PERSONALIZED CREDENTIAL NEGOTIATION BASED ON POLICY INDIVIDUALIZATION IN FEDERATION

A thesis submitted
to Kent State University in partial
fulfillment of the requirements for the
degree of Master of Science

by
Kailas B. Bobade

December 2009
Thesis written by
Kailas B. Bobade

M.S., Kent State University, USA, 2009

B.S., Dr. BAMU, India 2001

Approved by

Dr. Javed I. Khan______________, Advisor

Dr. Robert A. Walker__________, Chair, Department of Computer Science

Dr. John R.D. Stalvey__________, Dean, College of Arts and Sciences
# TABLE OF CONTENTS

LIST OF FIGURES .................................................................................................................. VI

LIST OF TABLES ..................................................................................................................... XI

DEDICATION .......................................................................................................................... XII

ACKNOWLEDGEMENTS ......................................................................................................... XIII

CHAPTER 1 INTRODUCTION ................................................................................................. 1

1.1 Definitions ......................................................................................................................... 3

CHAPTER 2 PROBLEM FORMULATION .............................................................................. 5

2.1 Objective ............................................................................................................................ 6

2.2 Negotiation System Model ............................................................................................... 7

   2.2.1 Centralized Negotiation System ................................................................................. 8

   2.2.2 Federated Negotiation System ................................................................................... 9

   2.2.3 Libertarian Negotiation System ............................................................................... 9

   2.2.4 Community Negotiation System ............................................................................ 10

2.3 Related Work .................................................................................................................... 11

2.4 Overview of Approach ...................................................................................................... 12

CHAPTER 3 PROPOSED SOLUTION .................................................................................. 14

3.1 Negotiation Process Model .............................................................................................. 14

3.2 Negotiation Protocol ........................................................................................................ 17

   3.2.1 Negotiation Message Exchange Sequence ............................................................ 18

   3.2.2 Negotiation Message Structure .............................................................................. 20
3.3 Architecture ....................................................................................................................................22
  3.3.1 Resource Vault............................................................................................................................23
  3.3.2 Policy Vault..................................................................................................................................24
  3.3.3 Negotiation System.......................................................................................................................25

3.4 Negotiation Algorithms ..................................................................................................................26
  3.4.1 Stateless Eager Attribute Negotiation (SEAN) algorithm.........................................................27
  3.4.2 Stateless Proxy Attribute Negotiation (SPAN) Algorithm .......................................................31

3.5 Efficient Negotiation Strategy .......................................................................................................39
  3.5.1 Sequential Negotiation ...............................................................................................................40
  3.5.2 Parallel Negotiation ....................................................................................................................41

CHAPTER 4 EXPERIMENTAL SETUP AND ANALYSIS.................................................................42

4.1 Examples - SEAN Algorithm ........................................................................................................43
  4.1.1 Successful Negotiation ..............................................................................................................45
  4.1.2 Unsuccessful Negotiation .........................................................................................................46
  4.1.3 Negotiation Results ....................................................................................................................47

4.2 Examples - SPAN Algorithm ..........................................................................................................49
  4.2.1 Proxy Negotiation ......................................................................................................................49
  4.2.2 Proxy Negotiation with backtracking .......................................................................................51
  4.2.3 Negotiation Results ....................................................................................................................53

4.3 Examples – KBAN Strategy ............................................................................................................54
  4.3.1 Sequential KBAN ......................................................................................................................55
LIST OF FIGURES

Figure 1. A Federation with three organizations namely A, B, and C. Each organization exchanges information using Communication point .................................................. 3

Figure 2. In Centralized Credential System members of every organization, in green color, stores credentials at centralized server shown in black color ......................... 8

Figure 3. In Federated Credential System every organization has federation Server, shown in black color, where members store their credentials. Federation Server acts as Negotiation Agent on behalf members ................................................................. 9

Figure 4. In Libertarian Credential System each member of every organization stores own credentials and negotiate with other members without mediator ......................... 10

Figure 5. In Libertarian Credential System each member of every organization stores own credentials and negotiate with other members without mediator ......................... 11

Figure 6. A) State Transition Diagram representing states of negotiation of an attribute.

B) States in blue are considered for protocol and algorithm given in this paper.

Protocol and algorithm are extensible for states in green ....................................... 15

Figure 7. State Transition Diagram for Negotiation Protocol ........................................ 19

Figure 8. Negotiation Message .................................................................................. 21

Figure 9. Negotiation Enabled Framework .................................................................. 23
Figure 10. A) First row has SEAN_negotiator routine which receives message from and send message to opposite party. B) Second row has SEAN_state_machine routine which changes negotiation state of attributes and also generates state of negotiation i.e. DEAL, NO_DEAL or NEGOTIATION. C) Third row has SEAN_rule_resolver routine which decides if a attribute can be released.............................................28

Figure 11. Driver Process and Resolve Process in SPAN ..................................................33

Figure 12. Dependency Graph Generator in SPAN ............................................................38

Figure 13. Offered Set routine in SPAN ...........................................................................38

Figure 14. Access Control Policies of ABC Inc/John, CDE Inc/Yang, KLM Inc/Bob, Alice, Sajid, and Pooja respectively from left-right and top-bottom .........................43

Figure 15. A) Messages exchanged during negotiation between Bob and Alice to schedule an Interview using Eager strategy. Messages are identified by numbers from 1-6. B) At the end of Negotiation, Agent will generate list of resources received for KLM/Bob. C) At the end of Negotiation, Agent will generate list of resources received for Alice. .................................................................45

Figure 16. A) Messages exchanged during negotiation between Bob and Sajid to schedule an Interview using Eager strategy. Messages are identified by numbers from 1-4. B) At the end of Negotiation, Agent will generate list of resources received for KLM/Bob. C) At the end of Negotiation, Agent will generate list of resources received for Sajid.................................................................47
Figure 17. It shows resources received by three managers’ namely Yang, John, Bob, and three students’ namely Alice, Pooja, and Sajid from left-right, top-bottom.

Figure 18. A) Messages exchanged during a negotiation between ABC Inc and Alice to schedule an interview using Proxy strategy. Messages are shown from 1-12. B) Dependency Graph resulted from negotiation. C) At the end of Negotiation; each SA will store resources information for its user indicating attributes released and received.

Figure 19. A) Messages exchanged during a negotiation between KLM Inc and Pooja to schedule an interview using Proxy strategy. Messages are shown from 1-12. B) Dependency Graph resulted from negotiation. C) At the end of Negotiation; each SA will store resources information for its user indicating attributes released and received.

Table 4. SPAN Negotiation Result

Figure 20. It shows resources received by three students’ namely Alice, Pooja, Sajid as well as three managers’ namely Yang, John, and Bob from left-right, top-bottom.

Figure 21. Messages exchanged during a negotiation between ABC Inc and Pooja to schedule an Interview using Proxy strategy based on sequential KBAN.

Figure 22. Messages exchanged during three negotiations between Alice-John, Pooja-John, and Sajid-John (one negotiation per row in each message) to schedule an Interview using Proxy strategy based on Parallel KBAN.
Figure 23. Messages exchanged in a negotiation between KLM Inc and Pooja to schedule an Interview using SEAN.

Figure 24. Messages exchanged in a negotiation between ABC Inc and Pooja to schedule an Interview using SEAN.

Figure 25. Messages exchanged in a negotiation between ABC Inc and Sajid to schedule an Interview using SEAN.

Figure 26. Messages exchanged in a negotiation between CDE Inc and Pooja to schedule an Interview using SEAN.

Figure 27. Messages exchanged in a negotiation between ABC Inc and Alice to schedule an Interview using SEAN.

Figure 28. Messages exchanged in a negotiation between CDE Inc and Sajid to schedule an Interview using SEAN.

Figure 29. Messages exchanged in a negotiation between CDE Inc and Pooja to schedule an Interview using SEAN.

Figure 30. Messages exchanged in a negotiation between ABC Inc and Pooja to schedule an Interview using SPAN.

Figure 31. Messages exchanged in a negotiation between ABC Inc and Sajid to schedule an Interview using SPAN.

Figure 32. Messages exchanged in a negotiation between CDE Inc and Sajid to schedule an Interview using SPAN.
Figure 33. Messages exchanged in a negotiation between CDE Inc and Alice to schedule an Interview using SPAN ................................................................. 75

Figure 34. Messages exchanged in an unsuccessful negotiation between CDE Inc and Pooja to schedule an Interview using SPAN ....................................................... 76

Figure 35. Messages exchanged in an unsuccessful negotiation between KLM Inc and Sajid to schedule an Interview using SPAN .......................................................... 77

Figure 36. Messages exchanged in a negotiation between KLM Inc and Alice to schedule an Interview using SPAN ................................................................. 78
# LIST OF TABLES

Table 1. Resource Vault ........................................................................................................... 24
Table 2. Policy Vault .................................................................................................................. 25
Table 3. SEAN Negotiation Result ............................................................................................ 49
Table 4. SPAN Negotiation Result ............................................................................................. 53
Table 5. Knowledge Based Attributed Negotiation Results ......................................................... 59
Table 6. Negotiation Results between three organizations and three students ....................... 60
Table 7. Communication Complexity ......................................................................................... 61
DEDICATION

Dedicated to my parents, sister and her family
ACKNOWLEDGEMENTS

I would like to acknowledge and extend my heartfelt gratitude to the following person who made the completion of this thesis possible.

- Dr. Javed Khan
- Dr. Hassan Peyravi
- Most especially to my family and friends who always encouraged me

Kailas Bobade

June 6 2008, Kent, Ohio
CHAPTER 1

Introduction

An organization uses centralized resource access management solution which permits or denies resources to the members based on identity owned by the organization. Increasingly these members are accessing resources of other organizations and same is true for the members of other organizations. This separation of the members from the organization's centralized access management system is a result of decentralization nature of the Internet - an integral part of today’s personal and business life. Such decentralized system necessitates members of one organization to duplicate personal information at each resource provider organization. This information duplication poses privacy risk for the members as well as increases cost and complexity of identity management for the resource provider organizations.

These challenges’ have given rise to a new distributed approach for resource access management solution known as Federation. It enables the members of an organization to access the resources from resource provider organizations without the need for providing personal information with each. Federation has made it possible by enabling an organization to identify as well as authenticate member once and then using that information across the rest of organization - sign locally, access globally. In this process resource provider organization receives only the minimum required information about the members and thus federated system has increases privacy of an individual. Despite increase in privacy, members have very little say in when, how and to what
extent information about them is communicated to others. And one of the reasons is one-size-fits-all approach taken by the federated systems such as Shibboleth [15] and Active Directory Federation System (ADFS). In this thesis, information is divided in various categories and among them attribute and credential are used interchangeably.

Federated members can enjoy true privacy if every member could provide her own information release policies. Such Individualized policies inherently triggers credential exchange between federated members or between federated member and resource provider organization. In the thesis we are proposing Individualized policies for federated members and its inherent requirement – a negotiation. At the heart of negotiation are states - Request, Response, and Accept – of credential involved in negotiates. These states and transition between them is presented in Negotiation Process Model. Based on this model we are providing an extension to existing federation model which includes Negotiation Protocol, and Negotiation Enabled Framework. Negotiation protocol defines message structure to carry negotiation information between negotiating parties resulting in stateless negotiation. Negotiation Enabled framework is available at each negotiation participants side and executes negotiation process till it ends in deal or no-deal.

In real society, intrinsic variability of circumstances demands choosing appropriate negotiation flavor as per need of an Individual and same is true for the members of federation. For example a federated member might be interested in knowing the outcome of a negotiation in less number of communication steps at the cost of releasing more information; while other member might be more cautious in releasing
information at the cost of more number of communication steps. We have accommodated these needs of federated members by providing two negotiation flavors with supporting examples and analysis. In addition to these negotiation flavors, we will discuss efficient and asynchronous negotiation strategies called as Knowledge Based Attribute Negotiation with supporting examples and their analysis.

1.1 Definitions

**Organization** - An organization is an association of group of individuals that share re-sources, capabilities, and information to achieve a common objective.

**Federation** - A federation is an association of organizations which enables members of one organization to seamlessly access services of the rest of the

![Figure 1. A Federation with three organizations namely A, B, and C. Each organization exchanges information using Communication point](image)
organizations without redundant member administration. Figure 1 show federation of three organizations where Organization ‘A’ releases information about her member to Organization ‘B’ and released information is considered valid in ‘B’.

Attributes – A piece of information associated with an entity is referred as an attribute of that entity. For example - Name, Designation, Role are attributes of a person while metadata is an attribute of a digital document.

Access Control Policy (ACP) – A set of rules which permits or denies access to the resource based on requester’s attributes is referred as Access Control Policy for that resource. A rule is represented in the disjunctive normal form (DNF) as $S_1 \leftarrow C_1 V C_2 \ldots C_j$ where clause $C_1 = R_1 \Lambda R_2 \ldots \Lambda R_K$, which means resource $S_1$ will be released when either of clauses $C_1$, $C_2$, or $C_j$ is satisfied. Here, clause $C_1$ requires all the resources $R_1$, $R_2$ and $R_K$ from the resource requesting party. We also represent ACP for an attribute ‘A’ as $P_A \leftarrow F_A (C_1 C_2 \ldots C_j)$ where $F_A$ is an expression with attributes of opposite party only and must to be satisfied to release ‘A’.

Negotiation – The process of requesting and releasing attributes between two or more parties to reach an agreeable objective is called Negotiation. In this thesis, negotiation is limited between two parties.
CHAPTER 2

Problem Formulation

Federated model provides substantial advantage to the organizations. Often an organization does not have to create user base but rather tap into already existing massive user-base of other organizations. Organizations can temporarily or permanently form coalitions. In such federation, service provider organizations receive member’s attributes issued by her home organization. These federated systems like Shibboleth have claimed to increase privacy. In a way it is true, but private person is actually absent from such system so does individualized privacy. Currently, both in Shibboleth and in private arrangements- the exchange policy is set by organizations- not by individuals. Individuals have very little knowledge- least any say, in how their information is being released by their home organization. However there are additional aspects of privacy. Alan Westin [17] has defined privacy as: “The right of individuals to determine for themselves when, how and to what extent information about them is communicated to others.” There could be various scenarios where a home organization has to disclose its member’s private information by her acquiescence. Let’s consider a federation of Universities and Companies own material.

In such federation students will apply for jobs in companies to schedule an interview. Companies need information like GPA, Transcript, SSN, and Email from students. University could release student’s information as per her acquiescence. But it is
common that students often prefer to release same information like Transcript to same or different company under different condition like company offering job in operating system or software engineering. Same could be true for the companies. Existing federated model doesn’t support any such facility, and so we can state that Federation Model doesn’t support privacy, which is the basis of federation, as defined by Alan Westin. To provide privacy as-per definition in Federated Model, it is imperative to have Individualized Policies for federated members. Impact of such personalization on the protocol is however non-trivial. As opposed to simple request reply based communication, such personalization inherently triggers exchange of information between them — a negotiation.

2.1 Objective

A federation has standard set of policies \( P_1, P_2…P_L \) where \( 1….L \) is a set of attributes for the members \( 1…….M \). This can be represented as 1) \( P_{LM} = P_{L(M-1)} = \ldots = P_{L1} \) 2) \( P_{(L-1)M} = P_{(L-1)(M-1)} = \ldots = P_L \ldots..L \) 3) \( P_{1M} = P_{1(M-1)} = \ldots = P_{11} \) where \( P_{LM} \) indicates policy for \( L^{th} \) attribute of \( M^{th} \) member.

We are proposing policies \( (P_1, P_2…P_K) \) for every federated member. These policies are represented as 1) \( P_{KM} \neq P_{K(M-1)} \neq \ldots \neq P_{K1} \) 2) \( P_{(K-1)M} \neq P_{(K-1)(M-1)} \neq \ldots \neq P_{K1} \ldots M \) 3) \( P_{1M} \neq P_{1(M-1)} \neq \ldots \neq P_{1L} \) where \( P_{KM} \) indicates \( K^{th} \) policy of the \( M^{th} \) member.

A member’s \( (M) \) policy for an attribute ‘A’ is \( P_{AM} \leftrightarrow F_{AM}(S_{1R} S_{2R} \ldots S_{KR}) \) where \( F_{AM} \) is an expression with the attributes of the opposite party \( (R) \) only. Attribute ‘A’ will be shown to the opposite party \( (R) \) if and only if \( F_{AM}(S_{1R} S_{2R} \ldots S_{KR}) \) is satisfied. A negotiation between \( (M) \) and \( (R) \) results in attribute exchange sequence \( E = N_{P1} \ldots N_{PJ} \).
where \( N_I \) is an attribute of either party (T) such that \( 1 \leq I \leq J \). \( N_I \) is shown to the other party only if and only if \( F_{NI_T} \) is satisfied.

### 2.2 Negotiation System Model

Although, we have poised the problem as credential exchange- but *personalized negotiation* is the generalized form of all mutual *transactions* be it financial, p2p barter exchange, or identity management. A music file sharer may want to share a music file of specific genre- such as mystic songs of Lalon in exchange for some verses of Rumi or Omar Khayum translated specifically in Hindi or Bengali he enjoys. A file sharer may want to know the email address of the other side, while the other side may want to know the ID of the other side before releasing its email address. A specific seller might need an ID which however, can be satisfied with a driver’s license, state id or a passport, on in other cases in another country it might be the id issued by *Ponchaet*. In another part of the world two long lost friends may use their old school roll number and name of a specific common friend or late teacher to establish identity without requiring their govt. issued ID. In some cases the identity and more fundamentally trust may be established with dynamic communication based on the fields in the ID, and verifying some common knowledge provided in the credentials. There are many such traditional models of identify/trust verifications in real society. Technology for all has not been explored yet. Yet there is force in our community to move- perhaps unwisely- towards a centralized ID, whereas this may not be a ‘fundamentally’ required. Rather we need to explore other means of ID trust and ID verifications for online world- and examples of which are abundant right in our real society. The technological maturity of the field is at its puberty.
A universal centralized system is rather impractical— even if attempted and will be eventually rejected because of many abnormal artifacts on the working of bigger societal dynamics. Let us now focus back to the problem of credential exchange and compare four abstract models.

### 2.2.1 Centralized Negotiation System

In Model One, a **centralized credential system** is envisioned— where a global (it does not matter even if it is implemented by a distributed system such as DNS) will contain all the credential elements of all the individuals. The policy of credential exchange is set by a global standard. Technology for Model One is ripe. Though, technological scalability and its complete social implications are unknown.

![Centralized Credential System Diagram](image)

**Figure 2.** In Centralized Credential System members of every organization, in green color, stores credentials at centralized server shown in black color.
2.2.2 Federated Negotiation System

In Model Two, individuals are members of collective entities called *federations*. In this *federated credential system* an organization collects credential elements of all the individuals on behalf of the individuals. Credential release is set by two sets of policies one non-real-time and one real-time; non-real-time one that is agreed between the individuals and her organization and another real-time one that is agreed between the organizations within the federation.

![Diagram](image)

*Figure 3. In Federated Credential System every organization has federation Server, shown in black color, where members store their credentials. Federation Server acts as Negotiation Agent on behalf members.*

2.2.3 Libertarian Negotiation System

Model Three is diametric to Model One. In this *libertarian credential system* each individual stores their credentials. Each individual also sets the specific policy under which they will release which credential to whom in real-time. There is no centralized policy. Network takes the role of an exchange or a conduit. And a network wide protocol supports the resolution of the policies reflecting individual preference when two or more
parties engage to interact. Network may provide limited storage to support asynchronous or synchronous exchange- but do not take part in policy setting.

![Network Diagram](image)

**Figure 4.** In Libertarian Credential System each member of every organization stores own credentials and negotiate with other members without mediator.

### 2.2.4 Community Negotiation System

In model Four, individuals are also organized as communities. In community credential system, each individual still owns and stores their credentials. Each individual sets their credential release policies and credential acceptance policy. Network takes the role of an exchange or a conduit. Network may provide limited storage to support asynchronous or synchronous exchange- but do not take part in policy setting. The community acts as a credential keeper/agent on behalf of the individuals that remains semantically transparent without altering the real-time policy set forth by its member individuals. Rather, it’s only purpose is to provide efficiency and asynchronous modes in communication. For example, when there are sufficient consensus individual policies then it may provide policy aggregation and choice of designated community policies.
Figure 5. In Libertarian Credential System each member of every organization stores own credentials and negotiate with other members without mediator.

2.3 Related Work

Individualized Privacy Policy Based Access Control [1] proposes model for Individualized privacy policies to authorize action on personal data, replacing traditional permission or role based access control. But it doesn’t provide interaction-mechanism between policy holders - an inherent need of Individualization.

Trust Negotiation [2, 3, 4] enables service provider and requester to establish trust through cautious, iterative, bilateral disclosure of credentials. Automated Trust Negotiation [2, 6] discusses eager and parsimonious negotiation strategy. In eager strategy, participants release their attributes as soon as their ACP is satisfied without a request for that attribute. It is simple, efficient and leads to successful negotiation whenever possible but at the cost of disclosing more attributes than necessary. In parsimonious strategy, participants exchange attributes requests with focus on achieving local minimality of disclosures. It is also succeeds whenever possible, but the attribute
requests can disclose sensitive information about the users attributes. A PRUNE [5] negotiation strategy prevents irrelevant credentials disclosure, but it discloses policies of the attributes involved in negotiation. Trust Negotiation in federation [13] has integrated trust negotiation in the federation. This integration reduces number of exchanges between federated members and service provider with each successful negotiation between them. But it doesn’t provide negotiation protocol, and multiple negotiation flavors.

Foundation for Intelligent Physical Agents [18] has provided specifications for Agent Communications Language message and message exchange interaction protocol for interacting agents and agent-based systems.

### 2.4 Overview of Approach

Individualized policies for every federated member inherently triggers negotiation between the members. Such negotiation requires is a contract which decides how to initiate, proceed, and end the negotiation. This contract referred as Negotiation Protocol is discussed in section 3.2. Negotiation Protocol defines sequence of negotiation messages such as negotiation initiation, negotiation strategy, actual negotiation and end of negotiation. Protocol also defines negotiation message structure which carries negotiation state so that neither party has to store it - a stateless negotiation process.

A stateless negotiation process and complex message sequence requires an automated system to work on behalf of the members. This system referred as Negotiation Agent interacts with members’ Individualized Policies for end-to-end automated negotiation process. Interaction between Negotiation Agent and Individualized Policies is coordinated using Negotiation Enabled Framework (NEF) in section 3.3. NEF has three
main components - Policy Vault (PV), Resource Vault (RV), and Negotiation Agent (NA). Negotiation Agent has three subsystems - Policy Resolver, Session Manager and Message Handler. Message handler interprets incoming message as per Negotiation Protocol and forward it to Session Manager for verification. After that, Policy Resolver resolves incoming message to release attributes or request attributes – as per ACP - from opposite party by interacting with Policy Vault and Resource Vault.

Policy resolver resolves incoming message based on Negotiation Algorithms, in section 3.4, namely Stateless Eager Attribute Negotiation Algorithm (SEAN) and Stateless Proxy Attribute (SPAN) Negotiation Algorithm. Policy Resolver executes an algorithm agreed by both parties during early stages of negotiation. In addition to Negotiation Algorithms, we have discussed Knowledge Based Attribute Negotiation (KBAN) techniques for efficient negotiation process by reducing messages exchanged between two parties. These techniques – semantically transparent to negotiating parties - are Sequential and Parallel Negotiation.
CHAPTER 3

Proposed Solution

At the heart of proposed solution is Negotiation Process Model in section 3.1 and based-on this model we have provided Negotiation Protocol in section 3.2. This protocol is executed by Negotiation Enabled Framework, in section 3.3, at the each end point of negotiation. As a part of Negotiation Enabled Framework, we will discuss Negotiation Algorithms viz. SEAN, and SPAN. In addition to this, we will discuss efficient negotiation strategies viz. Sequential Negotiation and parallel negotiation in this section.

3.1 Negotiation Process Model

In this model, a negotiation occurs between two parties. For each negotiation there is a Negotiation Initiator [NI] and a Negotiation Responder [NR]. This section discusses formal model of negotiation. Negotiation modeling is an uncharted area. In figure 6, we present a state-space model of any negotiation.

For each resource there is a requestor (R), and an owner (O). Generally, the negotiation starts by the requestor requesting a target resource from the owner. However, the target resource is released in lieu of other additional resources as per owners release policy associated with the resource. We define the states via which these resources are processed in an arbitrary negotiation by the state diagram given in Figure 6. In this model any resource is released via a negotiation process which transits through the states namely requested, offered, and accepted.
Figure 6. A) State Transition Diagram representing states of negotiation of an attribute. B) States in blue are considered for protocol and algorithm given in this paper. Protocol and algorithm are extensible for states in green.

When a party first requests a resource it is placed into requested state. When the other party decides to release it, it is brought to the offered state. When the requesting party finally accepts it is moved into accepted state. However, depending on the complexity of release policies a negotiation may transit through additional states namely pending, not_available, denied, available, extraneous, and rejected. If the second party’s release policy requires some other resource before the requested resource can be released then the original request moves to pending state. If the second party does not have the requested resource then it is placed into not_available state. If it has but would like to deny then it moves to denied state. There is also a state called available which is used in proxy negotiation. In proxy negotiation, both parties move their releasable resources to available state until successful negotiation becomes possible. Once both parties are sure that successful negotiation is possible, they will exchange values of resources which are in available state. If the original party finds some error with the value of released
resource then the resource is rejected. Figure shows the possible transition paths between these states.

We now explain the possible transitions by modeling a very simple negotiation between Alice (Negotiation Initiator) and Bob (Negotiation Responder) for an attribute ‘A’.

**Alice’s side:** Alice starts negotiation by requesting resource ‘A’ from Bob. At this time, state of negotiation of resource ‘A’ is *requested*.

**Bob’s side:** Once Bob receives request for ‘A’, as per ACP he will bring state of negotiation of that attribute from *requested* to one of the five states of negotiation namely available, pending, not_available, offered, and denied. After that, Bob will send that attribute back to Alice. As discussed, five transitions are possible that starts with *requested* state of negotiation. Each transition is explained below:

1) If access control policy of Bob is satisfied for resource ‘A’, then state of negotiation of ‘A’ transits from *requested* to *offered* as show by number 4. This transition indicates that Bob will release actual resource immediately unlike transition from *requested* to *available*.

2) If access control policy of Bob is satisfied for resource ‘A’, but the negotiation is proxy negotiation then state of negotiation of ‘A’ transits from *requested* to *available* as show by number 1. This transition indicates that Bob will not release actual resource before he becomes sure that successful negotiation is possible.

3) If access control policy of Bob needs a resource (for example ‘B’) from Alice to release ‘A’, then state of negotiation of ‘A’ transits from *requested* to *pending* shown
by number 2. For resource ‘B’ new state transition diagram will start with state of

*requested*.

4) If resource ‘A’ is not available with Bob, then state of negotiation of ‘A’
transits from *requested* to *not_available* as shown by number 3.

5) If resource ‘A’ is available but for some reasons Bob does not want to release
it, then the state of negotiation of ‘A’ transits from *requested* to *denied* as shown by
number 5.

**Alice’s side:** When Alice receives response from Bob, Alice will bring state of
negotiation of ‘A’ to *accepted, extraneous, rejected* or *requested* state.

1) If this is the resource Alice wanted, then it will go to accepted state and
negotiation will terminate. But if Alice has some alternative resource to request, then it
will request new resource from Bob.

2) If Bob has offered something which Alice did not request and do not want then
the state of negotiation transits from *offered* to *extraneous* state.

3) If there is error in the resource format or credibility then Alice will not accept it
and place it into *rejected* state.

4) If Alice wants to request same attribute again, then state of negotiation transits
to *requested* state again.

### 3.2 Negotiation Protocol

This section provides a prototype of protocol for bi-partite negotiation. A
negotiation proceeds through number of steps between the parties. The core steps in a
negotiation is session initiation, request, analysis of request and decision, counter request,
analysis of counter request and decision at the other side, and so on, and at the end closure. An ongoing negotiation may or may not result in a deal. But the decision and analysis algorithm must continue until a resolution-deal or no-deal is achieved. There could be various flavors of negotiations. One strategy might be to complete the negotiation with minimum message exchange, another might be to disclose minimum amount of credentials, even if it means more communication steps. We design the communication so that it can accommodate most of these likely strategies. Resolution process can also be classified as stateless or state-full. We are particularly interested in a stateless process i.e. framework would not require to maintain complex state information about an ongoing negotiation.

3.2.1 Negotiation Message Exchange Sequence

Each negotiation proceeds through six states namely Advertisement/solicitation, greetings, strategy, active, negotiation, and adieu states shown in figure-x. Important messages used in negotiation are Greeting, Advertisement, Solicitation, Strategy, Negotiation, Deal, No_Deal and Reporting. A point to note is that negotiation can move from START to ADVERTISEMENT or GREETINGS or STRATEGY state. Another point is, negotiation can move back and forth between ACTIVE state and ADVERTISEMENT or GREETINGS or STRATEGY state if necessary.

Negotiation moves from START to GREETING state, when one party wants to invite other for a negotiation session by offering their initial identities. This is an invitation to negotiate. After sending greetings message, each party waits for greetings message from the opposite party. Opposite party can accept or reject the invitation (and
Figure 7. State Transition Diagram for Negotiation Protocol

Negotiation proceed to ADIEU state). If both parties agree to proceed then negotiation enter into ACTIVE state. Here, negotiation can also move to ADVERTISEMENT or STRATEGY state. In ADVERTISEMENT state, a party sends advertisement (or solicitation) messages naming target resources (R) and then waits for the other party to reciprocate by sending solicitation (advertisement) message. If one party is interested to offer the target resource and the other party is interested to obtain it then negotiation moves to ACTIVE state otherwise moves to ADIEU state if any one of them is not interested. In STRATEGY state, a party accepts list of negotiation strategies from opposite party, based on the lists they can agree on a strategy and proceed to ACTIVE state (or jump to ADIEU state). Negotiation moves from ACTIVE to NEGOTIATION state when both parties have fulfilled their prerequisite like receiving each other’s identity.
or deciding strategy or deciding actual resource to negotiate. In NEGOTIATION state, the actual negotiation starts by sending negotiation message and protocol moves though a series of exchanges until negotiation becomes successful (Deal) or unsuccessful (No_Deal). After this negotiation moves ADIEU state. But in case of any error in NEGOTIATION state, it moves to END state. In ADIEU state both sides perform accounting/reporting about this negotiation session and then moves to the END state. Also, negotiation protocol moves to END state if any error condition occurs at any given state.

3.2.2 Negotiation Message Structure

Negotiation message has been designed to store all necessary state information about the resources involved in negotiation so participants don't have to store negotiation information otherwise requested. The message i.e. Negotiation Envelope in figure 8 has two main compartments namely header and body. Header carries information regarding session. Its fields are Action, Session Id, Timestamp, Initiator Strategy, Responder Strategy, and set of fields with security information. Strategy fields are useful to indicate negotiation strategy chosen by Negotiation Initiator and Negotiation Responder. Even though this paper is going to consider only one negotiation strategy, strategy fields would be useful when two parties will negotiate for negotiation strategy. Action field contains one of the messages namely Advertisement, Solicitation, Greeting, Strategy, Negotiation, Deal, No_Deal, Adieu, Error. Advertisement message includes target resources (R) in the message and then waits for the other party to reciprocate by sending solicitation (advertisement) message. Greeting message is used to invite other party for a negotiation
session by offering initial identities and expect same from other party. Strategy message is used to decide negotiation strategy between two parties. Negotiation message is used when two parties start exchanging resources. Deal message indicates successful negotiation and No_Deal indicates unsuccessful negotiation. Adieu message indicates reporting about negotiation session. Error message indicates any unexpected error in negotiation.

![Figure 8. Negotiation Message](image)

Body has two parts namely Initiator's Resource List (IRL) and Responder's Resource List (RRL). Each item in these lists has the following fields. Resource Descriptor (RID) is the name of resource. This uniquely identifies the resource. The Value field contains the value or an URL to the value of the resource. Then there are three flags each one byte. TYPE, LOC and FORM. TYPE of a listed resource can have
one of the following values Personal (P), Credential (C), Attribute (A) or Information (I).

The LOC field tells where the credential value is available. It can be disclosed in the message (M), or in the private space of the Initiators (I), or of Responder (R). If the value of LOC flag is M then the value or location is disclosed in the VALUE field. If it is disclosed in the message, then the FORM field indicates if the VALUE field is Text (T) or URL (U). Finally, the License field contains the granted disclosure policy for the resource. State field contains current state of negotiation of a resource during negotiation process. Extra field can carry information about negotiation resolution process of a resource, if necessary.

3.3 Architecture

This section presents an agent based framework for negotiation in federation. Paper considers setup where each member in federation is owner of the resources such as credentials, attributes, files etc. Owners can specify their preferences and choices regarding their resource release policies. To release a particular resource, they can demand other resources from the requestor. This leads to negotiation. Every owner’s respective home organization performs negotiation on behalf of them when fellow organizations request their resources.

The components of framework in figure 9 are Policy Vault (PV), Resource Vault (RV), and Negotiation Agent (NA). Negotiation Agent has three subsystems Policy Resolver, Session Manager and Message Handler.
3.3.1 Resource Vault

Resource Vault is the cache system for the resources. We classify four kinds of resources mostly based on their source of authentication and privacy requirements. A resource can be a personal information (P) (such as display-name, legal-name, email, phone number, address, date-of-birth, social-security number), credentials (C) (such as credit card, e-cash, signed-certificate, transcript, credit-report, letter-of-recommendations, awards), attributes (A) (such as rank, position title, age-group, medical-coverage), or any other-piece-of-information (I) which can be released and exchanged subject to negotiation (such as news, weather, rumor, joke-of-the-day, etc). Although, all are owned by the members and treated in the same way in the negotiation process but we use the term personal information (P) to refer to the resources generated by owner and has strong
privacy requirement. We use credentials to refer to the resources originally generated by third parties thus might be subject to additional external verification. We use attributes to refer to the information vouched by the home organization. Note that particular piece of information may appear in different categories in different organizations. For example some organization may treat gender to be personal information while some as attribute. Each resource is stored in the vault either as a value (such as name, or email) or as a file (such as CV, list of publication, a picture, e-book file) based on their granularity. Table 1 shows a vault. Each resource also carries a unique resource ID for unambiguous referencing in the negotiation process.

### 3.3.2 Policy Vault

The Policy Vault in Table 2 stores the release policies of owner. It allows owner to set the disclosure conditions for each of her resources. The owner can unconditionally release a resource, or require other resources from the other side as precondition. The conditions are expressed by a set of rules. Each rule is represented in the disjunctive normal form (DNF) such as $R_1 \leftarrow C_1 \lor C_2 \ldots \lor C_j$, where, clauses $C_1 = R_1 \land R_2 \ldots \land R_K$,

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Resource Descriptor</th>
<th>Resource Type</th>
<th>Resource Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Name</td>
<td>P</td>
<td>John</td>
</tr>
<tr>
<td>2</td>
<td>Transcript</td>
<td>C</td>
<td>Trans.html</td>
</tr>
<tr>
<td>3</td>
<td>Job_Title</td>
<td>A</td>
<td>Software Eng.</td>
</tr>
<tr>
<td>4</td>
<td>Joke</td>
<td>I</td>
<td>Joke.html</td>
</tr>
</tbody>
</table>
which indicates to release resource $R_1$, either of clauses $C_1$, $C_2$, or $C_j$ must be satisfied. Where, clause $C_1$ requires all resources $R_1$, $R_2$ and $R_K$ from the other side.

<table>
<thead>
<tr>
<th>Resource ID</th>
<th>Role in Negotiation</th>
<th>License</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NI</td>
<td>HF</td>
<td>$R_1 \land R_2$</td>
</tr>
<tr>
<td>2</td>
<td>NR</td>
<td>HF</td>
<td>$R_1 \land R_2 \land R_3$</td>
</tr>
<tr>
<td>3</td>
<td>NI</td>
<td>S</td>
<td>$R_1 \land R_2$</td>
</tr>
<tr>
<td>4</td>
<td>NR</td>
<td>S</td>
<td>$R_1 \land R_2 \land R_3$</td>
</tr>
</tbody>
</table>

Owner can specify two rules for each resource. Negotiation Agent will select appropriate rule based on whether owner is Negotiation Initiator (NI) or Negotiation Responder (NR). Owner can also specify License offered to requester of the resource. License decides whether requester can Cache (H), Store (S), and Forward (F) requested resource.

### 3.3.3 Negotiation System

Now, let’s understand interaction of these components during negotiation in brief. Negotiation agent will negotiate on behalf of member of home organization with other Service Provider organization in federation. When negotiation begins, Policy Resolver will select policy and as per that policy, request and release attributes. These set of resources are send to Session Manager to add information like timestamp, address of opposite negotiation participant etc. Session manager send these resources and other
information to Message Handler. Message Handler generates message from received resources and other information to send it to opposite negotiation participant.

On other participant’s side, Message Handler accepts incoming message. Message Handler disassembles it to get all attributes, session information, and sender’s details. These attributes, session information, and sender’s details are forward to session Manager. Session Manager verifies sender and session information. Session manager forwards received attributes to Policy Resolver. Policy Resolver chooses policy from Release Policy Base and resolves all requested attributes by opposite negotiation participant. As a result of this resolution process, Policy Resolver selects releasable attributes from the Resource Vault. Resolution process also decides which attributes to request from opposite negotiation participant. All these attributes along with opposite participant's attribute from incoming message gets forwarded to Session Manager. From session manager same process repeats as explained above.

### 3.4 Negotiation Algorithms

The above described negotiation protocol is capable of supporting various flavors of negotiations. We will two flavors of negotiation resolution algorithm i.e. Stateless Eager Attribute Negotiation (SEAN), and Stateless Proxy Attribute Negotiation (SPAN) algorithm that performs a proxy negotiation first without disclosing the actual resources to each other. It first determines if the deal is possible or not. By the time it determines if the deal is possible, it also determines the optimum resource exchange sequence to close the deal. This resolution algorithm maintains maximum privacy of resources and credentials however it also generally requires more communication steps. As the name of
algorithm indicates, we are interested in stateless negotiation so that each end-point
doesn’t have to store negotiation session information.

The users interested in negotiation with fewer communication steps at the cost of
disclosing more attributes can opt for SEAN, while users interested in proxy negotiation
or interested in disclosing minimum credentials in negotiation can opt for SPAN.

3.4.1 Stateless Eager Attribute Negotiation (SEAN) algorithm

This stateless eager attribute negotiation (SEAN) algorithm, in figure 10, is used
when negotiation participants choose eager strategy for negotiation. This paper provides
stateless eager strategy so that framework doesn’t have to store negotiation information.
All necessary negotiation information is carried by message itself.

This algorithm (SEAN) accepts rlist and PR as input, where rlist is resources
available in IRL and RRL. PR is policy of Negotiation Responder assuming that
algorithm is executing at Negotiation Responder’s end. Same algorithm also executes at
Negotiation Initiator’s end. Each time it executes it returns one of the three states of
negotiation namely DEAL, NO-DEAL, or negotiation namely DEAL, NO-DEAL, or
NEGOTIATION i.e. negotiation is continuing. If the negotiation is continuing, algorithm
also returns three things namely a set of newly offered resources (NEW_RELEASE), a
resource (NEW_REQUEST) which has been counter requested, and name of a resource
which is currently pending (NEW_PENDING). This is a resource which has been
requested and responder is currently resolving on its release.

Note that NEW_REQUEST and NEW_PENDING are used only when algorithm
executes on Negotiation Initiator’s side and Negotiation Responder’s side respectively,
Figure 10. A) First row has SEAN_negotiator routine which receives message from and sends message to opposite party. B) Second row has SEAN_state_machine routine which changes negotiation state of attributes and also generates state of negotiation i.e. DEAL, NO_DEAL or NEGOTIATION. C) Third row has SEAN_rule_resolver routine which decides if a attribute can be released
but not other way round. This is because in eager strategy only requester requests resource from responder, but responder never requests resource from requester, so requester will request target resource each time until it gets it (and for that it uses NEW_REQUEST) and responder will make requester’s request pending until it’s policy is satisfied (and for that is uses NEW_PENDING.)

*Execution of the Algorithm*

As explained in the last section, algorithm generates three state of negotiation namely DEAL, NO-DEAL, and NEGOTIATION. These states are generated as follows

1) When Service Provider releases the requested resource ‘A’ the negotiation becomes successful. This successful negotiation is indicated by DEAL state. This is checked at the beginning. The algorithm checks (lines 5-7) if OLD_RELSET, which contains all resources offered by the opposite party, contains the initial target. If so then the state becomes DEAL and it returns ( ({Ø},{Ø},{Ø}, DEAL ) ). As state has become DEAL, first three parameters are empty.

2) On the other hand, NO-DEAL or failure of a negotiation is At the end of current request (lines 18-21), if there is anything new to offer then it is added to newly released resources set (NEW_RELSET). It is then compared with the old released set (OLD_RELSET). If these two sets are same then negotiation state becomes NO-DEAL with return parameters (line 22) as ( ({Ø}, {Ø}, {Ø}, NO-DEAL ) ). But first request from Negotiation Initiator is exception to above mentioned rule.

In first request, if Negotiation Initiator doesn’t have anything to offer then OLD_RELSET and NEW_RELSET become same. But in this case negotiation is not
necessarily unsuccessful as Negotiation Responder might offer her own resources and thus negotiation can proceed. To take care of this exception, line 20 includes clause i.e. this is not start of negotiation.

3) If state is neither DEAL nor NO-DEAL, then state becomes NEGOTIATION. This state has two possible scenarios as explained below

**Case-A:** Requester, along with the request for target resource, offers those new resources which don’t need opposite party’s resources. If this is the case, then algorithm checks if requested resource’s policy can be satisfied using requester’s or Negotiation Initiator’s offered resources

1) If policy is satisfied, then resource owner will offer requested resource. This step is shown by lines from 11-14. Line 11 checks if A is requested resources and line 12 passes it to SEAN_rule_resolver routine. Line 1 in SEAN_rule_resolver routine determines if A can be offered and at line 3, A is returned using in RELSET. If A is offered then, A becomes part of NEW_RELSET shown by line number 14. In this case, NEW_PENDING will be empty at line 13 because ‘A’ will either go in offer or pending state from requested. If ‘A’ is the requested resource, then algorithm won’t release rest of the B) If policy isn’t satisfied, then resource owner will offer his resources to requester and in turn requester will offer his resources. This way negotiation proceeds. This process is shown by line 11-18. In this case, at line 13 NEW_PENDING will contain requested resource because its policy is not satisfied. So at line 14, NEW_RELSET would be empty. Line 16-18 will release all possible resources of responder by going through
routine SEAN_rule_resolver. These released resources are returned at line number 24. At this time two sets (OLD_RELSET and NEW_RELSET) are not same.

**Case-B:** The other case is that requester has requested resource from opposite party without offering any of his resource. Now if resource owner doesn’t have any resource to disclose, then negotiation will be terminated i.e. NO_DEAL. This is taken care by steps in point 2) as explained before. But if resource owner has resources to offer, then state of negotiation of requested resource will become pending by offering resources to requester. This way negotiation proceeds and it could follow 1) or 2) again. Initially it is same as 3) and then turns to 1) or 2).

As per Eager strategy, only Negotiation Initiator requests resources so line 9-10 will handle pending requests for Negotiation Initiator. While only Negotiation Responder has to handle incoming requests so line 11-14 are exclusively for Negotiation Responder.

### 3.4.2 Stateless Proxy Attribute Negotiation (SPAN) Algorithm

In proxy negotiation two sides first exchange the name of the resources, but do not exchange the actual resources to let each other know about their requirement constraints. In the process of this proxy negotiation both tracks the requirement dependencies. The goal is to determine if the deal is possible. If it is possible only then they perform the actual resource exchange. Since, by this time both the parties know the requirement dependencies in this final step they exchange resources using optimum sequence.

The key step is the proxy resolution performed by SPAN. SPAN is the algorithm that drives the negotiation, while Resolver is the heart of the negotiation process. We
describe them next. In this explanation we will assume that SPAN algorithm is running at
the responder unless otherwise specified. Though, identical SPAN runs at both ends in
with perfect symmetry.

*Driver Process*

The input to this algorithm (at the responders end), in figure 11, is the responder’s
release policy and the newly received message (from Initiator). It’s output (line 22) is
four sets- the first three are his own resources namely (1) those ready to be released
(NEW_RELSET), (2) not ready but pending (NEW_PENSET), (3) those he would like to
deny (NEW_DENSET), and (4) a list of new resources he would like to counter request
(NEW_REQSET). SPAN() calls the Resolver routine (line 16) for each requested
resource in the message list. Resolve routine determines if the resource can be released or
if not what other resources need to be counter request, or if it has to be denied by
consulting the release policy associated with the resource. Its inputs are (i) a resource in
the set of old requests (OLD_REQSET), (ii) resource lists in the message, and (iii) the
rules (from policy base) particularly linked to selected resource from old requests set. The
routine changes the states of the resources and it also may generate some new release set
(RELSET) and the counter request set (REQSET). Corresponding to the four new sets,
routine maintains four copies of old sets (OLD_REQSET, OLD_RELSET,
OLD_DENSET, OLD_PENSET) extracted directly from an incoming message (shown
by lines 1-4). The algorithm checks (lines 11-13) old sets to see if the negotiation has
resulted in a deal. Negotiation becomes successful if very first request i.e. Service
Request is now available in OLD_RELSET. Negotiation becomes unsuccessful if Service
Figure 11. Driver Process and Resolve Process in SPAN

```plaintext
SPAN_state_machine(rlist, P^s) {
1. OLD_REQSET = {A^s C rlist: A^s. state = REQUESTED}
2. OLD_RELSSET = {A C rlist: A. state = AVAILABLE}
3. OLD_DENSET = {A C rlist: A. state = DENIED}
4. OLD_PENSET = {A^s C rlist: A^s. state = PENDING}
5. RELSET = REQSET = {f}
6. nego_state = CONTINUE
7. If (\{A^s C rlist: A^s is service request\} and \{A^s C OLD_DENSET\})
8. nego_state = NO-DEAL
9. return (\{f\}, \{f\}, \{f\}, \{f\}, nego_state)
10. Else
11. If (\{A^s C rlist: A^s is service request\} and \{A^s C OLD_RELSSET\})
12. nego_state = DEAL
13. return (\{f\}, \{f\}, \{f\}, \{f\}, nego_state)
14. Else
15. For (each A^s in OLD_REQSET)
16. {REQSET}, {RELSSET}) =
17. SPAN_rule_resolver(A^s, rlist, RULE[ A^s])
18. NEW_PENSET = {NEW_PENSET} U
19. \{A^s C OLD_REQSET: \{REQSET \neq \{f\} \}
20. and \{RELSSET = \{f\}\}
21. NEW_REQSET = {NEW_REQSET} U
22. \{REQSET\}
23. NEW_RELSSET = {NEW_RELSSET} U
24. \{RELSSET\}
25. NEW_DENSET = {NEW_DENSET} U
26. \{A^s C OLD_REQSET: \{REQSET = \{f\} \}
27. and \{RELSSET = \{f\}\}
28. } // End of for from line 15
29. NEW_RELSSET = {NEW_RELSSET} U
30. OLD_RELSSET
31. return (NEW_REQSET}, {NEW_RELSSET}, {NEW_PENSET}, {NEW_DENSET}, nego_state)
} // End of else from line 14

SPAN_rule_resolver(A^s, rlist, Rule)
1. GARC = Count of Released attributes in M
2. If (Rule = = True)
3. LURA [A^s] = ~
4. A^s C RELSET
5. Else
6. While (Rule)
7. { Clause = Next Clause in Rule
8. If (1st time in Rule & CQ [A^s] \neq null)
9. {A^1 = CQ [A^s]
10. Clause = Clause in Rule which contains first A^1
11. and all attributes in that clause before A^1 are
12. released
13. switch (A^1.state)
14. Case Pending: Exit Rule
15. Case Denied: Clause = Next clause
16. Case Offered: Clause = Clause
17. } // End of if from line 9
18. Next_clause = False
19. While (Clause \| Next_clause = = False)
20. {A^1 = Next credential in Clause
21. If (A^1 ~ rlist)
22. A^1 REQSSET
23. CQ [A^2] = A^1
24. Exit Rule // End of if
25. If (A^1.state = = Denied)
26. If (ARC [A^1] > GARC)
27. A^1 C REQSSET
29. Exit Rule
30. Else // Else for if from line 22
31. Next_clause = True
32. } // End of While from line 16
33. If (Next_clause = = False)
34. A^2 C RELSET
35. LURA [A^2] = Current Clause
36. GARC = GARC + 1
37. Exit Rule
38. } // End of Else from line 5.
39. Return (\{RELSSET\}, \{REQSET\})
```
Request is in OLD_DENSET (shown by lines 7-9)). If none of the above conclusions can be made, then the Resolver routine is called (line 16) to process every resource in OLD_REQSET. This Resolve routine generates two sets namely REQSET and RELSET. REQSET contains resources that need to be requested from opposite negotiation party and RELSET contains resources that need to be released (by owner) as a part of negotiation process. This new REQSET, RELSET provided by the resolve routine and the already existed old sets generate the new sets (NEW_REQSET, NEW_RELSET, NEW_PENSET, and NEW_DENSET) for the outgoing message (shown by lines 17-21).

Resolve Process

The algorithm uses following important scratch variables. 1) **GARC (Global Attribute Release Count)**: This is the current total number of resources released by both parties. 2) **ARC (Attribute Release Count)**: When a resource is denied, its ARC is used to save the current GARC value. 3) **CQ (Counter Request)**: This is used to remember the name of the resource asked as counter request for each pending resource. 4) **LURA (Clause that causes release of attribute)**: This is for each released resource to save the clause in a rule that has resulted in its release to trace back the release sequence.

Each time the routine is invoked to analyze a requested resource; it one by one considers the rules associated with that resource. In each rule, clauses are processed from left to right and in each clause it considers resources from left to right. Routine’s processing is explained using following points:
STEP-1) If a requested resource doesn’t have CQ, it means that it is being requested for the first time, routine executes (lines 6-7, lines 15-21, lines 29-33) in the following way:

If the first resource in requested resource’s first clause is not released yet and also not pending, then this first resource will be counter requested (be part of the REQSET) and remembered in requested resource’s CQ (shown by lines 17-21). But if this first resource is already released (by opposite party), then the next resource in the same clause is counter requested and so on till all resources in one clause are exhausted. If all resources in one clause are released, which can be found out from Resource List available in the body of the message, then requested resource can be released. When released, GARC is incremented by 1 and the entire clause in saved in the LURA of the released resource (to later tracking the release sequence) shown by lines 29-33.

STEP-2) If a requested resource already has CQ, then routine executes (lines 6-12 and lines 6-21, 29-33) in the following ways:

2.A) If CQ of requested resource is still pending, then the routine will come out of the current rule and keep CQ of requested resource as it is (lines 6-12). Since, this request has been processed before therefore no new processing is needed and as a result routine returns.

2.B) If CQ of requested resource, however, has already been released (that could be found out in Resource List) by opposite party, then as explained in step-1 next resource from the same clause is considered for CQ and so on (lines 6-21,29-33).
**STEP-3)** If after step-1 or 2, there is a new counter request and if this new counter request (which is different from old counter request, if any) is already pending, then it will skip that clause and move on to next clause. This is because it indicates a cyclic dependency which currently cannot be resolved with current set of released resources. If rest of the clauses too has at least one resource whose request is already pending, then the requested resource is denied and the GARC is saved in its ARC (lines 34-35). But, if any one of the remaining clauses doesn’t have a resource whose request is pending, then that clause is processed according to either step number 1 or 2.

**STEP-4)** At the end of all the previous steps before finally placing the counter requests in the new request set, the routine will check if that resource has been previously denied by the other side. If this new counter request resource has already been previously denied then (line 22-28, 34-35):

4. A) That resource become CQ only if it’s ARC > GARC (lines 22-26). This is because since there are more releases now since the last denial- that previous cycle may not be a problem anymore.

4. B) That resource doesn’t become CQ if it’s ARC = GARC. This is because nothing has changed since last denial. In this case, the clause containing that resource is skipped for next clause (line 28). If rest of the clauses too have at least one denied resource with above condition true then the requested resource is left in denied state (line 34-35) and its ARC become current GARC.
**Release of Resources**

Once the proxy negotiation is successful then both the sides enters into the actual resource release phase. At this time, each party knows that the deal is possible and also in the message there is indication that which clause in which rule caused the release of which resource. Using this information each party generates logical dependency graph, so that they can found out exact sequence of exchanging value of resources. This graph is generated using LDgraph_Generator routine explained in 4.4.2.3.1. Once dependency graph is generated, each party finds out which resources in graph to offer depending on resource disclosed from opposite party. This is performed by Offered_Set routine explained in 4.4.2.3.2

Thing to note, both parties generate logical dependency graph each time message comes in from opposite party, so none of them actually remembers dependency graph.

**Dependency Graph Generator**

To generate Logical Dependency Graph, algorithm uses rlist as input. A node in dependency graph is presented as a resource whose state of negotiation is offered or available and other nodes as solved clause of its disclosure policy (with one edge from each resource in that clause connecting to a node). This way all resources in dependency graph are represented (lines 2-9). Initially in the resource release phase, dependency graph is generated from resources whose state of negotiation is available because none of the parties have offered any resources yet. But once parties start offering resources to each other, they will brought state of negotiation of resources from *available* to *offered*
Figure 12. Dependency Graph Generator in SPAN

(Because they offered value of resource), so then logical dependency graph is generated from resources whose state of negotiation is either available or offered (line 1).

Offering Resources

```c
1. Dgraph_Generator(figlist) {
   2.   release [] = {R ∈ figlist: R.state = AVAILABLE or R.state = OFFERED}
   3.   lura_count = sizeof(LURA)
   4.   For (I=0; I < lura_count; I++)
      5.     ldgraph [I][0] = relble [I]
   6.   lura [] = {resources that caused release of release[I]}
   7.   lura_count = sizeof(lura[])
   8.   temp = 0
   9.   for (J=1; J <= lura_count; J++)
      10.    ldgraph [I][J] = lura[temp++]
    }
11. return ldgraph
}
```

Figure 13. Offered Set routine in SPAN

Offered_Set routine executing at one end of negotiation finds out which resources can be offered to opposite party using i) logical dependency graph and ii) resources
offered by opposite party (i.e. old_offeredset) till that time of resource release phase, if any.

Initially, when no party hasn’t offered anything, one of the sides will offer his resources from dependency graph which are unprotected i.e. don’t have disclosure policy (lines 3-7). This is first step of exchanging actual resources. After this, opposite party will offer his resource from dependency graph according offered resources it has got from first party (line 8-14). Thus two parties keep on offering resources to each other till Negotiation Requester gets his target service.

### 3.5 Efficient Negotiation Strategy

Based on SPAN algorithm we will discuss two efficient negotiation strategies. These strategies are applicable for Community Negotiation System. In this system, community acts as an agent and credential keeper on behalf of the individuals without altering the real-time policy set forth by its member individuals.

When such two communities negotiate on behalf of their members, it provides an opportunity for communities to learn about policies of the members involved in negotiation. This knowledge of every passing negotiation can be used in new negotiation between same communities to reduce number of messages exchanged. This technique of increasing communication efficiency between communities is Knowledge Based Attribute Negotiation (KBAN).

For example members $A_1$….$A_M$ of community ‘A’ are negotiating with members $B_1$….$B_N$ of community ‘B’. A negotiation between any two members is represented as $N_{IK}$ where $I$ and $K$ are $1 \leq I \leq M$ and $1 \leq K \leq N$. Knowledge about the policies in one
negotiation \( N_{IK} \) can be used in the negotiations such as \( N_{(I+P)} (K) \) or \( N_{(I)} (K+V) \) where \( 1 \leq P \leq (M - I) \) and \( 1 \leq V \leq (N - K) \). This is based on assumption that negotiation \( N_{IK} \) takes precedence over \( N_{(I+1)} (K) \) or \( N_{(I)} (K+1) \). This approach to use negotiation knowledge is \textit{semantically transparent} to community members. Below we will discuss two KBAN strategies namely Sequential and Parallel negotiation.

### 3.5.1 Sequential Negotiation

In sequential negotiation, one community negotiates on behalf of her member one-by-one with all the members from opposite community. For example, Community 'A' negotiates on behalf of a member \( A_1 \) with the members \( B_1, ..., B_N \) from community ‘B’ and after that same process is repeated with the members \( A_2, ..., A_M \). In this sequential process, execution of the first negotiation (between \( A_1 \) and \( B_1 \)) helps communities accumulate knowledge about negotiator’s policies and this knowledge is used in the next unexecuted negotiation (between \( A_1 \) and \( B_2 \)). Thus with each executed negotiation both parties accumulate knowledge about negotiator's policies. This cumulative knowledge is used in every unexecuted negotiation between two communities till last negotiation (between \( A_M \) and \( B_N \)).

Negotiation knowledge provides insight into negotiators policies to help reduce a number of messages exchanged between two communities - an efficient negotiation. One example of this efficient negotiation strategy is discussed in section 4.3.1.
3.5.2 Parallel Negotiation

As word parallel indicates, two communities negotiate on behalf of their members (or set of members’ interested in negotiation) simultaneously instead of sequentially. For simultaneous execution, a community requests on behalf of all her members in one message and then expects opposite community to process all those requests and respond back in one message. This process of request-response continues till all negotiations are resolved as deal or no-deal.

In parallel negotiation, multiple negotiations in every processed message provides an opportunity to learn about negotiators policies and use that knowledge in the rest of unprocessed messages till the end of all negotiations. Based on Negotiation-Knowledge it is possible to reduce a number of messages exchanged between two communities- an efficient negotiation. One example of this efficient negotiation strategy is discussed in section 4.3.2.
CHAPTER 4

Experimental Setup and Analysis

This section illustrates how Individualized release policies can be handled by a complex federated setup. We consider a job seeker-hunter scenario in a federated setup consisting of group of Universities and Companies. Students enrolled in each university can store their individualized policies with their respective home universities. Also, various managers in the companies set up their requirements when they search for potential employees. In this setup, we consider three students Alice, Pooja and Sajid with home institution KSU and three hiring managers Bob, John, and Yang from KLM Inc, ABC Inc, and CDE Inc respectively. First we show access control policies of all six individuals, in figure 14, and then two negotiations in detail. Finally, section includes results and statistic for all 9 possible negotiations [19] between three students and managers.

Each message carries state information about the resources which gets involved in a negotiation. In our notation we group resources against their respective current states in a negotiation as $\text{STATE}^p\left(\{R_1:V_1\},\ \{R_2:V_2\}…\ \{R_n:V_n\}\right)$. $\text{STATE}$ can be the states of a resources specified in figure-6 such as REQ, AVL, PEN, DEN, OFF etc. Each argument is a pair where $R_n$ is the ID of the resource and $V_n$ is the special information about the resource. The superscript $p$ represents the party (negotiation initiator (I) or negotiation responder (R)) involved in the latest update of a state. For stateless eager strategy $V$ is
always empty. Top portion and bottom portion of each message corresponds to IRL and RRL as shown in figure 8.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Alias</th>
<th>Value</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Name</td>
<td>(l1)</td>
<td>ABC Inc</td>
<td>~</td>
</tr>
<tr>
<td>Job Location</td>
<td>(l1)</td>
<td>XYZ OH</td>
<td>(R2, AR10 ∧ AR2)</td>
</tr>
<tr>
<td>Job Title</td>
<td>(l1)</td>
<td>Soft Engg.</td>
<td>(R2, AR10, AR2)</td>
</tr>
<tr>
<td>Job Profile</td>
<td>(l1)</td>
<td>Resp.html</td>
<td>(R2, AR10, AR10)</td>
</tr>
<tr>
<td>Salary</td>
<td>(l1)</td>
<td>50k</td>
<td>(R2, AR2)</td>
</tr>
<tr>
<td>Benefits</td>
<td>(l1)</td>
<td>Benefit.hn</td>
<td>~</td>
</tr>
<tr>
<td>Bonus</td>
<td>(l2)</td>
<td>Yes</td>
<td>(R2, AR10)</td>
</tr>
<tr>
<td>Visa sponsorship</td>
<td>(l4)</td>
<td>Yes</td>
<td>~</td>
</tr>
<tr>
<td>Start Date</td>
<td>(l4)</td>
<td>M/D/Y</td>
<td>R1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Alias</th>
<th>Value</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company Name</td>
<td>(l1)</td>
<td>CDE Inc</td>
<td>(R3)</td>
</tr>
<tr>
<td>Job Location</td>
<td>(l1)</td>
<td>XYZ OH</td>
<td>(R3)</td>
</tr>
<tr>
<td>Job Title</td>
<td>(l1)</td>
<td>Soft Engg.</td>
<td>(R3, AR7)</td>
</tr>
<tr>
<td>Job Profile</td>
<td>(l1)</td>
<td>Resp.html</td>
<td>(R3, AR5, AR3)</td>
</tr>
<tr>
<td>Salary</td>
<td>(l1)</td>
<td>30k</td>
<td>(R3, AR2, AR4)</td>
</tr>
<tr>
<td>Benefits</td>
<td>(l4)</td>
<td>Benefit.hn</td>
<td>~</td>
</tr>
<tr>
<td>Bonus</td>
<td>(l4)</td>
<td>No</td>
<td>~</td>
</tr>
<tr>
<td>Visa sponsorship</td>
<td>(l4)</td>
<td>Yes</td>
<td>~</td>
</tr>
<tr>
<td>Start Date</td>
<td>(l4)</td>
<td>M/D/Y</td>
<td>~</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Alias</th>
<th>Value</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interview</td>
<td>(R1)</td>
<td>Yes</td>
<td>(I1 ∧ I3 ∧ I10)</td>
</tr>
<tr>
<td>Name</td>
<td>(R2)</td>
<td>Sajid</td>
<td>~</td>
</tr>
<tr>
<td>Email</td>
<td>(R3)</td>
<td><a href="mailto:sajid@k.edu">sajid@k.edu</a></td>
<td>(I1 ∧ I1)</td>
</tr>
<tr>
<td>Cell Phone</td>
<td>(R4)</td>
<td>123123123</td>
<td>(I4)</td>
</tr>
<tr>
<td>SSN</td>
<td>(R5)</td>
<td>0000000000</td>
<td>(I1, I5, I1)</td>
</tr>
<tr>
<td>School</td>
<td>(R6)</td>
<td>KSU</td>
<td>(I1)</td>
</tr>
<tr>
<td>Major</td>
<td>(R7)</td>
<td>Comp-Sci</td>
<td>(I1 ∧ I11)</td>
</tr>
<tr>
<td>Experience</td>
<td>(R8)</td>
<td>Expr.html</td>
<td>(I1 ∧ I5 ∧ V(I1))</td>
</tr>
<tr>
<td>Transcripts</td>
<td>(R9)</td>
<td>Trans.html</td>
<td>(I1 ∧ I4 ∧ L1)</td>
</tr>
<tr>
<td>Publications</td>
<td>(R10)</td>
<td>Publ.html</td>
<td>~</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Alias</th>
<th>Value</th>
<th>Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interview</td>
<td>(R1)</td>
<td>Yes</td>
<td>(I1 ∧ I3)</td>
</tr>
<tr>
<td>Name</td>
<td>(R2)</td>
<td>Sajid</td>
<td>~</td>
</tr>
<tr>
<td>Email</td>
<td>(R3)</td>
<td><a href="mailto:sajid@k.edu">sajid@k.edu</a></td>
<td>(I1 ∧ I1)</td>
</tr>
<tr>
<td>Cell Phone</td>
<td>(R4)</td>
<td>123123123</td>
<td>(I4)</td>
</tr>
<tr>
<td>SSN</td>
<td>(R5)</td>
<td>0000000000</td>
<td>(I1, I5, I1)</td>
</tr>
<tr>
<td>School</td>
<td>(R6)</td>
<td>KSU</td>
<td>(I1)</td>
</tr>
<tr>
<td>Major</td>
<td>(R7)</td>
<td>Comp-Sci</td>
<td>(I1 ∧ I11)</td>
</tr>
<tr>
<td>Experience</td>
<td>(R8)</td>
<td>Expr.html</td>
<td>(I1 ∧ I5)</td>
</tr>
<tr>
<td>Transcripts</td>
<td>(R9)</td>
<td>Trans.html</td>
<td>(I1 ∧ I4 ∧ L1)</td>
</tr>
<tr>
<td>Publications</td>
<td>(R10)</td>
<td>Publ.html</td>
<td>~</td>
</tr>
</tbody>
</table>

Figure 14. Access Control Policies of ABC Inc/John, CDE Inc/Yang, KLM Inc/Bob, Alice, Sajid, and Pooja respectively from left-right and top-bottom

4.1 Examples - SEAN Algorithm
This section illustrates two step-by-step negotiations when Negotiation Agent of Bob’s company initiates a search for potential candidates to fill-up the vacancy under Bob. First example describes how Negotiation Agent of Bob’s company KLM negotiations with the negotiation agent of Alice’s host institution KSU to successfully schedule an interview and second example describes unsuccessful negotiation between Negotiation Agent of Bob’s company KLM and negotiation agent of Sajid’s host institution KSU to schedule an interview.
4.1.1 Successful Negotiation

Figure 15. A) Messages exchanged during negotiation between Bob and Alice to schedule an Interview using Eager strategy. Messages are identified by numbers from 1-6. B) At the end of Negotiation, Agent will generate list of resources received for KLM/Bob. C) At the end of Negotiation, Agent will generate list of resources received for Alice.

Bob's and Alice's access control policies are shown in figure 14. Bob’s agent will start negotiation by requesting Interview (i.e. REQ¹(R₁)) from Alice’s agent and as per eager strategy, Bob's agent will offer attributes (i.e. OFF¹(I₆, I₀)) as shown in message 1 of figure 15 A).
Once Alice's agent receives message 1, according requested attribute's i.e. \textit{interview} disclosure rule, it will change state of negotiation of \textit{Interview} to pending (i.e. \textit{PEN}^R (R_1)). Agent also offers attributes (i.e. \textit{OFF}^R (R_2, R_7, R_10)), whose disclosure rule is satisfied by Bob's offered attributes (i.e. I_6 and I_9), as shown in message 2. Here notice that, Alice's agent keep Bob's offered attributes as it is from message 1 to message 2. This way negotiation proceeds and at the end of negotiation, agent stores negotiation information in terms of resources released and received for their respective user as shown in figure 15 B) and 15 C).

4.1.2 Unsuccessful Negotiation

Bob's and Sajid's access control policies are shown in figure 14. Bob’s agent will start negotiation by requesting \textit{Interview} (i.e. \textit{REQ}^I (R_1)) from Bob’s agent and as per eager strategy, Bob's agent will offer attributes (i.e. \textit{OFF}^I (I_6, I_9)) as shown in message 1 of figure 16 A).

Here, negotiation proceeds as explained in previous example. But when Sajid's agent finds that offered attributes in message 3 and message 4 are same, which indicates that Sajid has no new resources to offer in negotiation, it terminates negotiation. At the
Figure 16. A) Messages exchanged during negotiation between Bob and Sajid to schedule an interview using Eager strategy. Messages are identified by numbers from 1-4. B) At the end of Negotiation, Agent will generate list of resources received for KLM/Bob. C) At the end of Negotiation, Agent will generate list of resources received for Sajid.

end of negotiation, agent stores negotiation information in terms of resources released and received for their respective user as shown in figure 16 B) and 16 C).

4.1.3 Negotiation Results

Table 3 summarizes results of 9 negotiations between three managers and three students using SEAN algorithm. Negotiation result could be either successful (Y) or unsuccessful (N) negotiation. At the end of successful or unsuccessful negotiation, negotiation agent will generate list of resources received and released for its user. Figure 17 shows resources received by Yang, John, Bob, Alice, Pooja, and Sajid respectively.
Figure 17. It shows resources received by three managers' namely Yang, John, Bob, and three students' namely Alice, Pooja and Sajid from left-right, top-bottom.
Table 3. SEAN Negotiation Result

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Pooja</th>
<th>Sajid</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLM Inc/Bob</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>ABC Inc/John</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>CDE Inc/Yang</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

4.2 Examples - SPAN Algorithm

This section illustrates two step-by-step negotiations when Negotiation Agent of Bob’s company initiates a search for potential candidates to fill-up the vacancy under Bob. First example describes how Negotiation Agent of Bob’s company KLM negotiations with the negotiation agent of Alice’s host institution KSU to successfully schedule an interview and second example describes unsuccessful negotiation between Negotiation Agent of Bob’s company KLM and negotiation agent of Sajid’s host institution KSU to schedule an interview.

4.2.1 Proxy Negotiation

ABC’s and Alice’s access control policies are shown in figure 14. Figure 18 shows messages exchanged during negotiation between ABC Inc and Alice.

ABC’s agent will start negotiation by requesting Interview (REQ\(^1\)((R_1, \emptyset)) from Alice’s agent as shown in message 1.\(\emptyset\) indicates that \(R_1\), being requested for the first time, has no counter request (CQ). Bottom portion of this message i.e. Alice’s dataspace is empty. In response to message 1, Alice’s agent will counter request Company\_name (REQ\(^R\)(I_1, \emptyset)) as shown in message 2. It also changes state of negotiation of Interview from requesting to pending i.e. PEN\(^R\)(\{R_1, I_1\}).
Figure 18. A) Messages exchanged during a negotiation between ABC Inc and Alice to schedule an Interview using Proxy strategy. Messages are shown from 1-12. B) Dependency Graph resulted from negotiation C) At the end of Negotiation; each SA will store resources information for its user indicating attributes released and received.

In response to message 2, ABC’s agent will release *Company_name* i.e. $\text{AVL}^1(\{1_i, \sim\})$ and changes state of negotiation of *Interview* from pending to requesting ($\text{REQ}^1(\{R_i, I_i\})$).
in message 3. Note that, in message 2 CQ for R₁ was I₁; but I₁ is offered by ABC’s agent in message 3 and so in message 4 CQ for R₁ is I₃.

This way negotiation proceeds and in message 12, Alice’s agent will offer Interview AVLₐ (R₁, R₈, I₁, I₃) to ABC’s agent. In this offer, R₁ is released because of a clause that contains I₁ and I₃ attributes. After this each agent will generate Dependency Graph as show in 18 B) and will start exchanging actual attribute values according to generated graph. Dependency graph is set of nodes and edges. Edge goes from node A to B, when A causes release of B. Only initial node is empty i.e. ~

Finally, each agent will generate negotiation result for their participants as shown in figure 18 C). This result shows released and received attributes by each participants at the end of successful negotiation.

4.2.2 Proxy Negotiation with backtracking

In this section we will explain negotiation between KLM and Pooja which proceeds in the same way as previous one, but with backtracking. Message 3 shows request of R₇ i.e. REQI (R₁, I₃) and as per Pooja’s policy it needs I₃ to unlock R₇, but state of negotiation of I₃ is already pending and it requires R₇ to unlock. This creates cycle. To resolve this, Pooja’s agent will deny R₇ i.e. DENR (R₇, 0) as shown in message 4. Here, 0 indicates total number of attributes released by KLM and Pooja when R₇ is denied. In response to message 4, KLM’s agent will find out that R₇ is counter request of I₃. So even though R₇ is denied, I₃ can be unlocked using R₂. So KLM’s agent will request R₂ as shown in message 5.
Figure 19. A) Messages exchanged during a negotiation between KLM Inc and Pooja to schedule an interview using Proxy strategy. Messages are shown from 1-12. B) Dependency Graph resulted from negotiation C) At the end of Negotiation; each SA will store resources information for its user indicating attributes released and received.

This process of searching an alternative path to unlock I3 and thus breaking cyclic dependency is called as backtracking.

In backtracking, a denied attribute is requested again only if numbers of attributes released by both participants are more than number of attributes released by both
participants when same attribute was denied last time. As per this rule \( R_7 \) is again requested by KLM’s agent in message 9. Here, number of attributes released (i.e. 1) in message 8 are more than number of released attributes (i.e. 0) when \( R_7 \) was denied in message 4. Finally, in message 12 Pooja’s agent offers Interview i.e. \( AVL^R (\{R_2, ~\}, \{R_7, I_3\}, \{R_1, \{I_3, I_1\}\} \) to KLM’s agent to finish first phase of negotiation. In second phase, agents will exchange actual values for negotiated attributes using dependency graph as shown in figure 19 B). At the end of second phase, agents will generate negotiation result for their participants as shown in 19 C).

### 4.2.3 Negotiation Results

Table 4 summarizes results of 9 negotiations between three managers and three students using SPAN algorithm. Negotiation result could be either successful (Y) or unsuccessful (N) negotiation.

<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Pooja</th>
<th>Sajid</th>
</tr>
</thead>
<tbody>
<tr>
<td>KLM Inc/Bob</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>ABC Inc/John</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>CDE Inc/Yang</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

At the end of successful, negotiation agent will generate list of resources received and released for its user. Figure 20 shows resources received by three students namely Alice, Pooja, Sajid and three managers’ namely Yang, John, and Bob.
### Figure 20

It show resources received by three students’ namely Alice, Pooja, Sajid as well as three managers’ namely Yang, John, and Bob from left-right, top-bottom.

### 4.3 Examples – KBAN Strategy

In Knowledge Based Attribute Negotiation, two parties learn about negotiators policies and use that knowledge either in same negotiation or future negotiations to reduce a number of messages exchanged between them.
As explained in section 4 every message carries negotiation state information. In addition to this state information, for KBAN we have defined following two properties for attributes in negotiation. These properties are assigned and removed by either of the party during negation based on Negotiation-Knowledge.

1. **ARO** (Assumed releasable from opposite party)

   This property of an attribute/credential indicates that requesting party is expecting opposite party to release her attribute based on Negotiation-Knowledge.

2. **ARR** (Advanced ready to release)

   This property indicates that a party is ready to release her attribute/credential even it is not requested, but Negotiation-Knowledge indicates that opposite party will need this attribute for next immediate release. Negotiation message will keep track of ARR and opposite parties’ immediate release.

### 4.3.1 Sequential KBAN

![Diagram](image)

**Figure 21.** Messages exchanged during a negotiation between ABC Inc and Pooja to schedule an Interview using Proxy strategy based on sequential KBAN.
Here we assume that two negotiations between ABC Inc/John and Alice as well as ABC Inc/John and Sajid have been executed and those negotiations are used as Negotiation Knowledge for future next negotiation.

John is requesting interview REQI [(R1, Ø)] from Pooja as shown in message 1 of figure 21. To release R1, Pooja needs I3 from John, but Negotiation-Knowledge (from negotiation 4.2.1 and 7.2.2) shows that John needs R2 and R10 to release I3. Pooja can release R2 (from policy of R2) and indicates that in message 2 i.e. AVLR (R2 ARR, ~) using ARR property. Now assuming R2 is releasable, John needs R10 and Pooja needs I1 to release it. But again, Negotiation-knowledge shows that I1 is releasable. So Pooja indicates that in message 2 as AVL R ((R10 ARR, I1ARO)) and also request that REQR (I1 ARO ). This tells john that Pooja is ready to release R10 but assuming that I1 will be released in next message. At this point, I3 is releasable in next message based on Negotiation-Knowledge and so attribute after I3 is I1 which is already requested REQR(I1 ARO ) and expected to be releasable in next message. So R1 is also becomes releasable and Pooja indicates this by ARR in message 2 as (R1ARR, {I3ARO,I1ARO}). This tells John that R1 is advancing releasable (by Pooja) assuming I3 and I1 are releasable by John. In 3rd message John indicate that I1 and I3 are Available to release without any condition [AVLI [(I1,~) (I3,{R2 ,R10}) ] but based on commitment from Pooja that R2 and R10 are available to release i.e. AVLR [(R2 ARR, ~) (R10 ARR, I1ARO )]Once Pooja received this message, she verifies that I1 and I3 are Available to release without any conditions. At this stage negotiation has become successful and Pooja removes ARR and ARO property from credentials as shown in message 4.
4.3.2 Parallel KBAN

In this example, University (as a community) negotiates on behalf of three students namely Alice, Pooja, and Sajid with a Company on behalf of three managers namely John, Yang, and Bob. This example, for simplicity, has three one-to-one parallel negotiations instead of nine negotiations. Each message represents three negotiations - one in a row - between Alice-John, Pooja-John and Sajid-John respectively.

Initially Company requests an Interview from University in message 1. University resolves this message to change state of three internal requests to pending and also creates three requests as \( \text{REQ}^R(I_1, \emptyset) \) i.e. Alice needs Company_Name , \( \text{REQ}^R(I_3, \emptyset) \) i.e. Pooja needs Job_Title, and \( \text{REQ}^R(I_2, \emptyset) \) i.e. sajid needs Job_Location from John. Company responds back for these three internal requests in message 3.

Message 4th is interesting as it uses Negotiation Knowledge for Sajid-John negotiation. In 3rd message, university receives a third internal request, \( \text{REQ}^I(\{R_1, I_2\}, \{R_3, \emptyset\}) \), for \( R_3 \) from Sajid and to release it sajid needs \( I_1 \) and \( I_3 \). But from 3rd message it can be learned that \( I_1 \) is already releasable i.e. \( \text{AVL}^I(\{I_1, \sim\}) \) from John and can be assumed releasable in next message. So there is no need of separate request for \( I_1 \) but \( I_3 \).

This state of \( I_1 \) and \( I_3 \) is communicated to the opposite party as \( \text{PENR}\{R_3, (I1ARO, I3)\} \), in message 4. Also from 3rd message it can be learned that \( I_3 \) needs \( R_2 \) - \( \text{PENI}\{I3, R2\} \) and so sajid is ready-to-release it in advance i.e. \( \text{AVLR}\{R2ARR, \sim\} \) in message 4.
Figure 22. Messages exchanged during three negotiations between Alice-John, Pooja-John, and Sajid-John (one negotiation per row in each message) to schedule an Interview using Proxy strategy based on Parallel KBAN
Once company receives 4th message (will discuss only Sajid-John negotiation), it processes PENR (\{R1, I2\}, \{R3, (I1ARO,I3)\}). Here \{R3, (I1ARO ,I3)\} indicates that sajid is expecting I1 as released from John and requesting I3. So John releases it as AVL I(I1,~) in message 5. John needs R2 to release I3, but AVLR(\{R2ARR, ~\}) in message 4 indicates that Sajid is ready-to-release R2 so instead John requests R10 in message 5. In message 6, ARR property of R2 is removed.

Because of Negotiation-Knowledge in message 4 and 5, a number of messages exchanged between Sajid and John are reduced by 4 in comparisons with Sajid-John SPAN negotiation in 7.2.2. Also Negotiation-Knowledge reduces a number of message exchanged between Pooja and John by 2 in comparison with Pooja-John SPAN negotiation in 7.2.1

4.3.3 Negotiation Results

Negotiation results in table 5 shows that KBAN-Parallel and KBAN-Sequential are efficient compare to SPAN in-terms of number of messages exchanged between John and Pooja during a negotiation.

Table 5. Knowledge Based Attributed Negotiation Results

<table>
<thead>
<tr>
<th>Negotiating Parties</th>
<th>Algorithm</th>
<th>Number of Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC Inc (John) - Pooja</td>
<td>SPAN</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>KBAN- Parallel</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>KBAN- Sequential</td>
<td>04</td>
</tr>
</tbody>
</table>
4.4 Analysis of SEAN and SPAN

In this section, table 6 shows negotiation results between three organizations, and three students. Results are based on three important parameters namely Number of Rules fired, messages exchanged, and attributes released in a Negotiation using SEAN, and SPAN algorithm.

<table>
<thead>
<tr>
<th>Negotiating Parties</th>
<th>Negotiation Result</th>
<th>Number of Rules fired in a Negotiation</th>
<th>Number of Messages exchanged in a Negotiation</th>
<th>Number of attributes released in a Negotiation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Stateless Eager</td>
<td>Stateless Proxy</td>
<td>Stateless Eager</td>
</tr>
<tr>
<td>ABC-Alice</td>
<td>Success</td>
<td>10</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>ABC-Pooja</td>
<td>Success</td>
<td>9</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>ABC-Sajid</td>
<td>Success</td>
<td>14</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>CDE-Alice</td>
<td>Success</td>
<td>15</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>CDE-Pooja</td>
<td>Fail</td>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CDE-Sajid</td>
<td>Success</td>
<td>9</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>KLM-Alice</td>
<td>Success</td>
<td>10</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>KLM-Pooja</td>
<td>Success</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>KLM-Sajid</td>
<td>Fail</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

In every successful negotiation, SEAN releases more attributes as well as fires more number of rules than SPAN but with fewer messages. In unsuccessful negotiation like CDE-Pooja, and KLM-Sajid, SEAN releases attributes, but SPAN doesn’t release even single attribute.

This result shows importance of choosing appropriate negotiation flavor by federation members’ on-per-need basis. Using this framework, federated members eager to reach a deal can opt for SEAN at the cost of disclosing more attributes while cautious users, for example having more number of sensitive attributes, can opt for SPAN to avoid unnecessary disclosure of attributes.
Communication Complexity of SEAN and SPAN is shown below in table 7.

**Table 7. Communication Complexity**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Worst case Communication Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of messages</td>
</tr>
<tr>
<td>SPAN</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>SEAN</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>
CHAPTER 5

Conclusion

We have contributed a novel Negotiation Enabled Framework to enhance federated members’ privacy by providing Individualized Policies and Personalized Negotiation flavors. This Framework is supported by a negotiation protocol and two negotiation algorithms namely Stateless Eager Attribute Negotiation (SEAN) Algorithm and Stateless Proxy Attribute Negotiation (SPAN) algorithm. Negotiation protocol defines ordering of the messages and unique message structure that carries negotiation information. In algorithms, SEAN is optimum in the sense that it results in successful negotiation whenever possible in minimum number of communication steps. While SPAN is optimum in the sense that it discloses only necessary attributes for negotiation when both parties are sure that successful negotiation is possible. In addition to negotiation algorithms, we have contributed an efficient negotiation strategy called Knowledge Based Attribute Negotiation (KBAN). KBAN reduces number of communication steps in bipartite negotiation.

The privacy implications of this work are quite interesting. With previous systems like Shibboleth a member was no longer at the mercy of the foreign (Service provider) organization about the disclosure policy, but still was at the mercy of the on-size-fits-all release policy set by the home organization. Through this work the individual’s privacy is further enhanced. A member is now not at the mercy of home organization. Through this model the home organization acts more as a negotiation agent on behalf of the member-
than as her policy setter. If any organization still wants to add additional policies, then that is also enforceable just as a second policy tier. Interestingly, the work also enhances home organization’s flexibility. Now even home organization can set dynamic and targeted release policies, and let it resolve dynamically.
REFERENCES


APPENDIX A

Examples – SEAN Algorithm

5.1 Negotiation between KLM Inc/Bob and Pooja [DEAL]

Figure 23. Messages exchanged in a negotiation between KLM Inc and Pooja to schedule an Interview using SEAN
5.2 Negotiation between ABC Inc/John and Pooja [DEAL]

Figure 24. Messages exchanged in a negotiation between ABC Inc and Pooja to schedule an Interview using SEAN
5.3 Negotiation between ABC Inc /John and Sajid [DEAL]

Figure 25. Messages exchanged in a negotiation between ABC Inc and Sajid to schedule an Interview using SEAN
5.4 Negotiation between CDE Inc /Yang and Pooja [NO-DEAL]

![Diagram of negotiation between CDE Inc /Yang and Pooja]

Figure 26. Messages exchanged in a negotiation between CDE Inc and Pooja to schedule an Interview using SEAN

5.5 Negotiation between ABC Inc/John and Alice [DEAL]

![Diagram of negotiation between ABC Inc/John and Alice]

Figure 27. Messages exchanged in a negotiation between ABC Inc and Alice to schedule an Interview using SEAN
5.6 Negotiation between CDE Inc/Yang and Sajid [NO-DEAL]

CDE Inc/Yang (Initiator)       Sajid (Responder)

1. OFF$^1(I_6, I_9, I_{10})$
2. PEN$^R(R_1)$
3. REQ$^1(R_1)$
4. OFF$^1(I_6, I_9, I_{10}, I_1, I_2)$

Figure 28. Messages exchanged in a negotiation between CDE Inc and Sajid to schedule an Interview using SEAN.

5.7 Negotiation between CDE Inc/Yang and Pooja [NO-DEAL]

CDE Inc/Yang (Initiator)       Pooja (Responder)

1. OFF$^1(I_6, I_9, I_{10})$
2. PEN$^R(R_1)$
3. REQ$^1(R_1)$
4. OFF$^1(I_6, I_9, I_{10})$

Figure 29. Messages exchanged in a negotiation between CDE Inc and Pooja to schedule an Interview using SEAN.
APPENDIX B

Examples – SPAN Algorithm

5.8 Negotiation between ABC Inc/John and Pooja [DEAL]

Figure 30. Messages exchanged in a negotiation between ABC Inc and Pooja to schedule an Interview using SPAN.
5.9 Negotiation between ABC Inc/John and Sajid [DEAL]

Figure 31. Messages exchanged in a negotiation between ABC Inc and Sajid to schedule an Interview using SPAN
5.10 Negotiation between CDE Inc/Yang and Sajid [DEAL with Backtracking]

Figure 32. Messages exchanged in a negotiation between CDE Inc and Sajid to schedule an Interview using SPAN
Figure 33. Messages exchanged in a negotiation between CDE Inc and Alice to schedule an Interview using SPAN
5.12 Negotiation between CDE Inc /Yang and Pooja [NO-DEAL]

Figure 34. Messages exchanged in an unsuccessful negotiation between CDE Inc and Pooja to schedule an Interview using SPAN
5.13 Negotiation between KLM Inc/Bob and Sajid [NO-DEAL]

Figure 35. Messages exchanged in an unsuccessful negotiation between KLM Inc and Sajid to schedule an Interview using SPAN
5.14 Negotiation between KLM Inc/Bob and Alice [DEAL]

Figure 36. Messages exchanged in a negotiation between KLM Inc and Alice to schedule an Interview using SPAN