บทที่ 3

รายละเอียดวิธีการดำเนินงานวิจัย

บทนี้จะนำเสนอขอบเขตการวิจัย เครื่องมือและซอฟต์แวร์ที่ใช้ในการทดลอง โครงสร้างการทำงานของระบบ ขั้นตอนการทดลองและวิธีวัดผลการทดลอง ตามลำดับ ทั้งนี้เพื่อให้ได้แนวทางในการดำเนินการทำวิจัย และนำเสนอกระบวนการในการพัฒนาระบบโครงร่าง RBAC สำหรับการใช้ทรัพยากรจากหลายองค์กรร่วมกัน โดยใช้การกำหนดสิทธิแบบ XACMLและระบบหลายตัวแทน

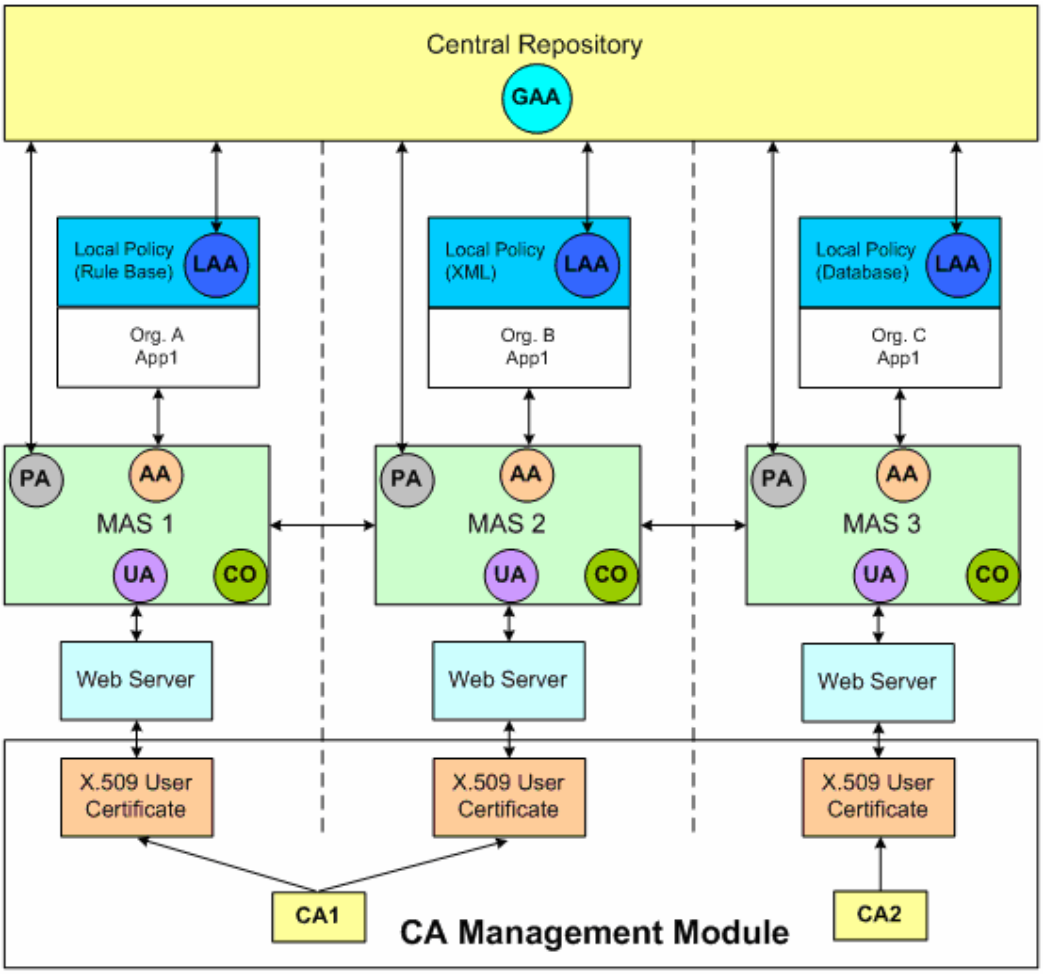
3.1 ขอบเขตการวิจัย

1. แอพพลิเคชั่นก่อนที่จะนำมาใช้งานกับระบบหลายตัวแทนจะมีการยืนยันตัวบุคคลด้วยชื่อผู้ใช้และรหัสผ่าน
2. แอพพลิเคชั่นก่อนที่จะนำมาใช้งานกับระบบหลายตัวแทนจะมีการกำหนดสิทธิเข้าใช้งานด้วย Rule Based Policy, XML Policy และ Database Policy
3. กำหนดให้มีระบบหลายตัวแทนจำนวน 3 ระบบที่มีการกำหนดสิทธิเข้าใช้งานที่แตกต่างกันด้วย Rule Based Policy, XML Policy และ Database Policy
4. การทำงานของระบบทั้งหมดจะทำงานอยู่ภายใต้สภาพแวดล้อมที่สมบูรณ์ โดยไม่มีความผิดพลาดของสภาพแวดล้อมเกิดขึ้น

จากภาพที่ 3.1 จะมีการใช้ระบบหลายตัวแทนจำนวน 3 ระบบ โดยแต่ละระบบจะมีการใช้รูปแบบในการยืนยันตัวบุคคลที่เหมือนกันคือการใช้ Token ในการยืนยันตัวบุคคลโดยใช้ 2 องค์ประกอบ และเป็นการเข้าใช้งานเพียงครั้งเดียว (SSO) แต่ในส่วนของการกำหนดสิทธิเข้าใช้งานจะมีการใช้รูปแบบการอนุญาตเข้าใช้งานที่แตกต่างกันซึ่งได้แก่ Rule Based Policy, XML Policy และ Database Policy โดย MAS1, MAS2 และ MAS3 จะมีการกำหนดสิทธิเข้าใช้งานแบบ Rule Based Policy, XML Policy และ Database Policy ตามลำดับ และในแต่ละ MAS จะกำหนดให้มีแอพพลิเคชั่นที่ทำงานภายใต้ MAS มีจำนวนอย่างละ 1 แอพพลิเคชั่น นอกจากนี้ยังมีการออกใบรับรองให้กับผู้ใช้ทุกคน โดยผู้ใช้ของ MAS1 และ MAS2 จะใช้ใบรับรองที่รับรองโดยผู้

ให้บริการออกใบรับรองรายเดียวกัน (CA1) ส่วนผู้ใช้ของ MAS3 จะใช้ใบรับรองที่รับรองโดยผู้ให้บริการออกใบรับรองรายอื่น(CA2)

ภาพที่ 3.1 แสดงภาพรวมของขอบเขตงานวิจัย



3.2 เครื่องมือและซอฟต์แวร์ที่ใช้ในการทดลอง

เครื่องมือและซอฟต์แวร์ที่นำมาใช้ในการพัฒนาระบบโครงร่าง RBAC สำหรับการใช้ทรัพยากรจากหลายองค์กรร่วมกัน โดยใช้การกำหนดสิทธิแบบ XACMLและระบบหลายตัวแทน มีรายละเอียดดังนี้

ตารางที่ 3.1 แสดงรายละเอียดของเครื่องมือและซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ

No.	Machine Types	Items	Details
1	Hardware	หน่วยประมวลผล	Intel Core Duo processor T7100 (1.8 GHz)
		หน่วยความจำ	4 GB
		ระบบปฏิบัติการ	Window Vista Ultimate
		ซอฟต์แวร์	- Tomcat (MAS1 Application) - Tomcat (MAS2 Applications) - Tomcat (MAS3 Applications) - Tomcat (Websettlement Applications) - Tomcat (RA System Applications) - Tomcat (ePayment Applications) - Tomcat (Central Repository Applications)
2	Virtual Machine (Run on No.1)	ระบบปฏิบัติการ	Window Server 2003 Standard Edition
		หน่วยความจำ	512 MB
		ซอฟต์แวร์	Certification Authority (CA)
3	Virtual Machine (Run on No.1)	ระบบปฏิบัติการ	Window Server 2003 Standard Edition
		หน่วยความจำ	512 MB
		ซอฟต์แวร์	LDAP

4		Token	eToken Pro32

ภาษาโปรแกรม

- Java

เครื่องมือพัฒนาระบบ

- Eclipse 3.2.0

3.3 โครงสร้างการทำงานของระบบ

3.3.1. ส่วนประกอบในการทำงานของระบบทั้งหมด

ส่วนประกอบในการทำงานของระบบ ARMORS ประกอบไปด้วยส่วนสำคัญอยู่ 6 ส่วน ได้แก่

1. CA Management Module ทำหน้าที่ออกใบรับรองอิเล็กทรอนิกส์ให้กับผู้ใช้
2. Web Server ทำหน้าที่ในการให้บริการเว็บและทำหน้าที่ในการยืนยันตัวบุคคลโดยใช้สององค์ประกอบของผู้ใช้
3. MAS ทำหน้าที่ในการยืนยันตัวบุคคลของผู้ใช้ผ่านทาง Agent ทำหน้าที่ในการตรวจสอบสิทธิของผู้ใช้ก่อนเข้าใช้งานแอพพลิเคชั่น ทำหน้าที่ในการบริหารจัดการลำดับการทำงานของคำร้องในการใช้งานแอพพลิเคชั่น โดยใน MAS จะประกอบไปด้วย Agent ทั้งหมด 4 ประเภทได้แก่
    a. User Agent (UA) ทำหน้าที่เสมือนเป็นผู้ใช้ในระบบ
    b. Policy Agent (PA) ทำหน้าที่เป็น Policy Decision Point (PDP) สำหรับ XACML และตรวจสอบสิทธิการใช้งานของแอพพลิเคชั่นตามสิทธิของบทบาทที่ได้กำหนดไว้แล้ว
    c. Collector Agent (CO) ทำหน้าที่เป็นตัวกลางระหว่าง Agent ต่างๆ กับ Log Server ในการเก็บบันทึกข้อมูลการทำงานของ Agent ทุกตัว
    d. Application Agent (AA) ทำหน้าที่เป็น Policy Enforcement Point (PEP) สำหรับ XACML และทำหน้าที่ในการบริหารจัดการลำดับการทำงานของคำร้องในการใช้งานแอพพลิเคชั่น

นอกจากนี้ MAS ยังต้องใช้เซิร์ฟเวอร์อีก 1 เซิร์ฟเวอร์ได้แก่ Log Server โดยทำหน้าที่ในการเก็บบันทึกข้อมูลการทำงานของ Agent ทุกตัว

4. Application ทำหน้าที่ในการให้บริการแอพพลิเคชั่น
5. Local Policy ทำหน้าที่ในการกำหนดสิทธิเข้าใช้งานแอพพลิเคชั่นแบบ Local โดยจะมี Agent ประเภท Local Admin Agent (LAA) ทำหน้าที่ในการทำ Local Policy Mapping ในแต่ละแอพพลิเคชั่นให้อยู่ในรูปของ XACML Policy แล้วส่งต่อไปยัง Global Admin Agent (GAA)

6. XACML Policy Integration ทำหน้าที่ในการรวบรวมการกำหนดสิทธิเข้าใช้งานของทุกแอพพลิเคชั่นในรูปแบบ XACML โดยจะมี Agent ประเภท Global Admin Agent (GAA) ทำหน้าที่ในการรวม XACML ที่ได้รับมาจาก LAA ในแต่ละแอพพลิเคชั่นให้มาเป็น XACML ส่วนกลาง

3.3.2. การจับคู่และการรวมนโยบาย (Policy Mapping and Integration)

ในส่วนของการทำ Policy Mapping and Integration นั้นจะเป็นการช่วยให้แต่ละระบบที่มีการกำหนดสิทธิที่แตกต่างกันจะสามารถทำงานร่วมกันได้ ด้วยการนำเสนอวิธีการ Mapping การกำหนดสิทธิรูปแบบต่างๆ ได้แก่ Rule-Based, XML-Based และ Database ของแต่ละ Local Policy ให้มาอยู่ในรูปแบบของ XACML โดย XACML ถือได้ว่าเป็นภาษากลางที่เป็นมาตรฐานในการกำหนดสิทธิได้ หลังจากทำการ Mapping แล้วจะมีการเก็บ Policy ไว้ใช้งานในครั้งต่อไปซึ่งจะเป็นข้อดีในการทำ Policy Administration and Coordination ซึ่งจะดีกว่าการทำ Policy Translation ในแง่ที่ไม่ได้เป็นการทำงานเพื่อใช้งานในแต่ละครั้งซึ่งในระยะยาวอาจจะไม่คุ้มค่าที่จะทำได้ และการกำหนด Authorization Model ในรูปแบบ RBAC ที่มีการ Integrate Policy เข้าด้วยกันซึ่งจะช่วยสนับสนุนการทำงานในกรณีที่มีจำนวน User มากๆ ในแต่ละ Organizational Domain ได้ดีกว่าแต่ละอันซึ่งเป็น RBAC อยู่แล้ว นอกจากนี้แล้วอัลกอลิทึมที่นำเสนอจะครอบคลุมถึงการ Mapping ที่สอดคล้องกับ XACML Syntax อย่างครบถ้วนและยังสามารถกำหนด Common Policy หรือ Conditions อื่นใด ที่สามารถ Enforce สำหรับ Collaborative Applications และจะมีการใช้ Rule Combining Algorithm เป็น ordered-deny-overrides ในการ Mapping นี้

โดยส่วนถัดไปจะอธิบายถึงแนวคิดในการทำการ Mapping ในแต่ละประเภทของ Policy โดยจะนำเสนออัลกอลิทึมและแผนในการจับคู่นโยบาย (Mapping Plan) ซึ่งมีรายละเอียดดังต่อไปนี้

3.3.2.1. การ Mapping Rule Based Policy

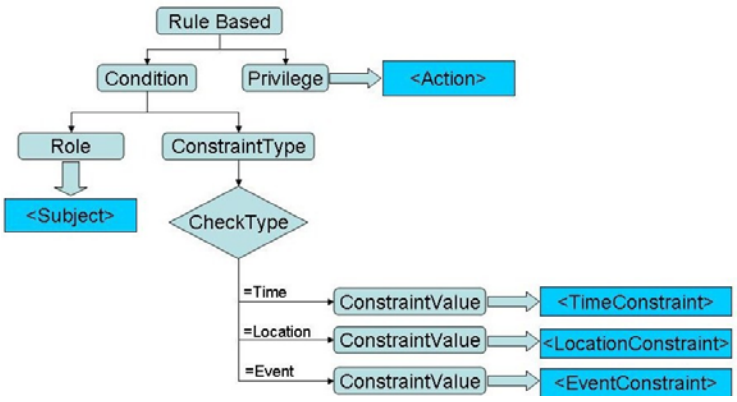
ลักษณะของ Rule Based Policy (R) จะเป็นเซ็ตของกฎต่างๆ ที่ใช้ในการกำหนดสิทธิการเข้าใช้งานให้แต่ละบทบาท(r) โดยจะมีส่วนประกอบ 2 ส่วนคือ Condition (c) และ Privilege (a) โดยส่วนของ Condition จะมีอีก 3 ส่วนคือ Role (role), ConstraintType (ct) และ ConstraintValue (cv) ซึ่ง ConstraintType ก็จะมีอยู่ 3 ประเภทด้วยกันได้แก่ Time Constraint, Location Constraint และ Event Constraint มีไว้สำหรับกำหนดเงื่อนไขในการเข้าใช้งานระบบ

นอกจากนี้จะกำหนดให้แต่ละกฎในนโยบายแบบ Rule Based จะต้องไม่มีการขัดแย้งกัน และมีการจัดเก็บนโยบายแบบ Rule Based ไว้เพียงที่เดียวกันทั้งหมด

โดยวิธีในการ Mapping จาก Rule Based Policy ไปเป็น XACML Policy นั้นจะมี 2 ขั้นตอนหลักด้วยกันได้แก่

1. การแยกส่วนประกอบของ Rule Based Policy (Decompose Rule Based Policy Element) คือทำการแยกส่วนประกอบย่อยของ Rule Based Policy ซึ่งได้แก่ Role, Privilege, ConstraintType และ ConstraintValue
2. การจับคู่กฎกับ XACML Tag (Map rule elements into XACML Tag) โดยสิ่งที่ต้องการจากขั้นตอนนี้คือ XACML Policy (P) ที่ซึ่งจะประกอบด้วยส่วนของ XACML Rule List (L) โดยมีวิธีการคือนำส่วนของ Role มาจับคู่กับแท็ก Subject ของ XACML, นำส่วนของ Privilege มาจับคู่กับแท็ก Action ของ XACML ส่วน Constraint จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ซึ่งได้แก่ Time Constraint, Location Constraint หรือ Event Constraint เพื่อให้ได้ XACML Rule List ที่จะนำมารวมกันเป็น XACML Policy

ภาพที่ 3.2 แสดงวิธีการ Mapping Local Policy ประเภท Rule Based



จากภาพที่ 3.2 จะแสดงให้เห็นว่าแต่ละ Rule Based จะมีส่วนประกอบ 2 ส่วนคือ Condition และ Privilege โดยส่วนของ Privilege จะถูกนำค่ามาจับคู่กับแท็ก Action ของ XACML แต่ส่วนของ Condition จะมีอีก 3 ส่วนคือ Role (role), ConstraintType (ct) และ ConstraintValue (cv) โดยส่วนของ Role จะถูกนำค่ามาจับคู่กับแท็ก Subject ของ XACML แต่

ส่วนของ Constraint ก็จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ให้สอดคล้องกับรูปแบบ XACML Policy ซึ่งได้มาจากการทำงานตามอัลกอลิทึม 1 ดังต่อไปนี้

Algorithm 1: Rule Based Policy Mapping

Input: R is a set of rules (r) constituting the Policy P

Step 1: Decompose Rule Based Policy Element

1: // R is a set of rules $\{r_1, \ldots, r_n\}$ where
r_i consists of a set of Conditions C $\{c_1, \ldots, c_n\}$ and a set of Actions A $\{a_1, \ldots, a_n\}$ //

2: R $\leftarrow$ a set of rules $r_1, \ldots, r_n$

3: $r_i \leftarrow (c_i, a_i)$

4: // c_i consists of set of Role $\{role_1, \ldots, role_n\}$, ConstraintType CT $\{ct_1, \ldots, ct_n\}$, ConstraintValue CV $\{cv_1, \ldots, cv_n\}$ //

5: $c_i \leftarrow (role_i, ct_i, cv_i)$

Step 2: Map rule elements into XACML Tag

6: L is a xacml rule list $\{l_1, \ldots, l_n\}$

7: For each r_i

8: // a is mapped into Action Tag //

9: a_i=> <Action>

10: // role is mapped into Subject Tag //

11: $role_i$ => <Subject>

12: // x is mapped according to the following conditions //

13: Switch (TypeOfConstraint(ct_i))

14: Case TimeConstraint :

15: cv_i => <TimeConstraint>

16: Case LocationConstraint :

17: cv_i => <LocationConstraint>

18: Case EventConstraint :

19: cv_i => <EventConstraint>

20: End Switch

21: $l_i \leftarrow$ XACMLRule (< Action >, <Subject>, <Condition>)

22: End For

23: // P_{Local} is a XACML Policy //

24: $P_{Local} \leftarrow$ XACMLRuleList (L)

25 Return P_{Local}

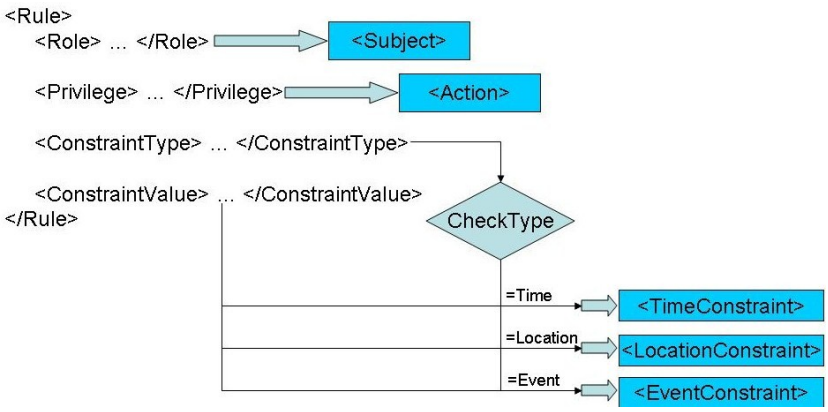
3.3.2.2. การ Mapping XML Policy

ลักษณะของ XML Policy (R) จะเป็นเซ็ตของกฎต่างๆ ที่ใช้ในการกำหนดสิทธิการเข้าใช้งานให้แต่ละบทบาท(r) โดยจะมีส่วนประกอบ 2 ส่วนคือ Condition (c) และ Element Privilege (a) โดยส่วนของ Condition จะมีอีก 3 Element คือ Role (role), ConstraintType (ct) และ ConstraintValue (cv) ซึ่ง ConstraintType ก็จะมีอยู่ 3 ประเภทด้วยกันได้แก่ Time Constraint, Location Constraint และ Event Constraint มีไว้สำหรับกำหนดเงื่อนไขในการเข้าใช้งานระบบ นอกจากนี้จะกำหนดให้แต่ละกฎในนโยบายแบบ XML จะต้องไม่มีการขัดแย้งกัน และมีการจัดเก็บนโยบายแบบ XML ไว้เพียงที่เดียวกันทั้งหมด

โดยวิธีในการ Mapping จาก XML Policy ไปเป็น XACML Policy นั้นจะมี 2 ขั้นตอนหลักด้วยกันได้แก่

1. การแยกส่วนประกอบของ XML Policy (Decompose XML Policy Element) คือทำการแยกส่วนประกอบย่อยของ XML Policy ซึ่งได้แก่ Role, Privilege, ConstraintType และ ConstraintValue
2. การจับคู่ XML Element กับ XACML Tag (Map XML elements into XACML Tag) โดยสิ่งที่ต้องการจากขั้นตอนนี้คือ XACML Policy (P) ที่ซึ่งจะประกอบด้วยส่วนของ XACML Rule List (L) โดยมีวิธีการคือ นำส่วนของ Role มาจับคู่กับแท็ก Subject ของ XACML, นำส่วนของ Privilege มาจับคู่กับแท็ก Action ของ XACML ส่วน Constraint จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ซึ่งได้แก่ Time Constraint, Location Constraint หรือ Event Constraint เพื่อให้ได้ XACML Rule List ที่จะนำมารวมกันเป็น XACML Policy

ภาพที่ 3.3 แสดงวิธีการ Mapping Local Policy ประเภท XML



จากภาพที่ 3.3 จะแสดงให้เห็นว่าแต่ละ Rule ของ XML จะมีส่วนประกอบ 4 Element คือ Role, Privilege, ConstraintType และ ConstraintValue โดยส่วนของ Role จะถูกนำค่ามาจับคู่กับแท็ก Subject ของ XACML และส่วนของ Privilege จะถูกนำค่ามาจับคู่กับแท็ก Action ของ XACML แต่ส่วนของ Constraint ก็จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ให้สอดคล้องกับรูปแบบ XACML Policy ซึ่งได้มาจากการทำงานตามอัลกอลิทึม 2 ดังต่อไปนี้

Algorithm 2: XML Policy Mapping

Input: R is a set of rules (r) constituting the Policy P

Step 1: Decompose XML Policy Element

1: //R is a set of rule $\{r_1, \ldots, r_n\}$ where

r_i consists of set of Element Role $\{role_1, \ldots, role_n\}$, Element Action A $\{a_1, \ldots, a_n\}$ and Element Condition C$\{c_1, \ldots, c_n\}$ //

2: R $\leftarrow$ a set of rules $r_1, \ldots, r_n$

3: $r_i \leftarrow (c_i, a_i)$

4: // c_i consists of set of Role $\{role_1, \ldots, role_n\}$, ConstraintType CT $\{ct_1, \ldots, ct_n\}$, ConstraintValue CV $\{cv_1, \ldots, cv_n\}$ //

5: $c_i \leftarrow (role_i, ct_i, cv_i)$

Step 2: Map XML elements into XACML Tag

6: L is a xacml rule list {l_1, …, l_n}

7: For each r_i

8: // a is mapped into Action Tag //

9: a_i=> <Action>

10: // role is mapped into Subject Tag //

11: $role_i$ => <Subject>

12: // x is mapped according to the following conditions //

13: Switch (TypeOfConstraint(ct_i))

14: Case TimeConstraint :

15: cv_i => <TimeConstraint>

16: Case LocationConstraint :

17: cv_i => <LocationConstraint>

18: Case EventConstraint :

19: cv_i => <EventConstraint>

20: End Switch

21: l_i ← XACMLRule (< Action >, <Subject>, <Condition>)

22: End For

23: // P_{Local} is a XACML Policy //

24: P_{Local} ← XACMLRuleList (L)

25 Return P_{Local}

3.3.2.3. การ Mapping Database Policy

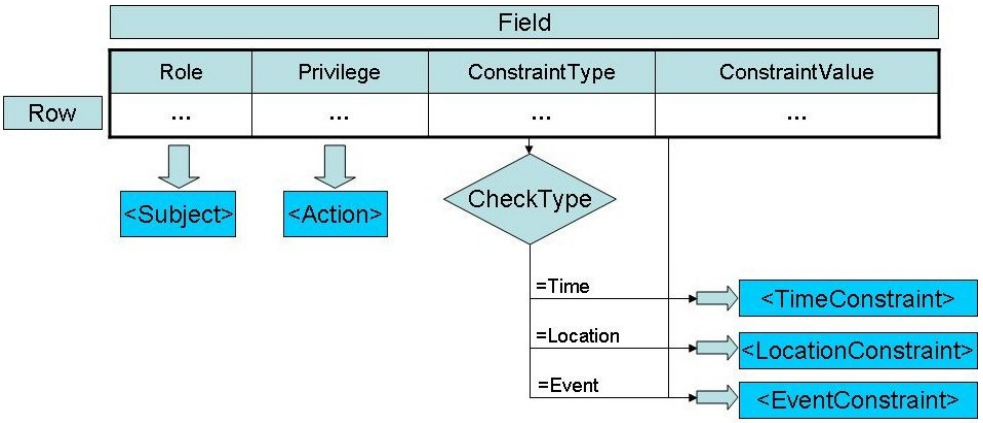
ลักษณะของ Database Policy (R) จะเป็นเซ็ตของกฎต่างๆ ที่ใช้ในการกำหนดสิทธิการเข้าใช้งานให้แต่ละบทบาท(r) โดยจะมีส่วนประกอบ 2 ส่วนคือ Condition (c) และ Field Privilege (a) โดยส่วนของ Condition จะมีอีก 3 Field คือ Role (role), ConstraintType (ct) และ ConstraintValue (cv) ซึ่ง ConstraintType ก็จะมีอยู่ 3 ประเภทด้วยกันได้แก่ Time Constraint, Location Constraint และ Event Constraint มีไว้สำหรับกำหนดเงื่อนไขในการเข้าใช้งานระบบ

นอกจากนี้จะกำหนดให้แต่ละกฎในนโยบายแบบ Database จะต้องไม่มีการขัดแย้งกัน และมีการจัดเก็บนโยบายแบบ Database ไว้เพียงที่เดียวกันทั้งหมด

โดยวิธีในการ Mapping จาก Database Policy ไปเป็น XACML Policy นั้นจะมี 2 ขั้นตอนหลักด้วยกันได้แก่

1. การแยกส่วนประกอบของ Database Policy (Decompose of Database Element) คือ ทำการแยกส่วนประกอบย่อยของ Database Policy ซึ่งได้แก่ Role, Privilege, ConstraintType และ ConstraintValue
2. การจับคู่ Field กับ XACML Tag (Map Field into XACML Tag) โดยสิ่งที่ต้องการจากขั้นตอนนี้คือ XACML Policy (P) ที่ซึ่งจะประกอบด้วยส่วนของ XACML Rule List (L) โดยมีวิธีการคือ นำส่วนของ Role มาจับคู่กับแท็ก Subject ของ XACML, นำส่วนของ Privilege มาจับคู่กับแท็ก Action ของ XACML ส่วน Constraint จะนำ ConstraintType มาแยกประเภทก่อนแล้วจึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ซึ่งได้แก่ Time Constraint, Location Constraint หรือ Event Constraint เพื่อให้ได้ XACML Rule List ที่จะนำมารวมกันเป็น XACML Policy

ภาพที่ 3.4 แสดงวิธีการ Mapping Local Policy ประเภท Database



จากภาพที่ 3.4 จะแสดงให้เห็นว่าแต่ละแถวของ Database จะมีส่วนประกอบ 4 Field คือ Role, Privilege, ConstraintType และ ConstraintValue โดยส่วนของ Role จะถูกนำค่ามาจับคู่กับแท็ก Subject ของ XACML และส่วนของ Privilege จะถูกนำค่ามาจับคู่กับแท็ก Action ของ XACML แต่ส่วนของ Constraint ก็จะนำ ConstraintType มาแยกประเภทก่อนแล้ว

จึงนำค่า ConstraintValue มาจับคู่กับแท็กที่จะกำหนด Constraint ประเภทต่างๆ ให้สอดคล้องกับรูปแบบ XACML Policy ซึ่งได้มาจากการทำงานตามอัลกอลิทึม 3 ดังต่อไปนี้

Algorithm 3: Database Policy Mapping

Input: R is a set of rules (r) constituting the Policy P

Step 1: Decompose of Database Element

1: //R is a set of rule {r_1, …, r_n} where
ri consists of Field Role {$role_1$, …, $role_n$}, Field Action A {a_1, …, a_n}
and Field Condition C{c_1, …, c_n} //

2: R ← a set of rules r_1, …, r_n

3: r_i ← (c_i, a_i)

4: // c_i consists of set of Role {$role_1$, …, $role_n$} , ConstraintType CT {ct_1, …, ct_n} , ConstraintValue CV {cv_1, …, cv_n} //

5: c_i ← ($role_i$, ct_i, cv_i)

Step 2: Map Field into XACML Tag

6: L is a xacml rule list {l_1, …, l_n}

7: For each r_i

8: // a is mapped into Action Tag //

9: a_i=> <Action>

10: // role is mapped into Subject Tag //

11: $role_i$ => <Subject>

12: // x is mapped according to the following conditions //

13: Switch (TypeOfConstraint(ct_i))

14: Case TimeConstraint :

15: cv_i => <TimeConstraint>

16: Case LocationConstraint :

17: cv_i => <LocationConstraint>

18: Case EventConstraint :

19: cv_i => <EventConstraint>

20: End Switch

21: $l_i \leftarrow$ XACMLRule (< Action >, <Subject>, <Condition>)

22: End For

23: // P_{Local} is a XACML Policy //

24: $P_{Local} \leftarrow$ XACMLRuleList (L)

25 Return P_{Local}

โดยหลังจากกระบวนการในการ Mapping Local Policy แบบต่างๆ ไปเป็น XACML Policy เสร็จเรียบร้อยแล้ว ในขั้นตอนสุดท้ายนี้คือกระบวนการในการรวม XACML Policy จากแต่ละที่ มาเป็น Global XACML Policy และจะมีการใช้ Policy Combining Algorithm เป็น ordered-deny-overrides ในการรวม XACML Policy เพื่อให้ระบบหลายตัวแทน (MAS) มาใช้งานในการตรวจสอบการกำหนดสิทธิเข้าใช้งาน ในแต่ละแอพพลิเคชั่น ซึ่งมีวิธีการดังแสดงในอัลกอลิทึมที่ 4 ดังต่อไปนี้

//ขั้นตอนสุดท้ายทำการรวม Local Policy ให้มาอยู่ในรูปของ Global XACML Policy

Algorithm 4: Combine Local Policy into Global XACML Policy

Input: P_{Local} is a set of rules (r) constituting the Policy P

$P_{Global'} \leftarrow$ Append (P_{Global}, P_{Local})

Return $P_{Global'}$

* เครื่องหมาย => หมายถึงการนำค่าทางซ้ายมา map ไปยัง tag ทางขวา

<Action> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดสิทธิ

<Subject> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดบทบาท

<Condition> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดเงื่อนไขโดยสามารถแบ่งเป็นประเภทดังต่อไปนี้

- <TimeConstraint> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดเงื่อนไขเกี่ยวกับเวลา
- <LocationConstraint> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดเงื่อนไขเกี่ยวกับสถานที่

- <EventConstraint> คือ Tag XACML ที่ทำหน้าที่ในการกำหนดเงื่อนไขเกี่ยวกับเหตุการณ์

ตัวอย่างเงื่อนไขเกี่ยวกับการกำหนดสิทธิ < Action >

สมมติให้ a_i เป็น Privilege ของ Local Policy ซึ่งมีค่าเท่ากับ read เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังนี้

```
<Actions>
    <Action>
      <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            read
        </AttributeValue>
        <ActionAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ActionMatch>
    </Action>
</Actions>
```

ตัวอย่างเงื่อนไขเกี่ยวกับการกำหนดบทบาท < Subject >

สมมติให้ $role_i$ เป็น Role ของ Local Policy ซึ่งมีค่าเท่ากับ admin เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังนี้

```
<Subjects>
    <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            admin
          </AttributeValue>
      <SubjectAttributeDesignator AttributeId="role"
```

```
                DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
</Subjects>
```

<u>ตัวอย่างเงื่อนไขเกี่ยวกับเวลา <TimeConstraint></u>

สมมติให้ Local Policy มีค่าของ ct_i เป็น time และค่าของ cv_i เป็น 0900 - 1700 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังนี้

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
        function:time-greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
            DataType="http://www.w3.org/2001/XMLSchema#time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
            09:00:00
        </AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
        function:time-less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
            DataType="http://www.w3.org/2001/XMLSchema#time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">
            17:00:00
        </AttributeValue>
```

```
    </Apply>
</Condition>
```

ตัวอย่างเงื่อนไขเกี่ยวกับสถานที่ <LocationConstraint>

สมมติให้ Local Policy มีค่าของ ct_i เป็น location และค่าของ cv_i เป็น 192.168.1.1 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังนี้

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
      <SubjectAttributeDesignator AttributeId="location"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
      192.168.1.1
    </AttributeValue>
</Condition>
```

ตัวอย่างเงื่อนไขเกี่ยวกับเหตุการณ์ <EventConstraint>

สมมติให้ Local Policy มีค่าของ ct_i เป็น Event และค่าของ cv_i เป็น Quantity < 100 เมื่อทำการ Mapping เป็น XACML Policy แล้วจะมีค่าเป็นดังนี้

```
<Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-than">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only">
      <SubjectAttributeDesignator AttributeId="Quantity"
          DataType="http://www.w3.org/2001/XMLSchema#integer"/>
    </Apply>
    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
      100
    </AttributeValue>
</Condition>
```

ในส่วนนี้จะเป็นการพิสูจน์ Mapping Algorithm ทุก Algorithm ที่นำเสนอว่ามีคุณสมบัติในด้านความครบถ้วนสมบูรณ์ (Existence and Completeness) และ ด้านความถูกต้อง (Correctness) โดยสามารถแสดงได้ดังต่อไปนี้

Definition 1

x => <XACML-Tag> หมายถึง การ Map แบบ one-to-one โดยการนำข้อมูล x มาเป็นข้อมูล ใน Tag <XACML-Tag> ของ XACML

พิสูจน์ Algorithm 1

1. ด้านความครบถ้วนสมบูรณ์ (Existence and Completeness)

Theorem 1: การแปลงองค์ประกอบของ กฎ (rule) ในนโยบายแบบ Rule-Based ที่ซึ่งจะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) แบบเดิมไปเป็นนโยบายแบบ XACML จะต้องมีจำนวนองค์ประกอบดังกล่าวที่เท่าเดิม จึงจะถือว่าการแปลง (mapping) นั้นมีความครบถ้วนสมบูรณ์

Proof 1: จาก Step 1 ใน Algorithm 1 จะมีการแยกองค์ประกอบของ rule เดิมออกเป็นส่วนต่างๆ และการ mapping จะเป็นลักษณะแบบ one-to-one ดังนั้น จำนวน action และ role จะถูก map ไปอยู่ใน XACML Tag ได้ทั้งหมด รวมถึง constraints ต่างๆ ก็จะถูก map เข้าไปในประเภทของ XACML Tag ที่สอดคล้องกันด้วย ดังนั้น ความครบถ้วนจึงเกิดขึ้นในทุกกรณีสำหรับทุกๆ rule

2. ด้านความถูกต้องในการแปลงนโยบาย (Correctness of Policy Mapping)

Theorem 2: อัลกอริทึมที่ใช้ในการ mapping จะถือว่ามีความถูกต้องในการแปลงนโยบายจาก Rule Based ไปเป็น XACML ก็ต่อเมื่อผลลัพธ์ที่ได้จากการแปลงมีความถูกต้องเช่นเดียวกันกับนโยบายแบบเดิมในทุกกรณี

Proof 2: ในการพิสูจน์นี้จะใช้วิธีการพิสูจน์แบบ Induction โดย

Assumption 1: ข้อมูล Action ใดๆ ของนโยบายแบบ Rule Based เป็น a_n จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based

จาก Assumption 1: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ Rule Based เป็น a_1 จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 1: สำหรับ n = n+1

จาก Assumption 1 จะได้ว่า ข้อมูล Action ของนโยบายแบบ Rule Based เป็น a_{n+1} จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Action ใดๆ ของนโยบายแบบ Rule Based จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based นั้นเป็นจริง .....(2.1)

Assumption 2: ข้อมูล Role ใดๆ ของนโยบายแบบ Rule Based เป็น $role_n$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_n$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based

จาก Assumption 2: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ Rule Based เป็น $role_1$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_1$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 2: สำหรับ n = n+1

จาก Assumption 2 จะได้ว่า ข้อมูล Role ของนโยบายแบบ Rule Based เป็น $role_{n+1}$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_{n+1}$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Role ใดๆ ของนโยบายแบบ Rule Based จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based นั้นเป็นจริง .....(2.2)

Assumption 3: ข้อมูล Constraint ใดๆ ของนโยบายแบบ Rule Based เป็น ct_n และ cv_n จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_n และ cv_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based

จาก Assumption 3: สำหรับ n = 1

ข้อมูล Constraint ของนโยบายแบบ Rule Based เป็น ct_1 และ cv_1 จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_1 และ cv_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1 และการแยกประเภทของ ct_1 ก่อนที่จะนำค่าของ cv_1 ไปทำการ mapping ดังแสดงใน Algorithm 1 บรรทัดที่ 13

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 3: สำหรับ n = n+1

จาก Assumption 3 จะได้ว่า ข้อมูล Constraint ของนโยบายแบบ Rule Based เป็น ct_{n+1} และ cv_{n+1} จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_{n+1} และ cv_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based เนื่องจาก Definition 1 และการแยกประเภทของ ct_{n+1} ก่อนที่จะนำค่าของ cv_{n+1} ไปทำการ mapping ดังแสดงใน Algorithm 1 บรรทัดที่ 13

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Constraint ใดๆ ของนโยบายแบบ Rule Based จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Rule Based นั้นเป็นจริง .....(2.3)

เนื่องจากนโยบายแบบ Rule Based จะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) ดังนั้นจากข้อความใน (2.1), (2.2) และ (2.3) จึงสามารถสรุปได้ว่า Theorem 2 จะเป็นจริงด้วยเช่นกัน

พิสูจน์ Algorithm 2

1. ด้านความครบถ้วนสมบูรณ์ (Existence and Completeness)

Theorem 3: การแปลงองค์ประกอบของ กฎ (rule) ในนโยบายแบบ XML ที่ซึ่งจะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) แบบเดิมไปเป็นนโยบายแบบ XACML จะต้องมีจำนวนองค์ประกอบดังกล่าวที่เท่าเดิม จึงจะถือว่าการแปลง (mapping) นั้นมีความครบถ้วนสมบูรณ์

Proof 3: จาก Step 1 ใน Algorithm 2 จะมีการแยกองค์ประกอบของ rule เดิมออกเป็นส่วนต่างๆ และการ mapping จะเป็นลักษณะแบบ one-to-one ดังนั้น จำนวน action และ role จะถูก map ไปอยู่ใน XACML Tag ได้ทั้งหมด รวมถึง constraints ต่างๆ ก็จะถูก map เข้าไปในประเภทของ XACML Tag ที่สอดคล้องกันด้วย ดังนั้น ความครบถ้วนจึงเกิดขึ้นในทุกกรณีสำหรับทุกๆ rule

2. ด้านความถูกต้องในการแปลงนโยบาย (Correctness of Policy Mapping)

Theorem 4: อัลกอริทึมที่ใช้ในการ mapping จะถือว่ามีความถูกต้องในการแปลงนโยบายจาก XML ไปเป็น XACML ก็ต่อเมื่อผลลัพธ์ที่ได้จากการแปลงมีความถูกต้องเช่นเดียวกันกับนโยบายแบบเดิมในทุกกรณี

Proof 4: ในการพิสูจน์นี้จะใช้วิธีการพิสูจน์แบบ Induction โดย

Assumption 4: ข้อมูล Action ใดๆ ของนโยบายแบบ XML เป็น a_n จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML

จาก Assumption 4: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ XML เป็น a_1 จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 4: สำหรับ n = n+1

จาก Assumption 4 จะได้ว่า ข้อมูล Action ของนโยบายแบบ XML เป็น a_{n+1} จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Action ใดๆ ของนโยบายแบบ XML จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML นั้นเป็นจริง .....(4.1)

Assumption 5: ข้อมูล Role ใดๆ ของนโยบายแบบ XML เป็น $role_n$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_n$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML

จาก Assumption 5: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ XML เป็น $role_1$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_1$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 5: สำหรับ n = n+1

จาก Assumption 5 จะได้ว่า ข้อมูล Role ของนโยบายแบบ XML เป็น $role_{n+1}$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_{n+1}$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Role ใดๆ ของนโยบายแบบ XML จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML นั้นเป็นจริง .....(4.2)

Assumption 6: ข้อมูล Constraint ใดๆ ของนโยบายแบบ XML เป็น ct_n และ cv_n จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_n และ cv_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML

จาก Assumption 6: สำหรับ n = 1

ข้อมูล Constraint ของนโยบายแบบ XML เป็น ct_1 และ cv_1 จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_1 และ cv_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1 และการแยกประเภทของ ct_1 ก่อนที่จะนำค่าของ cv_1 ไปทำการ mapping ดังแสดงใน Algorithm 2 บรรทัดที่ 13

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 6: สำหรับ n = n+1

จาก Assumption 6 จะได้ว่า ข้อมูล Constraint ของนโยบายแบบ XML เป็น ct_{n+1} และ cv_{n+1} จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_{n+1} และ cv_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML เนื่องจาก Definition 1 และการแยกประเภทของ ct_{n+1} ก่อนที่จะนำค่าของ cv_{n+1} ไปทำการ mapping ดังแสดงใน Algorithm 2 บรรทัดที่ 13

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Constraint ใดๆ ของนโยบายแบบ XML จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ XML นั้น เป็นจริง ……..……(4.3)

เนื่องจากนโยบายแบบ XML จะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) ดังนั้นจากข้อความใน (4.1), (4.2) และ (4.3) จึงสามารถสรุปได้ว่า Theorem 4 จะเป็นจริงด้วยเช่นกัน

<u>พิสูจน์ Algorithm 3</u>

1. ด้านความครบถ้วนสมบูรณ์ (Existence and Completeness)

<u>Theorem 5:</u> การแปลงองค์ประกอบของ กฎ (rule) ในนโยบายแบบ Database ที่ซึ่งจะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) แบบเดิมไปเป็นนโยบายแบบ XACML จะต้องมีจำนวนองค์ประกอบดังกล่าวที่เท่าเดิม จึงจะถือว่าการแปลง (mapping) นั้นมีความครบถ้วนสมบูรณ์

<u>Proof 5:</u> จาก Step 1 ใน Algorithm 3 จะมีการแยกองค์ประกอบของ rule เดิมออกเป็นส่วนต่างๆ และการ mapping จะเป็นลักษณะแบบ one-to-one ดังนั้น จำนวน action

และ role จะถูก map ไปอยู่ใน XACML Tag ได้ทั้งหมด รวมถึง constraints ต่างๆ ก็จะถูก map เข้าไปในประเภทของ XACML Tag ที่สอดคล้องกันด้วย ดังนั้น ความครบถ้วนจึงเกิดขึ้นในทุกกรณีสำหรับทุกๆ rule

2. ด้านความถูกต้องในการแปลงนโยบาย (Correctness of Policy Mapping)

Theorem 6: อัลกอริทึมที่ใช้ในการ mapping จะถือว่ามีความถูกต้องในการแปลงนโยบายจาก Database ไปเป็น XACML ก็ต่อเมื่อผลลัพธ์ที่ได้จากการแปลงมีความถูกต้องเช่นเดียวกันกับนโยบายแบบเดิมในทุกกรณี

Proof 6: ในการพิสูจน์นี้จะใช้วิธีการพิสูจน์แบบ Induction โดย

Assumption 7: ข้อมูล Action ใดๆ ของนโยบายแบบ Database เป็น a_n จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database

จาก Assumption 7: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ Database เป็น a_1 จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 7: สำหรับ n = n+1

จาก Assumption 7 จะได้ว่า ข้อมูล Action ของนโยบายแบบ Database เป็น a_{n+1} จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายของ a_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Action ใดๆ ของนโยบายแบบ Database จะสามารถ Map ไปยัง XACML Tag <Action> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database นั้นเป็นจริง .....(6.1)

Assumption 8: ข้อมูล Role ใดๆ ของนโยบายแบบ Database เป็น $role_n$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_n$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database

จาก Assumption 8: สำหรับ n = 1

ข้อมูล Action ของนโยบายแบบ Database เป็น $role_1$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_1$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 8: สำหรับ n = n+1

จาก Assumption 8 จะได้ว่า ข้อมูล Role ของนโยบายแบบ Database เป็น $role_{n+1}$ จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายของ $role_{n+1}$ เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Role ใดๆ ของนโยบายแบบ Database จะสามารถ Map ไปยัง XACML Tag <Subject> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database นั้นเป็นจริง .....(6.2)

Assumption 9: ข้อมูล Constraint ใดๆ ของนโยบายแบบ Database เป็น ct_n และ cv_n จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_n และ cv_n เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database

จาก Assumption 9: สำหรับ n = 1

ข้อมูล Constraint ของนโยบายแบบ Database เป็น ct_1 และ cv_1 จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_1 และ cv_1 เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1 และการแยกประเภทของ ct_1 ก่อนที่จะนำค่าของ cv_1 ไปทำการ mapping ดังแสดงใน Algorithm 3 บรรทัดที่ 13

ฉนั้นสำหรับ n = 1 จะเป็นจริง

จาก Assumption 9: สำหรับ n = n+1

จาก Assumption 9 จะได้ว่า ข้อมูล Constraint ของนโยบายแบบ Database เป็น ct_{n+1} และ cv_{n+1} จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายของ ct_{n+1} และ cv_{n+1} เช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database เนื่องจาก Definition 1 และการแยกประเภทของ ct_{n+1} ก่อนที่จะนำค่าของ cv_{n+1} ไปทำการ mapping ดังแสดงใน Algorithm 3 บรรทัดที่ 13

ฉนั้นสำหรับ n = n+1 จะเป็นจริง

ดังนั้นจึงสามารถสรุปได้ว่า ข้อมูล Constraint ใดๆ ของนโยบายแบบ Database จะสามารถ Map ไปยัง XACML Tag <Condition> ได้ โดยที่ยังคงความหมายเช่นเดียวกันกับที่กำหนดไว้ในนโยบายแบบ Database นั้น เป็นจริง .....(6.3)

เนื่องจากนโยบายแบบ Database จะประกอบไปด้วย สิทธิ (action), บทบาท (role) และเงื่อนไข (ConstraintsType และ ConstraintValue) ดังนั้นจากข้อความใน (6.1), (6.2) และ (6.3) จึงสามารถสรุปได้ว่า Theorem 6 จะเป็นจริงด้วยเช่นกัน

<u>พิสูจน์ Algorithm 4</u>

จาก Theorem 1, 2, 3, 4, 5 และ 6 จึงส่งผลให้ Algorithm 4 นั้นจะมีคุณสมบัติด้านความครบถ้วนสมบูรณ์ (Existence and Completeness) และ ด้านความถูกต้อง (Correctness) ด้วยเช่นกันเนื่องจากเป็นการนำข้อมูลจาก Algorithm 1, 2 และ 3 ที่มีความครบถ้วนและถูกต้องมารวมกัน

ดังนั้น Mapping Algorithm ทุก Algorithm ที่นำเสนอมีคุณสมบัติในด้านความครบถ้วนสมบูรณ์ (Existence and Completeness) และ ด้านความถูกต้อง (Correctness) ตามที่ได้พิสูจน์ไว้แล้ว

3.3.3. Local Policy Update and Global XACML Policy Syncronization

ในกรณีที่มีความต้องการในการเปลี่ยนแปลงแก้ไข Local Policy เช่น เพิ่ม, แก้ไข หรือ ลบ กฎใน Local Policy จะมี ขั้นตอนการทำงานของ Agent ดังนี้

1. เมื่อมีการเปลี่ยนแปลงการกำหนดสิทธิในส่วนของ Local Policy เกิดขึ้น จะทำการ Restart Service ในการ Mapping Local Policy เพื่อให้ LAA ทำการ Mapping Local Policy ใหม่แล้วส่ง XACML Policy ที่มีการเปลี่ยนแปลงไปให้ GAA ได้รับทราบ
2. เมื่อ GAA ได้รับ XACML Policy จาก LAA แล้ว GAA ก็จะทำการรวม XACML Policy ใหม่เพื่อให้ได้ XACML Policy ที่มีข้อมูลล่าสุด

3.3.4. รูปแบบข้อความที่ใช้ในการสื่อสารของระบบหลายตัวแทน

รูปแบบข้อความในการสื่อสารในระบบหลายตัวแทนสามารถแบ่งตามการสื่อสารกันในแต่ละ Agent ได้ดังต่อไปนี้

1. การส่งข้อความคำร้องขอของ UA ไปยัง AA โดยข้อความจะประกอบด้วย <user_id, session_id, app_id, privilege, timestamp> โดยมีรายละเอียดดังต่อไปนี้

   - user_id คือค่าประจำตัวของผู้ใช้ที่ทำการส่งคำร้องขอ
   - session_id คือของเซสชั่นที่ใช้ในการสื่อสารคำร้องขอ (ซึ่งโดยปกติแล้วจะมีการสุ่มค่าเซสชั่นในตอนที่เริ่มเซสชั่น)
   - app_id คือค่าประจำตัวของแอพพลิเคชั่นที่ถูกร้องขอโดยผู้ใช้
   - privilege คือสิทธิที่ผู้ใช้ต้องการใช้งานจากแอพพลิเคชั่น
   - timestamp คือเวลาที่ UA ทำการส่งคำร้องขอ

2. การส่งข้อความเพื่อแจ้งให้ GAA ทราบว่านโยบายของ LAA มีการเปลี่ยนแปลง โดยจะมีการส่งข้อความจาก LAA ไปยัง GAA โดยข้อความจะประกอบด้วย <laa_id, session_id, update_req, timestamp> โดยมีรายละเอียดดังต่อไปนี้

   - laa_id คือค่าประจำตัวของ LAA ที่ทำการส่งคำร้องขอ
   - session_id คือของเซสชั่นที่ใช้ในการสื่อสารคำร้องขอ (ซึ่งโดยปกติแล้วจะมีการสุ่มค่าเซสชั่นในตอนที่เริ่มเซสชั่น)
   - update_req คือการร้องขอให้ GAA ทำการเปลี่ยนแปลงนโยบาย เนื่องจากมีการเปลี่ยนแปลงเกิดขึ้น
   - timestamp คือเวลาที่ LAA ทำการส่งคำร้องขอ

3. การส่งข้อความร้องขอจาก AA ไปยัง PA จะใช้รูปแบบมาตรฐานของ XACML Request ดังแสดงในภาพที่ 3.5

ภาพที่ 3.5 ตัวอย่าง XACML Request

```
<Request>
  <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>maker</AttributeValue></Attribute>
    <Attribute AttributeId="location" DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>192.168.71.1</AttributeValue></Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="http://www.w3.org/2001/XMLSchema#anyURI">
    <AttributeValue>https://www.epayment.com:6443/epayment/</AttributeValue></Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="http://www.w3.org/2001/XMLSchema#string">
    <AttributeValue>create_transaction</AttributeValue></Attribute>
  </Action>
</Request>
```

4. การส่งข้อความตอบกลับจาก PA ไปยัง AA จะใช้รูปแบบมาตรฐานของ XACML Response ดังแสดงในภาพที่ 3.6

ภาพที่ 3.6 ตัวอย่าง XACML Response

```
<Response>
  <Result ResourceID="https://www.epayment.com:6443/epayment/">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

3.3.5. ขั้นตอนการทำงานของ MAS

จากในภาพที่ 3.1 จะแสดงขั้นตอนการทำงานของ MAS ซึ่งจะแสดงให้เห็นถึงการสื่อสารกันระหว่างส่วนประกอบต่างๆ โดยจำนวนขั้นตอนการทำงานของ MAS จะประกอบด้วยขั้นตอนทั้งหมด 8 ขั้นตอน โดยจะทำการแบ่งเป็น 2 ส่วนคือ ส่วน Setup และส่วน Runtime ซึ่งจะมีรายละเอียดดังต่อไปนี้

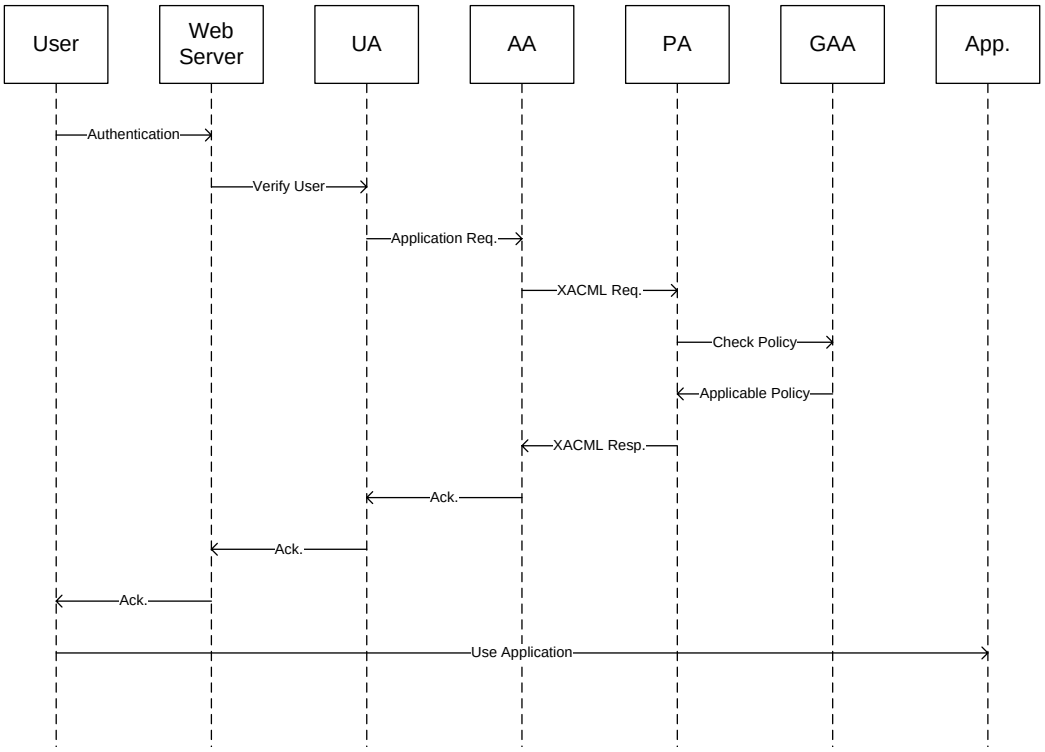
Setup Phase:

1. การยืนยันตัวบุคคลโดยใช้สององค์ประกอบ โดยผู้ใช้จะใช้สมาร์ทการ์ดหรือโทเคน ในการยืนยันตัวบุคคลผ่าน SSL กับเว็บเซิร์ฟเวอร์
2. โดยหลังจากที่ยืนยันตัวบุคคลโดยใช้สององค์ประกอบเรียบร้อยแล้ว เว็บเซิร์ฟเวอร์จะร้องขอไปยัง MAS ให้ทำการสร้าง UA ขึ้นมา โดย UA จะถูกจับคู่เข้ากับผู้ใช้เพื่อใช้ในการจัดการเกี่ยวกับคำร้องขอใช้แอพพลิเคชั่น โดยในส่วนนี้จะถือได้ว่า MAS เป็นส่วนที่ไว้วางใจได้ จึงได้ทำการสร้าง UA หลังจากที่ยืนยันตัวบุคคลได้สำเร็จ ซึ่งส่งผลให้ UA มีความไว้วางใจด้วยเช่นกัน
3. แล้ว UA จะทำการตรวจสอบใบรับรองอิเล็กทรอนิกส์และนำข้อมูลของชื่อผู้ใช้และรหัสผ่านของผู้ใช้ ซึ่งจะถูกเข้ารหัสด้วยกุญแจสาธารณะของผู้ใช้จาก LDAP มาถอดรหัสด้วยกุญแจส่วนตัวของผู้ใช้แล้วจึงนำมาเก็บไว้เพื่อใช้ในการยืนยันตัวบุคคลของผู้ใช้ก่อนเข้าใช้งานแอพพลิเคชั่นต่างๆ เพื่อสนับสนุนการเข้าใช้งานเพียงครั้งเดียว นอกจากนี้ยัง UA ยังทำการตรวจสอบผู้ให้บริการออกใบรับรองฯ ของผู้ใช้ด้วยว่าอยู่ใน Certificate Trust List หรือไม่ ซึ่งถ้าหากผู้ให้บริการออกใบรับรองฯ ของผู้ใช้ ไม่ได้อยู่ใน Certificate Trust List ผู้ใช้ก็จะไม่สามารถใช้งานแอพพลิเคชั่นได้

Runtime Phase:

4. หลังจากการตรวจสอบใบรับรองอิเล็กทรอนิกส์ และได้ข้อมูลของชื่อผู้ใช้และรหัสผ่านของผู้ใช้เรียบร้อยแล้ว UA จะทำการสร้างคำร้องขอไปยัง AA เพื่อขอเข้าใช้งานแอพพลิเคชั่น
5. หลังจากนั้น AA จะทำการส่ง XACML Request ไปยัง PA เพื่อให้ทำการตรวจสอบสิทธิของผู้ใช้ว่ามีสิทธิเข้าใช้งานแอพพลิเคชั่นที่ร้องขอมาได้หรือไม่
6. PA จะทำการตรวจสอบสิทธิการเข้าถึงของแอพพลิเคชั่น จากนโยบายศูนย์กลางที่อยู่ในรูปแบบของ XACML ผ่านทาง GAA (ซึ่งจะได้ XACML Policy มาจากกระบวนการ Mapping ที่ได้กล่าวไว้แล้วในหัวข้อ Policy Mapping and Integration)
7. หลังจากนั้น PA จะทำการส่งผลลัพท์ในรูปแบบของ XACML Response ไปยัง AA ให้รับทราบถึงสิทธิของผู้ใช้
8. เมื่อ AA ทราบแล้วว่าผู้ใช้มีสิทธิเข้าใช้งานแอพพลิเคชั่น AA จะทำหน้าที่รับผิดชอบการควบคุมการใช้งานแอพพลิเคชั่นจากผู้ใช้จำนวนมาก โดยทำการบริหารจัดการรายการของคำร้องขอตามลำดับ

ภาพที่ 3.7 แสดงขั้นตอนการทำงานของ MAS



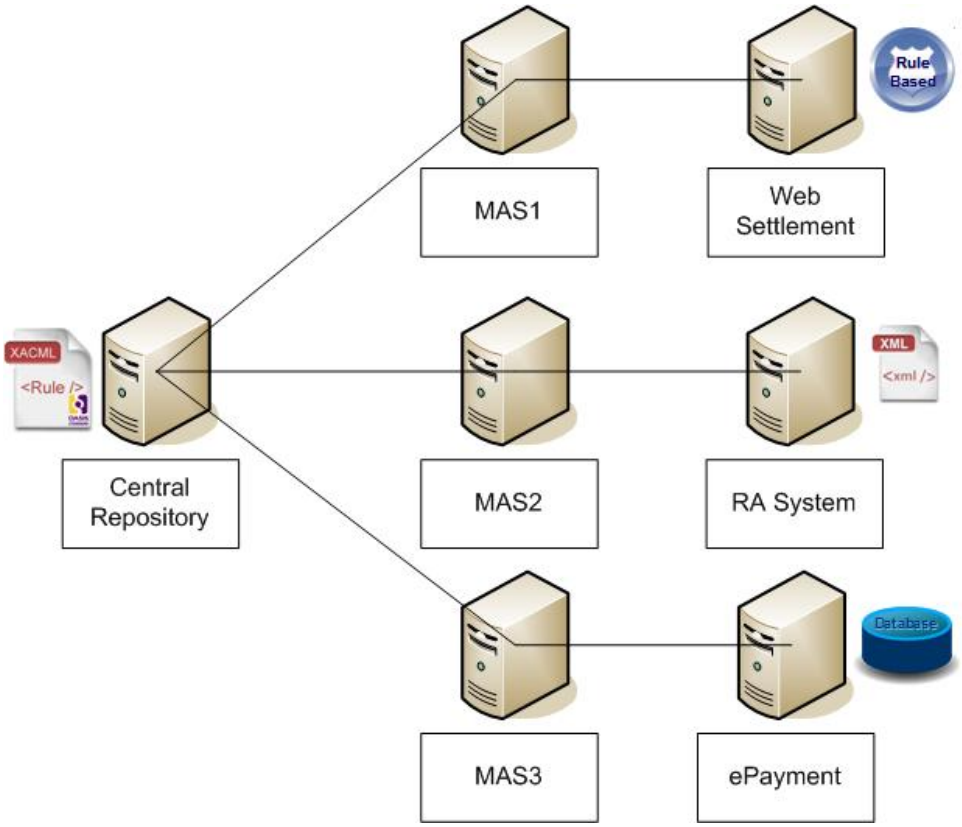
จากในภาพที่ 3.7 จะแสดงขั้นตอนการทำงานของ MAS ซึ่งจะแสดงให้เห็นถึงการสื่อสารกันระหว่างส่วนประกอบต่างๆ ในรูปแบบ Sequence Diagram ตามจำนวนขั้นตอนการทำงานของ MAS ทั้งหมด 8 ขั้นตอน ดังที่ได้กล่าวไว้แล้ว ซึ่งการสื่อสารกันระหว่างส่วนประกอบต่างๆ จะเป็นการสื่อสารกันแบบปลอดภัยเนื่องจากแต่ละตัวแทนจะมีคู่กุญแจเป็นของตัวเอง ทำให้ข้อมูลที่ส่งให้กันมีการเข้ารหัสเสมอ

3.4 ขั้นตอนการทดลอง

3.4.1. รูปแบบการทดลอง

ผู้วิจัยจะทำการทดสอบโดยใช้แอพพลิเคชั่นที่มีรูปแบบการอนุญาตเข้าใช้งานที่แตกต่างกันซึ่งได้แก่ Rule Based Policy, XML Policy และ Database Policy โดยจะมีการกำหนดเงื่อนไข (Constraint) ในรูปแบบ Role Constraint, Time Constraint, Location Constraint และ Event Constraint

ภาพที่ 3.8 แสดงภาพแอพพลิเคชั่นที่ใช้ในการทดลอง



โดยมีการใช้ rules ใน Rule-Based, XML และ Database อย่างละ 20 rule ในแต่ละแอพพลิเคชั่น ซึ่งแอพพลิเคชั่นที่จะนำมาใช้ในงานวิจัยนี้ได้แก่ Web Settlement, RA System

และ ePayment ซึ่งเป็นแอพพลิเคชั่นจากต่างองค์กรกัน โดยมีการกำหนดสิทธิเข้าใช้งานแบบ Rule-Based, XML และ Database ที่แตกต่างกันตามลำดับ

ในส่วนของการทดลองของงานวิจัยในชิ้นนี้ จะมีการใช้แอพพลิเคชั่นทั้งหมด 3 แอพพลิเคชั่นดังต่อไปนี้

1. Web Settlement เป็นแอพพลิเคชั่นที่ใช้ในการกระทบยอดข้อมูลการเงินระหว่างธนาคารต่างๆ เมื่อมีการโอนเงินระหว่างธนาคาร ซึ่งผู้ใช้งานในแอพพลิเคชั่น Web Settlement คือ ธนาคารที่ต้องการกระทบยอดข้อมูลการโอนเงินระหว่างธนาคารกัน โดยส่วนของการยืนยันตัวบุคคลจะใช้ชื่อผู้ใช้ร่วมกับรหัสผ่าน และส่วนของการกำหนดสิทธิเข้าใช้งานของระบบนี้ จะเป็นแบบ Rule-Based ซึ่งสามารถดูรายละเอียดได้ในภาคผนวก ก และ ข โดยหากนำมาทำการจำแนกความสัมพันธ์ของชื่อผู้ใช้กับบทบาท, บทบาท, สิทธิ และเงื่อนไขในการใช้งาน จะทำให้สามารถสรุปได้ดังตารางต่อไปนี้

ตารางที่ 3.2 แสดงความสัมพันธ์ของชื่อผู้ใช้กับบทบาทในแอพพลิเคชั่น Web Settlement

ชื่อผู้ใช้	บทบาท
user1	Operator1
user2	Operator2
user3	Operator3
user4	Administrator
user5	Auditor
user6	Operator1
user7	Operator2
user8	Operator3
user9	Administrator
user10	Auditor
user11	Operator1
user12	Operator2
user13	Operator3
user14	Administrator

user15	Auditor
user16	Operator1
user17	Operator2
user18	Operator3
user19	Administrator
user20	Auditor

ตารางที่ 3.3 แสดงบทบาทของแอพพลิเคชั่น Web Settlement

เลขที่	บทบาท
1.	Operator1
2.	Operator2
3.	Operator3
4.	Administrator
5.	Auditor

ตารางที่ 3.4 แสดงสิทธิของแอพพลิเคชั่น Web Settlement

เลขที่	สิทธิ
1.	Inquiry
2.	Settlement
3.	ApproveTransaction
4.	RejectTransaction
5.	ViewLog
6.	ViewAuditLog

ตารางที่ 3.5 แสดงกฎและเงื่อนไขของการใช้งานแอพพลิเคชั่น Web Settlement

เลขที่	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข
1.	Operator1	Inquiry	location	192.168.71.1
2.	Operator1	Inquiry	time	0600 - 1400
3.	Operator1	Settlement	location	192.168.71.1
4.	Operator1	Settlement	time	0600 - 1400
5.	Operator1	Settlement	event	amount < 10,000,000
6.	Operator2	Inquiry	location	192.168.71.1
7.	Operator2	Inquiry	time	0900 - 1700
8.	Operator2	Settlement	location	192.168.71.1
9.	Operator2	Settlement	time	0900 - 1700
10.	Operator2	Settlement	event	amount < 10,000,000
11.	Operator3	Inquiry	location	192.168.71.1
12.	Operator3	Inquiry	time	1100 - 1900
13.	Operator3	Settlement	location	192.168.71.1
14.	Operator3	Settlement	time	1100 - 1900
15.	Operator3	Settlement	event	amount < 10,000,000
16.	Administrator	ApproveTransaction	time	0600 - 1900
17.	Administrator	ApproveTransaction	event	amount < 10,000,000
18.	Administrator	RejectTransaction	time	0600 - 1900
19.	Administrator	ViewLog	time	0600 - 1900
20.	Auditor	ViewAuditLog	location	192.168.71.2

2. RA System เป็นแอพพลิเคชั่นที่ใช้ในการบริหารจัดการใบรับรองอิเล็กทรอนิกส์ ซึ่งผู้ใช้งานในแอพพลิเคชั่น RA System คือบริษัทที่ต้องการบริหารจัดการใบรับรอง โดยส่วนของการยืนยันตัวบุคคลจะใช้ชื่อผู้ใช้ร่วมกับรหัสผ่าน และส่วนของการกำหนดสิทธิเข้าใช้งานของระบบนี้ จะเป็นแบบ XML ซึ่งสามารถดูรายละเอียดได้ในภาคผนวก ค และ ง โดยหากนำมาทำการจำแนกความสัมพันธ์ของชื่อผู้ใช้กับบทบาท, บทบาท, สิทธิ และเงื่อนไขในการใช้งาน จะทำให้สามารถสรุปได้ดังตารางต่อไปนี้

ตารางที่ 3.6 แสดงความสัมพันธ์ของชื่อผู้ใช้กับบทบาทในแอพพลิเคชั่น RA System

ชื่อผู้ใช้	บทบาท
user1	Registration Authority Operator1
user2	Registration Authority Operator2
user3	Key Recovery Operator
user4	Registration and Recovery Operator
user5	Registration Authority Operator1
user6	Registration Authority Operator2
user7	Key Recovery Operator
user8	Registration and Recovery Operator
user9	Registration Authority Operator1
user10	Registration Authority Operator2
user11	Key Recovery Operator
user12	Registration and Recovery Operator
user13	Registration Authority Operator1
user14	Registration Authority Operator2
user15	Key Recovery Operator
user16	Registration and Recovery Operator
user17	Registration Authority Operator1
user18	Registration Authority Operator2
user19	Key Recovery Operator

user20	Registration and Recovery Operator

ตารางที่ 3.7 แสดงบทบาทของแอพพลิเคชั่น RA System

เลขที่	บทบาท
1.	Registration Authority Operator1 (RAO1)
2.	Registration Authority Operator2 (RAO2)
3.	Key Recovery Operator (KRO)
4.	Registration and Recovery Operator (RRO)

ตารางที่ 3.8 แสดงสิทธิของแอพพลิเคชั่น RA System

เลขที่	สิทธิ
1.	Issue
2.	Suspend
3.	Unsuspend
4.	Revoke
5.	Recovery

ตารางที่ 3.9 แสดงกฎและเงื่อนไขของการใช้งานแอพพลิเคชั่น RA System

เลขที่	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข
1.	RAO1	Issue	location	192.168.71.1
2.	RAO1	Issue	time	0800 - 1700
3.	RAO1	Suspend	location	192.168.71.1
4.	RAO1	Unsuspend	location	192.168.71.1
5.	RAO1	Unsuspend	time	0800 - 1700
6.	RAO1	Revoke	location	192.168.71.1
7.	RAO2	Issue	location	192.168.71.2

8.	RAO2	Issue	time	1100 – 2000
9.	RAO2	Suspend	location	192.168.71.2
10.	RAO2	Unsuspend	location	192.168.71.2
11.	RAO2	Unsuspend	time	1100 – 2000
12.	RAO2	Revoke	location	192.168.71.2
13.	KRO	Recovery	time	0800 – 2000
14.	KRO	Recovery	event	amount_cert < 5
15.	RRO	Issue	time	0800 – 2000
16.	RRO	Suspend	time	0800 – 2000
17.	RRO	Unsuspend	time	0800 – 2000
18.	RRO	Revoke	time	0800 – 2000
19.	RRO	Recovery	time	0800 – 2000
20.	RRO	Recovery	event	amount_cert < 10

3. ePayment เป็นแอพพลิเคชั่นที่ใช้ในการโอนเงินข้ามธนาคารแบบเรียลไทม์ ซึ่งผู้ใช้งานคือบริษัทที่ต้องการโอนเงินข้ามธนาคารแบบเรียลไทม์ โดยส่วนของการยืนยันตัวบุคคลจะใช้ชื่อผู้ใช้ร่วมกับรหัสผ่าน และส่วนของการกำหนดสิทธิเข้าใช้งานของระบบนี้ จะเป็นแบบ Database โดยหากนำมาทำการจำแนกความสัมพันธ์ของชื่อผู้ใช้กับบทบาท, บทบาท, สิทธิ และเงื่อนไขในการใช้งาน จะทำให้สามารถสรุปได้ดังตารางต่อไปนี้

ตารางที่ 3.10 แสดงความสัมพันธ์ของชื่อผู้ใช้กับบทบาทในแอพพลิเคชั่น ePayment

ชื่อผู้ใช้	บทบาท
user1	Maker1
user2	Maker2
user3	Checker
user4	Authorizer
user5	Maker1
user6	Maker2

user7	Checker
user8	Authorizer
user9	Maker1
user10	Maker2
user11	Checker
user12	Authorizer
user13	Maker1
user14	Maker2
user15	Checker
user16	Authorizer
user17	Maker1
user18	Maker2
user19	Checker
user20	Authorizer

ตารางที่ 3.11 แสดงบทบาทของแอพพลิเคชั่น ePayment

เลขที่	บทบาท
1.	Maker1
2.	Maker2
3.	Checker
4.	Authorizer

ตารางที่ 3.12 แสดงสิทธิของแอพพลิเคชั่น ePayment

เลขที่	สิทธิ
1.	CreateTransaction
2.	UpdateTransaction
3.	DeleteTransaction

4.	ViewTransaction
5.	ApproveTransaction
6.	RejectTransaction

ตารางที่ 3.13 แสดงกฎและเงื่อนไขของการใช้งานแอพพลิเคชั่น ePayment

เลขที่	บทบาท	สิทธิ	รูปแบบเงื่อนไข	ค่าของเงื่อนไข
1.	Maker1	CreateTransaction	time	0800 – 1900
2.	Maker1	CreateTransaction	location	192.168.71.1
3.	Maker1	CreateTransaction	event	amount < 1,000,000
4.	Maker1	UpdateTransaction	time	0800 – 1900
5.	Maker1	UpdateTransaction	location	192.168.71.1
6.	Maker1	DeleteTransaction	time	0800 – 1900
7.	Maker1	DeleteTransaction	location	192.168.71.1
8.	Maker2	CreateTransaction	time	0800 – 1900
9.	Maker2	CreateTransaction	location	192.168.71.2
10.	Maker2	CreateTransaction	event	amount < 5,000,000
11.	Maker2	UpdateTransaction	time	0800 – 1900
12.	Maker2	UpdateTransaction	location	192.168.71.2
13.	Maker2	DeleteTransaction	time	0800 – 1900
14.	Maker2	DeleteTransaction	location	192.168.71.2
15.	Cheker	ViewTransaction	time	0800 – 1900
16.	Cheker	ViewTransaction	location	192.168.71.3
17.	Cheker	ViewTransaction	event	amount < 5,000,000
18.	Authorizer	ApproveTransaction	time	0700 – 2000

19.	Authorizer	ApproveTransaction	event	amount < 5,000,000
20.	Authorizer	RejectTransaction	time	0700 – 2000

โดยแต่ละแอพพลิเคชั่นดังกล่าวเหล่านี้จะกำหนดให้มีจำนวนผู้ใช้คือ 20 ผู้ใช้ และแต่ละผู้ใช้มีความจำเป็นต้องใช้แอพพลิเคชั่นจากต่างองค์กรดังกล่าวนี้ด้วย

3.5 วิธีวัดผลการทดลอง

วิธีวัดผลการทดลองในงานวิจัยนี้แบ่งออกเป็น 2 หัวข้อหลัก ได้แก่ การตรวจสอบความถูกต้องในการทำงานของระบบ ARMORS และ การวัดระยะเวลาการทำงานของระบบ ARMORS กับระบบเดิม โดยแต่ละหัวข้อมีรายละเอียดดังนี้

3.5.1. การตรวจสอบความถูกต้องในการทำงานของระบบ ARMORS

ในส่วนนี้จะอธิบายถึงวิธีการตรวจสอบความถูกต้องในการทำงานของระบบ ARMORS โดยสามารถแบ่งการตรวจสอบความถูกต้องออกเป็น 2 ส่วนดังต่อไปนี้

3.5.1.1. การตรวจสอบความถูกต้องในส่วนของการยืนยันตัวบุคคล

ในส่วนของการยืนยันตัวบุคคลในงานวิจัยนี้เป็นการยืนยันตัวบุคคลด้วยสององค์ประกอบ โดยองค์ประกอบแรกคืออุปกรณ์ USB-Token ซึ่งภายในอุปกรณ์นี้จะมีการบรรจุใบรับรองอิเล็กทรอนิกส์ไว้ และองค์ประกอบที่สองคือรหัสผ่าน หรือ PIN ของอุปกรณ์ USB-Token ซึ่งในการยืนยันตัวบุคคลแบบสององค์ประกอบนี้จะขาดอย่างใดอย่างหนึ่งไปไม่ได้ นอกจากนี้ยังได้มีการตรวจสอบสถานะของใบรับรองอิเล็กทรอนิกส์อีกด้วยว่าถ้าสถานะของใบรับรองอิเล็กทรอนิกส์เป็นปกติจึงจะสามารถยืนยันตัวบุคคลผ่าน และถ้าสถานะของใบรับรองอิเล็กทรอนิกส์เป็นการพักการใช้งานหรือถูกยกเลิกก็จะไม่สามารถยืนยันตัวบุคคลผ่าน ดังนั้นจะทำการทดสอบดังรายละเอียดตามตารางดังต่อไปนี้ เพื่อดูผลลัพท์ในแต่ละกรณีว่าสามารถยืนยันตัวบุคคลผ่านหรือไม่

ตารางที่ 3.14 กรณีที่จะใช้ทดสอบความถูกต้องในส่วนของการยืนยันตัวบุคคล

สถานะใบรับรองฯ	อุปกรณ์ USB-Token	PIN
สถานะเป็นปกติ	ใช้	ถูกต้อง
สถานะเป็นปกติ	ใช้	ไม่ถูกต้อง
สถานะเป็นปกติ	ไม่ใช้	ถูกต้อง
สถานะเป็นปกติ	ไม่ใช้	ไม่ถูกต้อง
พักการใช้งานหรือถูกยกเลิก	ใช้	ถูกต้อง

พักการใช้งานหรือถูกยกเลิก	ใช้	ไม่ถูกต้อง
พักการใช้งานหรือถูกยกเลิก	ไม่ใช้	ถูกต้อง
พักการใช้งานหรือถูกยกเลิก	ไม่ใช้	ไม่ถูกต้อง

3.5.1.2. การตรวจสอบความถูกต้องในส่วนของการกำหนดสิทธิเข้าใช้งาน

ในแต่ละบทบาทของผู้ใช้จะมีการกำหนดเงื่อนไขในการเข้าใช้งานไว้ ซึ่งหากได้ทำการกำหนดให้เงื่อนไขทั้งหมดคือเซ็ต C โดยมีสมาชิกเป็น c_1 ถึง c_n และ n คือจำนวนเงื่อนไขทั้งหมด ดังนั้นถ้าหากเงื่อนไขของทุกสมาชิกตามที่กำหนดไว้ในเซ็ต C เป็นจริงทั้งหมดแล้วผลลัพท์ที่ได้จะเป็นจริง และถ้าหากเงื่อนไขของบางสมาชิกในเซ็ต C มีค่าความจริงเป็นเท็จแล้วผลลัพท์ที่ได้จะเป็นเท็จ

ดังนั้นจะทดสอบโดยการให้ผู้ใช้เข้าใช้งานระบบเดิมและระบบ ARMORS ด้วยข้อมูลนำเข้าชุดเดียวกันซึ่งอาจมีใช้เงื่อนไขที่เป็นจริงหรือเท็จก็ได้ แล้วดูผลลัพท์ในการเข้าใช้งานของแต่ละระบบว่าผลลัพท์การเข้าใช้งานของผู้ใช้ถูกต้องตรงกันตามนโยบายที่ได้กำหนดไว้หรือไม่ ซึ่งหากว่ามีการกำหนดข้อมูลนำเข้าที่เหมือนกันแล้วผลลัพท์การเข้าใช้งานของผู้ใช้ก็จะต้องตรงกันจึงจะถือได้ว่าระบบARMORS สามารถทำงานได้อย่างถูกต้องในส่วนของ การกำหนดสิทธิเข้าใช้งาน

3.5.2. การวัดระยะเวลาการทำงานของระบบ ARMORS กับระบบเดิม

ในส่วนนี้จะอธิบายถึงวิธีการวัดระยะเวลาการทำงานของระบบ ARMORS กับระบบเดิม โดยจะทำการวัดระยะเวลาตั้งแต่ขั้นตอนของการยืนยันตัวบุคคล จนถึงขั้นตอนการกำหนดสิทธิเข้าใช้งาน กระทั้งผู้ใช้สามารถใช้งานได้สำเร็จ โดยผู้วิจัยจะทำการทดลองซึ่งจะกำหนดให้มีจำนวนผู้ใช้ตั้งแต่ 100 แล้วเพิ่มขึ้นครั้งละ 100 จนถึง 1,000 แล้วจะนำผลลัพท์ที่ได้มาทำการเปรียบเทียบระยะเวลาระหว่างระบบ ARMORS กับระบบเดิม