Chapter 5

5. Proposed Framework

Sharing relevant and authorized health data in interoperable e-health environments requires identification and mitigation of security gaps accruing out of these collaborations. Identification of authorized requester and sender needs to be ascertained to establish legitimate sharing environment between healthcare organizations. Disparities in working nomenclature of each organization often results in irreconcilable ACPs. Moreover, union of ACPs often generate multiple and redundant rule sets thus congesting the state-space search leading to delayed decision of permitting/denying access to the data.

The framework must encompass heterogeneity of workflow and frequently changing demands of health professionals. Focusing on the above-mentioned issues, an access control framework is proposed that can integrate the policies by ranking user and resource attribute on the basis of hierarchical distances between the user and resource hierarchies of the healthcare organizations. The framework devises a Hierarchical Similarity Analyzer (HSA) that assign a security level to each user and resource attribute and fine-grains existing ACPs for better control and authorization on sharing of EHRs. The framework is designed in a manner that does not demand complex change in the existing policy control of the organization. It enables minimization of rules based on similarities between rule attributes that are obtained on merging of disparate policy sets.

The method of calculating the hierarchical distance between two attributes is adopted from a similarity-based policy filtering algorithm [183]. It is a lightweight approach that safely reduces the policies by pruning design rules. Our framework utilizes a part of this algorithm for allocating the security level to the attributes ensuring secured integration of access control policies and sharing of EHRs. The framework is designed in a manner that does not demand complex change in the existing policy control of the organization. It enables minimization of rules based on similarities between rule attributes that are obtained on merging of disparate policy sets.

The method of calculating the hierarchical distance between two attributes is adopted from a similarity-based policy filtering algorithm [183]. It is a lightweight approach that safely reduces the policies by pruning design rules. Our framework utilizes a part of this algorithm for allocating the security level to the attributes ensuring secured integration of access control policies and sharing of EHRs. The framework is designed in a manner that does not demand complex change in the existing policy control of the organization. It enables minimization of rules based on similarities between rule attributes that are obtained on merging of disparate policy sets.

The method of calculating the hierarchical distance between two attributes is adopted from a similarity-based policy filtering algorithm [183]. It is a lightweight approach that safely reduces the policies by pruning design rules. Our framework utilizes a part of this algorithm for allocating the security level to the attributes ensuring secured integration of access control policies and sharing of EHRs. The framework is designed in a manner that does not demand complex change in the existing policy control of the organization. It enables minimization of rules based on similarities between rule attributes that are obtained on merging of disparate policy sets.

Table 5.1 enlists the notations used in the algorithms described in figures 5.2 and 5.3.
It formally represents the authorization to perform specific set of operations by evaluating the attributes and constraints of given rules and policies under the similarity indexes obtained. Attributes, each holding multiple values are matched with each other and if a satisfying similarity is achieved, that rule becomes a probable candidate for generating a logical permit or deny decision. Similarity algorithm basically acts as a filter in producing justifiable subsets of matching rules and policies for sharing data between independent healthcare units.

5.1 Hierarchy Similarity Analyzer

Two or more healthcare units need to collaborate in order to allow access to each other’s data repository. Heterogeneity of data and schema prevails in each unit, thus sharing becomes a complex issue. An XACML based [186] access-control framework (Figure 5.1) is proposed that defines Hierarchy Similarity Analyzer algorithm to find the similarities between each attribute of the rules defined in disparate policies (P1, P2, .... , Pn). The HSA (Hierarchy Similarity Analyzer) calculates the similarity score for all matched and unmatched attributes based on the hierarchical distance of each attribute belonging to the subject (users) and resource (EHR) hierarchy. The rules are fine-grained by allocating a Security level (SL) for

![Figure 0.1: Proposed Access Control Framework for XACML Policies](image-url)
The computational steps for Hierarchy Similarity Analyzer are as follows:

For the sake of generality, the categorical or numerical similarity array obtained using the similarity algorithm [183] is referred as A. The steps for implementing HSA are as follows:

1. Take array A as input and store it in vector C in sorted order.
2. Store the indices of A and C as m and n respectively.
3. Find unique values in the obtained matrix, i.e.
   a. \([C, m, n] = \text{unique} (A)\)
4. Convert the matrix into 1-D array such that:
5. \(A = [](1 \times n)\) for all the values in A
6. Sort the 1D array in ascending order
7. Find the occurrence of each unique value and match with its position in the matrix.
8. Assign SL to each value accordingly

It is assumed that A can return unique set of values irrespective of the type of array i.e. categorical or numerical array. A is an intersection of rows and columns and a similar set of rows and columns are returned in C in sorted order. M returns the position of each value in the array A which is used to assign the unique values obtained in C accordingly. The functionality of HSA can be understood with the help of HSA algorithm described in figure 5.2.

HSA algorithm generates a unique SL for each user in all the collaborative healthcare units. Similarity scores obtained on the calculated hierarchical distances in user or resource hierarchies is taken as input in the HSA. Line 1 creates a 1-D array \(S_{\text{score}}\) to store the similarity scores of both the policies. Line 2 declares and defines the size of 1-D array \(S_{\text{uniq}}\) that would store the unique similarity scores from \(S_{\text{score}}\). Line 3 and 4 begins the iteration in hierarchy of hospital A and hospital B respectively. Line 5 iterates for every element in the
hierarchy of A and B. Line 6-10 checks if the current value in the array Score exist in the array S_uniq and set a flag value to 1 and break the loop otherwise and set flag to 0. Line 11 passes the control back to Line 5. Line 12-15 appends the S_score in the array S_uniq and increments its position by 1. Line 15 passes the control to Line 4. Line 16 passes the control to Line 3. Line 17 passes the control to another procedure Sort that sorts the array S_uniq in descending order. Line 18 then calls other procedure Assign_Security_Level that converts and assigns the similarity scores in S_score from the sorted array S_uniq. Line 19 returns the assigned SL for each user in the hierarchies of hospitals A and B.

Security level (SL): Hierarchy Similarity Analyzer (HSA) calculates the similarity score and assigns it as security level to each subject and resource attribute in the rule set of the policy. Security level cannot be less than zero and also cannot exceed the maximum hierarchy level of the hospital. SL is assigned to each user and resource in user and resource hierarchy respectively.

Security level (SL) is assigned to each user and resource attribute and has an upper bound limiting its range to the maximum hierarchy level specified in the respective organization. The SL if set to 0 in any rule signifies that the user or the resource has no matching entry in other rules and hence are not the probable candidates to collaborate and share data.

\[\forall u : Uset(A, u) \rightarrow SL, \text{ such that } 0 \leq SLu \leq UHierarchylevel\]

(Irrespective of user roles and responsibilities, SLA \(\equiv SLB\), if UHierarchyA = UHierarchyB)

\[\forall r : EHRset(A, r) \rightarrow SL, \text{ such that } 0 \leq SLr \leq RHierarchylevel\]

(Irrespective of type of resource in resource hierarchies, SLAr \(\equiv SLBr\) if RHierarchyA = RHierarchyB)

Algorithm: Hierarchy Similarity Analyzer (H_A, H_B)

<table>
<thead>
<tr>
<th>Algorithm: Hierarchy Similarity Analyzer (H_A, H_B)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td><strong>Output:</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3 for i= 1 (\rightarrow) H_{LevelA}</td>
</tr>
<tr>
<td>4 for j= 1 (\rightarrow) H_{LevelB}</td>
</tr>
<tr>
<td>5 for l=1 (\rightarrow) H_{LevelA} x H_{LevelB}</td>
</tr>
<tr>
<td>6 if S_uniq(l) = S_score (i, j) then</td>
</tr>
<tr>
<td>7 set flag to 1</td>
</tr>
<tr>
<td>8 break loop</td>
</tr>
</tbody>
</table>
9 else
10 set S_flag to 0
11 end for
12 if S_flag = 0 then
13 S_uniq (k) = S_score (i, j)
14 Increment k by 1
15 end for
16 end for
17 Sort (S_uniq (k))
18 Assign_Security_Level (S_score (H_{LevelA}, H_{LevelB}), S_uniq (), k)
19 return S_score (H_{LevelA}, H_{LevelB})

Sort (S_uniq (k))
1 for i = 1 -> k
2 set S_max to S_uniq (0)
3 for j = i -> k
4 if S_max < S_uniq (j)) then
5 S_max = swap (S_max, S_uniq (j))
6 end for
7 S_uniq (i) = S_max
8 end for

Swap(S_max, S_uniq (j))
1 set S_temp to S_max
2 S_max = S_uniq (j)
3 S_uniq (j) = S_temp
4 return S_max

Assign_Security_Level (S_score (H_{LevelA}, H_{LevelB}), S_uniq (), k)
1 for p =1 -> k or k greater than 1
2 set S_val to S_uniq (k)
3 for i = 1 -> H_{LevelA}
4 for j = 1 -> H_{LevelB}
5 if S_score (i,j) = S_val then
6 S_score (i,j) = k
7 end for
8 end for
9 decrement k by 1 and increment p by 1
10 end for

Figure 0.2: HSA Algorithm to obtain Security Level for Health Providers

5.2 Setting Authorizations

There exist multiple causes for sharing patient’s health data within different health facilities. Patient may change his health provider, can be transferred from his primary health unit to other super-speciality hospital or may have been admitted in case of emergency in some distant hospital. Sharing in such scenarios impose a problem of determining the authorization of the users to whom the data would be exposed.
The analyzer expands the user’s association with other users as well as itself. This is done to obtain the unique values as existence of two different SLs for the same association of rules is just not possible. It establishes well-defined authorization and refines the ACPs before merging to obtain decisions on sharing of EHRs in disparate but interoperable healthcare environment.

Authorizations are set within the upper bound of users in an organization. Figure 6.3 describes the algorithm devised for setting the authorization in the available set of rules. The users at the top level hierarchy hold higher authority and control over the data. Not all rule specifications demand the inclusion of authorization attribute. Hence, authorization is set as per the rules and policies of each healthcare organization. For some rule R in policy P, an authorization is subset of the authorized users of the system.

$$\forall p \exists r: (Pset (AP, p), Rset (AR, r)) \Rightarrow Authset(A,p,r), such \ that \ Authset (A) \subset Uset(A)$$

$$\forall p \exists r: (Pset (BP, p), Rset (BR, r)) \Rightarrow Authset(B,p,r), such \ that \ Authset (B) \subset Uset(B)$$

Authorization algorithm describes the logical decision in establishing authorization as per the attributes values in the specified rules. Authorization can be set to doctor, patient or the user whose security level is higher than the security level of the user specified in the rule. Setting authorization also takes into preference the action required on the data. If the action is ‘read’ and the security level of user is equivalent with the security level of EHR, which means that both represent same level of access, authorization can be set to patient or doctor. If the action is ‘write’ then it is important to determine the security level higher than that of the stated user and the authorization is set accordingly. The algorithm also keeps track of the maximum level of hierarchy defined in an organization. While determining the security level of the higher position, it is ensured that the level must not exceed the maximum limit. If the user specified in the rule has the maximal security level, authorization is set to itself or patient in such cases.

---

**Algorithm: Authorization**($sl_{user}$, $sl_{ehr}$)

1 $S_{Auth} = (doctor, patient, sl_{user})$
2 If $sl_{user} = sl_{ehr}$ and action= read then $S_{Auth} = patient$ or doctor
3 If $sl_{user} = sl_{ehr}$ and action= write then $S_{Auth} = sl_{user} + 1$
4 If $sl_{user} < sl_{ehr}$ and action= read then
5 Length_A = $H_{level}$
6 If $sl_{user} < $Length_A then
7 $S_{Auth} = sl_{user} + 1$
8 else if $sl_{user} = A$ then
9 $S_{Auth} = patient$
10 If $sl_{user} < sl_{ehr}$ and action= write then
Authorization algorithm takes input of SLs of users and resource to determine the authority. It also takes a third parameter i.e. action that the user wishes to perform on the resource. This algorithm specifies only two very common actions (read, write) that the users usually demand on any resource. The algorithm can be easily extended to include all possible type of actions that the hospital permits its users on accessing the records of the patient.

5.3 Significance of the Proposed Framework

EHR is often shared among fellow professionals for providing quality and timely care to the patient. Maintaining confidentiality of these records is primarily a part of their code of conduct. Security cannot be ascertained merely by identifying authorized users, groups or availability of possible resources required to be accessed. No access control policy can completely express the real-world workflow as it encompass dynamically imposed conditions that require dynamic modifications to the policy-sets.

HSA calculates SL on the basis of the hierarchical distances between each user in their user hierarchies. It is assumed that the users at each level of hierarchy in an organization tend to depict similar behaviour in terms of their roles and responsibilities. Keeping this in view, hierarchical distances are calculated between each user-pair and SL is assigned accordingly.

It restricts the user-access of data to only specific security level, as per the position in the user hierarchy. The framework provides a second level of security by setting up authorizations along with SLs. The authorizations ensure the accountability and authority of the respective user allowing/disallowing access to EHR. Simple security rule ‘No read up and No write down’ and the logical combination of security level and action attributes specified in the rule, is applied to establish authorization. The authorization establishes higher level of control and authority on taking decision to access the data.
Depending on the size and technical constraints, each organization defines and deploys its own access control schema exhibiting heterogeneity in organizational rules and policies. The health professionals in each organization are grouped in a hierarchy based on their roles and responsibilities. Keeping in view the top-down and bottom-up differences in the access privileges, similarities between varied users is calculated on the basis of their hierarchical distances in respective hierarchies. The restrictions are imposed such that the user can share only the permissible amount of data under set authorizations. Based on the generated request, the rules are verified and an appropriate decision (Permit/Deny) is performed. The proposed framework promises to provide the following benefits.

**A user can have more than one SL for more than one operation** - Healthcare environment experience ‘Multiple Roles at a time’ assigned to the health providers. Hence, grant of access must be categorized as per varied roles and responsibilities assigned to a health provider. HSA assigns SL to the users and resources that may reflect differently in each policy. A relationship between users and resources is set according to the user’s position and authority in an organization.

**Allow sharing of only required amount of EHR** - An organization bifurcate patient’s EHR into various heads. An EHR hierarchy is then provided to HSA that generates SLs for each head and sub-head. This permits to enhance confidentiality without compromising minimum availability of EHR.

**Determine authority before accessing the data** - HSA binds EHR with different security levels. Further, authorizations are determined and assigned with each user-resource pair in the rule set of ACPs to restrict the access to the user not defined under that authorization.

**Rule reduction on merging of disparate policy sets** - HSA works on the matching rules in the policy sets and rejects all non-matching rules. This reduces the final set of rules verified to obtain quick and justified decision of permitting/denying access to the requested data.

**Allow to incorporate dynamically changing roles and responsibilities of health professionals** - Frequent shift of duties is a common phenomenon observed with health professionals. With shift in duties, the roles and privileges on data access changes dynamically. HSA can administer such changes as it generates SLs taking the user hierarchical positions as input. Wherever there is a change in hierarchy, SLs will automatically change.
Table 0.1: Notations/Terminologies for Proposed Framework

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{dist}(H_A, H_B)$</td>
<td>Similarity Scores based on two heterogeneous hierarchical structures $H_A, H_B$</td>
</tr>
<tr>
<td>$S_{score}(H_{LevelA}, H_{LevelB})$</td>
<td>Array to hold Security Level for each user in $H_A, H_B$</td>
</tr>
<tr>
<td>$S_{uniq}(H_{LevelA} \times H_{LevelB})$</td>
<td>Array to store unique similarity score from $S_{score}(H_{LevelA}, H_{LevelB})$</td>
</tr>
<tr>
<td>$S_{flag}$</td>
<td>Tag set to check and validate the state of $S_{uniq}$</td>
</tr>
<tr>
<td>Sort ($S_{uniq}(k)$)</td>
<td>A routine to arrange the values in $S_{uniq}$ in ascending order</td>
</tr>
<tr>
<td>Assign_Security_Level</td>
<td>Rotate in the complete hierarchy of the users or resources and assign the respective value in $S_{uniq}$ as SL</td>
</tr>
<tr>
<td>$S_{auth}$</td>
<td>Set of users that can authorize other users share the data</td>
</tr>
<tr>
<td>$sl_{user}$</td>
<td>Security Level of the respective user</td>
</tr>
<tr>
<td>$sl_{ehr}$</td>
<td>Security Level of the respective resource</td>
</tr>
<tr>
<td>Length$_{A}$</td>
<td>A variable to hold the value of maximum hierarchy level in an organization</td>
</tr>
<tr>
<td>$H_{Level}$</td>
<td>Maximum hierarchy level in an organization</td>
</tr>
<tr>
<td>HSA</td>
<td>Hierarchy Similarity Analyzer generates SL</td>
</tr>
<tr>
<td>SL</td>
<td>Security Level determining access rights in policy sets of disparate hospitals</td>
</tr>
</tbody>
</table>

5.4 Operational Scenario

Assuming interoperable systems in place[33], the main focus will be on determining the similarity between users’ hierarchy of two independent healthcare units. Each attribute of a rule in one policy is matched with similar attribute of a rule in other policy. A rule similarity score is thus obtained on all such rules that clearly identify highest similarity rules in two different policies. Similarity score enable strengthening of authorization between the intended users of EHRs. Also, higher similarity score ensures trustworthy relation between the users and also enhances confidence in administering security while sharing of desired resources. It enhances the secured decision-making in providing access to sensitive health information of the patient to users outside the domain of the organization.

This section gives a picturesque view of access control policies of healthcare organizations being merged using the proposed framework. ACPs embed the organizational security structure of protecting the sensitive and valuable health records from illegitimate access. ACPs are vulnerable to exposure and data leakages due to natural emergence of conflicts and discrepancies creeped up on integrating them in interoperable environment. Moreover, the solutions sought for determining secured sharing of EHRs must be verified on parameters deemed necessary to achieve. Hence, merging of ACPs must be verified and tested to identify the possible conflicts and threats to prevent any such leakages to occur. This chapter further includes the simulator adopted for verifying the proposed framework and justify its significance in identifying the existence of policy conflicts popped while merging of such ACPs.
5.4.1 Interoperable Healthcare Environment

Framing of access control policies uses a real-time workflow of the hospitals as observed through meetings and interviews with the hospital administrative staff and IT managers. The policies express the roles and the type of actions permitted or denied considering the working nomenclature of the hospital. A clarity and soundness of how much information needs to be revealed and permissions granted to the receiver is sought.

A hospital staff comprises of department head, consultants, senior and junior physicians, paramedical and administrative staff. Patient hops at various units or facilities during his visit to the hospital. Health provider record or view relevant health and other details during providing health care. Electronic Health Records (EHR) is a collection of patient’s personal, clinical and other billing details that is categorized under demographic, clinical and financial heads. The demographic data comprises of personal details and the details of the person whom the patient nominates to operate on his behalf in case of any contingency. The Clinical data groups the symptoms, laboratory tests, reports and prescriptions under different heads. History holds patient’s previous records; Physical records the general health statistics of patient like BP, weight, Sugar etc. The Diagnosis records the symptoms, test reports and the advice given by the doctor. All billing and insurance details are stored under Financial head. This information is accessed by various healthcare users ranging from top management to bottom level hierarchy in the organization.

The Request space in figure 5.4 infers the queries of the users requesting EHR access in interoperable healthcare organizations. Query tuple ordered as [Subject, SL_U, Resource, SL_R, Action, [Condition], Effect], is a subset of ABAC attributes defined in XACML-based
policy sets. \( P_i \), \( P_j \) and \( P_k \) represent the policy-sets of various healthcare organizations. Each policy exhibit similar and unique set of rules when merged with other policies. The existence of dissimilar rules often results in contradictory or non-matching rules and thus generates policy conflicts.

A request is made by a hospital to share the data, i.e. EHR, from other hospital. It is assumed that the hospitals are in legal agreement to collaborate and share their data. The hospitals participating in such collaboration is termed as ‘requester’ used for the hospital that is demanding the data and ‘sender’ used for the hospital entertaining the request.

**User and Resource Hierarchy**

The two hospitals, namely, HA and HB, have a set of user and resource (data) hierarchy usually in top-down manner such that the users at top level hold higher responsibilities, control and authorization on the data. The resource hierarchy represent the heads and sub-heads under which the respective health data of patient is stored. User Hierarchy (Figure 4.7) positions all the users i.e. health providers and patient, that access EHR in part or depending on their role and responsibility(s) in the organization.

\[
\text{UHierarchy}_{HA} = \text{User}_{HA} \in \text{Uset}(HA), \text{ where } (\forall n: (\text{Uset}(HA) = \{u_{a1}, u_{a2}, u_{a3}, ..., u_{an}\}))
\]

\[
\text{UHierarchy}_{HB} = \text{User}_{HB} \in \text{Uset}(HB), \text{ where } (\forall m: (\text{Uset}(HB) = \{u_{b1}, u_{b2}, u_{b3}, ..., u_{bn}\}))
\]

Resource Hierarchy (Figure 4.8) holds the EHRs of patients and arranges them in various heads like, patient’s personal data is recorded under ‘Demographic data’, diagnostic and prescriptions details into ‘Clinical Data’ and so on. Each of these heads is further divided into sub-heads as deemed feasible by the respective hospital.

\[
\text{RHierarchy}_{HA} = \text{EHR}_{set}(HA), \text{ where } (\text{EHR}_{set}(HA) \Rightarrow \forall i: (\text{EHR}_{HA}(HA, r) = \{ar_{i1}, ar_{i2}, ar_{i3}, ..., ar_{ijn}\}))
\]

\[
\text{RHierarchy}_{HB} = \text{EHR}_{set}(HB), \text{ where } (\text{EHR}_{set}(HB) \Rightarrow \forall k: (\text{EHR}_{HB}(HB, r) = \{br_{k1}, br_{k2}, br_{k3}, ..., br_{kkn}\}))
\]

**5.5 Verification Tool**

The issues discussed so far identify policy-conflict and rule-reduction as important parameters to be considered while integrating two or more disparate access control policies. It requires designing and testing of the proposed solution with respect to these parameters. The testing and verification deals with implementing the proposed solution and generate test cases satisfying the set conditional constraints and properties. The objective of testing is to check the proposed solution to verify the generated request through an exhaustive state-space
search. The literature reveals various verification tools ACPT [187], Margrave [188], Alloy [189], PoliVer [190] developed or used by researchers in the past for similar applications. Each of the tools can be used to verify the access policies using various one or the other fault-detection techniques such as logical formulas [191] of temporal logic (Computational Tree Logic (CTL) and Linear-time Temporal Logic (LTL)), data mining techniques, mutation testing etc. The prerequisite of each tool differs in their requirement for producing the test results.

Verification of access control systems prevalent in the real-world scenarios would provide a logical reasoning of the suitability and compatibility of applied system in satisfying the secured sharing of data in interoperable EHR-systems. The policies must be analyzed to determine their correctness and power with respect to the exposure of sensitive data to only the legitimate and authorized users. The main focus of verification is to identify the existence of irrelevant roles and unauthorized exposure of data resulting from merging of two disparate policies thus resulting in security risk. The verification process can be manual and automatic. Considering the size and complexity of the environment and the number of variables under test, it becomes logically impossible to test and verify each possible combination manually. Automated analysis and verification of access control policies gives promising and plausible results in far less time as compared with the manual testing. Moreover, automated tools can handle very large and complex state-space relations encompassing multiple variables closely-knit together. Various tools with their own advantages and limitations exist for verification of XACML access control policies.

Access Control Policy Testing (ACPT) tool developed by NIST is used to verify the policies fine-grained using the proposed framework. ACPT bridges the gap between disparate policies by generating enforceable policies directly from policy models. The tool allows static as well as dynamic verification of the generated policies thereby helping in the identification and reduction of faults in the policies. ACPT integrates NuSMV [192] for symbolic model checking and for generating combinatorial tests. NuSMV considers the defined or set states and the specified properties as input. NuSMV is linked to a MiniSat SAT solver to detect and falsify the defined properties through the generation of counterexample (Fig. 5.2) for predefined set of states. Counterexamples illustrate semantic differences between the two policies. More specifically, each counterexample represents a request that evaluates to a different response when applied to the two policy versions.
XACML policies comprise of three major components: Policies, Rules and Conditions. Hence, verifying the integration of two or more policies with respect to rules bounded under conditional constraints is sought. Three major verifications sought are: Policy coverage, rule coverage and Condition Coverage. The coverage is defined as the possibility of the stated component contributing to the decision obtained on the generated request.

**Policy Coverage:** The number of policies under test divided by the total number of policies existing in the given environment.

**Rule Coverage:** The number of rules corresponding to the request made divided by the total number of rules existing in the given environment.

**Condition Coverage:** The number of Boolean conditions involved for the policies and rules under test divided by two times the number of conditions.

A policy tester must thrive to generate requests that achieve 100% policy, rule and condition coverage. ACPT is capable to fulfil all three requirements and generates t-way covering array by taking the given variables as input. It is capable to generate 2, 3 and 4-way covering arrays. The rows of the covering array cover all t-way combinations of parameter values for incorporating into Symbolic Model Verifier (SMV) property specifications that can be processed by the NuSMV model checker. As multiple rules generate multiple decisions for the same request due to rule overlapping, ACPT specifies the precedence between rules by
providing rule combination algorithms, namely, first applicable, permit override and deny override to choose from. First-applicable follows the decision in the first rule to report permit or deny. In Permit override, permit decision takes precedence whereas in Deny override, deny decision takes the precedence. The framework is verified and tested against policy conflicts and undefined resource access using any of these algorithms. First Applicable Rule combining algorithm is adopted to derive the verification decisions on merging of two policies. It produces verification reports (Fig. 5.3) where if the property is true in the policy no test cases are generated. On the other hand, if the property is violated, counterexamples are generated.

5.5.2 Rule Combining Algorithms

The rule combining algorithms facilitate to produce a single decision on combining of different policy sets where each policy is capable enough to produce a decision of permitting/denying access to the resource. The decision obtained from each individual policy would end up in having a set of different decisions thus making it difficult to select the most viable of them all. Hence, a rule combining algorithm capable of merging disparate set of policies and generating a combined decision is most required. Further, these algorithms enable to identify and resolve policy-conflicts levied in merged rule and policy sets. Selecting the feasible algorithm depends on the priority set by the respective administrators. XACML provides a variety of rule combining algorithms to choose from.

Deny Overrides

This rule combining algorithm gives preference to the deny decision. To elaborate, if a decision obtained in two rules contradicts in such a manner where one rule generates a permit whereas other rule denies access on the same resource, the final decision would be ‘Deny’. This type is highly recommended in the applications demanding stringent confidentiality with compromising on availability of data.

Permit Overrides

This rule combining algorithm gives preference to the permit decision. To elaborate, if a decision obtained in two rules contradicts in such a manner where one rule generates a permit whereas other rule denies access on the same resource, the final decision would be ‘Permit’. This type is highly recommended in the applications that are flexible in easily making the
data available. Confidentiality and privacy factors may be compromised if not properly dealt with in this setup.

**First Applicable**

This rule combining algorithm provides the decision by checking the first occurrence of the decision in any rule. It that rule permits the access the overall decision generated by this algorithm would be ‘Permit’ and if the first rule denies the access, then irrespective of other rules permitting the same resource to be accessed, the final decision would be ‘Deny’. This type is feasible for the environment having huge volume of policies and rules. Checking for each combination is computationally expensive in such environments. Hence, a good choice can be made by applying first applicable algorithm that balances the confidentiality and availability of data access by its implicit behavior.

**5.5.3 Property Verification**

A property is a condition specified to verify the policy(s) under test. More generally, it is a method of investigating the behavior of the policy. Though, the verification can happen in the absence of the properties also, due to the large and complex policy rules, it becomes difficult to handle and trace faults leading to security gaps in the existing policy(s). Correctness and accuracy of ACPs is dependent on the quality of properties [193] devised to check the ACPs. The problem becomes graver when two or more policies are integrated with the purpose of sharing their resources within required security constraints. Property verification can be static as well as dynamic. ACPT applies SMV to specify the properties which are then validated by NuSMV model checker. The checker can apply SMV checker to verify the properties in either way.

**Static Verification:** The policy and the property are represented as Finite State Machine (FSM) model and temporal logics in SMV. NuSMV is used to check the policy against the stated properties using CTL. NuSMV checks the specified state against all the states to detect if any state is violated. If it can detect a violated state, NuSMV generates a counterexample for the authors to correct the violations or discrepancies manually using their own knowledge decision.

**Dynamic Verification:** In dynamic verification, the checker can identify the defects in the policy and validate whether the actual output is equivalent to the desired output. Dynamic
verification assures the correctness of the policy during its evaluation on the generated request. The states can be specified dynamically and can change as per the requirement set by the policy checkers.

ACPT generates test cases based on structural and combinatorial coverage. Structural coverage refers to whit-box testing technique that covers every possible state that can be reached in verifying the policy sets, at least once. Combinatorial coverage refers to black-box testing technique that concentrates on the given inputs and desired outputs to verify the policy sets against the properties specified. Structural coverage can prove to be time-consuming and resource-extensive thereby resulting in time-delay and sound technical knowledge for its application. Combinatorial testing on the other hand does reduce the size of the inputs but lacks to gain sufficient level of confidence on policy correctness. Thus, a combination of structural coverage followed by combinatorial coverage would gain in rule reduction and identification of hidden faults that otherwise may be missed while verifying the policies against the given properties.

A sample property used for verifying the ACPs in healthcare domain is shown below. Suppose there are two roles, Family_Doc and Matron. The Family-Doc can write into the diagnostic records of the patient whereas the same is denied for the Matron. The policy set of two healthcare units where the subject is Lab_Tech are merged to verify against the following property:

Property 1: Spec ((Role = Lab_Tech) & (EHR= Cl_Diag) & (Action=Write) -> decision=Deny)

Property 2: Spec ((Role = Lab_Tech) & (EHR= Cl_Diag) & (Action=Write) -> decision=Permit)
The results obtained in figure 5.3 clearly show that the properties defined in each policy contradict with each other. The decision obtained individually allows Lab_Tech of one health unit to access patient’s diagnostic data whereas the same is denied to the Lab_Tech of the second health unit according to the states in their respective policy sets. NuSMV verification obtained on merging of the policy sets finds the property satisfying all possible states in one healthcare unit whereas generates a counterexample for the second healthcare unit where a conflicting state is detected.