PAPER *Special Section on Information and Communication System Security*

Inconsistency Resolution Method for RBAC Based Interoperation

Chao HUANG[†], ***Student Member***, **Jianling SUN**[†a)], ***Nonmember***, **Xinyu WANG**[†], ***Member***, ***and*** **Di WU**[††], ***Nonmember***

SUMMARY In this paper, we propose an inconsistency resolution method based on a new concept, *insecure backtracking role mapping*. By analyzing the role graph, we prove that the root cause of security inconsistency in distributed interoperation is the existence of insecure backtracking role mapping. We propose a novel and efficient algorithm to detect the inconsistency via finding all of the insecure backtracking role mappings. Our detection algorithm will not only report the existence of inconsistency, but also generate the inconsistency information for the resolution. We reduce the inconsistency resolution problem to the known Minimum-Cut problem, and based on the results generated by our detection algorithm we propose an inconsistency resolution algorithm which could guarantee the security of distributed interoperation. We demonstrate the effectiveness of our approach through simulated tests and a case study.

key words: *role based access control, security inconsistency, role mapping, inconsistency detection, inconsistency resolution*

1. Introduction

With the pervasive application of network technology, more and more systems adopt the distributed architecture for higher quality of services. However, due to the complexity of distributed systems, the validation of the security policy is extremely significant and complicated [9]. At present, many applications achieve the interoperation among distributed domains via coordinating and integrating the local security policy of each individual domain to form the overall security strategy. It's efficient and convenient to ensure the security of interoperation in this way.

However, the establishment of interoperable security policy may cause the inconsistency between the local security policy and the overall security strategy [10]. For the secure distributed interoperation, the global policy which supports the inter domain data access must be consistent with the access control policies of each individual domain. In particular, secure interoperation requires enforcement of the following two principles [12]:

- *Principle of Autonomy*. Any access permitted within an individual domain must also be permitted under secure interoperation.
- *Principle of Security*. Any access not permitted within an individual domain must also be denied under secure interoperation.

If the above two principles can not be met, the distributed interoperation will introduce security inconsistencies to the system, and the whole access control system will be exposed to risks [1].

Role based access control (RBAC) [3] is widely used in current enterprize systems [2], since it supports role hierarchy, separation of duty constraint, cardinality constraint and so on [5]. Furthermore, RBAC is capable of modeling a wide range of access control policies including Discretional Access Control (DAC) and Mandatory Access Control (MAC) [4], [6]. RBAC supports the distributed interoperation via introducing Role Mapping (RM) [9]. Gong [12] gives the time complexity analysis of the problem to resolve the security inconsistency, and also presents a simple security inconsistency detection algorithm, whose time complexity is $O(N^3)$, wherein N is the total number of roles in the systems.

For enterprise systems, Gong's method is not efficient enough for practical usage. Pointing to this practical efficiency problem, Wang [15] presents a minimal inconsistency detection algorithm, which only takes into consideration the roles involving in the role mapping, thus the time complexity of inconsistency detection is reduced to $O(n^3)$, wherein n is the number of roles involving in the role mapping. Compared to Gong's method, the efficiency of Wang's inconsistency detection algorithm is improved a lot since n is usually much smaller than N.

Basit [9] proposes an Integer Program (IP) based inconsistency detection method. Basit designs a set of rules, which can guide the process of converting the RBAC policy to a set of IP constraints, and the possible inconsistencies will be inferred through complicated IP programming algorithms. For the large enterprise systems, converting all the policies to IP rules is pretty expensive and complicated, thus the practical application of Basit's method is restricted. Besides above, the works in [7], [11], [14] also investigate the security inconsistency problem and [16] discusses the constraint consistency.

For all of the above methods concerning the security inconsistency problem, none of them presents an integrated approach to resolve the inconsistency, since the inconsistency detection is only the first step and the inconsistency resolution is the ultimate goal. Based on the detection results, manual resolution is not impossible, however, for the

Manuscript received July 31, 2009.
Manuscript revised December 18, 2009.

[†]The authors are with Computer College of Zhejiang University, Hangzhou, Zhejiang, 310027, China.
[††]The author is with State Street Corporation, Zhejiang, China.
a) E-mail: Email: sunjl@zju.edu.cn

DOI: 10.1587/transinf.E93.D.1070

large enterprise application it's too tedious to dig into the complicated security policies to find a solution. Hence more advanced way is needed.

Pointing to the above problems, in this paper, we propose an automated inconsistency resolution method based on a new concept, *insecure backtracking role mapping*. Our major contributions in this paper include:

(1) We classify the security inconsistencies, and prove that the sufficient and necessary condition of security inconsistency is the existence of insecure backtracking role mapping.
(2) Based on role graph, we propose a more efficient inconsistency detection algorithm. Via comparing the detection methods in [12], [15], we demonstrate that our approach is much more efficient.
(3) We reduce the inconsistency resolution problem to the known Minimum-Cut problem. Then based on the existing Minimum-Cut algorithm, we propose an inconsistency resolution algorithm. We demonstrate the effectiveness via simulated experiments and a case study.

The rest of this paper is organized as follows: Sect. 2 details the concept of security inconsistency. Section 3 presents the inconsistency detecting method. Section 4 introduces the inconsistency resolving approach, which is based on the existing Minimum-Cut algorithm. In Sect. 5, we demonstrate the simulated test results of our inconsistency detection algorithm via comparing to the methods in [12] and [15], and also show the effectiveness of our inconsistency resolution algorithm. Section 6 presents a case study to demonstrate the practical usage of our approach in real applications. Lastly, we give the conclusion and outline the future work.

2. Security Inconsistency

Figure 1 shows the interoperation across multiple domains

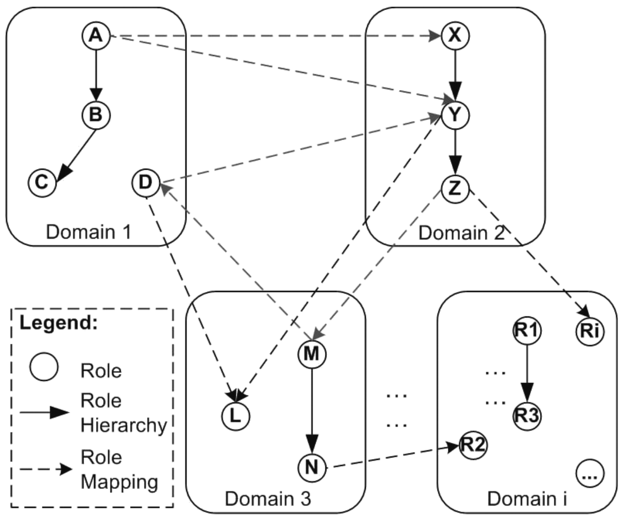

Fig. 1 Security inconsistency example.

via role mappings. For role mapping $A \rightarrow X$, role A is regarded as a senior role and role X is regarded as a junior role in comparison to A, i.e. the user to whom role A is assigned in Domain 1 will have all the permissions of role X when accessing to Domain 2. This is an efficient and convenient way to achieve the distributed interoperation while leveraging the existing security policies at the same time. Introducing the role mapping means that the current role system could be kept intact, thus the current enterprise investment could be preserved.

However, role mapping is not coming into being without any cost, since there may exist security inconsistencies after introducing role mappings. In Sect. 1, two principles to guarantee the security of interoperation have been presented, *Autonomy* and *Security*. Since the interoperation is realized via adding role mapping, the original role system of the individual domain has no change. Thus the compliance with the principle of *Autonomy* can always be achieved. However, since Role Mapping is a kind of transitive relation, as shown in Fig. 1, both of the following two situations violate the principle of *Security* and cause the security inconsistencies:

(1) Via role mapping *AY*, *ZM*, *MD* and role hierarchy *YZ*, role *A* in Domain 1 obtains the permissions of role *D*, which is not allowed before the interoperation (before introducing the role mapping).
(2) Via role mapping *DY*, *ZM*, *MD* and role hierarchy *YZ*, role *Z* obtains its senior role *Y*'s permission, i.e. the *Security* principle is violated.

The first kind of inconsistency is called *unfounded security inconsistency*, and the second kind is called *founded security inconsistency*.

Theorem 1: There are only two kinds of security inconsistencies in distributed interoperation, one kind is unfounded security inconsistency and the other kind is founded inconsistency.

Proof: see the Appendix A.

3. Detecting Security Inconsistency

Detecting security inconsistency is the precondition of inconsistency resolution. Although there have been a lot of work concerning the inconsistency detection, none of the current methods could provide sufficient inconsistency information for the resolution. In this section, we propose a new inconsistency detection algorithm which will not only report the existence of inconsistencies, but also generate the inconsistency information for future resolution. Our approach is based on the analysis of role graph.

Definition 1: Role Hierarchy edge

Role hierarchy is a binary relation set H_i on the node set V_i of the role system G_i, for each element $(u_i, v_i) \in H_i$, $u_i \in V_i$ $v_i \in V_i$, u_i is the senior role to v_i. H_i is an anti-reflexive and anti-symmetric binary relation, i.e. $\forall u_i \in V_i, v_i \in V_i$, if

$(u_i, v_i) \in H_i$, then $(v_i, u_i) \notin H_i$. We call the (u_i, v_i) as *role hierarchy edge* (*RHE*). H_i^+ is the transitive closure of H_i.

Definition 2: Single domain role system graph

The role set of domain i is represented as R_i, and the role hierarchy set is represented as H_i. Thus the role system of domain i can be represented as a directed graph $G_i =< V_i, E_i >$, wherein the node set $V_i = R_i$ and $E_i = H_i$, and E_i^+ represents the transitive closure on the directed edge set E_i.

Definition 3: Role mapping edge

Role mapping M is the binary relation set on $\cup_{i=1}^n V_i$, for each element $(u, v) \in M$, then $u \in V_i$, $v \in V_j$ and $i \neq j$. M is anti-reflexive, i.e. a role can't be mapped to itself. We call the $(u, v) \in M$ as *role mapping edge* (*RME*).

Definition 4: Global role system graph

Global role system across multi-domain is represented as the graph $G =< V, E >$, wherein the node set is $V = \cup_{i=1}^n V_i$, and the directed edge set is $E = \cup_{i=1}^n E_i \cup M$, and M is the role mapping set.

Definition 5: Trivial role mapping cycle

For the role $u \in V_i$, $v \in V_j$ and $i \neq j$, if $(u, v) \in M$ and $(v, u) \in M$, then the cycle via (u, v) and (v, u) is called trivial role mapping cycle. Role u is the *equivalent role* to v in the global role system, and vise versa.

Definition 6: Role hierarchy path, role mapping path

Both role hierarchy and role mapping are transitive relations, hence in the global role system G, if there is a connected path from node u to different node v, then we call (u, v) as role transitive path, we call u, v as start role and end role separately. If there is no role mapping edge on the path (u, v), we call it *role hierarchy path*, otherwise, it is called *role mapping path*.

Definition 7: Backtracking role mapping path

For the role mapping path (u, v) which is not the trivial role mapping cycle, if u and v belong to the same domain, then we call (u, v) as a backtracking role mapping path. We call $< u, v >$ as a backtracking role mapping role pair.

Definition 8: Insecure backtracking role mapping path

In the global role system, there is a backtracking role mapping path (u, v) assuming the same role system to which both u and u belong is $G_i =< V_i, E_i >$, if $(u, v) \notin E_i^+$, then (u, v) is called insecure backtracking role mapping path (*IBRMP*) and is called insecure backtracking role pair (*IBRP*).

Theorem 2: The sufficient and necessary condition of security inconsistency is that there exist insecure backtracking role mapping paths.

Proof: see the Appendix B.

```
Algorithm1: IDALG ( G' )
Input: G', the minimal role system graph
Output: insecure backtracking role pair (IBRP) set
1  initialize set s
2  initialize list l
3  foreach role node u in G'
4    mark u as URNode
5    span( u )
6    while l is not empty
7      retrieve the first element (u,v) in l
8      remove (u,v) from l
9      if v is Rnode and u ≠ v
10       if u and v belong to the same domain G_i
11         if u and v is not connected in G_i
12           add <u , v> to s
13 return s
```

Fig. 2 Inconsistency detecting algorithm.

```
Procedure 1: span (node u )
1  mark u as the visited node
2  foreach edge (u,v) in G'
3    if u is the RNode
4      mark v as RNode
5    else
6      if edge (u,v) is role mapping edge
7        mark v as RNode
8      else
9        mark v as URNode
10       add (u,v) to l as the first list element
```

Fig. 3 Procedure for inconsistency detecting algorithm.

Definition 9: Minimal Role Graph (*MRG*)

$G' =< V', E' >$ is the minimal role graph, wherein: $V' = \{u | \exists v, (u, v) \in M \vee (v, u) \in M\}$, $E' = M \cup \{(u, v) | (u, v) \in H_i^+ \wedge u \in V' \wedge v \in V'\}$.

A minimal role graph is essentially a sub-graph of the global role system graph. It only includes the roles and role hierarchies, which are involved in the role mapping. As the interoperation shown in Fig. 1, hierarchies (X, Y) and (Y, Z) are included in the corresponding minimal role graph, and hierarchies (A, B) and (B, C) are not included, because of $B \notin V'$ and $C \notin V'$.

Theorem 3: A minimal role graph contains the same insecure backtracking role mapping paths as the global role system graph does.

Proof: see the Appendix C.

According to Theorem 3, the inconsistency detection could be simplified via performing the detection on the minimal role graph. Based on Theorem 2, we give the inconsistency detecting algorithm (*IDALG*) in Fig. 2 and Fig 3.

For the convenience of description, we assume the following abbreviations:

- *RNode*: the node which is reached via role mapping edge.

- *URNode*: the node which is reached without any role mapping edge.

The core of the *IDALG* is to build a depth first search tree first, via determining whether the connections exist among the roles nodes in each single domain. The *span()* procedure marks the nodes which are reached via role mapping edges. After invoking *IDALG*, if the returned set s is empty, then there is no security inconsistency, otherwise the elements in s are the insecure backtracking role pairs, which cause the security inconsistencies.

The time complexity of *IDALG* is: $T_1 = O(n \times (m + m)) = O(nm)$, wherein $n = |V'|$, i.e. the vertex number in the minimal role graph, $m = |E'|$, i.e. the edge number in the minimal role graph. The time complexity of detection algorithm in [12] is: $T_2 = O(N^3)$, wherein $N = |V|$, i.e. the vertex number in the global role system graph. The time complexity of detection algorithm in [15] is: $T_3 = O(n^3)$, $n = |V'|$, i.e. the vertex number in the minimal role system graph.

For a RBAC role system graph, the maximum edge number is: $m = \frac{n(n-1)}{2}$, thus $m < n^2$. Furthermore, according to the case study report in [8], for a practical role system, m is much smaller than n^2, i.e. $m << n^2$, with $n < N$, thus we can get:

$$T_1 < T_3 < T_2,$$

i.e. our algorithm is the most efficient one based on theoretical analysis. We will verify this in the simulated test section.

4. Resolving Security Inconsistency

Based on the inconsistency detection algorithm, the set containing all of the insecure backtracking role pairs could be obtained. According to Theorem 2 and 3, eliminating those corresponding insecure backtracking role mapping path will resolve the inconsistency in the global role system. The mapping relationship from insecure backtracking role pair to insecure backtracking role path is one to many, i.e. one *IBRP* can be mapped to several *IBRMP*s. However, one *IBRMP* can only be mapped to exactly one *IBRP*. Pointing to the above factors, we design an inconsistency resolution algorithm based on Minimum-Cut (MC) theory [17]. The content of MC problem is not the focus of this paper, so we just explain the basics of MC to introduce our inconsistency resolution algorithm.

Definition 10: Minimum-Cut (Min-Cut), [17]

If in the digraph G there exist a set of edges, eliminating which will make vertex s unreachable to vertex t, i.e. there is no pathway from s to t, then the set of edges are called cut set of digraph G. The cut set with the minimal sum of edge's capacity (weight) is called the minimal cut of G.

For digraph G, the vertex set is V, if $S \subset V$ and $T = \overline{S} = V - S$, assuming $X = (S, T)$ containing all the edges

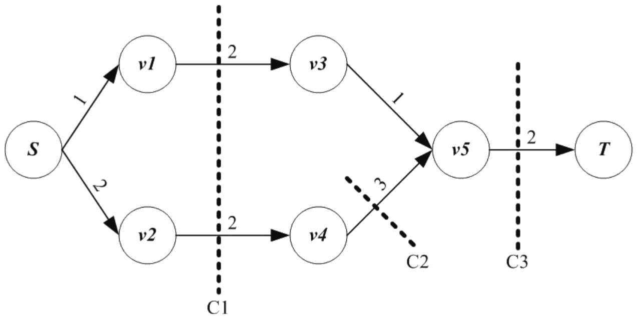

Fig. 4 Sample Min-Cut of S and T.

whose starting vertex is contained in set S and ending vertex is contained in set T, i.e.

$$X = \{(v_i, v_j)|v_i \in S, v_j \in T\},$$

then X is a cut of G, since eliminating the edges in X will make any vertex in S disconnected to T.

For all of the cuts, if cut X' satisfying:

$$\min\left\{\sum_{(v_i,v_j)\in X} w_{ij}|X \text{ is the cut set of } G\right\},$$

wherein w_{ij} is the capacity (weight) of edge (v_i, v_j), then X' can be call the Min-Cut of G.

Figure 4 shows the sample cuts of S and T, in which there are three marked sets:

$$C1 = \{(v1, v3), (v2, v4)\}$$

$$C2 = \{(v4, v5)\}, C3 = \{(v5, T)\}.$$

According to Definition 10, $C2$ is not the cut set for S and T, since removing $(v4, v5)$ does not disconnect the path from S to T. Both $C1$ and $C3$ are the cuts, and the cut value of $C1$ and $C3$:

$$|C1| = \sum_{(v_i,v_j)\in C1} w_{ij} = 2 + 2 = 4, \quad |C3| = 2.$$

$C3$ is a smaller cut than $C1$, and furthermore we can see that $C3$ actually is the cut with smallest cut value, thus $C3$ is the Min-Cut for S and T of the example in Fig. 4.

Now we reduce the inconsistency resolution problem to Min-Cut problem.

Definition 11: Min-Cut of *IBRP*

For a role system $G =< V, E >$, assuming $< V, E >$ is the insecure backtracking role mapping pair, to find:

$$\min\left\{\sum_{(v_i,v_j)\in X} w_{ij}|X \text{ is the cut set of } G\right\},$$

wherein $w_{ij} = \begin{cases} 1 & \text{if}(u, v) \text{ is } RME \\ +\infty & \text{if}(u, v) \text{ is } RHE \end{cases}$.

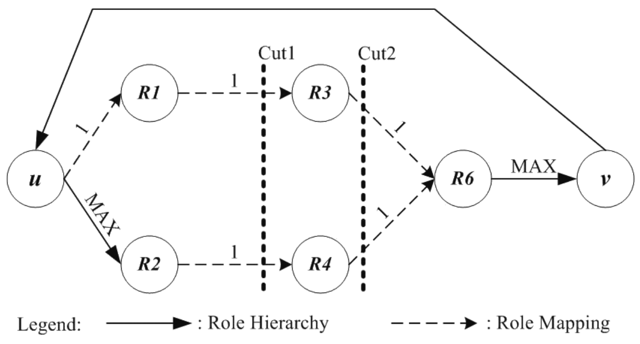

Fig. 5 Sample Min-Cut of insecure backtracking role pair.

Utilizing the min-cut method to resolve security inconsistency provides the great flexibility to choose the way of inconsistency resolution, since different capacity weight settings will yield different resolution strategies. In most situations, the resolution needs to keep the role hierarchy, since the autonomy principle should be complied with.

The capacity of role hierarchy is set to $+\infty$, in this way any resolution candidate which tries to eliminate role hierarchy edges will be not selected, since the minimum cut algorithm will choose a smaller one. The capacity of role mapping edge is set to one, which indicates role mapping edge could be eliminated in the final inconsistency resolution, and the one which needs to eliminate least number of role mapping edges will be selected based on minimum cut method. The less the eliminated edges are, the more the data could be shared. Due to the above settings, the inconsistency could be resolved at the same time complying with the autonomy principle and maximizing the data sharing.

Based on the results from detection algorithm in Fig. 2, the insecure backtracking role pairs could be obtained. For each pair (u, v), $\{u\}$ can be regarded as S, and $\{v\}$ can be regarded as T, applying the minimum cut algorithm and according to the specification of the algorithm in [13], S will not be connected to T after removing the minimum cut edge sets. According to definition 8, the insecure backtracking role mapping paths will be removed. We have proved in Theorem 2 that eliminating the insecure backtracking role mapping paths will eliminate the security inconsistency. Thus minimum cut problem is applicable to resolve the security inconsistency.

For the inconsistency resolution, the common rule is not to change the role hierarchy relationships in each individual system, whereas to modify the role mapping in order to obey the consistency restriction. The objective of interoperation is to share data and information, thus eliminating role mapping edges is against the objective. In order to comply with the consistency restriction and maximize the data sharing, the number of eliminated role mapping edges should be minimized. For each insecure backtracking role pair, it is required to find the corresponding Min-Cut, i.e. the minimum set of role mapping edges which should be eliminated to guarantee the consistency.

Figure 5 shows the insecure backtracking role pair $< u, v >$, and the corresponding insecure backtracking role mapping path. The inconsistency exists since a junior role u obtains the permission from its senior role v via role mappings $(u, R1)$ $(R1, R3)$ $(R3, R6)$ and hierarchy $(R6, v)$. The cut shown in Fig. 5 is as following:

$$Cut1 = \{(R1, R3), (R2, R4)\}, |Cut1| = 1 + 1 = 2,$$

$$Cut2 = \{(R3, R6), (R4, R6)\}, |Cut2| = 1 + 1 = 2,$$

since the role hierarchy needs to be preserved. Thus the marked $Cut1$ or $Cut2$ are the minimal cuts for $< u, v >$, and eliminating the edges in $Cut1$ or $Cut2$ will resolve the inconsistency.

There has been a lot of research work to solve the MC problem, we choose the one proposed in [13], since it's easy to implement and pretty efficient. Our inconsistency resolution algorithm (*IRALG*) is shown in Fig. 6 and Fig 7. The algorithm consists of two major steps:

(1) Before the inconsistency resolution, the weight of role hierarchy edges and role mapping edges in G' needs to be set.

```
Algorithm 2: IRALG (G', IBRPSet)
Input: minimal role graph G'
Input: IBRPSet, insecure backtracking role pair set
Output:
1   create an empty list L1
2   add all the vertex of G' to list L2
3   while ( L2 is not empty)
4     retrieve the first element u from L2
5     remove u from L2
6     search (u)
7     while (L1 is not empty)
8       retrieve the first element (u, v) from L1
9       remove (u, v) from L1
10      remove vertex v from L2
11      if(edge (u, v) is role mapping edge)
12        set w(u, v) = 1
13      else
14        set $w(u,v) = +\infty$
15        if (v is not VISITED vertex)
16          search (v)
17  create an empty edge set es
18  foreach (element $<u,v>$ in IBRPSet)
19    set $S = \{u\}$ and set $T = \{v\}$
20    invoke the Min-Cut(G', S, T) algorithm in [13]
21    record the cut edge set as CS
22    add CS to es
23  return es
```

Fig. 6 Inconsistency resolving algorithm.

```
Procedure 2: search(u)
Input: vertex u
1   mark u as the VISITED vertex
2   foreach edge (u, v) whose starting vertex is u
3     add (u, v) to L1 as the first element
```

Fig. 7 Procedure for inconsistency resolving algorithm.

(2) For each insecure backtracking role pair $< u, v >$, we set $S = \{u\}$ and $T = \{v\}$, then invoke the existing Min-Cut algorithm to calculate the minimal cut. The edges in the minimal cut are the ones need to be eliminated for resolving the inconsistency.

Two implementation issues need to be addressed:

- In the practical implementation, $+\infty$ can be implemented via setting the value as *MAX_INTEGER*. The Min-Cut algorithm [13] needs to be changed a little bit. During each iteration of flow calculation, the weight value of those edges that are marked as *MAX_INTEGER* will not be reduced. In this way, the role hierarchy edge will not be considered as a candidate eliminating edge.
- Since the algorithm is based on the existing Min-Cut algorithm, the extensibility is pretty good. If the security manager considers that some role hierarchy edges can be eliminated to keep the consistency of the access control policy, he only has to make a little change to the weight of those role hierarchy edges, setting a reasonable integer value instead of setting as *MAX_INTEGER*, then such role hierarchy edges will be considered as the candidate eliminating edge during the Min-Cut calculation. On the other side, if the security manager thinks that some role mapping edges must be preserved, he just needs to set the weight value of those edges as *MAX_INTEGER*.

The Min-Cut algorithm proposed in [13] is a very efficient one, whose time complexity is $O(nm\log(n))$, wherein n is the vertex number in G' and m is the edge number. Thus the overall time complexity of inconsistency resolution algorithm is:

$$T = O(m + n) + O(kmn\log(n)) = O(kmn\log(n)),$$

wherein k is the number of elements in insecure backtracking role pair set generated in Algorithm 1.

5. Simulated Test and Discussion

The purpose of the experimental evaluation is three-fold:

(1) To evaluate whether our algorithm is efficient enough for large enterprise application. Now that in [15] Wang has demonstrated his method (Minimal inconsistency detection, *MIDALG*) is more efficient than the one in [12], we only compare our algorithm to Wang' method.
(2) To validate that our algorithm works in a wide variety of environments ranging from small to large organization, and comprising of complex role mapping relationship.
(3) To evaluate the efficiency as well as effectiveness of our inconsistency resolution algorithm. We want to verify whether removing all the edges generated by our resolution algorithm would eliminate all the inconsistencies via applying both our detection algorithm and Wang's method again.

We create a test data generator, which performs as follows:

(1) A set of domains are created, the number of which is set to a certain maximum number *ND*.
(2) For each domain, a set of roles are created randomly, the number of which is set to a maximum number *NR*, and based on each role set, the role hierarchies are added randomly under a certain ratio *RRH* (equals to *NRH*/*NR* wherein *NRH* is the number of role hierarchies).
(3) We randomly choose several roles to be involved in the interoperation, the number of which is set to *NI*, and the role mappings are added across the domains randomly over the chosen interoperation role set under a certain ratio *RRM* (equals to *NRM*/*NI*, wherein *NRM* is the number of role mappings).

Our generator uses a simple randomization strategy. Thus the possible security inconsistencies are embedded in the global role systems randomly, which allows us to test in an unbiased manner.

Our experiment focuses on the impact of different *RRM* to the performance of our algorithms. Thus we set the parameters, which are related to the local role system, such as *NR*, *RRH*, as fixed values and vary the parameters which influence the interoperation, including *NI* and *RRM*. Based on the case study report of a large European Bank in [8], we set the parameters as follows:

- $ND = 20$
- $NR = 1000$
- $NI = 200, 400, \ldots, 1800, 2000$
- $RRH = 0.5$
- $RRM = 0.1, 0.2, 0.3, 0.4, 0.5$, and *WC*

The case study in [8] has shown that a role system with role hierarchy ratio *RRH* of 0.5 has been pretty complex. The role mapping ratio varies from 0.1 to *WC* for the worst case, which means for the certain NI, the number of role mapping *NRM* among the interoperation roles reaches the maximum possible value, i.e. $NRM = \frac{NI(NI-1)}{2}$, the maximum mapping edge number for a role graph. Thus our experiment could cover the worst theoretical situation.

The experimental data is collected on a machine with following configuration:

- Single Intel Core 2 T7500 @ 2.20 GH.
- 2 G main memory.
- Operating system is Microsoft XP SP4.
- Java with JRE 1.6.0_13.

Table 1 and Table 2 show the CPU time records of our detection algorithm *IDALG* and Wang's *MIDALG* with the different *NI* and *RRM*. The result shows that our algorithm has better performance than *MIDALG* in most situations. Even in the worst case (*WC*), our approach still runs a little faster than *MIDALG*.

Figure 8 shows the comparison of the two algorithms.

Table 1 Execution time of algorithm *MIDALG*.

NI	***T1*** (milliseconds)					
RRM=	0.1	0.2	0.3	0.4	0.5	*WC*
200	49	49	49	51	51	54
400	328	329	329	488	489	513
600	1101	1101	1117	1135	1119	1175
800	2628	2663	2662	2694	2679	2813
1000	5256	5272	5306	5290	5307	5572
1200	9212	9246	9230	9263	9330	9797
1400	14597	14695	14713	14665	14715	15451
1600	21851	21853	22002	22085	22103	23208
1800	31058	31355	31078	31244	31426	32997
2000	43434	42927	42715	42898	42967	45115

Table 2 Execution time of algorithm *IDALG*.

NI	***T2*** (milliseconds)					
RRM=	0.1	0.2	0.3	0.4	0.5	*WC*
200	9	15	20	32	49	53
400	44	87	122	250	315	497
600	99	193	322	437	543	1139
800	193	365	570	772	967	2728
1000	289	595	899	1205	1658	5405
1200	421	862	1247	1748	2276	9503
1400	596	1105	1691	2328	2991	14990
1600	746	1523	2309	3111	3919	22518
1800	986	1935	2800	3857	4957	32018
2000	1198	2179	3560	4714	5729	43779

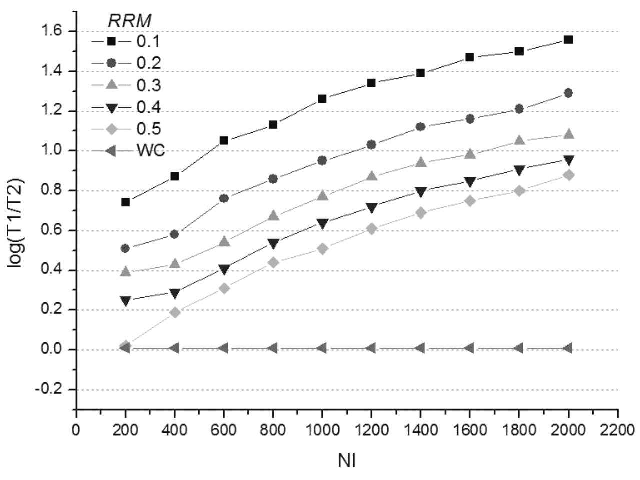

Fig. 8 Execution time comparison of inconsistency detection algorithms.

We calculate the value of $\log(T1/T2)$, which shows the execution time comparison more clearly. The value of $\log(T1/T2)$ can be categorized into the following three cases:

(1) $\log(T1/T2) > 0$, which means $T1 > T2$, i.e. *IDALG* runs faster than *MIDALG*.
(2) $\log(T1/T2) = 0$, which means $T1 = T2$, i.e. *IDALG* runs as fast as *MIDALG*.
(3) $\log(T1/T2) < 0$, which means $T1 < T2$, i.e. *IDALG* runs slower than *MIDALG*.

From Figure 8, the following observations could be obtained:

- *IDALG* runs faster than *MIDALG*, even in the worst case.
- The efficiency advantage of our approach will become more remarkable as the number of interoperation roles increases, which means our approach is more suitable for the large enterprise application. Although such kind of advantage will become less remarkable as the number of role mapping edges increases, as long as *NI* is large enough, our approach still runs faster than *MIDALG* or at least as fast as *MIDALG*.

Table 3 Execution time of algorithm *IRALG*.

NI	***T*** (milliseconds)					
RRM=	0.1	0.2	0.3	0.4	0.5	*WC*
200	1	1	2	3	3	17
400	1	2	4	6	8	45
600	2	4	6	8	10	78
800	3	5	8	11	14	97
1000	4	7	11	14	18	134
1200	4	9	13	18	23	162
1400	5	10	16	21	27	181
1600	6	12	18	25	31	203
1800	7	14	21	28	36	236
2000	8	16	24	32	40	277

We execute *IRALG* based on the results generated by our *IDALG*. Table 3 shows the execution time records of *IRALG* with different *NI* and *RRM*. We can see that execution time of our resolution algorithm increases as *NI* and *RRM* become larger. However, even in the worst case the running time is still pretty efficient. Another observation is that although from the time complexity analysis, it seems that *IRALG* takes more execution time than *IDALG*, the actual execution time result is contrary to the expectation.

The reason is that although the time complexity of *IRALG* is $O(kmn\log(n))$, the actual impact factors of the execution time of *IRALG* are k which is the number of inconsistencies, m' which is the number of vertexes involved in the insecure backtracking role mapping path, and n' which is the number of edges involved in the *IBRMP*. Because of $m' << m$ and $n' << n$, the actual execution time is much less than expected.

The number of resolved inconsistencies is not presented here, since the result only reflects the mock test data, and the result is pretty simple. The number of resolved inconsistencies is the same as the one detected by *IDALG*, i.e. all of the detected consistencies are resolved by our *IRALG*. The detailed resolved information is presented in Case Study section with real application data.

The number of resolved inconsistencies is not presented here, since the result only reflects the mock test data, and the result is pretty simple. The number of resolved inconsistencies is the same as the one detected by *IDALG*, i.e. all of the detected consistencies are resolved by our *IRALG*. The detailed resolved information is presented in Case Study section with real application data.

6. Case Study

The purpose of case study is to show the effectiveness of

Table 4 Access control information of four systems.

Systems	*NR*	*NRH*	*RRH*	*NI*	*NRM*	*RRM*
FXS	1988	457	0.23	281	31	0.11
TATS	3970	1349	0.34	403	36	0.09
GMS	3012	934	0.31	323	39	0.12
RPS	1131	215	0.19	155	11	0.07
Summary	10101	2955	0.27	1162	117	0.097

our approach in practical applications. Our approach has been applied in a global bank State Street Corporation. Inconsistency management for distributed interoperation is a part of the bank's enterprise application integration (EAI) project, whose purpose is to integrate the legacy enterprise distributed systems as a seamless combination. We present four major systems involved in the EAI project:

- FXS: a foreign exchange order management system.
- TATS: a trading audit trial system.
- GMS: a global software management system.
- RPS: a resource planning system.

The access control information of the above four systems is listed in Table 4, from which we can see that the role hierarchy ratio is less than 0.4 and the average value is 0.27. The role mapping ratio is less than 0.2 and the average value is 0.097. Both *RRH* and *RRM* are less than the values we set in simulated test.

The practical application data of our approach are listed in Table 5. Due to the space limitation, we only list 10 records of the detection and resolution. And we define the following acronyms:

- *DIN* means detected inconsistency number.
- *RIN* means resolved inconsistency number.
- *DRMN* is the number of deleted role mapping.

From Table 5 (the unit of Time is millisecond), we could see that, our detection method still performs much more efficient than *MIDALG*. which takes more than 7 seconds to do a cycle of checking, while *IDALG* only needs less than 0.5 second. The security checking is not a one time job, every time the security policy is changed, the inconsistency checking will be performed. The more efficient the detection algorithm is, the more frequently the checking can be performed. Besides detection, the resolution algorithm could resolve all the practical inconsistencies efficiently with a low cost.

7. Conclusions

We propose an inconsistency resolution method based on a new concept, insecure backtracking role mapping. Via analyzing the role graph, we prove that the root cause of security inconsistency in distributed interoperation is the existence of insecure backtracking role mapping. To detect the inconsistency, a novel and efficient algorithm is presented, which will not only report the existence of inconsistency, but also generate the inconsistency information for future resolution. We reduce the inconsistency resolution problem to the known Minimum-Cut problem, and propose an inconsistency resolution algorithm which could guarantee the security of distributed interoperation.

The research on the security consistency concerning the access control constraints in the interoperation will be our future work.

Table 5 Results of applying our approach to practical systems.

	MIDALG		*IDALG*		*IRALG*		
NO	Time*	*DIN*	Time*	*DIN*	Time*	*RIN*	*DRMN*
1	7684	3	432	3	24	3	5
2	7201	1	381	1	23	1	2
3	7535	2	452	2	22	2	2
4	7891	2	437	2	23	2	5
5	7603	1	398	1	23	1	2
6	7493	3	454	3	28	3	4
7	7725	1	432	1	19	1	1
8	7697	3	455	3	29	3	5
9	7402	1	389	1	21	1	2
10	7239	2	392	2	22	2	5

*Time: unit is milliseconds

Acknowledgements

This research is sponsored by StateStreet Corporation and conducted at StateStreet Zhejiang University Technology Center. The content of this paper is the opinions and conclusions of the authors only and do not necessarily represent the position of StateStreet Corporation and its subsidiaries, affiliates, officers, directors or employees.

References

[1] P. Centonze, G. Naumovich, and S.J. Fink, "Role-based access control consistency validation," Proc. International Symposium on Software Testing and Analysis, pp.121–132, 2006.

[2] D.F. Ferraiolo, R. Chandramouli, G. Ahn, and S.I. Gavrila, "The role control center: Features and case studies," Proc. 8th ACM Symposium on Access Control Models and Technologies, pp.12–20, 2003.

[3] R. Sandhu, "Role based access control," Adv. in Computer Science, vol.48, no.1, pp.38–47, 1998.

[4] N. Li and M.V. Tripunitara, "Security analysis in role-based access control," ACM Trans. Inf. Syst. Secur., vol.9, no.4, pp.391–420, 2006.

[5] D.F. Ferraiolo, R. Sandhu, and S. Gavrila, "Proposed NIST standard for role-based access control," ACM Trans. Inf. Syst. Secur., vol.4, no.1, pp.224–274, 2001.

[6] W. Essmayr, S. Probst, and E. Weippl, "Role-based access controls: Status, dissemination, and prospects for generic security mechanisms," Electronic Commerce Research, vol.4, no.1, pp.127–156, 2004.

[7] M. Shehab, E. Bertino, and A. Ghafoor, "SERAT: SEcure role mApping technique for decentralized secure interoperability," 10th ACM Symposium on Access Control Models and Technologies, pp.159–167, 2005.

[8] A. Schaad, J. Moffett, and J. Jacob, "The role-based access control system of a European bank: A case study and discussion," Proc. 6th ACM Symposium on Access Control Models and Technologies, pp.3–9, 2001.

[9] S. Basit, J. James, and B. Elisa, "Secure interoperation in a multidomain environment employing RBAC policies," IEEE Trans. Knowl. Data Eng., vol.17, no.11, pp.1557–1577, 2005.

[10] E.C. Lupu and M. Sloman, "Conflicts in policy-based distributed

systems management," IEEE Trans. Softw. Eng., vol.25, no.6, pp.852–869, 1999.

[11] L. Chen and J. Crampton, "Inter-domain role mapping and least privilege," Proc. 12th ACM Symposium on Access Control Models and Technologies, pp.157–162, 2007.

[12] L. Gong and X. Qian, "Computational issues in secure interoperation," IEEE Trans. Softw. Eng., vol.22, no.1, pp.43–52, 1996.

[13] D.D. Sleator and R.E. Tarjan, "A data structure for dynamic trees," Proc. 13th Annual ACM Symposium on Theory of Computing, STOC '81, pp.114–122, ACM, New York, NY, 1981.

[14] H.J. Huang and H. Kirchner, "Secure interoperation in heterogeneous systems based on Colored Petri Nets," (to appear), available from http://hal.inria.fr/docs/00/39/69/52/PDF/conflicts.pdf, 2009.

[15] X. Wang, J. Sun, X. Yang, C. Huang, and D. Wu, "Security violation detection for rbac based interoperation in distributed environment," IEICE Trans. Inf. & Syst., vol.E91-D, no.5, pp.1447–1456, May 2008.

[16] C. Huang, J. Sun, X. Wang, and Y. Si, "Security policy management for systems employing role based access control model," Information Technology J., vol.8, no.5, pp.726–734, 2009.

[17] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Intoduction to Algorithms, Second ed., MIT Press, 2005.

Appendix A: Proof of Theorem 1

The relationship between the two roles R_i and R_j in the same single domain role system can be categorized into three situations:

(1) R_i is a senior role to R_j
(2) R_j is a senior role to R_i
(3) There is no hierarchy relation between R_i and R_j, which means R_i and R_j are independent roles.

After adding role mappings for the multi-domain interoperations, if there are security inconsistencies, for situation 1) and 2), since the autonomy principle can be guaranteed in the process of constructing the role mapping, thus the hierarchy relation will still exist after the interoperation, the only insecure situation is that junior role obtains the permission of its senior role, i.e. founded inconsistency. For situation 3), the only insecure situation is that the hierarchy relation between R_i and R_j comes into being, which doesn't exist before interoperation. This kind fits into the unfounded security inconsistency. To sum up, Theorem 1 is proved.□

Appendix B: Proof of Theorem 2

(1) *Sufficiency*: according to Definition 8, the existence of insecure backtracking role mapping path leads to the security inconsistency consequently.
(2) *Necessity*: according to Theorem 1, in the global multi-domain environment, there are only two kinds of security inconsistencies, i.e. unfounded and founded. For both of the inconsistencies, there must be a single role system G_i, in which there are two different role nodes u and v. In G_i, u and v are not connected, i.e. no path exists between u and v. On the other side, in global role system G, u and v are connected, i.e. path (u, v) exists, and path (u, v) must go through at least one individual role system G_j, otherwise, u and v should be connected in G_i. From G_i to G_j, role mapping edge is the only way, thus path (u, v) contains at least one role mapping edge. Thus path (u, v) with the start and end nodes in the some domain contains the role mapping edge. According to Definition 8, (u, v) is the insecure backtracking mapping path. Given the above, theorem is proved.□

Appendix C: Proof of Theorem 3

Assuming the sets of insecure backtracking role mapping path in minimal role graph and global role system graph are *minIBRMP* and *gloIBRMP* respectively. Then Theorem 3 can be converted to prove that *minIBRMP* equals to *gloIBRMP*.

(1) Based on the definition of *minIBRMP*, we have *minIBRMP* $\subseteq$ *gloIBRMP*, since minimal role graph is constructed from global role graph without adding any extra role, role hierarchy or role mapping.
(2) Hypothesizing that *gloIBRMP* $\nsubseteq$ *minIBRMP*, and assuming the insecure backtracking role mapping (u, v), is the one $(u, v) \in$ *gloIBRMP* and $(u, v) \notin$ *minIBRMP*. According to the definition of *IBRMP*, there must be a role mapping edge (u', v') on the path (u, v). According to the construction of minimal role graph G', $u' \in V'$ and $v' \in V'$. It is connected from u to u' (since it's on the same path), and there is only role hierarchy or role mapping to connect u and u', if the connecting edge is role mapping, then $u \in V'$ and $(u, u') \in E'$; if the connecting edge is role hierarchy edge, then $(u, u') \in H_i^+ \Rightarrow (u, u') \in E'$. The similar proof could be given to (v', v). Thus all the vertexes on path (u, v) are included in V' and all the edges on path are included in E'. According to Definition 9, $(u, v) \in$ *minIBRMP*, which violates the assumption. The hypothesis does not hold, *gloIBRMP* $\subseteq$ *minIBRMP*.

Thus *minIBRMP* $\subseteq$ *gloIBRMP* and *gloIBRMP* $\subseteq$ *minIBRMP* infer *minIBRMP* $=$ *gloIBRMP*. Theorem is proved.□

Chao Huang received the B.E. degree in the department of Computer Science and Engineering from Zhejiang University (China) in 2005. He is a research assistant and Ph.D. candidate in Zhejiang University now. His primary research interests include distributed software architecture, and enterprise security architecture.

Jianling Sun received the B.E. and D.E. degrees of Computer Application from Zhejiang University (China) in 1990 and 1993 respectively. Since October 1993, he has been a faculty member of the College of Computer Science and Engineering at Zhejiang University. His current research interests include database systems, software engineering,and software architecture.

Xinyu Wang received the B.E. degree in the department of Computer Science and Engineering from Zhejiang University (China) in 2002. From 2002 to 2007, he was a research assistant in the Zhejiang University, where he got the D.E. degree from the college of Computer Science. Since the June 2007, he has been the lecturer in the same university. His primary research interests include Software engineering, distributed software architecture and distributed computing. He is a member of IEEE CS.

Di Wu received the B.E. degree in the department of Computer Science and Engineering from Zhejiang University (China) in 2001. He received his Ph.D. degree of computer science at Department of Computer Science and Engineering, Zhejiang University in 2006. Since 2007, he has been a software engineer in the State Street Technology Zhejiang. His primary research interests include distributed software architecture, distributed computing and enterprise security architecture.