

Received June 5, 2016, accepted June 21, 2016, date of publication June 27, 2016, date of current version July 22, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2585185

Automated Policy Combination for Secure Data Sharing in Cross-Organizational Collaborations

LI DUAN^{1,2}, YANG ZHANG¹, SHIPING CHEN², SHUAI ZHAO¹, SHIYAO WANG¹, DONGXI LIU², REN PING LIU^{2,3}, (Senior Member, IEEE), BO CHENG¹, AND JUNLIANG CHEN¹

¹State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

²Data61, Commonwealth Scientific and Industrial Research Organization, Marsfield, NSW 2122, Australia

³University of Technology Sydney, Ultimo, NSW 2007, Australia

Corresponding author: L. Duan (duanli@bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61372115, Grant 61132001, and Grant 61501048, in part by the China Postdoctoral Science Foundation funded project under Grant 2016T90067 and Grant 2015M570060, and in part by the National Grand Fundamental Research 973 Program of China under Grant 2013CB329102.

ABSTRACT During business collaborations, multiple participating organizations often need to share data for common interests. In such cases, it is necessary to combine local policies from different organizations into a global one in order to manage access to the shared data. However, local policies of organizations may be different or even conflicting, due to diverse rules and rule combining algorithms chosen. Few existing methods for policy combination are able to automatically combine multiple local policies into a global one. In this paper, we propose a bottom-up approach to address the issues of multiple policy combinations. The key idea is to first classify the rules based on attribute constraints in each policy, and then reduce the rules of the corresponding classes to one with the same attribute constraints. The reduced rules are then combined into a new global policy by choosing the appropriate rule combining algorithm in XACML. The latter ensures compliance with each of the local policies at syntax and semantic levels. To validate our approach, we develop a proof-of-concept implementation of the automated policy combination. Experimental results demonstrate that our approach is highly scalable and supports a number of attribute constraints in each local policy.

INDEX TERMS XACML, collaboration, data sharing, policy combination, access control policy.

I. INTRODUCTION

A. MOTIVATION

Organizations often collaborate with each other in order to provide better services to customers [1]. Service oriented computing (SOC) provides a promising paradigm for business collaborations. The main objectives in such collaborations are data sharing, where the shared data may be sensitive, such as patient's medical record in healthcare information system (HIS) [2], [3]. Hence, the focus on protecting data privacy and security is becoming a crucial requirement [4], [5].

Access control is one of the most important parts of data privacy and security. Its goal is to prevent unauthorized access to the protected data [6]. However, realizing access control for shared data [7] is challenging due to the multiple collaborative organizations involved. In order to address this challenge, it is necessary for the participating organizations to establish a common access control policy, which is a global access control policy that can be accepted by all collaborative organizations. Creating such a policy is usually carried

out through certain principles (e.g., compromise, negotiation [8]) among all the participating organizations. Taking service combination for example, the policy of combined service is generated by integrating all the component service policies [9]. Thus, the key of deciding access control policy for shared data is to combine local policies from different participating organizations into a global one.

Generally speaking, there are two levels of policies for data sharing [10]: one is coarse-grained data level, that is organizational level, where data can be files or database as well as other information; the other is fine-grained data level related to data structure. In this paper, we mainly focus on the coarse-grained organizational data level. In the environment of cross-organizational collaborations, the shared data is usually owned and managed by various organizations. To protect their data, different participating organizations may choose different elements and access control constraints to independently specify policy rules to regulate how their data can be used. Such differences may result in misunderstandings

among organizations. Furthermore, these organizations may define different or even conflicting policy rules for shared data [11]. For example, one rule allows to carry out certain operations on data, but other one does not allow to carry out the same operations on it. As such, how to model and integrate policy rules as well as how to resolve the conflicts among these rules are key challenges in policy combination.

To address the above mentioned challenges of policy combination, the first task is to specify access control policy requirements of each participating organization. Policy languages play an important role in expressing these requirements. Various types of policy languages have emerged, such as XACL [12], EPAL [13] and the eXtensible Access Control Mark-up Language (XACML) [14]. They have provided certain approaches to combine policies. However, they mainly focus on supporting the pre-specified policy combining algorithms, such as permit-override, deny-override and so on. These policy languages are insufficient to support the complex semantics of policy combination for data sharing. For example, they do not specify more restrictive policy combining algorithms, e.g., the combined policy permits a request when all the policies permit it, and denies a request when any one of policies denies it, which will be the new principles used to combine policies in this paper.

Among these existing policy languages, XACML is the most popular one. It provides the most flexible approach to manage all the elements of each policy. XACML supports attribute-based access control model [15], which makes attribute-based constraint rules become one of the popular access control methods in a distributed collaboration environment. Thus, we focus on attribute-based policy combination. Additionally, XACML allows one policy consist of more than one attribute-based constraint rules. To combine these rules, XACML specifies some rules and policy combining algorithms, which contain permit-override, deny-override, permit-unless-deny and deny-unless-permit and so on. However, if one organization utilizes different XACML rule combining algorithms to combine its rules, it will obtain different policy. For example, if organization A adopts permit-override to combine its rules, the result is to permit a request if any rule permits it. Whereas, if organization A adopts deny-override to combine its rules, the result is to deny a request if any rule denies it. Thus, it is necessary to consider rule combining algorithm used in each policy during policy combination.

The attribute constraint is to put restricts on an attribute. The process of policy combination is very complex, because of many attribute constraints existed in a policy. It is helpful to construct a policy scheme by means of algebraic theory [17], which can be used to describe the behaviors of policy rule combination, and to verify the correctness of rule combination [19], [20]. Recently, there have been many policy combination algebraic systems, such as policy combination language (PCL) based on automata theory presented in [16], fine-grained integration algebra (FIA) based on logical expressions [21], access control system based on propositional algebra [29] and so on. These algebra systems can

deal with limited attribute constraints and provide theoretical support for our research.

Determining the global policy for the shared data from multiple organizations is a challenging problem. From a request point of view, in order to determine whether a request is allowed to access the shared data, it should determine whether the request matches the global policy of the collaborating organizations. That requires combining local policies from different organizations into a global one. The combined policy not only needs to be in full compliance with the policies from all these organizations, but also must be accepted by all organizations. The access control policy for shared data is usually established through negotiations or reconciliation [8] among various participating organizations. It is important to select an adequate XACML rule combining algorithm in the new global policy. Moreover, when receiving a number of various local policies, few automated policy combination tools exist that can automatically generate a global policy in XACML, other than a policy decision.

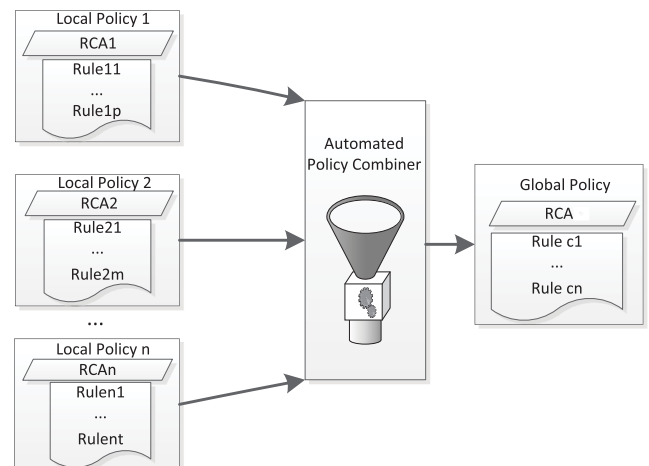


FIGURE 1. Multiple policy combination architecture.

In this paper, we present a policy combination architecture shown in Fig.1. We adopt a rule reduction approach and develop an automated tool, which can be used to generate a global policy by combining various policies from different organizations. Based on XACML standard specification, we extend the FIA algebraic operator system [23], [24] by defining reducing operators to formally specify each policy rule to support a wide range of attribute values in a policy. In the Policy Combination Architecture of Fig. 1, for multiple policies P_1, P_2, \dots, P_n , each policy P_i ($1 \leq i \leq n$) consists of a set of policy rules (R_{i1}, \dots, R_{im}), and also has a rule combining algorithm RCA_i . In this architecture, we first classify the rules based on the attribute constraints. We then reduce the rules of the corresponding classes to one with the same constraints. After the reduction, the comparison of the conditional attributes (e.g., the attributes defined in the conditions of each rule) is carried out by means of predefined reducing operations in the first step. The decision of

a policy applied to a request relies on the decisions of its composing rules [25]. Thus, a new global policy is created by combining the reduced rules, and selecting an appropriate rule combining algorithm *RCA*, which is chosen according to the algorithms used in all the participating organizations. The rule-reduction-based approach makes the combined policy more restrictive, that is, the combined global policy permits a request only when all the policies permit it, denies a request when any one of policies denies it. The creation of a global policy is conducted in the Automated Policy Combiner as shown in the central part of Fig. 1.

Compared to the idea of policy decomposition approach in [24], which adopts a top-down approach to decompose a global policy into local rules, we adopt a bottom-up approach to decompose the rules included in a policy into different classes according to their attribute constraints. The rules in each class have the same attribute constraints, and we use attribute-based combination approach to combine these rules. For the permitting rules, our tool creates the logical intersection of these rules as a reduced rule with permit effect in the global policy. For the denying rules, our tool creates the logical union of these rules as a reduced rule with deny effect in the global policy. For the conflicting rules, our tool first transforms them into ones with the same effects, and then reduces these rules by choosing proper reducing operators.

Finally, the combined global policy is obtained by traversing all the attribute constraints and combining the reduced rules by choosing an appropriate rule combining algorithm specified in XACML. The generated policy ensures compliance with each of the local policies at syntax and semantic levels.

Our previous work on policy combination was reported earlier in SCC2015 research track [39], where we assumed that all collaborating organizations adopted the same rule combining algorithms to specify their policies. This paper expands that work by considering different rule combining algorithms in different local policies, as well as adding the following contents. First of all, we discuss how to combine policies that have conflicting rules and different rule combining algorithms. Secondly, we describe the related work in more detailed. Thirdly, we improve our multiple policy combining algorithms and provide its proof-of-concept implementation. Beyond that, we carry out an experimental evaluation of our policy combination tool.

Our contributions in this paper are as follows: (1) We adopt bottom-up approach to decompose the rules of a policy into different classes based on the attribute constraints. The rules in a class have the same constraints. Condition-based attribute combination is used to combine these rules in a class. (2) We present a rule-reduction approach to combine rules with the same attribute constraints. The reduced rules are combined in a global policy by choosing a rule combining algorithm in XACML. (3) We develop a proof-of-concept implementation for our policy combination algorithm, which is applied in a practical case study. (4) Finally, we have carried out an experimental evaluation of the policy combination tool.

Experimental results validate our approach, and demonstrate the scalability of our automated policy combination algorithm.

B. ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows. Section 2 firstly presents related work on policy combination, and then reviews the principal concepts of XACML policy. Section 3 introduces basic definitions and rule combination operators that will be used in this paper, as well as the logical expressions of rule combining algorithms in XACML policy specification. Section 4 introduces policy expression. Section 5 presents our policy combination approaches, mainly consist of the detailed procedures and related algorithms of generating a global policy. Section 6 presents our implementations. Section 7 concludes this paper.

II. RELATED WORK AND XACML OVERVIEW

In this section, we firstly survey related work on policy combination, and then present the principal concepts in XACML policy.

A. RELATED WORK

Recently, there have been much work on the issues of policy combination [21], [22], [28], [30]. Existing policy combination approaches usually carried out in the aspects of policy specification languages, policy combination algebra theory and data sharing-based policy negotiation. Thus, we will carry out literature reviews from the above three aspects.

We first discuss work related to policy languages. Existing policy specification languages consisting of XACL [12], EPAL [13], XACML [14] have provided some approaches to combine policies. However, they only support the pre-specified policy combining algorithms, such as permit-override, deny-override and so on, which are insufficient to support the complex semantics of policy combination for data sharing. Compared to these specification languages, we have presented a new policy combination principle that is not specified in XACML. However, we have formally expressed some definitions based on the semantics of XACML policy.

We then review work related to policy combination algebra. The algebra theory is the most expressive approach for describing the behaviors of policy combination. The earliest work was by Mclean [27] introduced grid-based policy combination framework of mandatory access control. Bonatti et al. [28] proposed set theory-based access control policy combination algebra, in which a set of subject, object and action attribute tuples are used to define access control policies, and logic operations (e.g., addition, conjunction, subtraction) are used to express policy combination. This work provides the foundation for the following policy combination research. Wijesekera and Jajodia [29] proposed a propositional algebra approach for combining policy, in which policy is formally expressed as a nondeterministic transformers set of assignment permission. They described authorization rules and combinational logics by means of

propositional operations. Mazzoleni et al. [30] presented an algebra system for combining fine-grained authorization policies for different participating organizations. The common limitation of above work is that only the simple and limited attribute constraints can be dealt with in these policy combination algorithm. Compared to these work, we have presented one policy combination algebra system, which not only supports complex computation of attribute values, but also supports a number of attribute constraints defined in each policy.

Additionally, there are also some other approaches to combine policies. Ferraiol et al. [16] proposed policy combination approach based on policy machine, but they did not present policy combination logics. Backes et al. [17] introduced a 3-valued algebra for combining policies, that replies to the requests either with “Permit”, “Deny”, or “Not Applicable”. They also introduced algebraic operations (e.g., addition, conjunction, subtraction, negation, constraint) and their properties. Their works was based on EPAL policy specification. Li et al. [31] introduced a policy combination language (PCL) to model each policy. Few of these work generate XACML policies as result of policy combination. Negotiation-based policy combination has been suggested for multiple policies combination [8], [32], [33], [34], [35]. In [8], similarity-based policy adaptation approach was proposed to avoid conflicts in authorization rules, and negotiation-based approach was adopted to combine policies. However, if some organizations are unwilling to negotiate with others, their approach cannot be applied.

Rao et al. [21], [22] proposed a fine-grained integration algebraic system and presented an approach of generating the actual combination XACML policy. In their approach, a policy is defined by the set of requests that the policy applies to, that is, a policy can be expressed as the set of requests that are permitted by the policy, the set of requests that are denied by the policy and the set of requests that are Notapplicable. Second, they presented a series of policy combination operators, and described the combination semantics by requests set. Later, the generated policy is translated into an XACML policy by using a multi-terminal binary decision diagram. This method supports the complex policy combination semantics such as policy jumps. This work have complete algebra theory and experimental results. Among these policy combination works, to the best of our knowledge, this is the only one that focuses on generating the actual policy. Compared with their work, we extended the fine-grained integration algebraic with reducing operators, and our work also focuses on generating an XACML policy. Unlike their work that they do not consider rule combining algorithms in the combined policy, our work focus on discussing the results of policy combination with different rule combination algorithms in XACML. What is more, their works focus on formal specification of the policy combination, and there is no proper tool to support the automatic combination of multiple policies to a global policy in XACML. In this paper, we define the mapping operators between various kinds of attribute constraints to support more

attribute variables in each policy. Moreover, we develop an automated policy combination tool.

B. XACML POLICY OVERVIEW

XACML is an OASIS standard language for specifying access control policies. It can not only express the properties of subjects, actions, objects and environments, but also make an evaluation on the request. When dealing with policy combination, the first task is to construct unified policy model, which is based on security requirements of each organization. XACML defines some rule combining algorithms, which are used to resolve conflicts and redundancy in a policy. In this part, we firstly review XACML policy model, then introduce the definitions used in this paper as well as the rule combining algorithms in XACML and their logical expressions [14], [36].

In XACML there are the PEP (Policy Enforcement Point), PDP (Policy Decision Point) and PIP (Policy Information Point), which can dynamically evaluate an access request and make a decision according to resources, requested information and condition constraints. In general, a subject requests an action to be executed on a resource through PEP, and the policy decides whether the request is denied or permitted to execute that action in PDP.

The elements in an XACML policy mainly contain a policy target, a set of rules, a rule combining algorithm and obligations. The policy target specifies a set of requests that the policy is applicable to. It defines a set of attribute constraints characterizing subjects, objects and actions and environment that the policy apply to. A rule, as the smallest element in policy, consists of the target, a condition and an effect. It can be applied to define authorization constraints. The rule target has the same structure as the policy target. It identifies a group of requests that the rule is applicable to. The condition specifies restrictions on the attributes in the target, which supports attribute-based access control. In the constraints, access policies can be expressed as the conditions against the properties and actions. The effect specifies whether the request actions should be “permitted” or “denied”. A rule is specified by only one effect. If an access request matches the rule target and satisfies the conditions, the rule is applicable to the request and yields the decision specified by the effect element. The rule combining algorithm is applied to resolve conflicts and avoid redundancy among applicable rules in a policy. It specifies how to combine the rules with different effects to generate one policy with one effect, that is, a policy is generated by integrating different rules. Here, we consider four kinds of popular rule combining algorithms, that are permit-override, deny-override, permit-unless-deny and deny-unless-permit. For example, permit-override combining algorithm shows that a policy permits a request in case at least one of its rules permits it. Obligations represent actions to be executed in conjunction with the enforcement of an authorization decision. The policy set is a set of XACML policies. In this paper, the obligations are out of our considerations. In the following, we motivate our work by an example of

XACML policies in a Health Information System (HIS), that will be used throughout the paper.

HIS Example: in a medical database, a large sum of the dispersed medical data is recorded by different organizations. In order to protect patient’s privacy, each organization has its policy for its recorded data. However, some treatments require data sharing across multiple organizations. Taking Organization A (OrgA), Organization B (OrgB), Organization C (OrgC) and Organization D (OrgD) as examples. For the medical data, assume that the policies of four Organizations are P_1, P_2, P_3 and P_4 respectively. The detailed policy descriptions are as follows:

- P_1 states that doctors are allowed to write medical data if their trust level is greater than or equal to 8.
- P_2 states that doctors and nurses are allowed to write medical data if their trust level is greater than or equal to 6. However, any doctors are not allowed to write medical data if their seniority is less than or equal to 10.
- P_3 states that doctors can read and write medical data if their seniority is greater than or equal to 7, and any doctors with trust level greater than or equal to 4 are authorized to write medical data. However, nurses are not allowed to write medical data if their trust level is less than or equal to 6.
- P_4 states that doctors and nurses can read and write medical data if their trust level is greater than or equal to 3, and any doctors with seniority less than or equal to 5 are not authorized to write medical data.

Each local policy used in our example can be written in an XACML framework as shown in Policy1.xml (Fig.2), where $\langle policyId \dots \rangle$ is the policy identifier, $\langle RuleCombiningAlgId = \dots \rangle$ specifies the rule combining algorithm. The policy P_1 has one rule R_{11} , the effect *Permit*, the target $\langle Target \rangle$ and the condition constraints $\langle Condition \rangle$.

III. POLICY COMBINATION OPERATORS

To facilitate the combination of their policies, all organizations should specify their policies by using the same language, like XACML. XACML supports attribute-based access control policy, in which attributes are used to specify the constraints on subjects, actions, objects and environments. In this section, we firstly present some definitions related to a policy, and then construct a policy combination algebraic framework by introducing rule combination operators. Table 1 lists the notations to be used.

A. BASIC DEFINITIONS

In order to specify policy, we present the following definitions related to policy specification. In a policy, the attributes are a set of constraints characterizing subjects, objects, actions, and environments that the policy applies to. A subject is a requestor who requests to carry out operations on objects. An object is a resource (e.g., files, data) to be protected from unauthorized access. An action is an operation.

```

Policy1.xml
<Policy PolicyId="Policy1" RuleCombiningAlgId=".....">
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch MatchId="...function:rfc822Name-match">
          <AttributeValue DataType="...#string">
            Doctor
          </AttributeValue>
          <SubjectAttributeDesignator AttributeId="...:subject-id"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="...:anyURI-equal">
          <AttributeValue DataType="... #anyURI">
            http://datashare.org/orgA/data/
          </AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
  </Target>
  <Rule RuleId="CommitRule" Effect="Permit">
    <Target>
      <Actions>
        <Action>
          <ActionMatch MatchId="...:string-equal">
            <AttributeValue DataType="... #string">
              Write
            </AttributeValue>
            <ActionAttributeDesignator AttributeId="...:action-id"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="...:integer-greater-than-or-equal">
      <Apply FunctionId="...:integer-one-and-only">
        <EnvironmentAttributeDesignator AttributeId=".../ >
      </Apply>
      <AttributeValue DataType="... #integer">8</AttributeValue>
    </Condition>
  </Rule>
  <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
    
```

FIGURE 2. Policy P_1 .

TABLE 1. Notations.

s, a, o	subject-id, action-id, object-id
att	attribute name
$\alpha \in \{=, \neq, >, >=, <, <=, \in\}$	attribute operation
$e \in \{Y, N\}$	the effect of a rule
$V(att)$	the domain of the attribute att
$v(att)$	one value of the attribute att
$r = ((att, v(att)))$	an access control request
$S = (S_1, S_2, \dots, S_K)$	a tuple of subject attributes
$A = \{act_1, \dots, act_m\}$	a set of actions
$E = (E_1, E_2, \dots, E_T)$	a tuple of environment attributes
$C = \{C_1, C_2, \dots, C_t\}$	condition attribute constraints
$P = \{P_1, P_2, \dots, P_n\}$	a set of policies
$R_i = \{R_{i1}, R_{i2}, \dots, R_{im}\}$	a set of rules
RCA	rule combining algorithm
PO	Deny Overrides
DO	Deny Override
PD	Permit-unless-Deny
DP	Deny-unless-Permit

The environment (e.g., security level, trust level) is the condition within which a requestor is to be evaluated.

Definition 1 (Attributes): A subject attribute S_k is denoted by $S_k = (s, V_k(s))$, an action attribute denoted by $A_h = (a, V_h(a))$, an object attribute denoted by $O_m = (o, V_m(o))$ and an environment attribute denoted by $E_t = (att_t \alpha V(att_t))$, s, a and o are attribute subject-id, action-id and object-id, $V(s), V(a)$ and $V(o)$ are the domain of subjects, actions and objects, att is the attribute name, α is the attribute operation, $V(att)$ is the attribute value.

In the above definition, the attributes of action act are enumerated values, such as *read*, *write*. The attribute value domains V can be constant, numerical interval (closed interval or open interval) and set. When it is a constant, the operation $\alpha \in \{=, \neq, >, <\}$, where $>$ expresses partial order, such as priority. When an attribute value is an interval, $\alpha \in \{=, \neq, >, \geq, <, \leq, \in\}$, and it is used for numerical comparison. When the attribute value is a set, $\alpha \in \{=, \neq, \in\}$. Each operation α has the inverse operation, notated as $\neg \alpha$. For example, when an operation $\alpha = “\leq”$, its inverse operation is $\neg \alpha = “>”$. In XACML, the conditions in a rule show the constraints that a request subject should satisfy to carry out the corresponding actions on objects. Based on this, the policy rule is defined as follows.

Definition 2 (Policy Rule): An attribute-based policy rule is formally defined as $R(e) = \langle S_k \wedge A_h \wedge O_m \wedge C \rangle$, where e is rule effect, either “permit (Y)” or “deny (N)”, that is to say, $e \in \{Y, N\}$, $C = \langle C_1, C_2, \dots, C_n \rangle$ are the set of attribute constraints, each $C_i = (att_i \alpha V(att_i))$. This rule means that the subjects from $V_k(s)$ are permitted or denied to carry out the actions $V_h(a)$ on objects $V_m(o)$ when its attribute values satisfy the corresponding attribute predicates.

In HIS Example, for simplicity, we use the notations “doc”, “nur”, “re”, “wr”, “sen”, “secl” and “trul” to replace “doctor”, “nurse”, “read”, “write”, “seniority”, “security level” and “trust level” respectively. In this paper, we assume that the default access object is medical data, so we remove the object in the definitions. From the above definition, the first rule in P_2 can be notated as $R(N) = \langle (s, doc) \wedge (a, wr) \wedge (sen \leq 10) \rangle$, other elements defined in P_i , $i \in \{1, 2, 3, 4\}$ are shown in Table 2.

TABLE 2. Policy example.

policy	rule	subject	action	object	condition	effect
P_1	R_{11}	doc	wr	med	$trul \geq 8$	permit
P_2	R_{21}	doc	wr	med	$sen \leq 10$	deny
	R_{22}	doc,nur	wr	med	$trul \geq 6$	permit
P_3	R_{31}	doc	re, wr	med	$sen \geq 7$	permit
	R_{32}	doc	wr	med	$trul \geq 4$	permit
	R_{33}	nur	wr	med	$secl \leq 6$	deny
P_4	R_{41}	doc	wr	med	$sen \leq 5$	deny
	R_{42}	doc,nur	re, wr	med	$trul \geq 3$	permit

A request contains all the attribute information required to access data, consisting of subject attributes, action attributes, environment attributes and other information such as the trust level, the current time. A request is denoted by $r = \{(s, v(s)), (a, act), (att, v(att))\}$.

Definition 3 (Request Matching): For a request $r = \{(s, v(s)), (a, act), (att, v(att))\}$ and a rule $R(e) = \langle (s, V(s)) \wedge (a, V(a)) \wedge (att \alpha V(att)) \rangle$, the request r matches the rule $R(e)$ if and only if $v(s) \in V(s)$ and $act \in V(a)$ as well as $v(att) \alpha V(att)$.

For the rules with the same attribute constraints, there are two kinds of relations between them: compatible and conflicting. If two rules have the same effect, they are compatible, otherwise, they are conflicting.

Definition 4 (Conflicting Rules): Let R_i and R_j be two rules, $R_i(e_i) = \langle (s, V_i(s)) \wedge (a, V_i(a)) \wedge (att_i \alpha_i V_i(att_i)) \rangle$, and $R_j(e_j) = \langle (s, V_j(s)) \wedge (a, V_j(a)) \wedge (att_j \alpha_j V_j(att_j)) \rangle$. R_i and R_j are conflicting rules if and only if $att_i = att_j$, $e_i \neq e_j$ and $\alpha_i = (\neg) \alpha_j$.

In HIS example, R_{11} in P_1 and R_{22} in P_2 have the common trust level constraint *trul* and the same effect *permit*, so R_{11} and R_{22} are compatible rules. On the contrary, R_{21} in P_2 and R_{31} in P_3 have the common seniority constraint *sen*, but they have different effects, so R_{21} and R_{31} are conflicting rules.

The 3-valued algebra presented in [22] supports attribute-based policy rules, when a request applies to a XACML policy, the decision is one of Permit (Y), Deny (N) or NotApplicable (NA). The symbol notations in [22] are adopted in order to describe a policy.

Definition 5 (Policy): A policy P is a triple $\langle R_Y^P, R_N^P, R_{NA}^P \rangle \triangleq f_{RCA}(R_1(e), R_2(e), \dots, R_m(e))$, f_{RCA} denotes the combination operators of $(R_1(e), R_2(e), \dots, R_m(e))$ under RCA. R_Y^P, R_N^P, R_{NA}^P denotes respectively the set of requests permitted, requests denied, and not applicable by the policy P , where $R_\Sigma = R_Y^P \cup R_N^P \cup R_{NA}^P$, $R_Y^P \cap R_N^P = \emptyset$, $R_N^P \cap R_{NA}^P = \emptyset$, $R_Y^P \cap R_{NA}^P = \emptyset$.

With the above concepts, we present below the algebraic operations to support rule combination in a policy.

B. RULE COMBINATION OPERATOR

Policy combination could be carried out by combining all the policy rules. In this section, we extend FIA policy combination algebra system PCA with reducing operator δ , which is denoted as $PCA = \langle P, \&|_\delta, \oplus, \ominus, \odot, \neg, \bar{c} \rangle$, where P is the set of policies, each policy includes a set of rules, $\&$ is the operators of rule and policy combinations, δ is a rule reduce operator, \bar{c} is an condition constraint. “ \oplus , \ominus , \odot ” are binary operators, “ \neg ” is a unary operator. In order to build the policy combination framework, formal operational semantics are introduced as follows:

Definition 6 (“ \oplus ” Operator): $R_1(e) \oplus R_2(e)$ represents a new policy rule $R(e)$, which means that if a request matches either $R_1(e)$ or $R_2(e)$, then the request matches $R_1(e) \oplus R_2(e)$. The formal definition is:

$$R(e) = R_1(e) \oplus R_2(e) \Leftrightarrow R(e) = \{r | r \in R_1(e), \text{ or } r \in R_2(e)\}$$

For example, for P_1 , $R_{11}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\}$, for P_3 , $R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\}$, then the composition rule $R(Y) = R_{11}(Y) \oplus R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\}$.

Definition 7 (“ \neg ” Operator): $\neg R(e_1)$ represents a new policy rule $R(e)$, which means that if a request satisfies $R(e_1)$, then the request does not satisfy $\neg R(e_1)$. The formal formula as follows:

$$R(e) = \neg R(e_1) \Leftrightarrow R(e) = \{r | r \notin R(e_1)\}$$

For example, for P_2 , $R_{21}(N) = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 10)\}$, then the negation of $R_{21}(Y)$ is $R(Y) = \{(s, doc) \wedge (a, wr) \wedge (sen > 10)\}$.

Definition 8 (“ \odot ” Operator): $R_1(e_1) \odot R_2(e_2)$ represents a new rule $R(e)$, which means that if a request matches $R_1(e_1)$

and $R_2(e_2)$, then the request matches $R_1(e_1) \odot R_2(e_2)$. The formal formula as follows:

$$R(e) = R_1(e_1) \odot R_2(e_2) \Leftrightarrow \{r | r \in R_1(e_1) \text{ and } r \in R_2(e_2)\}$$

For example, for P_2 , $R_{22}(Y) = \{(s, (doc, nur)) \wedge (a, wr) \wedge (trul \geq 6)\}$, for P_3 , $R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\}$, then the composition rule $R(Y) = R_{22}(Y) \odot R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 6)\}$.

Definition 9 (“ \ominus ” Operator): $R_1(e_1) \ominus R_2(e_2)$ represents a new policy rule $R(e)$, which means that if a request matches $R_1(e_1)$, but does not match $R_2(e_2)$, then the request matches $R_1(e_1) \ominus R_2(e_2)$. The formal formula as follows:

$$R(e) = R_1(e_1) \ominus R_2(e_2) \Leftrightarrow \{r | r \in R_1(e_1), r \notin R_2(e_2)\}$$

For example, for P_2 , $R_{21}(N) = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 10)\}$, for P_3 , $R_{31}(Y) = \{(s, doc) \wedge (a, (re, wr)) \wedge (sen \geq 7)\}$, then the permitted part in combined rule $R(Y) = R_{31}(Y) \ominus R_{21}(N) = \{(s, doc) \wedge (a, wr) \wedge (sen > 10)\}$, the denied part in combined rule $R(N) = \{(s, doc) \wedge (a, wr) \wedge (sen < 7)\}$.

Definition 10 (Condition Constraint): \bar{c} is a condition constraint of a rule R , $R|_{\bar{c}}(e) = S|_{\bar{c}} \wedge A|_{\bar{c}} \wedge C|_{\bar{c}}$, where $S \wedge A \wedge C \subseteq R(e)$ and $S \wedge A \wedge C$ satisfy constraints \bar{c} .

Intuitively, the rule constraint is to put some restricts on a rule R , and to delete the parts that do not satisfy \bar{c} , the size of rule R is narrowed down. In a policy rule, one subject may have several attribute constraints. When imposing restrictions on subjects, we can view it as subject constraints in a policy rule. Rule combination based on subject constraints can be reduced to a condition attribute-based values computation. For example, for P_2 , $R_{22}(Y) = \{(s, (doc, nur)) \wedge (a, wr) \wedge (trul \geq 6)\}$, for P_3 , $R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\}$, when computing the combined rule of $R_{22}(Y)$ and $R_{32}(Y)$, we put the restrictions on the common subject domains, that is to say, \bar{c} is $s = doc$.

Form the above definitions, we can see that the operations \oplus , \ominus , \odot have the semantics of set-union, set-difference and set-intersection, which support resolving the most common issues of policy combination. However, these operators only support the limited attribute constraints. Thus, we define a reduce operator as follows.

Definition 11 (“ $\&|_\delta$ ” Operator): $P_1 \&|_\delta P_2$ represents a new policy, δ is used to reduce two value domains of the same attributes into one as the reducing results of two attribute values under “ δ ” operator.

The operator “ δ ” can be used between the sets, constants, numerical interval. For example, for the condition constraints $trul \geq 6$ in $R_{22}(Y)$, and $trul \geq 4$ in $R_{32}(Y)$, the reducing result of the attribute $trul$ in two rules under “ δ ” operator notated as $\delta_\cap(trul \geq 6, trul \geq 4) = (trul \geq 6)$.

Assuming that the value ranges of a condition attribute att in P_1, P_2, \dots, P_n are $(att, V_1(att)), (att, V_2(att)), \dots, (att, V_n(att))$ respectively, the intersection and union of these condition attribute values can be expressed formally as $\delta_\cup = \bigcup_{k=1}^n V_k(att)$ and $\delta_\cap = \bigcap_{k=1}^n V_k(att)$, \ominus expresses no any attribute is considered, so we can regard \ominus as $dom(c)$.

C. RULE COMBINING ALGORITHMS

XACML has four basic rule combining algorithms. They are “Deny Overrides (DO)”, “Permit Overrides (PO)”, “Deny-unless-Permit (DP)”, “Permit-unless-Deny(PD)”. Rule combining algorithms are used to resolve conflicts among applicable rules. For example, if a policy P contains two rules, R_1 permitting a doctor to access the medical data when his seniority is more than 8, R_2 allowing a doctor to access the medical data when his seniority is less than 10. For an access request r with seniority 9, if it applies to R_1 , the request is permitted, whereas if it applies to R_2 , the request is denied, thus R_1 and R_2 are conflicting rules in P . If P chooses PO principle to combine rules, then $P = R_1 \oplus R_2 = R_1$; if P chooses DO principle to combine rules, then $P = R_1 \oplus R_2 = R_2$.

The combination problem of multiple policies P_1, P_2, \dots, P_n can be expressed formally as $P_1 \& P_2 \& \dots \& P_n$. Taking $P_1 \& P_2$ for example, in last paper [39], we discussed multiple policy combination when all collaborative organizations adopt the same rule combining algorithms, that is diagonal parts in Table 3. In this paper, we further allow each collaborative organization to have different rule combining algorithms as shown by “*” parts in Table 3.

TABLE 3. Policy combination matrix.

$P_1 \& P_2$	PO	DO	DP	PD
PO	PO	*	*	*
DO	*	DO	*	*
DP	*	*	DP	*
PD	*	*	*	PD

Assume that $P = (R_1(Y), \dots, R_i(Y), R_1(N), \dots, R_j(N))$, $i + j = n$, where the set of permitted rules in P is notated as $R(Y) = (R_1(Y), \dots, R_i(Y))$, the set of denied rules in P is notated as $R(N) = (R_1(N), \dots, R_j(N))$.

The rules in a policy can be evaluated according to rule combining algorithms. In order to describe policy expressions, we present the semantics and the formal logical expressions of these algorithms as follows.

1) PERMIT OVERRIDE (PO)

The result is “permit” if any rule evaluates to “Permit”, the combined result is “Deny” if no rule evaluates to “Permit” and at least one policy evaluates to “Deny”. Otherwise, the result is “NotApplicable”. Its logical expression is

$$P_{PO} \Leftrightarrow \begin{cases} R_Y^P = \{r | r \in \bigoplus_{t=1}^i R_t(Y)\} \\ R_N^P = \{r | r \in \{(\bigoplus_{t=1}^j R_t(N)) \ominus (\bigoplus_{t=1}^i R_t(Y))\}\} \\ R_{NA}^P = \{r | r \notin R_Y^P, \text{ and } r \notin R_N^P\}. \end{cases}$$

In the same manner, we present the logical expressions of other three kinds of rule combining algorithms in a policy, which is shown in Table 4.

TABLE 4. Logical expression of rule combining algorithms (RCA).

Effect \ RCA	PO	DO	PD	DP
Permit	$\exists r \in R_Y^P$	$\neg \exists r \in R_N^P \wedge \exists r' \in R_Y^P$	$\exists r \in R_Y^P \vee R_{NA}^P$	$\exists r \in R_Y^P$
Deny	$\neg \exists r \in R_Y^P \wedge \exists r' \in R_N^P$	$\exists r \in R_N^P$	$\exists r \in R_N^P$	$\exists r \in R_N^P \vee R_{NA}^P$
NA	$\neg \exists r \in R_Y^P \wedge \neg \exists r \in R_N^P$	$\neg \exists r \in R_Y^P \wedge \neg \exists r \in R_N^P$	-	-

2) DENY OVERRIDE (DO)

Deny overrides is the opposite of permit overrides. The result is “Deny” if any rule is encountered that evaluates to “Deny”. The combined result is “Permit” if no rule evaluates to “Deny” as well as at least one rule evaluates to “Permit”. Otherwise, the result is “NotApplicable”.

$$P_{DO} \Leftrightarrow \begin{cases} R_Y^P = \{r|r \in \bigoplus_{i=1}^i R_i(Y) \ominus (\bigoplus_{i=1}^j R_i(N))\} \\ R_N^P = \{r|r \in \bigoplus_{i=1}^j R_i(N)\} \\ R_{NA}^P = \{r|r \notin R_Y^P, \text{ and } r \notin R_N^P\}. \end{cases}$$

3) DENY-UNLESS-PERMIT (DP)

The result is “Permit” if any policy evaluates to “Permit”, otherwise, the result is “Deny”. “NotApplicable” must never be the result.

$$P_{DP} \Leftrightarrow \begin{cases} R_Y^P = \{r|r \in \bigoplus_{i=1}^i R_i(Y)\} \\ R_N^P = \{r|r \in \bigoplus_{i=1}^i (\neg R_i(Y))\}. \end{cases}$$

4) PERMIT-UNLESS-DENY (PD)

The result is “Deny” if any policy evaluates to “Deny”, otherwise, the result is “Permit”. “NotApplicable” must never be the result.

$$P_{PD} \Leftrightarrow \begin{cases} R_Y^P = \{r|r \in \bigoplus_{i=1}^j (\neg R_i(N))\} \\ R_N^P = \{r|r \in \bigoplus_{i=1}^j R_i(N)\} \end{cases}$$

IV. POLICY OPERATORS

Assuming that each organization adopts attribute-based policy, the attributes mainly focus on subject, action and condition. All organizations have the condition attributes and all the condition attribute values are characterized by the same set.

Definition 12 (“&” Operator): $P_1 \& P_2$ represents a new access control policy P , which states that if a request satisfies the permitted rules of both P_1 and P_2 , then the request satisfies the permitted rules in $P_1 \& P_2$; if a request satisfies the denied rules of both P_1 and P_2 , then the request satisfies the denied rules in $P_1 \& P_2$, i.e., the request satisfy the denied rules in either P_1 or P_2 . The formal formula is as follows:

$$P = P_1 \& P_2 \Leftrightarrow \begin{cases} R_Y^P = R_Y^{P_1} \cap R_Y^{P_2} \\ R_N^P = (R_N^{P_1} \setminus R_Y^{P_2}) \cup (R_N^{P_2} \setminus R_Y^{P_1}). \\ R_{NA}^P = \text{others} \end{cases}$$

From the rule combining algorithms, we can see that each policy could formally describe the combined results of rules. Thus, each policy could be shown by using rules and rule combination algebraic operators. What is more, a policy is also used to validate whether the combined policy satisfy individual policy of collaborative organizations. The combined policy has the following properties:

Property 1: Let P_1, P_2, \dots, P_n be set of policies, then $P_1 \& P_2 \& \dots \& P_n$ is also a policy.

This result is intuitive, so we omit the related derivation here.

If policies P_1, P_2, \dots, P_n have same subject constraints, then we have the following properties.

Property 2: Let $P(x_1|\bar{c}, x_2|\bar{c}, \dots, x_n|\bar{c}) = (x_1|\bar{c} \cap x_2|\bar{c} \cap \dots \cap x_n|\bar{c})$ be a policy. If a request r satisfies multiple access rules R_1, R_2, \dots, R_n and a policy constraint \bar{c} , then we have $r \in P(R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c})$.

Proof \Rightarrow : for any request $r, r \in R_1, \dots, r \in R_n$ and r satisfies constraint \bar{c} , so $r \in R_1|\bar{c}, \dots, r \in R_n|\bar{c}$, thus $r \in P(R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c})$;

\Leftarrow : for any $r \in P(R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c}), r \in (R_1|\bar{c} \cap R_2|\bar{c} \cap \dots \cap R_n|\bar{c})$, then $r \in R_1|\bar{c}, \dots, r \in R_n|\bar{c}$, so r satisfies each of R_1, R_2, \dots, R_n and the constraint \bar{c} .

Property 3: Let $P(\neg x_1|\bar{c}, x_2|\bar{c}, \dots, x_n|\bar{c}) = (\neg x_1|\bar{c} \cap x_2|\bar{c} \cap \dots \cap x_n|\bar{c})$ be a policy. If a request r satisfies multiple access rules R_2, \dots, R_n and a policy constraint \bar{c} , but does not satisfy R_1 , then we have $r \in P(\neg R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c})$.

Proof \Rightarrow : for any request $r, r \notin R_1, r \in R_2, \dots, r \in R_n$ and r satisfies constraint \bar{c} , so $r \in (\neg R_1|\bar{c}), r \in R_2|\bar{c}, \dots, r \in R_n|\bar{c}$, thus $r \in P(\neg R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c})$;

\Leftarrow : for any $r \in P(\neg R_1|\bar{c}, R_2|\bar{c}, \dots, R_n|\bar{c}), r \in (\neg R_1|\bar{c} \cap R_2|\bar{c} \cap \dots \cap R_n|\bar{c})$, then $r \in (\neg R_1|\bar{c}), \dots, r \in R_n|\bar{c}$, so r satisfies each of $\neg R_1, R_2, \dots, R_n$ and the constraint \bar{c} .

Any two policy combination results are always obtained by combining the reduced rules under the appropriate rule combining algorithm chosen in XACML.

Theorem 13: Let P_1 and P_2 be two policies, P_1 chooses RCA_1 and P_2 chooses RCA_2 as their rule combining algorithms respectively. There exists a rule combining algorithm RCA such that $RCA_1.P_1 \& RCA_2.P_2 \approx RCA.(P_1 \& P_2)$, where $RCA_1, RCA_2, RCA \in \{PO, DO, PD, DP\}$.

The rule combining algorithms RCA_1 and RCA_2 are included in the set $\{PO, DO, PD, DP\}$. There are two cases:

Proof: (1) If both P_1 and P_2 use the same rule combining algorithms, the conclusion is obvious, that is, $RCA_1 = RCA_2$, then $RCA = RCA_1 = RCA_2$ such that $RCA_1.P_1 \& RCA_2.P_2 = RCA.(P_1 \& P_2)$. For example, we assume that $RCA_1 = PO, RCA_2 = PO$, from the logical expression of a policy

with different rule combining algorithms, we can see that $R_Y^{P_1} = \bigoplus_{t=1}^i R_t(Y)$, $R_N^{P_1} = \{(\bigoplus_{t=1}^j R_t(N)) \ominus (\bigoplus_{t=1}^i R_t(Y))\}$, $R_Y^{P_2} = \bigoplus_{t=1}^i R_t(Y)$, $R_N^{P_2} = \{(\bigoplus_{t=1}^j R_t(N)) \ominus (\bigoplus_{t=1}^i R_t(Y))\}$. Thus, $RCA = PO$. Similarly, other results could be conducted, that is, when $RCA_1 = CA_2 \in \{DO, PD, DP\}$, $RCA = RCA_1 = RCA_2 \in \{DO, PD, DP\}$.

(2) If both P_1 and P_2 use different rule combining algorithms, that is, $RCA_1 \neq RCA_2$, then we try to find an appropriate RCA such that $RCA_1.P_1 \& RCA_2.P_2 \approx RCA.(P_1 \& P_2)$. Assume that $RCA_1 = PO$, $RCA_2 = DO$, from the logical expression of a policy with different rule combining algorithms, we can see that $R_Y^{P_1} = \bigoplus_{t=1}^i R_{1t}(Y)$, $R_N^{P_1} = (\bigoplus_{t=1}^j R_{1t}(N)) \ominus (\bigoplus_{t=1}^i R_{1t}(Y))$, $R_Y^{P_2} = \bigoplus_{t=1}^i R_{2t}(Y) \ominus (\bigoplus_{t=1}^j R_{2t}(N))$, $R_N^{P_2} = \bigoplus_{t=1}^j R_{2t}(N)$, $R_Y^{P_1} \cap R_Y^{P_2} = \bigoplus_{t=1}^i R_{1t}(Y) \cap \bigoplus_{t=1}^i R_{2t}(Y) \ominus (\bigoplus_{t=1}^j R_{2t}(N))$. When $P_1 \& P_2$ chooses DO as its rule combining algorithm, $R_Y^{P_1 \& P_2} = \bigoplus_{t=1}^i (R_{1t}(Y) \cap R_{2t}(Y)) \ominus (\bigoplus_{t=1}^j (R_{1t}(N) \cap R_{2t}(N))) \subseteq R_Y^{P_1} \cap R_Y^{P_2}$. Thus, we can choose the more appropriate algorithm $RCA = DO$ such that $PO.P_1 \& DO.P_2 \approx DO.P_1 \& P_2$. Similarly, other results could be conducted. \square

V. POLICY COMBINING APPROACH

In this section, we present our approach to combining multiple policies. The major goal of this approach is to automatically generate a new global policy on the strength of a set of attribute-based access control policies, which are specified by multiple collaborative organizations, respectively.

Since our approach can ensure to comply to each policy from different organizations, the generated policy would be acceptable for each collaborative organization. The overview of our approach is shown in Fig. 3 and it mainly consists of the following three steps:

- Step 1 (Policy rule specification): all the collaborative organizations should adopt an unified scheme and individually specify their local policies.
- Step 2 (Rule classification and reduction): classifying all the rules from different policies according to the same attribute constraints, and then reducing the rules in each class into a new one as a rule in the global policy.
- Step 3 (New policy generation): choosing an appropriate rule combining algorithm to combine the rules in a global policy.

A. THE PROCEDURE OF POLICY COMBINATION

Our automated multiple policy combination starts with receiving a set of access control policies P_1, P_2, \dots, P_n , and end up with returning a new global policy P , rather than with returning a policy decision. The detailed description of Step 1 and Step 2 is presented in a pseudo-code algorithm for computing $P = P_1 \& P_2 \& \dots \& P_n$, as shown in Algorithm 1.

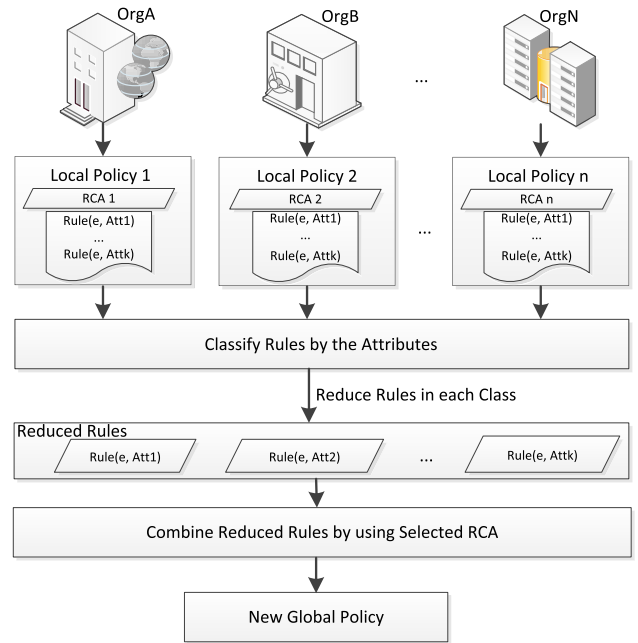


FIGURE 3. Approach overview for multiple policies combination.

The detailed description of Step 3 is presented in the other pseudo-code algorithm for choosing an appropriate rule combining algorithm in XACML shown in Algorithm 2. Next, we present the details step by step, and take HIS as an example to illustrate the above algorithms. The detailed procedure of policy combination is as follows:

Step 1 (Policy Rule Specification): The first step is to present all the rule expressions in policy of each collaborative organization, which should adopt the unified specification for the shared data to individually specify their local policy requirements through the algebraic operators we presented in Section 3. The rules with effect in a local policy can be defined by attribute-based authorization rules, notated as $R(e) = \{(s, V(s)) \wedge (a, V(a)) \wedge (att, V(att))\}$. This step corresponds to the top layer of our framework as shown in Fig 3.

Step 2 (Rule Classification and Integration): We first classify all the rules according to the some condition constraints, i.e., the rules included in each class have the same condition constraints. Then we compare the effects of the rules in a class, there are three cases: (1) all the rules have the same “Permit” effects, (2) all the rules have the same “Deny” effects, (3) some rules have the “Permit” effects and others have the “Deny” effects. Later, we transfer the rules with different effects to the rules with the same effects, and adopt a mapping operator in our presented algebraic system to reduce the rules into a set of new rules in a global policy. This step is formalized as computing $P = P_1 \& P_2 \& \dots \& P_n$ (Algorithm 1). We compute the values of subject attribute and action attribute in the rules with the same condition constraints and the same effects. The detailed procedure of this step is as follows.

Algorithm 1 Algorithm for Computing $P_1 \& P_2 \& \dots \& P_n$ **Require:** n policies P_1, P_2, \dots, P_n **Ensure:** $P = P_1 \& P_2 \& \dots \& P_n$

```

1: For any rule  $R_i \in P_i$  do
2: Begin
3: {Noted  $R_i$  as visited;
4: if  $R_i.att$  defined into  $P_j$  ( $1 \leq j \neq i \leq n$ ) then
5:   let  $R_k.att \in P_j$  be the rule such that  $R_k.att = R_i.att$ 
6:   Noted  $R_k.att$  as visited
   /*Next step to compute the value of  $R.att$  in  $P$ */
7:   Do case
8:   case  $R_i.e = Y$ 
9:     let  $S = S_i \cap S_k$  and  $A = A_i \cap A_{kj}$  be non-empty
10:    if  $R_{kj}.e = Y$  then
11:      Add $R(e) = S \wedge A \wedge (att, V_i(att) \cap V_k(att))$  to  $P(Y)$ 
12:    else
13:      Add $R(e) = S \wedge A \wedge (att, V_i(att) \cap (\neg V_k(att)))$  to  $P(Y)$ 
14:    endif
15:   case  $R_i.e = N$ 
16:   if  $R_k.e_k = Y$  then
17:     Add $R(e) = S \wedge A \wedge (att, V_k(att) \cap (\neg V_i(att)))$  to  $P(Y)$ 
18:   else
19:     Add  $R_i(e) = S_i \wedge A_i \wedge (att, V_i(att))$  to  $R(N)$ 
20:     Add  $R_k(e) = S_k \wedge A_k \wedge (att, V_k(att))$  to  $R(N)$ 
21:   endif
22:   endcase
23: else
24:   /* $R_i.att$  not defined into  $P_2$ */
25:   if  $R_i.e = N$  then
26:     Add  $R_i(e) = S_i \wedge A_i \wedge (att, V_i(att))$  to  $R(N)$ 
27:   end if;
28: }
29: endfor;
30: return  $P = \langle R(Y), R(N) \rangle$ ;
31: END

```

For any condition constraint att in a policy P_i , we first find all the rules with the constraint att (Lines 2-6 in Algorithm 1). The combination results of the rules contains the following four cases according to the effects of rules.

Case (1): The effects of all the rules included in a class are ‘‘Permit’’, these rules have the condition constraints $(att, V_k(att))$ ($1 \leq k \leq n$). We compute the values of subject attribute and the action attributes. For example, for a subject attribute s , we find all the attribute domains in each rule. Assume $((s, V_1(s)), (a, V_1(a))) \in R_1, ((s, V_2(s)), (a, V_2(a))) \in R_2, \dots, ((s, V_n(s)), (a, V_n(a))) \in R_n$, we compute $\bigcap_{k=1}^n V_k(s)$ and $\bigcap_{k=1}^n V_k(a)$ as the values of the subject s and the action a separately. For each $(att, V_k(att)) \in R_k(Y)$, we compute $(att, \bigcap_{k=1}^n V_k(att))$ as a condition constraint of the subject s in a combined rule R , which is shown in Lines 7-10. Then the combination results in a class with the condition constraint $c = att$ are added to $R(Y)$, so we have

$$R(Y)|_{c=att} = \{(s, \bigcap_{k=1}^n V_k(s)) \wedge (a, \bigcap_{k=1}^n V_k(a)) \wedge (att, \bigcap_{k=1}^n V_k(att))\}.$$

Case (2): The effects of some rules included in a class are ‘‘Permit’’, and the effects of other rules are ‘‘Deny’’.

Without loss of generality, we can assume that $(att, V_k(att)) \in P_k(Y), k = 1, \dots, n-1$ and $(att, V_n(att)) \in R_n(N)$, if the subject attribute $(s, \bigcap_{k=1}^n V_k(s))$ and the action attribute $(a, \bigcap_{k=1}^{n-1} V_k(a))$ are existing, we compute $(att, \bigcap_{k=1}^{n-1} V_k(att) \cap \neg V_n(att))$ as the condition constraints in the reduced rule $R(Y)$, as shown in Lines 11-17. So we have

$$R(Y)|_{c=att} = \{(s, \bigcap_{k=1}^n V_k(s)) \wedge (a, \bigcap_{k=1}^n V_k(a)) \wedge (att, \bigcap_{k=1}^{n-1} V_k(att) \cap (\neg V_n(att)))\}.$$

Case (3): The effects of all the rules with attribute constraint $(att, V_i(att))$ ($1 \leq i \leq n$) in a class are ‘‘Deny’’. All rules should be reduced to the new rule set in the global policy as shown in Lines 18-25.

Case (4): For the condition constraint $(att, V(att))$ in a policy P_i , we can not find the same condition constraints in other policies. We add the rule with the constraint $(att, V(att))$ into the rule set in the global policy (Lines 2-6 in Algorithm 1).

When traversing all the condition constraints defined in the rules from each organization, we obtain the reduced rules concluded in the global policy.

Algorithm 2 Choosing RCA in P **Require:** A set of RCAs $\{PO, DO, PD, DP\}$ **Ensure:** Rule combining algorithm RCA in P

```

1: For any rule combining  $RCA_1$  in  $P_1$  and  $RCA_2$  in  $P_2$ ;
2: if  $RCA_1 \neq RCA_2$  then
3:   if  $RCA_1 = PO$  and  $RCA_2 \in \{DO, PD, DP\}$  then
4:      $RCA = RCA_2$ ;
5:   if  $RCA_1 = PD$  and  $RCA_2 \in \{DO, DP\}$  then
6:      $RCA = DO$ ;
7:   if  $RCA_1 = DO$  and  $RCA_2 = DP$  then
8:     Error;
9: else
10:    $RCA = RCA_1 = RCA_2$ ;
11: end if
12: endFor
13: return Rule combining algorithm  $RCA$  in  $P$ .

```

Step 3 (New Policy Generation): Choosing the appropriate rule combination algorithm RCA to address combination issues of the new rule set as shown in Algorithm 2. RCA is chosen according to the rule combining algorithms used in each organization. Each organization can choose four kinds of rule combining algorithms, so there are 16 cases of algorithm choices for any two policies P_1 and P_2 . If two policies have the same rule combining algorithms, the global policy has the same algorithm as shown in Lines 9-11. If two policies have different rule combining algorithms RCA_1 and RCA_2 , RCA in the global policy is the same to either RCA_1 or RCA_2 , shown in Lines 2-8. There is a special case, if one policy uses rule combining algorithm DO , and the other uses rule combining algorithm DP , the two policies cannot be combined due to the conflict logics. What is more, the chosen RCA can be identified by existing rule combining algorithms in XACML policy specification. This step is processed by the bottom component in our framework. Here, we only present choosing

algorithm of *RCA* for two policies, for multiple policies, *RCA* can be obtained through multiple iterations.

B. CASE STUDY

In HIS example, the global policy of OrgA, OrgB, OrgC and OrgD is obtained by combining each local policy from these organizations. Based on our presented fine-grained policy combination algorithm, we first formally express all the rules in each local policy, that is Step 1 as follows.

Step 1: Specifying all the rules in each policy P_1 , P_2 , P_3 and P_4 . In policy P_1 , there is one rule R_{11} . Its effect is permit, thus, R_{11} can be formally expressed as

$$R_{11}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\}.$$

In policy P_2 , there are two rules R_{21} and R_{22} . The effect of rule R_{21} is deny, and the effect of rule R_{22} is permit, thus, R_{21} and R_{22} can be formally expressed as

$$R_{21}(N) = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 10)\};$$

$$R_{22}(Y) = \{(s, doc, nur) \wedge (a, wr) \wedge (trul \geq 6)\}.$$

Analogously, in policy P_3 , R_{31} , R_{32} and R_{33} can be formally expressed as

$$R_{31}(Y) = \{(s, doc) \wedge (a, (re, wr)) \wedge (sen \geq 7)\};$$

$$R_{32}(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\};$$

$$R_{33}(N) = \{(s, nur) \wedge (a, re) \wedge (secl \leq 6)\}.$$

In policy P_4 , R_{41} and R_{42} can be formally expressed as

$$R_{41}(N) = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 5)\}.$$

$$R_{42}(Y) = \{(s, (doc, nur) \wedge (a, (re, wr)) \wedge (trul \geq 3)\}.$$

Step 2: Reducing all the rules into a new rule set in the global policy.

For the above rule expressions, we can see that the rules R_{11} in P_1 , R_{22} in P_2 , R_{32} in P_3 and R_{42} in P_4 have the common trust level constraint $trul$ and the same effect *permit*, so R_{11} , R_{22} , R_{32} and R_{42} are compatible rules. Thus, δ -operator should be the interaction of the attribute values of $V_{11}(trul)$, $V_{22}(trul)$, $V_{32}(trul)$ and $V_{42}(trul)$. These four rules R_{11} , R_{22} , R_{32} and R_{42} could be reduced into $R_1(Y)$ as a rule in the global policy, that is

$$R_1(Y) = R_{11} \& R_{22} \& R_{32} \& R_{42} = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\} \&_{\delta} \{(s, (doc, nur) \wedge (a, wr) \wedge (trul \geq 6)\} \&_{\delta} \{(s, doc) \wedge (a, wr) \wedge (trul \geq 4)\} \&_{\delta} \{(s, (doc, nur) \wedge (a, (re, wr)) \wedge (trul \geq 3)\} = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\}.$$

For the condition attribute sen , there is no constraint sen in P_1 , the rules R_{21} in P_2 , R_{31} in P_3 and R_{41} in P_4 have the common seniority attribute constraint sen , but they have different effects, so R_{31} are conflicting with R_{21} and R_{41} . These rules could be reduced into $R_2(Y)$ (or $R_2(N)$) as a rule included in the global policy. We consider $R_2(Y)$ here. Thus, δ -operator should be the subtraction of the attribute values of $V_{31}(sen)$ and $V_{21}(sen)$ and $V_{41}(sen)$, that is

$$R_2(Y) = \{(s, doc) \wedge (a, (re, wr)) \wedge (sen \geq 7)\} \&_{\delta} \{(s, doc) \wedge (a, wr) \wedge (sen \leq 5)\} \&_{\delta} \{(s, doc) \wedge (a, wr) \wedge (sen \leq 10)\} = \{(s, doc) \wedge (a, wr) \wedge (sen > 10)\}.$$

There is a special case that for the attribute constraint $secl$ in rule R_{33} with the effect “Deny”, we cannot find the same constraint in other policies. In this case, R_{33} could be reduced into $R_3(N)$. That is,

$$R_3(N) = R_{33}(N) = \{(s, nur) \wedge (a, re) \wedge (secl \leq 6)\}.$$

```

Policy2.xml
<Policy PolicyId="Policy1" RuleCombiningAlgId=".....">
  <Rule RuleId="Rule11" Effect="deny">
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="...function:rfc822Name-match">
            <AttributeValue DataType="...#string">
              Doctor
            </AttributeValue>
            <SubjectAttributeDesignator AttributeId="...:subject-id"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
    </Target>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="...:anyURI-equal">
          <AttributeValue DataType="...#anyURI">
            http://datashare.org/orgB/data/
          </AttributeValue>
          <ResourceAttributeDesignator AttributeId="resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <Action>
        <ActionMatch MatchId="...:string-equal">
          <AttributeValue DataType="...#string">
            Write
          </AttributeValue>
          <ActionAttributeDesignator AttributeId="...:action-id"/>
        </ActionMatch>
      </Action>
    </Actions>
  </Rule>
  <Rule RuleId="Rule12" Effect="Permit"/>
    <Target>
      <Subject AttributeValue = " Doctor " and " Nurse" >
        <Resource AttributeValue = "http://datashare.org/orgB/data/" >
          <Action AttributeValue = " Write " >
            </Target>
            <Condition FunctionId="...: integer-greater-than-or-equal ">
              <Apply FunctionId="...:integer-one-and-only">
                <EnvironmentAttributeDesignator AttributeId= trul />
              </Apply>
              <AttributeValue DataType="... #integer">6</AttributeValue>
            </Condition>
          </Rule>
        </Policy>

```

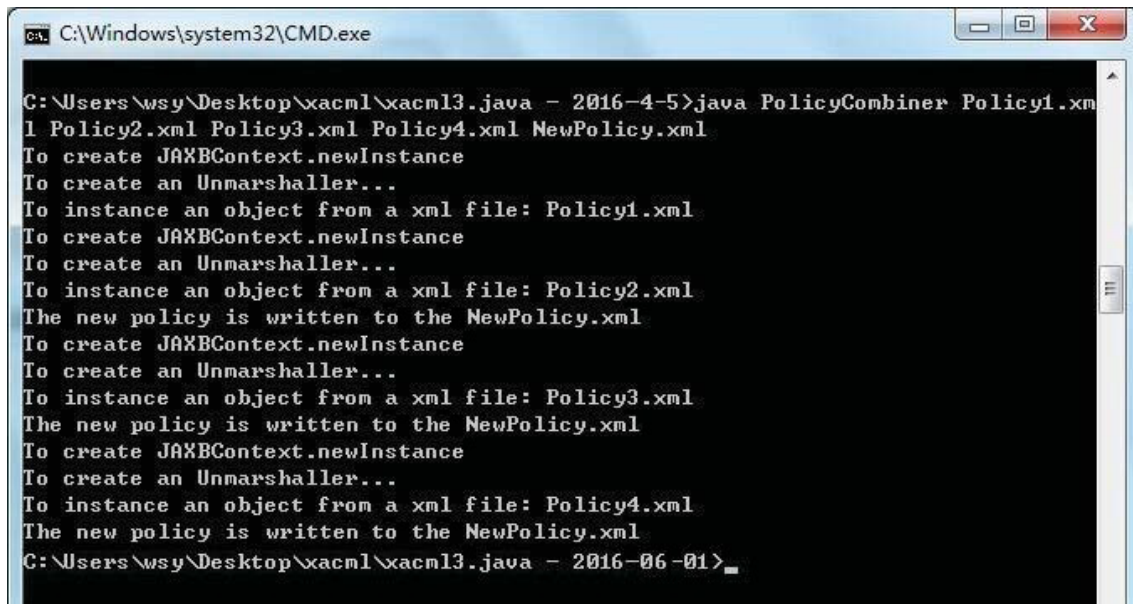
FIGURE 4. Policy P_2 .

Thus, the global policy $P = \{R_1(Y), R_2(Y), R_3(N)\}$.

Step 3: Choosing the optimum algorithm to combine the reduced rules.

For P_1 , P_2 , P_3 and P_4 , assume that P_1 , P_2 and P_4 are generated by using PO rule combining algorithm to combine their own rules, and P_3 is generated by using DO rule combining algorithm to combine R_{31} , R_{32} and R_{33} . That is, $RCA_1 = PO$, $RCA_2 = PO$, $RCA_3 = DO$ and $RCA_4 = PO$. From the aspect of requests, a policy can be the set of all the permitted requests, all the denied requests and all the NotApplicable requests. Thus, for each policy P_i ($1 \leq i \leq 4$), we have $P_i = \langle R_Y^{P_i}, R_N^{P_i}, R_{NA}^{P_i} \rangle$, we compute the intersection of subjects from P_1 , P_2 , P_3 , P_4 .

In HIS Example, policy $P_1 = \langle R_Y^{P_1}, R_N^{P_1}, R_{NA}^{P_1} \rangle$, in which $R_Y^{P_1} = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\}$. Policy $P_2 = \langle R_Y^{P_2}, R_N^{P_2}, R_{NA}^{P_2} \rangle$, where $R_Y^{P_2} = \{(s, (doc, nur)) \wedge (a, wr) \wedge (trul \geq 6)\}$, $R_N^{P_2} = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 10)\}$. Policy $P_3 = \langle R_Y^{P_3}, R_N^{P_3}, R_{NA}^{P_3} \rangle$, where $R_Y^{P_3} = \{(s, doc) \wedge (a, wr) \wedge ((sen \geq 7) \wedge (trul \geq 4))\}$, $R_N^{P_3} = \{(s, nur) \wedge (a, re) \wedge (secl \leq 6)\}$. Policy $P_4 = \langle R_Y^{P_4}, R_N^{P_4}, R_{NA}^{P_4} \rangle$, where $R_Y^{P_4} = \{(s, (doc, nur) \wedge (a, (re, wr)) \wedge (trul \geq 3)\}$, $R_N^{P_4} = \{(s, doc) \wedge (a, wr) \wedge (sen \leq 5)\} \ominus ((s, (doc, nur) \wedge (a, (re, wr)) \wedge (trul \geq 3)))$, the common subject is doctor in P_1 and P_2 .



```

C:\Windows\system32\CMD.exe

C:\Users\wsy\Desktop\xacml\xacml3.java - 2016-4-5>java PolicyCombiner Policy1.xml
1 Policy2.xml Policy3.xml Policy4.xml NewPolicy.xml
To create JAXBContext.newInstance
To create an Unmarshaller...
To instance an object from a xml file: Policy1.xml
To create JAXBContext.newInstance
To create an Unmarshaller...
To instance an object from a xml file: Policy2.xml
The new policy is written to the NewPolicy.xml
To create JAXBContext.newInstance
To create an Unmarshaller...
To instance an object from a xml file: Policy3.xml
The new policy is written to the NewPolicy.xml
To create JAXBContext.newInstance
To create an Unmarshaller...
To instance an object from a xml file: Policy4.xml
The new policy is written to the NewPolicy.xml
C:\Users\wsy\Desktop\xacml\xacml3.java - 2016-06-01>

```

FIGURE 5. Policy Combiner.

DO is chosen as an algorithm for combining new rules in a global policy.

From the above discussion, the policy combination result is a new policy P ,

$$R_Y^P = R_1(Y) \oplus R_2(Y) \ominus R_3(N),$$

$$R_N^P = R_3(N)$$

where

$$R_1(Y) = \{(s, doc) \wedge (a, wr) \wedge (trul \geq 8)\};$$

$$R_2(Y) = \{(s, doc) \wedge (a, wr) \wedge (sen > 10)\};$$

$$R_3(N) = \{(s, nur) \wedge (a, re) \wedge (secl \leq 6)\}$$

That is to say, doctors are allowed to write medical data if their seniority is more than 10 years, and their trust level is greater than or equal to 8. However, any nurses are not allowed to write data when their security level is less than or equal to 6.

C. COMPLEXITY ANALYSIS

The generation of a global policy mainly involves two parts, one is to compute $P = P_1 \& P_2 \& \dots \& P_n$ (Algorithm 1) and the other is to choose rule combining algorithm used in P (Algorithm 2).

We first consider the policy combination algorithm (Algorithm 1), where n is the number of policies. Let N_r denote the number of rules in all the policies, and N_{att} denote the number of attribute constraints in all the policies. Let N_{ri} denote the number of the rules in a policy P_i ($1 \leq i \leq n$) and the number of all the rules is $N_r = \sum_{i=1}^n N_{ri}$, so Lines 1-3 can be executed in time $O(N_{att} \cdot N_{r1})$. Let N_{ci} denote the number of the attribute constraints on the rules in a policy P_i ($1 \leq i \leq n$), so for each P_j ($2 \leq j \leq n$), Lines 4-6 can be executed in time $O(N_{att} \cdot (\sum_{j=2}^n N_{rj}))$. Thus, Lines 1-6 can be executed in time $N_r \cdot N_{att}$. Let N_{si} denote the number of all the subjects in a policy P_i ($1 \leq i \leq n$), according to

idea of hash-based approach to computing intersection set, time complexity of computing intersection set of the subjects is linear, so in Line 8, $S = S_i \cap S_k$ can be executed in time $O(\sum_{i=1}^n N_{si}) \triangleq O(N_s)$, where N_s denotes the number of common subjects defined in all policies. Similarly, $A = A_i \cap A_{kj}$ can be executed in time $O(N_a)$, where N_a denotes the number of common actions defined in all policies. So Lines 7-27 can be executed in time $O(N_s) \cdot O(N_a)$. One policy has only one rule combining algorithm, so the executed time of RCA choosing algorithm in a global policy (i.e., Algorithm 2) is $O(1)$. Hence, the overall complexity of policy combining algorithm procedure is $O(N_r \cdot N_{att} \cdot N_s \cdot N_a)$. Complexity results show that our policy combination approach is efficient.

VI. IMPLEMENTATION

To demonstrate the concept, we implemented the above algorithms in Java as a simple policy combination tool. We also carried out experiments to evaluate the performance of generating a common policy in terms of the number of policies, as well as the number of rules in each policy.

A. POLICY COMBINATION TOOL

We adopted the built-in Java Architecture for XML Binding (JAXB) [37] tool (i.e. xjc) of Oracle JDK 7.0_79 to generate a set of xacml java code from XACML v3.0 schema [38], i.e. xacml-core-v3-schema-wd-17.xsd. Then we used the generated xacml java code to implement the policy combination tool. The current implementation can take multiple policies in XACML as inputs, e.g. Policy1.xml is shown in Fig. 1, it states that doctors are allowed to write medical data if their seniority is greater than or equal to 8.

Policy2.xml is shown in Fig. 4, which states that doctors and nurses are allowed to write medical data if their trust level

```

NewPolicy.xml
<Policy PolicyId="Policy1" RuleCombiningAlgId="deny-override">
  <Rule RuleId="Rule1" Effect="permit">
    <Target>
      .....
      <Subject AttributeValue = " Doctor " >
        <Resource AttributeValue = "http://datashare.org/orgA/data/" >
        <Resource AttributeValue = "http://datashare.org/orgB/data/" >
        <Resource AttributeValue = "http://datashare.org/orgC/data/" >
        <Resource AttributeValue = "http://datashare.org/orgD/data/" >
        <Action AttributeValue = " Write " >
      .....
    </Target>
    <Condition FunctionId="...: integer-greater-than-or-equal ">
      <Apply FunctionId="...:integer-one-and-only">
        <EnvironmentAttributeDesignator AttributeId= doctor-trul />
      </Apply>
      <AttributeValue DataType="... #integer">8</AttributeValue>
    </Condition>
  </Rule>
  <Rule RuleId="Rulec2" Effect="Permit"/>
  <Target>
    .....
    <Subject AttributeValue = " Doctor " >
      <Resource AttributeValue = "http://datashare.org/orgA/data/" >
      <Resource AttributeValue = "http://datashare.org/orgB/data/" >
      <Resource AttributeValue = "http://datashare.org/orgC/data/" >
      <Resource AttributeValue = "http://datashare.org/orgD/data/" >
      <Action AttributeValue = " Write " >
    .....
  </Target>
  <Condition FunctionId="...: integer-greater-than ">
    <Apply FunctionId="...:integer-one-and-only">
      <EnvironmentAttributeDesignator AttributeId= doctor-sen />
    </Apply>
    <AttributeValue DataType="... #integer">10</AttributeValue>
  </Condition>
</Rule>
<Rule RuleId="Rulec3" Effect="deny"/>
  <Target>
    .....
    <Subject AttributeValue = " Nurse " >
      <Resource AttributeValue = "http://datashare.org/orgA/data/" >
      <Resource AttributeValue = "http://datashare.org/orgB/data/" >
      <Resource AttributeValue = "http://datashare.org/orgC/data/" >
      <Resource AttributeValue = "http://datashare.org/orgD/data/" >
      <Action AttributeValue = " Read " >
    .....
  </Target>
  <Condition FunctionId="...: integer-less-than-or-equal ">
    <Apply FunctionId="...:integer-one-and-only">
      <EnvironmentAttributeDesignator AttributeId= nurse-secl />
    </Apply>
    <AttributeValue DataType="... #integer">6</AttributeValue>
  </Condition>
</Rule>
</Policy>
  
```

FIGURE 6. Combined Policy of P_1 , P_2 , P_3 and P_4 .

is greater than or equal to 6. However, any doctors are not allowed to write medical data if their seniority is less than or equal to 10. For space limitation, we omit Policy3.xml and Policy4.xml. Policy Combiner can automatically combine the four policies into a new policy, i.e. NewPolicy.xml as shown in Fig. 6, which shows the combined result is that doctors are allowed to write medical data if their seniority is greater than 10 and their trust level is greater than or equal to 8. However, any nurses are not allowed to write data when their security level less than or equal to 6. Fig. 5 shows the process of the automatic process of combining four policies Policy1, Policy2, Policy3 and Policy4 into one NewPolicy. Optimizing this tool to support more rule combining algorithms for combining multiple policies is one of our future works.

B. POLICY GENERATION PERFORMANCE

In order to evaluate our policy combination tool and the performance of our algorithms, we measured the average processing time of generating a local policy with different number of rules, as well as generating a new global policy with different number of policies. All the experiments were carried out on a Pentium(R) Dual-Core CPU 3.20GHZ PC with 4G RAM.

Each local policy is generated by the different rule combining algorithms PO, DO, PD, DP. Fig.7 shows the average

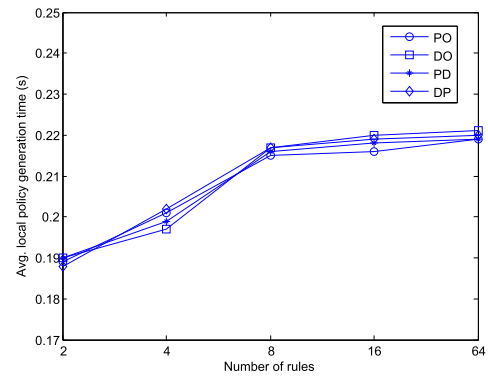


FIGURE 7. Local policy generation.

processing time of generating one local policy by combining the random numbers of policy rules, which are specified by using the attribute constraints and some logical expressions. We can observe in Fig.7 that different rule combining algorithms have little affects on the time of local policy generation. What is more, as the number of rules are increased, the curves of processing time maintain stable, which show each local policy can deal with a random number of rules and has strong extensibility.

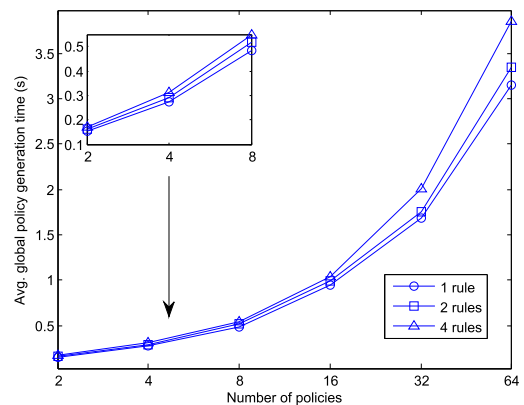


FIGURE 8. Global policy generation.

Fig.8 shows the average processing time of generating one global policy from multiple local policies. The number of policies rose to 64, each local policy is generated by some rules range among one, two and four. The test results reported in Fig.8 show that our policy combination approach can handle a large number of attribute constraints in each collaborative policy.

VII. CONCLUSIONS

In this paper, in order to combine XACML policies for data sharing among multiple organizations, we proposed a rule reducing approach and developed a proof-of-concept implementation of the automated policy combination. The rules with different condition attribute constraints have different effects. For the rules with the common attribute constraints, we compared the attribute values and the effects of

these rules. Under this comparison, rule combination was reduced to the attribute-based combination. The final reduced rule set was obtained after the attribute constraints traversed through all attributes involved in the rules. Then, the reduced rules were combined into a new global policy by choosing the appropriate rule combining algorithm in XACML. We considered the scenarios that organizations were defined by four kinds of rule combining algorithms. Our approach maintained various policies compliance in both of syntax level and semantic level, and also supported a number of attribute constraints in each local policy.

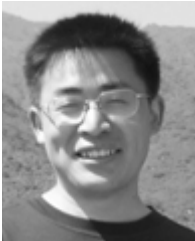
Our future work will focus on comparing the effectiveness and extensibility of existing policy combination approaches, and find the most efficient approach with low cost to combine policies of cross-organization collaborations.

REFERENCES

- [1] H. Tong, J. Cao, S. Zhang, and M. Li, "A distributed algorithm for Web service composition based on service agent model," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2008–2021, Dec. 2011.
- [2] B. Alhaqhani and C. Fidge, "Access control requirements for processing electronic health records," in *Proc. Bus. Process Manage. Workshops*, 2008, pp. 371–382.
- [3] C. Clifton et al., "Privacy-preserving data integration and sharing," in *Proc. 9th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, 2004, pp. 19–26.
- [4] Y.-J. Hu and J.-J. Yang, "A semantic privacy-preserving model for data sharing and integration," in *Proc. Int. Conf. Web Intell., Mining Semantics*, 2011, Art. no. 9.
- [5] OCareCloudS. (2014). *OCareCloudS—Overview Projects—iMinds*. [Online]. Available: <http://www.iminds.be/en/research/overview-projects/p/detail/ocareclouds-2>
- [6] D. D. He and J. Yang, "Authorization control in collaborative healthcare systems," *J. Theoretical Appl. Electron. Commerce Res.*, vol. 4, no. 2, pp. 88–109, 2009.
- [7] M. Decat, D. Van Landuyt, B. Lagaisse, and W. Joosen, "On the need for federated authorization in cross-organizational e-health platforms," in *Proc. 8th Int. Conf. Health Informat.*, vol. 8, pp. 540–546, Jan. 2015.
- [8] S. S. Yau and Z. Chen, "Security policy integration and conflict reconciliation for collaborations among organizations in ubiquitous computing environments," in *Proc. 5th Int. Conf. UIC*, 2008, pp. 3–19.
- [9] B. Carminati, E. Ferrari, and P. C. K. Hung, "Security conscious Web service composition," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Sep. 2006, pp. 489–496.
- [10] F. Liang, H. Guo, S. Yi, and S. Ma, "A multiple-policy supported attribute-based access control architecture within large-scale device collaboration systems," *J. Netw.*, vol. 7, no. 3, pp. 524–531, 2012.
- [11] L. Iliadis, M. Papazoglou, and K. Pohl, Eds. "Resolving policy conflicts—Integrating policies from multiple authors," in *Advanced Information Systems Engineering Workshops (Lecture Notes in Business Information Processing)*, vol. 178. Cham, Switzerland: Springer, 2014, pp. 310–321.
- [12] S. Hada and M. Kudo. *XML Access Control Language: Provisional Authorization for XML Documents*. [Online]. Available: <http://xml.coverpages.org/xaclspec200102.html>
- [13] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter, (2003). *Enterprise Privacy Authorization Language (EPAL 1.2), Submission to W3C*. [Online]. Available: <http://www.w3.org/2003/p3p-ws/pp/ibm3.html>
- [14] OASIS XACML TC. (Jan. 2013). *eXtensible Access Control Markup Language (XACML) Version 3.0*. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
- [15] X. Zhang, Y. Li, and D. Nalla "An attribute-based access matrix model," in *Proc. ACM Symp. Appl. Comput.*, 2005, pp. 359–363.
- [16] D. F. Ferraioli, S. Gavrila, V. Hu, and D. R. Kuhn, "Composing and combining policies under the policy machine," in *Proc. 10th ACM Symp. Access Control Models Technol. (SACMAT)*, New York, NY, USA, 2005, pp. 11–20.
- [17] M. Backes, M. Dürrmuth, and R. Steinwandt, "An algebra for composing enterprise privacy policies," in *Proc. 9th Eur. Symp. Res. Comput. Secur. (ESORICS)*, vol. 3193. 2004, pp. 33–52.
- [18] L. Y. Wang, D. Wijesekera, and S. Jajodia, "A logic-based framework for attribute based access control," in *Proc. ACM Workshop Formal Methods Secur. Eng. (FMSE)*, New York, NY, USA, 2004, pp. 45–55.
- [19] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies," in *Proc. 27th ICSE*, 2005, pp. 196–205.
- [20] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone, "Analysis of XACML policies with SMT," in *Principles of Security and Trust*. Berlin, Germany: Springer, 2015, pp. 115–134.
- [21] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "An algebra for fine-grained integration of XACML policies," in *Proc. ACM Symp. Access Control Models Technol.*, 2009, pp. 63–72.
- [22] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "Fine-grained integration of access control policies," *Comput. Secur.*, vol. 30, nos. 2–3, pp. 91–107, Mar./May 2011.
- [23] M. Siponen and A. Vance, "Neutralization: New insights into the problem of employee information systems security policy violations," *MIS Quart.*, vol. 34, no. 3, pp. 487–502, Sep. 2010.
- [24] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo, "Policy decomposition for collaborative access control," in *Proc. ACM Symp. Access Control Models Technol.*, 2008, pp. 103–112.
- [25] K. Brown, M. Hayes, D. Allison, M. A. M. Capretz, M. Sazio, and R. Mann, "Fine-grained filtering to provide access control for data providing services within collaborative environments," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 6, pp. 1445–1466, Apr. 2015, doi: 10.1002/cpe.3167.2013.
- [26] S. Walraven, B. Lagaisse, E. Truyen, and W. Joosen, "Dynamic composition of cross-organizational features in distributed software systems," in *Distributed Applications and Interoperable Systems*. Berlin, Germany: Springer, 2010, pp. 183–197.
- [27] J. Mclean, "The algebra of security," in *Proc. IEEE Symp. Secur. Privacy*, Apr. 1988, pp. 2–7.
- [28] P. Bonatti, S. D. C. di Vimercati, and P. Samarati, "An algebra for composing access control policies," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 1, pp. 1–35, Feb. 2002.
- [29] D. Wijesekera and S. Jajodia, "A propositional policy algebra for access control," *ACM Trans. Inf. Syst. Secur.*, vol. 6, no. 2, pp. 286–325, May 2003.
- [30] P. Mazzoleni, B. Crispo, S. Sivasubramanian, and E. Bertino, "XACML policy integration algorithms," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 1, pp. 852–869, Feb. 2008.
- [31] N. Li et al., "Access control policy combining: Theory meets practice," in *Proc. ACM SACMAT*, 2009, pp. 135–144.
- [32] V. D. Gligor, H. Khurana, R. K. Koleva, V. G. Bharadwaj, and J. S. Baras, "On the negotiation of access control policies," in *Proc. 9th Int. Workshop Secur. Protocols*, 2001, pp. 188–201.
- [33] P. McDaniel and A. Prakash, "Methods and limitations of security policy reconciliation," *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 3, pp. 259–291, Aug. 2006.
- [34] H. Wang, S. Jhat, M. Livny, and P. D. McDaniel, "Security policy reconciliation in distributed computing environments," in *Proc. 5th IEEE Int. Workshop Policies Distrib. Syst. Netw. (POLICY)*, Jun. 2004, pp. 137–146.
- [35] H. Gimpel, H. Ludwig, A. Dan, and B. Kearney, "PANDA: Specifying policies for automated negotiations of service contracts," in *Service-Oriented Computing—ICSOC*. Berlin, Germany: Springer, 2003, pp. 287–302.
- [36] C. D. P. K. Ramli, H. R. Nielson, and F. Nielson, "The logic of XACML," in *Formal Aspects of Component Software*. Berlin, Germany: Springer, 2012, pp. 205–222.
- [37] *Java Architecture for XML Binding (JAXB)*. [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/index-140168.html>
- [38] *xacmlcorev3schemawd17.xsd*. [Online]. Available: <http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd>
- [39] L. Duan et al., "Automated policy combination for data sharing across multiple organizations," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun./Jul. 2015, pp. 226–233.



LI DUAN received the M.Sc. degree from the Mathematical School, Zhengzhou University, Zhengzhou, China. She is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. She is also pursuing the Joint-Training Ph.D. degree with Data61, CSIRO, Australia. Her main research interests include services computing, services security and privacy of distribution system, and policy combination.



YANG ZHANG received the Ph.D. degree in computer applied technology from the Institute of Software, Chinese Academy of Sciences, in 2007. He is currently with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China. His team makes scientific research on mobile service platform. He has authored papers concern anonymous routing protocols, anonymous authentication protocols, design and implementation of anonymous systems, and pseudonym systems. His research interests include security and privacy of anonymous systems.



SHIPING CHEN received the Ph.D. degree in computer science from the University of New South Wales, Sydney, NSW, Australia. From 1990 to 1999, he worked on real-time control, parallel computing, and CORBA-based Internet gaming systems in research institutes and the IT industry. Since joining CSIRO in 1999, he has worked on a number of middleware-related research and consultant projects, including software architecture, software testing, software performance modeling, and trust computing. He is currently a Research Scientist with Digital Productivity Flagship, CSIRO, Australia, and also an IT Professional with over 20 years of research experience and combined R&D skills.



SHUAI ZHAO is a Post-Doctoral Fellow the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His current research interests include Internet of Things and service computing.



SHIYAO WANG is currently pursuing the M.S. degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. She is majoring in computer science and technology. Her research interests include service computing and mobile service platform.



DONGXI LIU was a Researcher with the University of Tokyo from 2004 to 2008. He joined CSIRO in 2008. His current research focuses on the processing of encrypted data with the fully homomorphic encryption (FHE) scheme invented by him. His FHE scheme shows that the noise management techniques essential for the existing FHE schemes are not necessary. His FHE scheme is practically efficient and simple to understand and implement. The aim of his current research is to support secure outsourced computations on untrusted computing platforms, such as a public cloud.



REN PING LIU (M'09–SM'14) received the B.E. (Hons.) and M.E. degrees from the Beijing University of Posts and Telecommunications, China, and the Ph.D. degree from the University of Newcastle, Australia. He was a Principal Scientist with CSIRO, where he led wireless networking research activities. He is currently a Professor with the School of Computing and Communications, University of Technology Sydney, where he leads the Network Security Laboratory, Global Big Data Technologies Centre. He has authored over 100 research publications. His research interests include Markov analysis and QoS scheduling of wireless networks. He received the Australian Engineering Innovation Award and the CSIRO Chairman's Medal. He has supervised over 30 Ph.D. students. He is the Founding Chair of the IEEE NSW VTS Chapter. He served as the TPC Chair for BodyNets2015, ISCIT2015, and WPMC2014, and the OC Co-Chair for VTC2017-Spring, BodyNets2014, ICUWB2013, ISCIT2012, and SenSys2007, and on the Technical Program Committee in a number of IEEE conferences. He specializes in protocol design and modeling, and has delivered networking solutions to a number of government agencies and industry customers.



BO CHENG received the Ph.D. degree in computer science from the University of Electronics Science and Technology, China, in 2006. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include service computing, Internet of Things, and multimedia communications.



JUNLIANG CHEN is currently a Professor with the Beijing University of Posts and Telecommunications. His research interests are in the area of service creation technology. He was selected as a member of the Chinese Academy of Science in 1991, and a member of the Chinese Academy of Engineering in 1994.

...