algorithms for certain set of access control mining problem with complexity analysis.

- ❖ In case of infeasibility, solution algorithms are presented to make it feasible under given criteria.

Background

- To the best of our knowledge: feasibility analysis of access control policy mining is proposed for the first time
  - Hence, no directly related background work

- Some access control policy mining works
  - Role Mining
  - ABAC policy mining [from authorization, RBAC, log data, sparse log] etc.

- Our study includes 3 types of Access Control System
  - Enumerated Authorization System (EAS)
  - RBAC System
  - ABAC System

Background

EAS is a tuple <U, O, OP, AUTH, $checkAccess_{EAS}$>

- ❖ U, O, and OP are finite sets of users, objects and operations, respectively
- ❖ AUTH $\subseteq$ U x O x OP

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 read operation on Obj2

Background

RBAC system

- is a tuple <U, O, OP, Roles, RPA, RUA, RH, $checkAccess_{RBAC}$>

- Roles is finite set of roles
- RH is the role hierarchy relation [RH': reflexive transitive closure of RH]
- RPA : Role Permission Assignment
- RUA: Role User Assignment
- Permission is an object-operation pair
- $authPerm(r) = \{p \in RPA(r') | (r, r') \in RH'\}$, where $r,r' \in Roles$
- $authUser(r) = \{u \in RUA(r') | (r', r) \in RH'\}$ where $r,r' \in Roles$

- $checkAccess_{RBAC}(u{:}U, o{:}O, op{:}OP) \equiv \exists r \in Roles.(u \in authUser(r) \wedge (o, op) \in authPerm(r)$

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]

Equivalency

Two access control systems are equivalent iff

- ❖ U, O, and OP are equal for both systems
- ❖ ∀(u,o,op) ∈ UxOxOP. $\text{checkAccess}_{\text{system1}}$ (u,o,op) ≡ $\text{checkAccess}_{\text{system2}}$ (u,o,op)

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

Background

- **ABAC system** is a tuple <U, O, OP, UA, OA, UAValue, OAValue, RangeSet, RuleSet, $\text{checkAccess}_{ABAC}$ >

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 one separate rule for each operation, $\{\text{Rule}_{read}, \text{Rule}_{write}\}$
- ***ABAC system is incomplete in Example 3 (No rules given!)***

ABAC rule structure

For any operation op ∈ OP, $Rule_{op}$ grammar

- $Rule_{op}$ ::= $Rule_{op}$ ∨ $Rule_{op}$ | (Atomicexp)
- Atomicexp ::= Atomicuexp ∧ Atomicoexp | Atomicuexp | Atomicoexp
- Atomicuexp ::= Atomicuexp ∧ Atomicuexp | uexp
- Atomicoexp ::= Atomicoexp ∧ Atomicoexp | oexp
- uexp ∈ {ua(u) = value | ua ∈ UA ∧ value ∈ Range(ua)}
- oexp ∈ {oa(o) = value | oa ∈ OA ∧ value ∈ Range(oa)}

- **$checkAccess_{ABAC}$ (a:U, b:O, op:OP) ≡ $Rule_{op}$(a:U, b:O)**

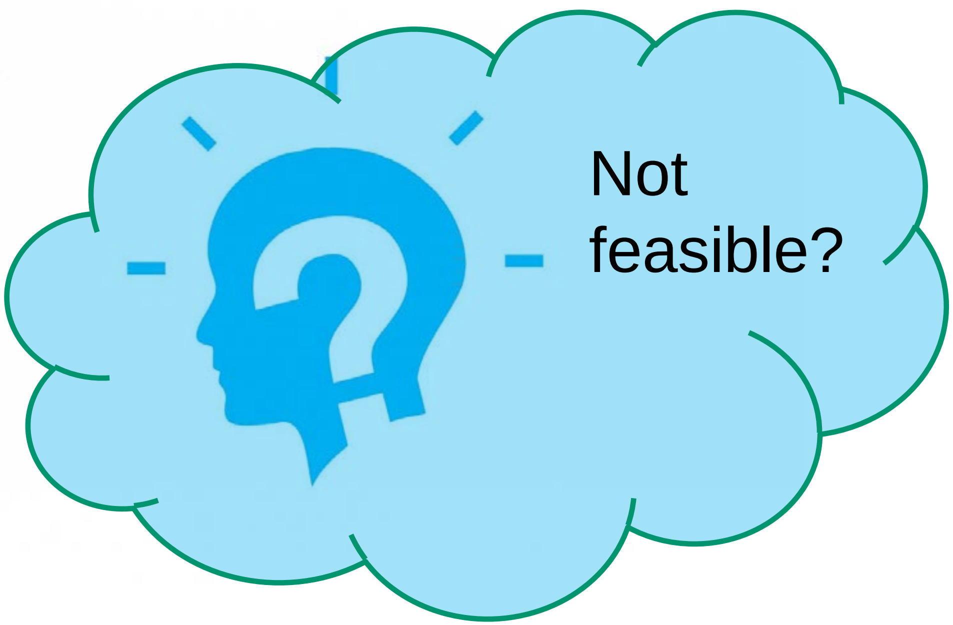
*** Illustrated ABAC rule examples can be found in later slides

Contribution

EAS + Incomplete ABAC system
(Paper 1)

ABAC policy mining is feasible or not

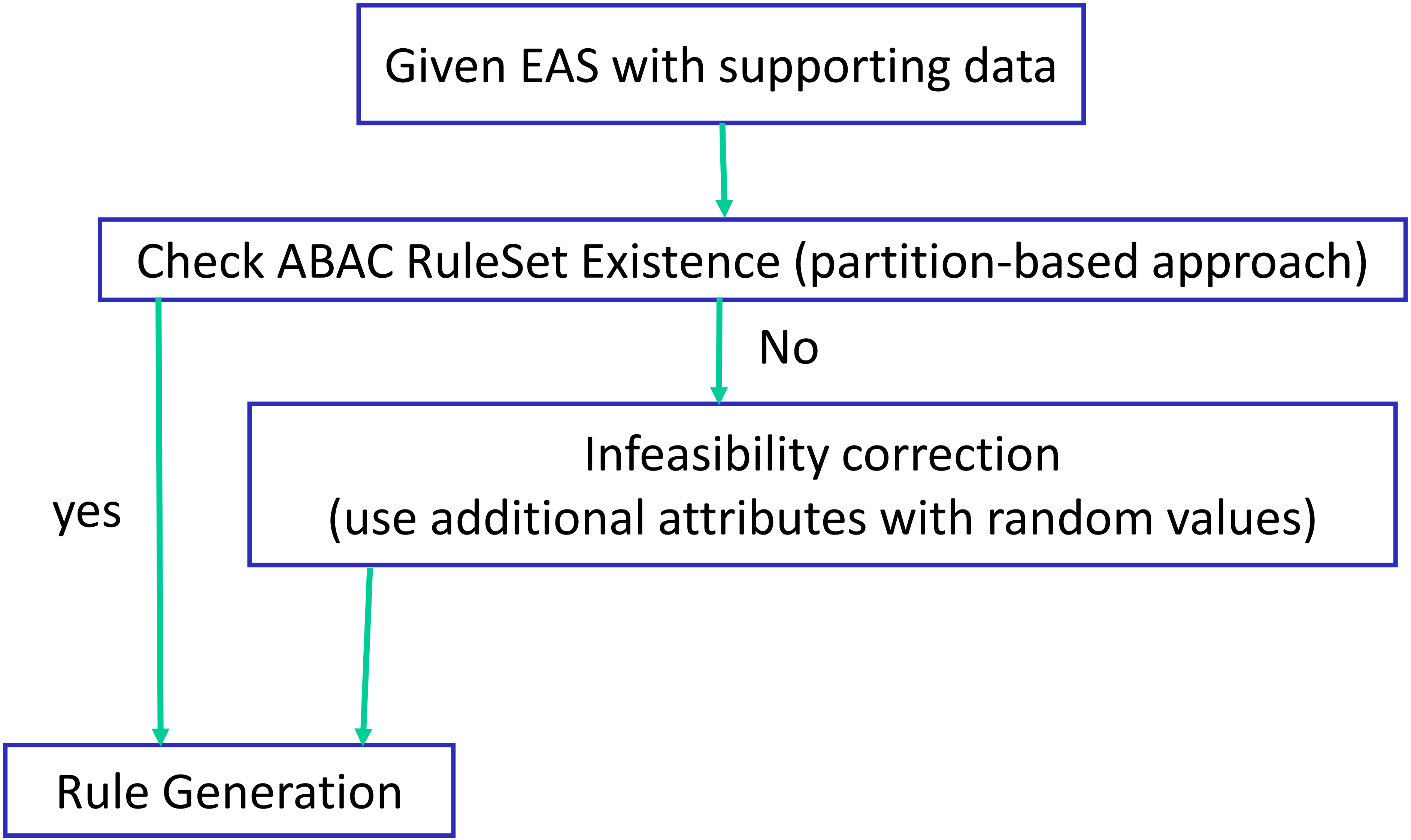
RBAC + Incomplete ABAC system
(Paper 2)



Use additional attribute to solve infeasibility

On the Feasibility of Attribute-Based Access Control Policy Mining

Paper 1 Workflow



Review: ABAC

Age

Salary

User

Name

Version No

Type

Object

Attribute-Based Access Control (ABAC) limits user to object access by using properties of both user and objects, namely “attribute”.

ABAC policy mining

Access Control System

Authorization (AUTH)
(John, obj1, write)
(Lina, obj2, write)

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}

Does an equivalent ABAC system exist for the given access control system and supporting data?

Example A: no ID

Access Control System

Authorization (AUTH)
(John, obj1, write)
(Lina, obj2, write)

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}

No IDs → Not possible
no way to separate John from Ray and Tom

Example A: with ID

Access Control System

Authorization (AUTH)
(John, obj1, write)
(Lina, obj2, write)

Equivalent ABAC system

Supporting Data

UAValue			
User (U)	uID	Position	Dept.
John	u1	Officer	CS
Lina	u2	Student	CS
Ray	u3	Officer	CS
Tom	u4	Officer	CS

OAValue		
Object (O)	oID	Type
Obj1	o1	File
Obj2	o2	Printer

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

Entity IDs → Always possible

Rule_write =

$(uID(U)=u1 \land oID(O)=o1) \lor (uID(U)=u2 \land oID(O)=o2)$

Example B

Access Control System

Authorization (AUTH)
(John, obj1, write)
(Ray, obj1, write)
(Tom, obj1, write)
(Lina, obj2, write)

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

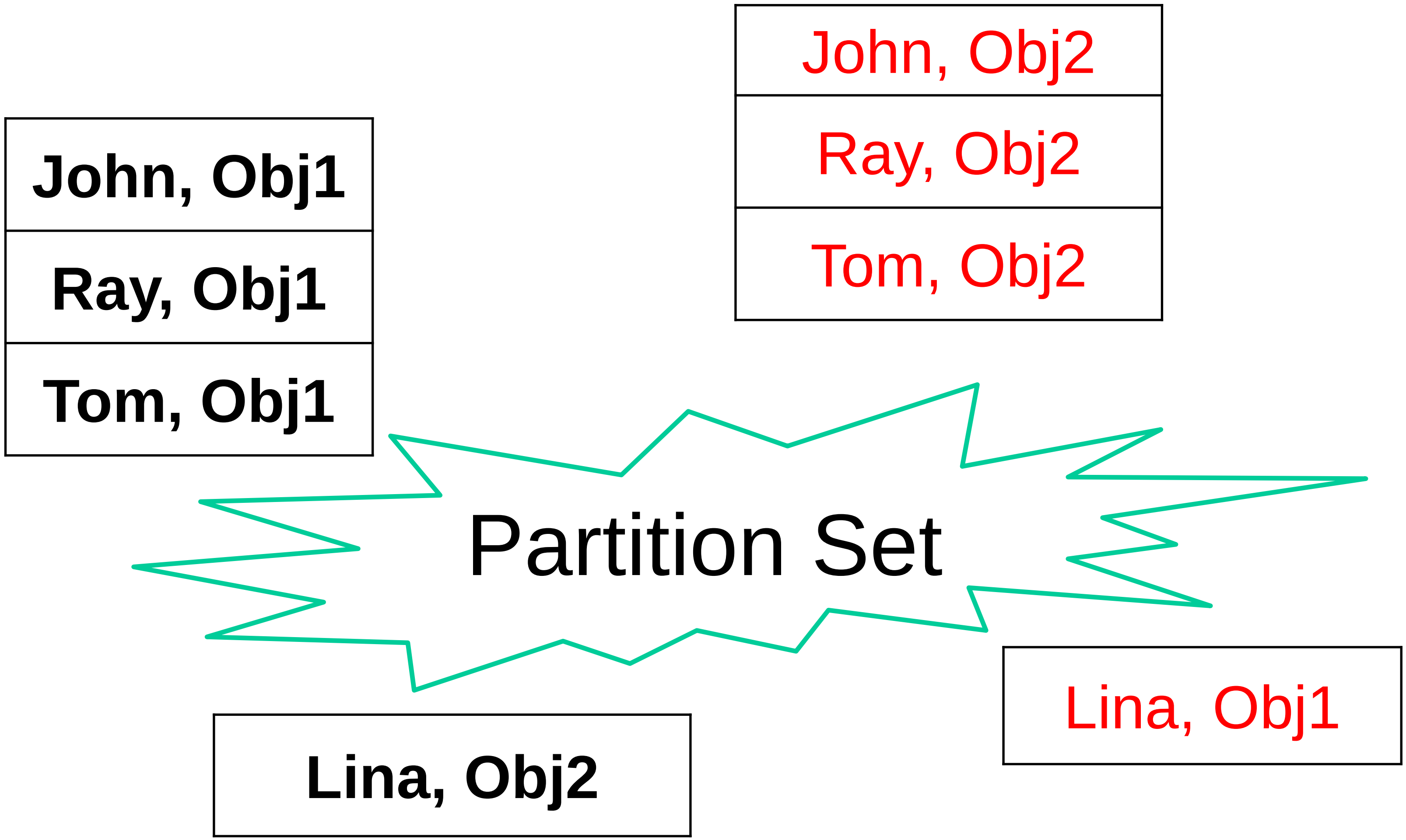
RangeSet

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

Determine the feasibility before rule generation!

Our solution: Partition-based strategy

Partition set: example B



Partition set is conflict-free w.r.t. write → Yes

Partition set: example A

Conflict

John, Obj1
Ray, Obj1
Tom, Obj1

John, Obj2
Ray, Obj2
Tom, Obj2

Partition Set

Lina, Obj1

Lina, Obj2

Authorization (AUTH)
(John, obj1, write)
(Lina, obj2, write)

Ray and Tom has no authorization compared to example B

Infeasibility correction

Partition

conflict

Partition the sets of users and objects present
e.g., {John, Ray, Tom} is partitioned as {John} and {Ray, Tom}

conflict-free, (UA, OA)

Generate a conjunctive clause

if needed
a. Add an attribute to UA (exU)
b. Add an attribute to OA (exO)

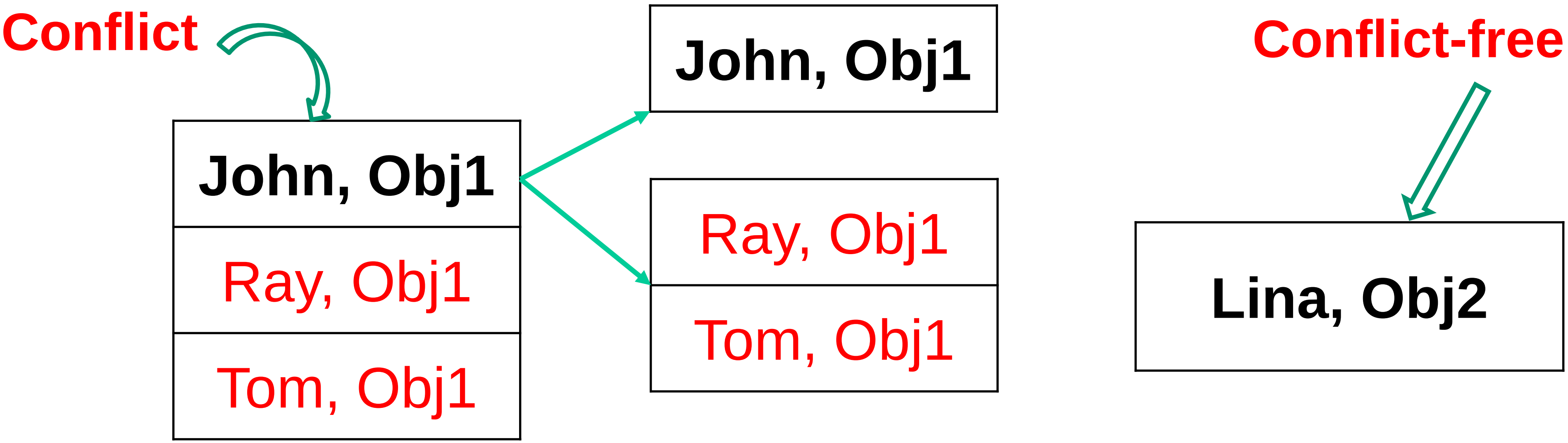
conflict-free
(UA U exU), (OA U exO)

Use unique random values to identify
e.g., exU(John)=a, exU(Ray)=b and exU(Tom)=b.
exO is not required here!

OR to Rule_{op}

Exact Solution can be achieved many ways

Example A: Rule Generation



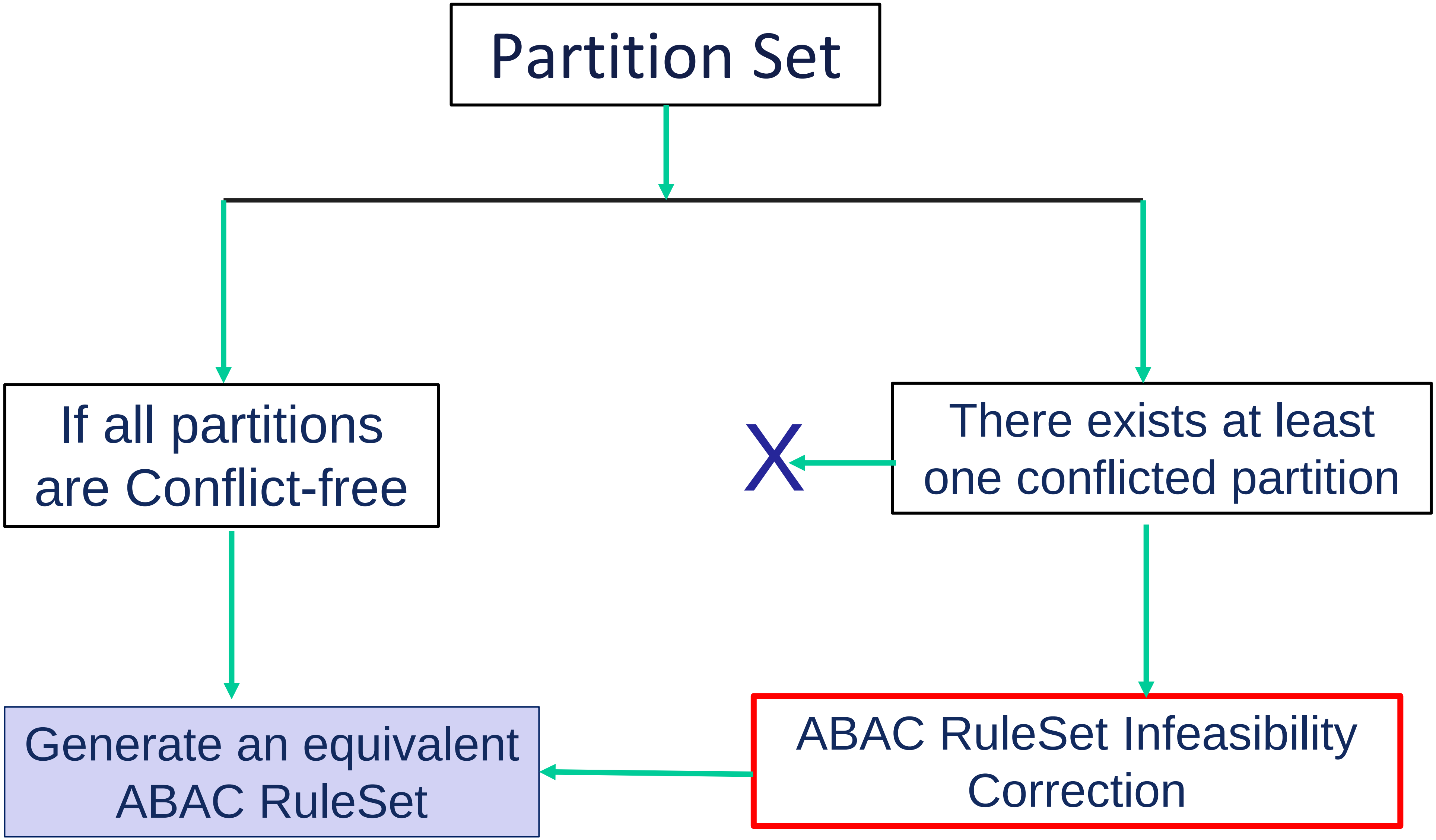
Rule_{write} ≡ (Position = officer AND Dept = CS AND exU = a AND Type = File) OR (Position = student AND Dept=CS AND Type = Printer)

ABAC system

<U, O, OP, UA, OA, RangeSet, UAValue, OAValue,{Rule_{write}}, $\text{checkAccess}_{ABAC}$>

Equivalent ABAC system generation is always possible!

At a Glance



Partition Set
If all partitions are Conflict-free
There exists at least one conflicted partition
X
Generate an equivalent ABAC RuleSet
ABAC RuleSet Infeasibility Correction

Unrepresented Partition

{Stud., Fac., Off.}

{CS, EE}

{F., Pr., Sc.}

<u>Represented: 4</u>
e.g., (Off., CS, F.),
(Stud., CS, Pr.)

<u>Unrepresented: 14</u>
e.g., (Fac., CS, Pr.),
(Stud., EE, Pr.)

Outcome of peculiarity in attribute value assignment

Summary: Paper 1

- **Formalized notion: feasibility of ABAC policy mining for the 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 is: $O(|OP| \times (|U| \times |O|)^3)$

- **<u>Challenges</u>**
- Can you replace random values?
- More compact set of rule generation
- Exact solution:
  - Reduce number of split partitions
  - Change number of attributes required
  - Changing existing attribute set, possible?
- Approximate Solution
  - Change authorization
  - Change existing attribute value assignment

References

1. Shuvra Chakraborty, Ravi Sandhu and Ram Krishnan, *On the Feasibility of Attribute-Based Access Control Policy Mining.* In Proceedings of the 20th IEEE Conference on Information Reuse and Integration (IRI), Los Angeles, California, July 30-August 1, 2019, 8 pages.