Semantics-Based Approach for Dynamic Evolution of Trust Negotiation Protocols in Cloud Collaboration

Seung Hwan Ryu[1,⋆], Abdelkarim Erradi[1], Khaled M. Khan[1], Saleh Alhazbi[1], and Boualem Benatallah[2]

[1] Department of Computer Science & Engineering,
Qatar University, 2713, Doha, Qatar
{deepryu@gmail.com, erradi,k.khan,salhazbi@qu.edu.qa}
[2] School of Computer Science & Engineering,
University of New South Wales, Sydney, NSW, 2051, Australia
boualem@cse.unsw.edu.au

Abstract. Many techniques for addressing trust negotiation issues is little concerned with managing the dynamic evolution of trust negotiation protocols (policies), particularly in cases where there exist ongoing negotiations when a protocol has been changed. We propose an approach that automatically determines how consequences of changing a protocol affect ongoing negotiations. In particular, our approach allows to capture the semantics and intention of protocol changes, memorize and apply them in effectively analyzing the impact of protocol changes on negotiations.

Keywords: Trust Negotiation, Semantics, Content, Evolution, Protocol.

1 Introduction

Collaboration environments have been widely adopted for diverse domains, from scientific domains to end-user communities on the Web. Recently the resources sharing among people in collaborative environments have been managed using cloud computing platforms [1]. In cloud collaboration environments, making access control decisions to resources managed in cloud platforms is a hard task because of the size and dynamics of the users [2,3].

Trust negotiation has been proposed as a viable authorization solution for addressing the issue [7,3]. A *trust negotiation protocol*[1] describes a negotiation process between negotiation parties, in the sense that it specifies which credentials (e.g., digital versions of passports or credit cards) a service provider and users should exchange for the users to access protected resources [6].

Although existing approaches for addressing trust negotiation issues have made significant progress (see [3] for a recent survey), little work has been done on the problem of *dynamic protocol evolution*, which refers to managing the ongoing negotiations when an existing protocol has been changed. In

⋆ Most of the work was done when the author was as a postdoc at Qatar University.
[1] In this paper we use "trust negotiation protocol" and "protocol" interchangeably.

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 518–526, 2014.

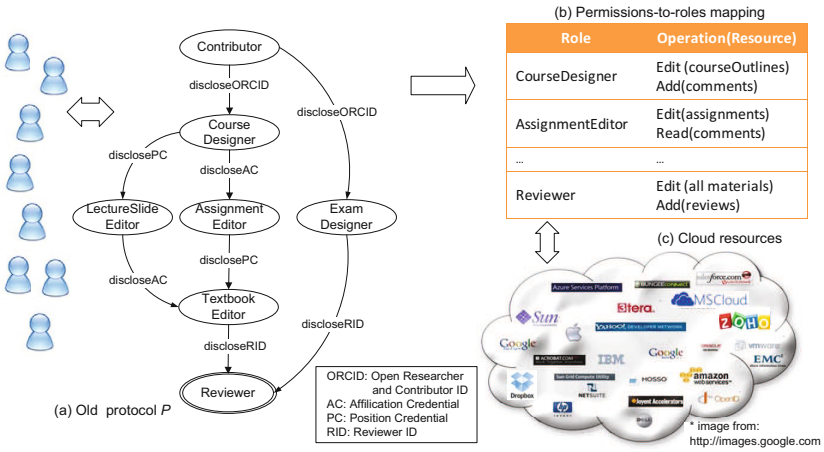

Fig. 1. Protocol P for an education material co-authoring service in cloud collaboration environments. Credentials are disclosed when exchanging messages.

dynamic collaborative environments, trust negotiation protocols can constantly over time because the collaboration contexts change. To tackle the problem, previous works [4,5] proposed techniques that analyze how ongoing negotiations (instances) are impacted by protocol changes to determine the successful migration of negotiations, e.g., migration to a new protocol. For this, the authors focused on *protocol level constraints* on the message (or credential) sequences exchanged between negotiation parties. In the constraints, they have *not* considered the actual message contents exchanged in the message sequences.

What is missing is a technique that takes into account *both* the sequence of messages and their contents as they are *interdependent* and *interrelated*. The consideration of message contents between sequences would be beneficial for improving the rate of successful replacements, which is critical in minimizing discomfort and disruptions to active negotiations. In this paper, we extend the previous works towards more comprehensive management of the dynamic protocol evolution. In particular, we make the following contributions:

- We present *composite change operators* that allow to express the semantics behind applied changes, i.e., a changed message sequence is semantically equivalent with an original one in terms of message contents (Section 3).
- We show how to *enhance* the change impact analyses of previous works by considering the change semantics (Section 4).
- We present the promising results that show we could achieve up to 89% *successful migration improvement*, compared with the prior works (Section 5).

2 Preliminaries

In what follows, we explain the protocol model for representing trust negotiation protocols and then present an example scenario.

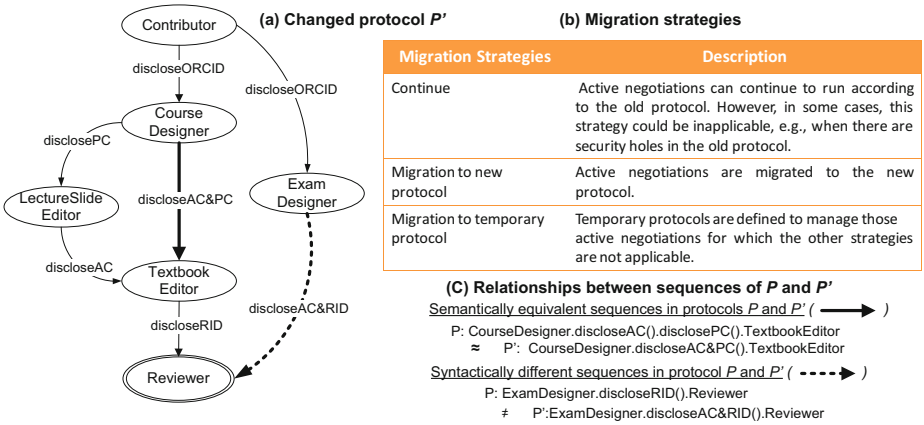

Fig. 2. Changed protocol *P'* for the education material co-authoring service and some migration strategies applicable to ongoing negotiations

2.1 Trust Negotiation Protocols Modeling

Following previous works [4,6], we model protocols as a finite state machine (FSM). FSM consists of states and transitions. *States* represent the levels of trust that a service provider establishes during its interaction with clients. As in [6], we map permissions (privileges) for accessing certain resources (e.g., service operations) to specific roles (e.g., researchers or administrators), instead of individual users, and then map such roles to states. Thus, once a client reaches at a state, she can access some resources with the role associated with the state. On the other hand, *transitions* are triggered by messages sent by clients to the provider. In our model, states are labeled with an assigned role while transitions with a message, corresponding to the invocation of a service operation.

2.2 An Example: Education Material Co-authoring Collaboration

Consider an education community in cloud collaboration environments. The community users from different countries, such as lecturers, senior lecturers, and professors, could collaboratively work with each other and share their knowledge and experiences in preparing education materials (e.g., lecture slides, assessments, online textbooks, etc). These education materials are (i) stored and managed in the cloud resources, such as Google docs, Dropbox, etc; (ii) shared and reused by the community users for their teaching purposes. The education community provides an education material co-authoring service that consists of several operations, which allow the collaborators to access the education resources in the cloud. The collaborators can invoke different service operations, based on the levels of trust established between the service provider and users.

Figure 1(a) shows a trust negotiation protocol for the co-authoring service. The protocol P states that any user is initially in the Contributor state (role). From there, after providing the ORCID (Open Researcher and Contributor ID),

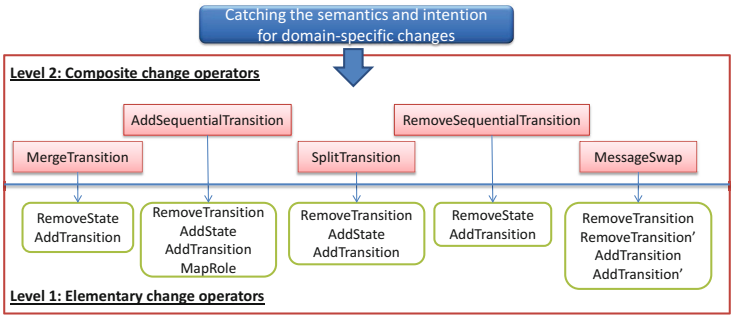

Fig. 3. Layered Protocol Change Operators

lecturers can proceed to state TextbookEditor by sending their position credential (e.g., User.position=Lecturer) and affiliation credential (e.g., User.organization= QU), while senior lecturers proceed to the same state by sending the credentials in the opposite order. In addition, professors can proceed to state ExamDesigner by disclosing their ORCID. Finally, the contributors disclose the reviewer ID (RID) to proceed to state Reviewer and then can access and review all the education materials by invoking the "Edit(all materials)".

3 Layered Change Operators for Evolving Protocols

This section describes two layered protocol change operators, particularly the operators used for catching the semantics and intention of protocol changes.

3.1 Elementary and Composite Change Operators

We distinguish two types of change operators, based on different levels of granularity of changes: elementary and composite change operators (Figure 3). The elementary change operators are used for generic and fine-granular protocol changes, e.g., adding a state or a transition. Such operators have been suggested in the past [4,6]. However, it is "not" possible to represent the semantics and intention behind applied changes using the operators. To fill this gap, we propose the composite change operators that can be applied for making some domain or service-specific changes. For example, if a protocol manager swaps the order in which she receives credentials in a given protocol, she still wants to make sure that both credentials have been received, regardless of their disclosed sequence. The composite change operators are derived by grouping elementary change operators that are executed in sequence.

MergeTransition (State s, State t, Messages $m_1, ..., m_n$): This operator merges n transitions with messages $m_1, ..., m_n$ respectively into a single transition with one message m. It can be applied when (i) a protocol manager wants to modify a message sequence in protocol P, which consists of several messages required to proceed from state s to state t, into another sequence in protocol P', which consists of only one single message; (ii) she regards the sequences as equivalent, from the semantic point of view, due to their contents.

Example 1. Consider the old protocol (Figure 1(a)) and the new protocol (Figure 2(a)). Assume that a protocol manager changes the sequence (CourseDesigner. discloseAC().disclosePC().TextbookEditor) of protocol P to the sequence (CourseD esigner.discloseAC&PC().TextbookEditor) of protocol P' by merging two messages into one message. She applies the composite operator "MergeTransition" to express the intention on that the two sequences are equivalent semantically.

SplitTransition (State s, State t, Message m): This operator splits a transition with message m into n transitions with $m_1, ..., m_n$. It is applied in the opposite case of the situation in which the "MergeTransition" operator is applied.

AddSequentialTransition(State s, State t, Message sm): This operator adds sequentially a transition with message sm between source state s and target state t. It could be used when protocol P' requires an extra message from s to t, which protocol P does not support.

RemoveSequentialTransition (State s, State t, Message sm): This operator removes a sequential transition with message sm between source state s and target state t. It is applied when protocol P needs to receive two messages for clients to access target state t from source state s while protocol protocol P' only requires one of them to grant the access to the same state.

MessageSwap (State s, State t, Message $m1$, Message $m2$) This operator swaps two messages m_1 and m_2 between two states s and t. It is applied in situations where, though the order of exchanged messages is swapped, a protocol manager only makes sure that two credentials has been submitted, regardless of the order.

4 Analysis Considering Both Message Sequences and Their Contents

In contrast to the previous works that only rely on the old and new protocols for syntactically comparing message sequences in the change impact analysis, we exploit the semantic equivalence between sequences in terms of message contents.

4.1 Compatibility Properties as Migration Decision Points

As requirements for determining whether an ongoing negotiation is migrateable, we identify two different degrees of compatibility: sequence and credential compatibility. Sequence compatibility is used for detecting the *syntactic equivalence* between two message sequences, irrespective of their message contents (i.e., disclosed credentials), while the credential compatibility for identifying the *semantic equivalence* between two different sequences, based on the message contents. The compatibility properties are further divided as follows:

- *Forward Sequence Compatibility (FSC)* means that the correct interaction of active negotiations with a given service should be guaranteed after they are migrated to a given new protocol.

  Example 2. In the old and new protocols *P, P'*, if a negotiation in state CourseDesigner of protocol P is migrated to the same state of protocol P', a violation of FSC might occur as it could fail to interact with the changed sequence (path): (CourseDesigner.discloseAC&PC().discloseRID().Reviewer).

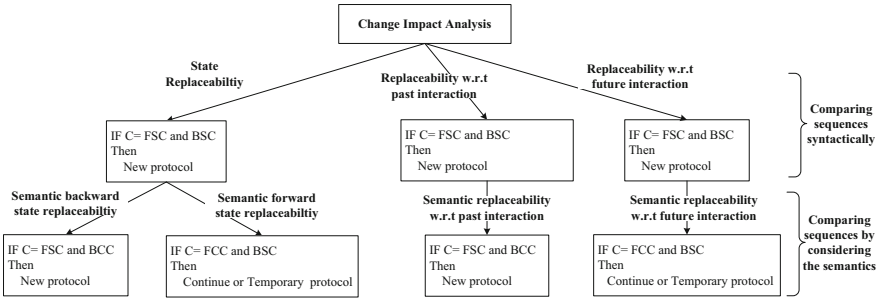

Fig. 4. Replaceability classes and their corresponding migration rules

- *Backward Sequence Compatibility (BSC)* means that, when a negotiation is migrated to the new protocol, the message sequence (followed by the negotiation so far) must be compatible in the context of new protocol.
  Example 3. In protocols *P*, *P'*, assume there is a negotiation in state TextbookEditor of protocol *P*. If the negotiation has followed the path (Contributor.discloseORCID().discloseAC()...TextbookEditor), a violation of BSC occurs as the sequence is not acceptable by the new protocol *P'*.
- *Forward Credential Compatibility (FCC)* means that a negotiation can disclose all required credentials when it is migrated to a new protocol, even though it is not guaranteed to correctly interact with a given service.
  Example 4. In Example 2, the negotiation is not guaranteed to correctly interact with the changed path, but it satisfies the FCC property as it can send all the required credentials {AC, PC, RID} in the context of protocol *P'*.
- *Backward Credential Compatibility (BCC)* means that a negotiation has already disclosed all required credentials, even if it followed the message sequence that is incompatible in the context of new protocol.
  Example 5. In Example 3, while the negotiation has followed the path incompatible with the new protocol, it satisfies the BCC property as it already sent all the credentials {ORCID, AC, PC} required in protocol *P'*.

4.2 Analyzing Change Impacts by Different Replaceability Classes

The change impact analysis is based on the notion of replaceability (that is, determining under what circumstances a new protocol can replace an old one to satisfy the above compatibility properties). Each replaceability class can be represented as a migration rule: if [condition] then [conclusion] (Figure 4). Here, the condition part corresponds to the compatibility properties and the conclusion part to the possible migration strategies, with meaning that all negotiations satisfying the properties are handled with the specified strategies.

State-based Replaceability When a new protocol can replace an old protocol, all negotiations can be safely migrated to the new protocol [5]. If protocols are not replaceable, we look at the current states of negotiations as follows.

State replaceability: This analysis class determines the change-transparent states where their forward paths (message sequences from the states to the final state) and backward paths (message sequences from the initial state to the states) are all the same in old and new protocols. As a migration strategy, we can migrate all the negotiations in these states to the *new* protocol as they satisfy both of the FSC and BSC properties, e.g., in Figure 1(a), like state LectureSlideEditor.
Semantic backward state replaceability: Though a certain state is affected by some changes, we can regard it as a semantically replaceable state, if all the changes are the semantic ones, meaning that the changed sequences are semantically equivalent with the original ones in terms of their message contents. For example, in Figure 1(a), this analysis returns the state TextbookEditor.
Semantic forward state replaceability: Unlike the previous semantic analysis, this analysis determines the states that have the same backward paths, but the *different forward paths* (all of the different paths are changed semantically). Note that the negotiations in such states cannot be migrated directly to the new protocol as they may not be guaranteed to interact with the semantically changed path. As possible migration strategies, they can continue to run under the old protocol, if it is acceptable.

Interaction Path-based Replaceability Note that there are some situations where we cannot determine the successful migration of negotiations at the state level, e.g., when migrating negotiations in a certain state might cause the violations of compatibility properties. In this case, to extract the migrateable ones from such a state, we further look at their past and future interactions as follows.
Replaceability with respect to a past interaction: This analysis is performed on the negotiations in the states that have the same forward paths, but *different backward paths* (one of them is changed *syntactically* with the elementary operators). Given a negotiation, the analysis checks whether its past interaction is compatible in the context of new protocol. For example, in protocols *P, P'*, among the negotiations in the state Reviewer, we extract the ones that only followed the *unaffected* backward path (Contributor.discloseORCID().discosePC().discloseAC().discloseRID().Reviewer), since they satisfy the FSC and BSC properties.
Semantic replaceability with respect to a past interaction: Even though a certain negotiation followed an affected backward path, this analysis classifies it as a migrateable one, if it took one of the semantically changed backward paths.
Replaceability with respect to a future interaction: After all the previous analyses, there remain negotiations in certain states, which followed compatible backward paths, but are *not* guaranteed to correctly interact with the new protocol. To identify the negotiations that will take the unaffected forward paths, we can infer the possible future interactions of them by applying some data mining techniques to the past interactions of negotiations (see [4] for the details).
Semantic replaceability with respect to a future interaction: Though a certain negotiation is expected to follow some affected forward paths, this analysis further examines whether it will only take one of the semantically changed paths. If so, the negotiation can be migrated to a temporary protocol.

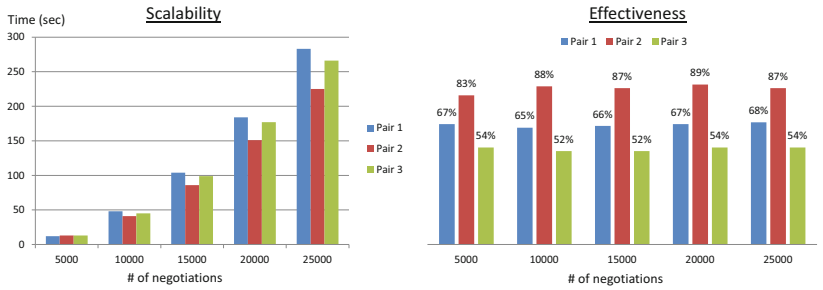

Fig. 5. Evaluation results. X-axis represents the number of simulated negotiations. Y-axis represents the time (sec) taken for the replaceability analysis in Figure 5(a) and the successful migration rate improvement in Figure 5(b).

5 Evaluation

We now present the evaluation results that show how our system can be effectively utilized in the dynamic evolution of protocols.

Evaluation Methodology: We evaluated the performance of change impact (replaceability) analysis in terms of the scalability and effectiveness. For this, we defined three pairs of protocols (each pair consisting of old and new protocols) with different number of states. For example, in pair one, an old protocol had 18 states and 19 transitions and a new protocol 16 states and 17 transitions. We populated the system with a number of artificial negotiations (e.g., 5k, 10k, etc).

Results: Figure 5(a) shows the time taken to perform the replaceability analyses from Section 4.2 for the three pairs of protocols. For example, for 15k negotiations, it took about 100 seconds in performing all the replaceability analyses on the negotiations and determining which ones are migrateable. As we can see from the figure, the time taken to complete the analyses grows linearly with respect to the number of negotiations. In the second evaluation, we measured how much the rate of successful migration could be improved by considering the change semantics as an additional knowledge in the replaceability analysis. Figure 5(b) shows the improvement rate for the protocol pairs. For instance, for 20k negotiations simulated in the pair two, we could obtain the improvement rate 89% ($\frac{(91-48)}{48} = 89$, where the 48% is the successful migration rate from the previous works and the 91% is the rate from this work). In the figure, we can see that we would achieve better migration rate by taking into account the semantics behind protocol changes.

6 Conclusion

This paper proposed an approach for considering both message sequences and their contents in managing the dynamic protocol evolution problem. Particularly, we presented composite change operators for expressing the semantic equivalence between message sequences. We also proposed the change impact analysis that considers the change semantics as an additional knowledge.

Acknowledgments. This publication was made possible by a grant from the Qatar National Research Fund; award number NPRP 7-481-1-088. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of QNRF.

References

1. Chard, K., Bubendorfer, K., Caton, S., Rana, O.F.: Social cloud computing: A vision for socially motivated resource sharing. IEEE T. Services Computing 5(4), 551–563 (2012)
2. Lee, A.J., Winslett, M., Basney, J., Welch, V.: The traust authorization service. ACM Trans. Inf. Syst. Secur. 11(1) (2008)
3. Noor, T.H., Sheng, Q.Z., Zeadally, S., Yu, J.: Trust management of services in cloud environments: Obstacles and solutions. ACM Comput. Surv. 46(1), 12 (2013)
4. Ryu, S.H., Casati, F., Skogsrud, H., Benatallah, B., Saint-Paul, R.: Supporting the dynamic evolution of web service protocols in service-oriented architectures. ACM Transactions on the Web 2(2) (2008)
5. Skogsrud, H., Benatallah, B., Casati, F., Toumani, F.: Managing impacts of security protocol changes in service-oriented applications. In: ICSE (2007)
6. Skogsrud, H., Nezhad, H.R.M., Benatallah, B., Casati, F.: Modeling trust negotiation for web services. IEEE Computer 42(2) (2009)
7. di Vimercati, S.D.C., Foresti, S., Jajodia, S., Paraboschi, S., Psaila, G., Samarati, P.: Integrating trust management and access control in data-intensive web applications. TWEB 6(2), 6 (2012)