An XML-based Protocol for Improving Trust Negotiation between Web Services

Yunxi Zhang and Tanko Ishaya
Internet Computing, University of Hull, Scarborough Campus,
Filey Road, Scarborough, YO11 3AZ, UK
Tel: +44 (0) 1723 357235
Yunxi.Zhang@2007.hull.ac.uk & T.Ishaya@hull.ac.uk

ABSTRACT

This paper aims to propose an XML-based protocol for two Web Services to utilise TN to establish a trust relationship between them. The main contribution of this protocol is towards preventing failed TN caused by the file format interoperability problem by checking file formats before TN processes. This will increase communication efficiency between WS.

Categories and Subject Descriptors

D.2.12 [Software Engineering]: Interoperability – distributed objects, interface definition languages, data mapping.

General Terms

Design, Reliability, Verification, Security

Keywords

Trust Negotiation, Web Services, XML-based Protocol, File Format Interoperability

1. INTRODUCTION

Trust Negotiation (from now on referred to as TN in this paper) has been developing for more than one decade. The original notion of TN was proposed by [21] as an innovative approach to allow two unknown entities to establish a trust relationship by exchanging their credentials. However, with the development of TN, credentials are no longer the only information transmitted between negotiating parties. Further information such as policies may also be exchanged as other requirements for TN. Thus, TN can be described as: a selected set of information including credentials (or even a partial attributes within credentials), declarations, policies or other needed information exchanged between entities to establish a trust relationship in a bilateral, iterative and cumulative process [24, 25, 28].

A variety of conceptual models and systems using TN have been incorporated into different kinds of open distributed systems such as Web Services [18, 19], Semantic Web [13, 14, 29], P2P systems [23], mobiles and pervasive computing [5, 10] and general open systems such as the Internet.

In Web Services (WS), obviously, the concept of TN can be adopted as an approach for establishing a trust relationship between two WS. However, a uniform protocol for supporting TN in WS is yet to be properly developed. Furthermore, although various file formats (i.e. credential, declaration and policy files) have been designed for TN, the file format interoperability problem continues to be a challenge. Therefore, approaches for the establishment of a trust relationship are required to further support communication efficiencies between WS. This paper aims to propose an XML-based protocol for two WS to utilise existing TN approaches to establish a trust relationship between them. The aim of the protocol is to facilitate a commonly accepted file format procedure for two WS to commit to and use for an improved TN process. The main contribution of this protocol is towards preventing failed TN caused by the file format interoperability problem by checking file formats before TN processes. The protocol has been implemented and has shown an increase in communication efficiency between WS.

The structure of remainder of this paper is organised as follows: Section 2, presents a review of approaches used in the provision of TN. Section 3 defines a case scenario being used in developing a proposed protocol. Section 4 describes the protocol, which is implemented in Section 5. Section 6 concludes the paper with a description of further work being carried out.

2. A REVIEW OF TN APPROACHES

The literature on TN has been presented to highlight different approaches currently being used. The basic components of TN as covered in a variety of literature have been presented as follows:

∞ **Strategy**: is to help two negotiators to determine the best way to exchange necessary information required to reach an agreement [21, 22].

∞ **Digital Credentials** (short for credentials): is a representation of the combination of attributes of the owner, which can be signed using the owner's private key and be verified by the owner's public key [16].

∞ **Declaration**: which contains extra information not included in the credentials to aid the process of TN [1].

∞ **Policy**: there are two notions of policy: access control policy and disclosure policy [1]. An access control policy claims the rules to grant the permission to access the requested resource. A disclosure policy states the rules for disclosing credential(s) to the opponent [6].

∞ **Protocol and standard**: the uniform syntax used to allow negotiators to distinguish the real exchanged information (such as credentials, policies), which can aid in easy incorporation of new negotiators to join the existing TN networks without consulting individual transmitting formats for any two negotiators [12].

In addition to these main aspects, two extra constellations (conceptual frameworks/models and implementations/real systems) are introduced to support the process of the entire TN.

2.1 Strategy

The initial two strategies for TN are eager strategy and parsimonious strategy [21]. The eager strategy allows participating parties to exchange as many as possible credentials required to reach a successful negotiation without the concern about whether they are relevant or not. On the other hand, the goal of the parsimonious strategy is to disclose credentials to a minimum due to the need for protecting the sensitive information within them. These strategies are defined in more detail in [22] for the purpose of managing the exchange of credentials.

A new strategy named Prudent Negotiation Strategy (PRUNES) was developed to guarantee that all the potential success of TN will not be missed [24]. Policy graphs and two automated strategies (relevant credentials set strategy and all relevant policies strategy) were presented to reduce the probability of the disclosure of sensitive policies [15]. Seamons et al [17] identified a new approach disclosing sensitive information of a negotiation party based on its behaviour reacting to the requests for certain credentials from the opponent. Two strategies were designed by [26] for operating with UniPro policies. He and Zhu [3] argued that previous strategies may fail unnecessarily with high communication or computation costs, and then proposed a strategy based on Negotiation Petri Net (SNPN) by integrating TN into the negotiation Petri Net architecture to make up these weaknesses. In order to compensate the flaw in [3], a MSC strategy was proposed by [4] to enable the SNPN to be able to choose the minimum cost disclosure sequence.

2.2 Credentials

An informal credential language was developed by [22] consisting of two parts named Property-based Authentication Language (PAL) and Role-based Authorisation Language (RAL). Winsborough and Li [27] identified that prior assumptions about credential languages are too simplified to fulfil the use in practice. Therefore, they introduced the RT_0, a member of a family of role-based trust management languages. X-TNL, an XML-based language serving TN for Web was presented by [1]. An Attributed-based Trust Negotiation Language (ATNL) that can specify credentials based on RT has been provided in [11], a family of Role-based Trust management languages and a protocol designed as an extension of TTG [27].

2.3 Declaration

A notion of declaration was presented in an XML format called X-TNL declaration [1]. It can store personal information to help to establish a trust relationship for negotiators more effectively. The main difference between an X-TNL credential and an X-TNL declaration is that the former one has to be certified by a trusted third party but the latter one does not.

2.4 Policy

The notion of the policy graph that is an acyclic graph with a node representing a credential disclosed by the opponent and a node representing the expected resource (e.g. credential or policy) has been introduced in [16]. To prevent the fact of possession-sensitive/non possession-sensitive credentials from being inferred by the opponent mentioned in [17], the notion of Attribute acknowledgement policies (Ack policies) as an alternative approach was recommended in [27]. X-TNL, an XML-based language used in TN for Web, can represent the policy languages cooperating with X-TNL certificates (credentials and declarations) [1]. To improve the efficiency of previous compliance checkers, [7] developed CLOUSEAU, a policy language compliance checker for TN based on the platform TrustBuilder2. PeerTrust, a language, to present access control policies for semantic webs for a distributed elearning environment in particular was suggested by [13]. The Attributed-based Trust Negotiation Language (ATNL) specifying policies has been described in [11]. Olmedila et al [14] developed a metapolicy language, which can specify policy filtering and credential search. An extension of WS-related policy frameworks was suggested to enable them to specify TN policies in WS [8].

2.5 Protocol and Standard

A protocol for TrustBuilder, a framework for TN, was introduced in [25], in order to address the strategy interoperability. Within the protocol, a message is a disclosure of a set of local credentials, local policies or a local resource. A failure message is defined if a message is an empty set. The notion of Trust Target Credentials (TTG) protocol to support credential languages, Ack policies and distributed credential storage is present in [27]. An Extended Trust Target Graph (ETTG) protocol was proposed by [11] as an extension of TTG in [27] to support ATNL and the cryptographic protocols by adding the characteristics of ATNL policy languages to ETTG. Another protocol for TN, which adopted two operators for existing policy languages, OSBE schemes to address cyclic dependency on policies and suggested two novel concepts named policy credentials and deriving information for resource has been recommended in [12].

2.6 Current Conceptual Frameworks for TN

Olmedila et al [14] presented their architecture for a service requester and a service provider to determine the likelihood of a proper trust establishment at run time for Semantic WS. A TN scheme for mobiles [10] requires that each protocol should be linked to a strategy.

2.7 Existing Implementations of TN

TrustBuilder is the first real system designed to support basic functionalities (credential management, policy development and negotiation strategies) for TN in open distributed systems such as Web [29]. Trust-Serv is a framework [18, 19] for supporting TN within WS specifically. Ye et al [23] suggested their model to implement TN in Peer-to-Peer (P2P) systems. Squicciarini et al [20] introduced the system named PP-Trust-X, an extension of Trust-X to support P3P. TrustBuilder2, a reconfigurable framework was proposed and developed by [9]. PROTUNE, a rule-based TN system in support of functionalities such as deployment efforts, user friendliness, communication efficiency and interoperability has been presented in [2].

2.8 Problems in TN

Obviously, concluded from the review above, a lack of a protocol for WS to support TN still exists. Moreover, since a multitude of files formats (i.e. credential, declaration and policy formats) have been designed for TN, the file format interoperability/unreadability problem is undoubtedly becoming an important issue to be addressed. Otherwise, even if two WS own the correct credentials, declaration and policy files, they can still not use TN to establish a trust relationship as a result of the inability to read different file formats.

3. A WS CASE SCENARIO

In this section, a scenario is presented to describe how the notion of TN can be adopted within WS. Two universities in the UK, *University A* and *University B* run their portals (*Portal A* and

Portal B as WS) to provide a virtual learning environment for their students to access learning materials (e.g. lecture notes, books, a student's assignments for a course or a student's personal design of systems etc.). Both universities' portals allow any UK university students to access these learning materials upon the verification of the credentials representing the student ID. Alice, a postgraduate student of *University A* studying in the department of Computer Science, is intending to read the notes on 'Trust Negotiation' provided by Bob, a PhD student of *University B*. Bob is willing to show these notes to others, but considering the fact that the content of the notes are unlikely to be understood by students other than those studying at the similar level in the department of Computer Science, and the criterion that Bob wishes to evaluate whether a student is eligible to read the notes is based on the class of the student's bachelor degree. Thus, he decides to declare two policies specifying that first, the requester must come from the department of Computer Science; second, the student must own a credential representing a bachelor's degree with first class honours. Alice is willing to disclose her student ID credential, but is reluctant to submit her bachelor's degree information to strangers unconditionally even though she does possess a bachelor's degree with first class honours. Hence, she also asserts a policy that credentials authenticating the opponent's identity of a student in the department of Computer Science of a UK university has to be disclosed first before she divulges her bachelor's degree. Fortunately, Bob also holds a credential of his student ID and is willing to show this credential without any restrictions.

The description of the entire process of TN (divided into eight steps after Alice logs into *Portal A*) is presented as follows:

Step 1: Through *Portal A*, Alice finds out that there are notes on "Trust Negotiation" supplied by Bob belonging to *Portal B* that are of interest to her. Upon Alice choosing the notes and clicking on the submit button, *Portal A* sends the request to *Portal B* for a permission to access Bob's notes with a credential representing *Portal A*.

Step 2: Upon retrieving the message from *Portal A* with its credential, *Portal B* verifies the credential to determine that it is a request to access Bob's resource; then it identifies that Bob has already implemented two policies for protecting the access to his notes. So *Portal B* sends back a message to *Portal A* containing Bob's first policy requesting a credential demonstrating that the requester is a student in the department of Computer Science of a UK university.

Step 3: *Portal A* checks Alice's policies and finds out that Alice is willing to show her credential of the student ID containing the information that Alice is a student in the department of Computer Science of a UK university, and sends this credential to *Portal B*.

Step 4: *Portal B* verifies Alice's credential of the student ID to confirm that Alice is a UK university student and agrees that Bob's first policy has been satisfied by this credential. Then, it looks up in Bob's polices and discovers that there is a second policy demanding a bachelor's degree with first class honours. So, it sends a request to *Portal A* requesting such a credential.

Step 5: Upon receiving this request, *Portal A* does not send Alice's credentials that can represent her bachelor's degree with first honours immediately since a policy has been stated by Alice that an opponent's credential has to be submitted to prove that the requester must be a UK university student. Thus, a request containing Alice's policy is sent to *Portal B* for such a credential.

Step 6: *Portal B* validates Bob's policy to verify that Bob is willing to disclose his student ID credential to any one, so sends this credential to *Portal A*.

Step 7: When *Portal A* receives Bob's credential verifying that Bob is a UK university student, and he is a student studying in the department of Computer Science fulfilling Alice's policy, *Portal A* then discloses Alice's credential representing her bachelor's degree with first honours to *Portal B*.

Step 8: After verifying that Alice's credential can meet Bob's second policy and no further policies required by Bob, *Portal B* finally grants the permission to Alice to access Bob's notes and TN terminates in success.

However, if two WS use different file formats and they can only read their own file formats, TN process will terminate in step 2 as the credential file format sent from *Portal A* cannot be recognised by *Portal B*. Although they may use other approaches to establish a trust relationship, the communication efficiency will clearly be affected. The protocol we proposed cannot address the file formats unreadability problem thoroughly, but can prevent the occurrence of failed TN caused by such a problem. The proposed protocol will also increase the communication efficiency.

4. AN XML-BASED PROTOCOL FOR WS

Within this protocol, the design of attributes with specific values are preferred in each header element rather than the design of sub elements due to the consideration from the perspective of efficiency that a WS requester and a WS provider can check up on the values of attributes to determine what message to send in the next step more efficiently.

The whole protocol is devised into two stages of TN: preparation stage and negotiation stage. The rationale to separate the preparation stage from the negotiation stage is to guarantee that two WS can recognise and be agreed on a common credential, declaration and policy file format for the negotiation stage. The significance of a preparation process was noted in [30] that it could help to achieve a higher probability of a successful negotiation. As mentioned in the literature review, various credential and policy languages can be used to conduct TN. But within the field of WS, a WS provider cannot predict which WS requester will send a request to it to access a particular resource. Furthermore, in reality, a WS provider may not recognise all different file formats. Therefore, even though two WS do possess the correct credentials to satisfy the opponent's policies, TN may still fail due to the inability to read opponent's file formats. This will obviously influence the efficiency of trust establishment between WS, since two WS have to cease TN and to use another approach to establish a trust relationship. Thus, in order to avoid the unnecessary extra steps in potential failed TN, the preparation stage is designed to allow two WS to exchange information of their supported file formats to identify whether there is at least one common file format that can be recognised by both WS. If there is, then determine to opt for one file format to start TN; otherwise, give up using TN and look for another approach to establish a trust relationship.

4.1 Preparation Stage Protocol

The protocol within the preparation stage includes four header blocks for two WS to communicate to check whether there is at least one file format in common supported by the communicating parties. The first and second header blocks are leveraged for two WS to decide whether a default file format can be supported in common. The third and fourth header blocks are designed for two

WS to choose one of the common file formats when the default file format is not recognised by the WS provider.

4.1.1 RequestingAction

The <RequestingAction> header block is harnessed by a WS requester to initiate a TN request for a targeted resource owned by a WS provider. The following describes the attributes and elements.

/RequestingAction: this is the header block for a WS requester to send a request to inform a WS provider what it is intending to do.

/RequestingAction/@actionName: this attribute is to denote what the WS requester intends to do. The values in this protocol can be: *RR* (for Request Resource), AS (for Activate Service) or other values defined in a specific WS.

/RequestingAction/@approachForTrustEstablishment: this attribute is to suggest TN to be used to establish a trust relationship between two WS. The value can only be TN.

/RequestingAction/@defaultFileFormat: this attribute is to indicate the default file format the WS requester supports. The value of this attribute can be any format name present or future.

/RequestingAction/{any}: this mechanism provides extensibility to allow WS to define new elements or information to provide extensibility. A fault should be given if the elements are not recognised.

/RequestingAction/@{any}: this mechanism provides extensibility to allow WS to define any new attributes. A fault should be given if the attributes are not recognised.

Note: The symbol {any} and @{any} within all the header blocks express the same meaning.

Note: We recommend that the uniformity of the name of each specific file format be established so that it can be straightforward for both negotiators to determine whether they are able to support a file format or not. Otherwise, even though two negotiators do support the same file format, they may misunderstand that no common file format can be supported resulting in missing the opportunity to utilise TN if the name of the common supported file format known by two WS is different.

An example is shown below:

```
<RequestingAction defaultFileFormat=”X-TNL”
actionName=”RR”
approachForTrustEstablishment=”TN”>Requestin
g For some resources</RequestingAction>
```

4.1.2 RespondingAction

The <RespondingAction> header block is employed by the WS provider in response to the <RequestingAction> header block. The following describes the attributes and elements.

/RespondingAction: this element is generated by the WS provider to send back the information in response to the <RequestingAction> header block from the WS requester.

/RespondingAction/@TNIsSupported: this attribute is denoted to inform the WS requester whether TN can be supported. The value can only be yes or no. If the value is yes, then TN is possible to be used, since the file format has not been decided yet. If the value is no, a fault message should be sent back to the WS requester.

/RespondingAction/@defaultFileFormatIsSupported: this attribute is used to inform the WS requester whether the default file format in <RequestingAction> block can be supported. The value can only be yes or no. If the value is yes, then the default file format can be recognised by both WS. If the value is no, then the default file format cannot be supported by the WS provider.

/RespondingAction/Fault: this element is used to inform the WS requester that TN cannot be supported by the WS provider when the value of the attribute TNIsSupported is no.

/RespondingAction/RequestingFileFormat: this is an optional element to request other file formats supported by the WS requester when the value of the attribute TNIsSupported is yes and the value of the attribute defaultFileFormatIsSupported is no.

/RespondingAction/RequestingFileFormat/@otherFileFormatIsNeeded: this attribute is to request the WS requester whether there is(are) other file format(s) that can be supported. The value of this attribute is only yes, because if the WS provider can support TN and does not recognise the default file format, it means that the WS provider can recognise at least one other file format. Hence, the value must be yes.

/RespondingAction/{any}

/RespondingAction/@{any}

Note: If the value of both attributes TNIsSupported and defaultFileFormatIsSupported are both yes, then other two header blocks <SupportedFormatForTN> (see section 4.1.3) and <RespondingForSupportedFormatForTN> (see section 4.1.4) do not have to be used and TN can be getting started, since two WS are agreed on the use of TN and the same file format.

An example is shown below:

```
<RespondingAction TNIsSupported=”yes”
defaultFileFormatIsSupported=”No”>
        <RequestingFileFormat
        otherFileFormatIsNeeded=”yes”/>
</RespondingAction>
```

4.1.3 SupportedFormatForTN

The <SupportedFormatForTN> header block is used by the WS requester to send a list of other recognisable file formats to the WS provider to identify a common supported file format. The following describes the attributes and elements.

/SupportedFormatForTN: this is the header element used by the WS requester to send other supported file formats for TN.

/SupportedFormatForTN/@otherFileFormatIsExisted: this attribute is used to notify the WS provider of whether there is(are) any other file format(s) supported for TN by the WS requester. The value of it can only be yes or no. If the value is yes, other file formats should be sent to the WS provider. If the value is no, it means that no more file formats can be identified by the WS requester.

/SupportedFormatForTN/Format: this is an optional element used when the value of the attribute otherFileFormatIsExisted is yes. One or more than one of this element can be set under the <SupportedFormatForTN> header block.

/SupportedFormatForTN/Format/@name: this attribute is to denote the name of other file formats when the <Format> element is used.

/SupportedFormatForTN/Fault: this optional element is to finish the communication with the WS provider if there is no other file format(s) that can be supported by the WS requester when the value of the attribute otherFileFormatIsExisted is no.

/SupportedFormatForTN/{any}

/SupportedFormatForTN/@{any}

An example is shown below:

```
<SupportedFormatForTN
otherFileFormatIsExisted="yes">
       <Format name="X-TNL"/>
</SupportedFormatForTN>
```

4.1.4 RespondingForSupportedFormatForTN

The <RespondingForSupportedFormatForTN> header block is used in response to the message in the <SupportedFormatForTN> block. It allows the WS provider to choose one of other supported file formats. The following describes the attributes and elements.

/RespondingForSupportedFormatForTN: this element is created by the WS provider to send a response to the WS requester about whether one of the other file formats can be supported.

/RespondingForSupportedFormatForTN/@otherFileFormatIsSupported: this attribute is to judge whether the WS provider can support at least one of the other file formats. The value can only be yes or no. If the value is yes, it means that the WS provider can support at least one of the other file formats. If the value is no, then none of the other file formats can be supported by the WS provider.

/RespondingForSupportedFormatForTN/Fault: this optional element is used to inform the WS requester that none of the other file formats can be supported when the value of the attribute otherFileFormatIsSupported is no or when receiving a fault message in the <SupportedFormatForTN> block from the WS requester informing that it does not support other file formats.

/RespondingForSupportedFormatForTN/Format: this is an optional element, which is used when at least one of the other file formats can be supported by the WS provider when the value of the attribute otherFileFormatIsSupported is yes. Only one format can be set under the <RespondingForSupportedFormatForTN> element, because considering the fact that sometimes more than one file format can be supported by two WS, then the WS provider should always select the ideal one file format, which can be analysed quicker than others to increase the analysis efficiency.

/RespondingForSupportedFormatForTN/Format/@name: this attribute is to record the name of the format.

/RespondingForSupportedFormatForTN/{any}

/RespondingForSupportedFormatForTN/@{any}

An example is shown below:

```
<RespondingForSupportedFormatForTN
otherFileFormatIsSupported="yes">
       <Format name="X-TNL" />
</RespondingForSupportedFormatForTN>
```

Through an analysis of all possible actions between the WS provider and the WS requester during the preparation stage, the process of this stage can be concluded into five different cases depicted in sequence diagrams as shown in figures 1-5 as follows:

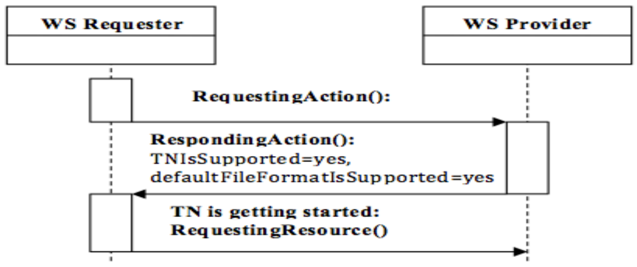

Figure 1. Case 1: WS Provider supports Both TN and default file Format

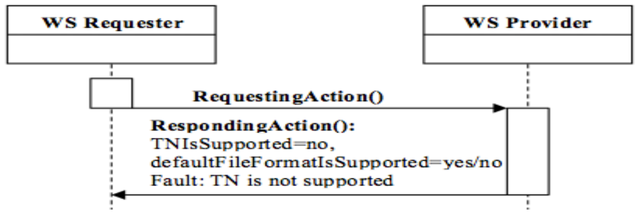

Figure 2. Case 2: WS Provider does not support TN

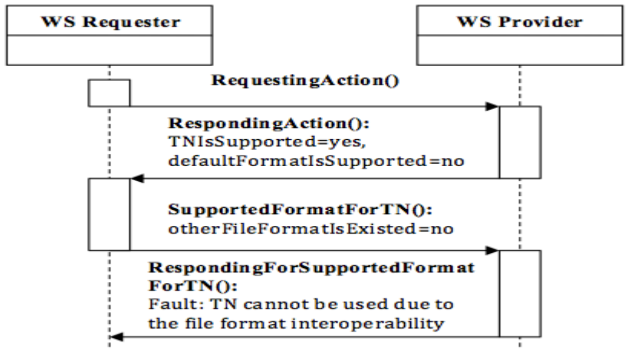
WS Requester
WS Provider
RequestingAction()
RespondingAction():
TNIsSupported=yes,
defaultFormatIsSupported=no
SupportedFormatForTN():
otherFileFormatIsExisted=no
RespondingForSupportedFormat
ForTN():
Fault: TN cannot be used due to
the file format interoperability

Figure 3. Case 3: WS Provider only supports TN but does not support the default file format and WS Requester does not support other file format(s)

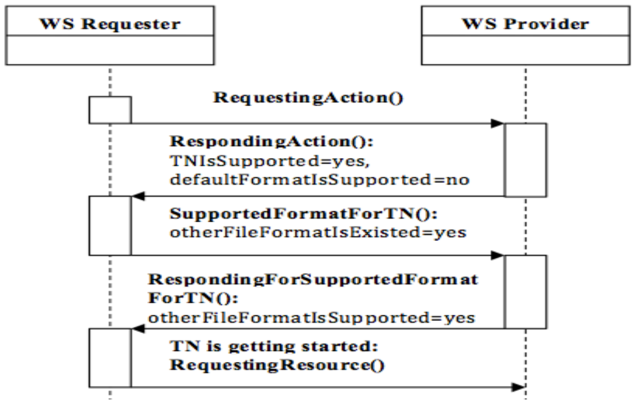

Figure 4. Case 4: WS Provider only supports TN but does not support the default file format and WS Requester supports other format(s). Fortunately, WS Provider supports at least one of other file format(s)

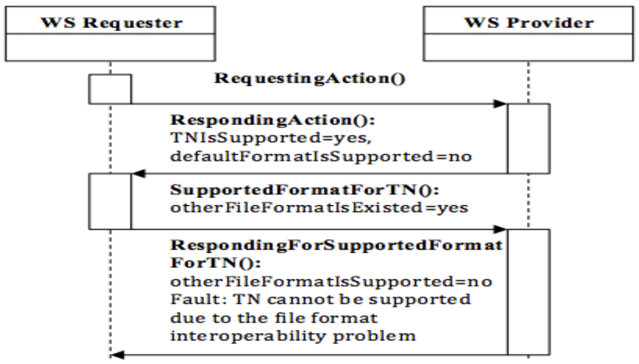
WS Requester
WS Provider
RequestingAction()
RespondingAction(): TNIsSupported=yes, defaultFormatIsSupported=no
SupportedFormatForTN(): otherFileFormatIsExisted=yes
RespondingForSupportedFormatForTN(): otherFileFormatIsSupported=no Fault: TN cannot be supported due to the file format interoperability problem

Figure 5. Case 5: WS Provider only supports TN but does not support the default file format and WS Requester supports other format(s). Unfortunately, WS Provider does not support other file format(s)

4.2 Negotiation Stage Protocol

After two WS are agreed about a common file format by using TN to establish a trust relationship in the preparation stage, the negotiation stage is triggered to start the real TN. Within the negotiation stage, two messages are special, which are the first message <RequestingResource> (see section 4.2.1) and the last message <ResultForRequestingResource> (see section 4.2.5). The first message is sent from the WS requester to inform the WS provider of the details about the target resource. The last message is sent from the WS provider to inform the WS requester whether TN is successful or not. Other messages transferred between two WS are only categorised to: credential block, declaration block and policy block. The real file content can be contained inside these blocks since a variety of these file formats have been designed by other researchers. As SOAP messages transferred between WS are XML-based messages, we recommend that one current existing TN language format called X-TNL language [1] be appropriate to be used inside our protocol, since it is XML-based format, which can be embedded into SOAP messages directly without any need of file format interpretation (e.g. convert from a non XML-based format to an XML-based format), and it can support credential, declaration and policy files for TN. Note that credentials, declarations and policies are also treated as a kind of resource in this protocol.

4.2.1 First Message: RequestingResource

The <RequestingResource> header block is sent from the WS requester to describe the details about the requested resource. The following describes the attributes and elements.

/RequestingResource: this header element is used by the WS requester to characterise the detailed information about the target resource within the WS provider.

/RequestingResource/@remoteResourceName: this attribute is used to specify the name of the remote resource.

/RequestingResource/@remoteResourceOwner: this is an attribute to indicate the owner's name of the remote resource.

/RequestingResource/@localRequesterName: this is an attribute to denote the local requester's name.

/RequestingResource/{any}

/RequestingResource/@{any}

An example is given below:

```
<RequestingResource
remoteResourceName="Notes of Trust
Negotiation" remoteResourceOwner="Bob"
localRequesterName="Alice"/>
```

4.2.2 PolicyForTN

The <PolicyForTN> header block is used to contain the real policy file content to inform the other WS about what kind of attributes or credentials are needed. The following describes the attributes and elements.

/PolicyForTN: this is the header element to contain the content of a policy file.

/PolicyForTN/@format: this attribute is used to confirm whether the format of the policy is the one accepted by two WS in common.

/PolicyForTN/@localPolicyName: this attribute is to record the name of the local policy.

/PolicyForTN/@remoteResourceOwner: this attribute is to specify the owner's name of the remote resource.

/PolicyForTN/@localPolicyOwnerName: this attribute is to indicate the local policy owner's name.

/policyForTN/@protectedLocalResource: this attribute is to denote which local resource this policy is protecting.

/PolicyForApproach/{any}

/PolicyForApproach/@{any}

Note: the content of a policy file should be embedded inside the <PolicyForTN> block.

An example is shown below:

```
<PolicyForTN format="X-TNL"
localPolicyName="Notes of Trust Negotiation
Policy" remoteResourceOwner="Alice"
localPolicyOwnerName="Bob"
protectedLocalResource="Notes of Trust
Negotiation">
The content of an XML-based policy file
</PolicyForTN>
```

4.2.3 CredentialForTN

The <CredentialForTN> header block is used to contain the real credential files. The following describes the attributes and elements.

/CredentialForTN: this header element is to contain the content of a credential file.

/CredentialForTN/@format: this attribute is used to confirm whether the format of the credential is the one agreed by two WS.

/CredentialForTN/@localCredentialName: this attribute is to indicate the name of the local credential.

/CredentialForTN/@localCredentialType: this attribute is to denote the type of the local credential.

/CredentialForTN/@localCredentialOwner: this attribute is to show the local credential owner's name.

/CredentialForTN/@meetRemotePolicy: this attribute is to specify the name of the remote policy from the other WS, which this credential can satisfy.

/CredentialForTN/@meetRemotePolicyOwner: this attribute is to specify the remote policy owner's name.

/CredentialForTN/Fault: this optional element is used when there is no local credential that can be submitted to satisfy the remote policy from the other WS.

/CredentialForTN/{any}

/CredentialForTN/@{any}

Note: the content of a credential file should be embedded inside the <CredentialForTN> block.

An example is shown below:

```
<CredentialForTN format="X-TNL"
localCredentialName=”StudentID_Alice”
localCredentialOwner=”Alice”
localCredentialType=”StudentID”
meetRemotePolicy=”Bob’s_Notes_PolicyOne”
meetRemotePolicyOwner=”Bob”>
The content of an XML-based credential file
</CredentialForTN>
```

4.2.4 DeclarationForTN

Basically, the <DeclarationForTN> block is similar to the <CredentialForTN> block including the same attributes where the word “credential” should be changed to “declaration”; therefore the details of this block are omitted here.

4.2.5 Last Message: ResultForRequestingResource

The <ResultForRequestingResource> header block is the last message sent from the WS provider to inform the WS requester whether the resource can be accessed or not. In other words, if the resource can be revealed, the approach for establishing a trust relationship is successful; otherwise, it is failed. The following describes the attributes and elements.

/ResultForRequestingResource: this header element is used to contain the result information

/ResultForRequestingResource/@isSuccessful: this attribute is used to inform the WS requester whether the resource can be accessed or not. The value can only be yes or no. If the value is yes, the resource can be accessed. Otherwise, the resource cannot be revealed.

/ResultForRequestingResource/Fault: this optional attribute is used to inform the WS requester about the reasons why TN is not successful, when the value of the attribute isSuccessful is no.

/ResultForRequestingResource/{any}

/ResultForRequestingResource/@{any}

5. SCENARIO IMPLEMENTATION BY ADOPTING THE PROTOCOL

The scenario mentioned earlier has been implemented by using Apache Axis2, a new generation WS/SOAP/WSDL engine [15]. In order to implement the scenario, two computers have been deployed as WS by using Apache Axis2 1.6.0 contained in Apache Tomcat 7.0.6 running on Mac OS 10.6, which operate as a WS requester and a WS provider respectively. However, due to the weakness of supporting communication in multiple steps between two WS by Axis2, TCP-monitors are used as proxies in both WS to aid in transferring SOAP messages continuously, otherwise transferring errors will occur when sending the 5th or the 6th message. The reason of the occurrence of such errors is perhaps that Axis2 is mainly designed for supporting several-steps communication between a client and a WS so that multiple-steps communication between two WS may not be well supported.

With the implementation, we assume that both two WS use TN to establish a trust relationship and support at least one common file format for TN. The file format chosen for TN is the X-TNL language [1], because it is designed in XML-based format, which can be embedded into SOAP messages directly [see section 4.2]. Through the incorporation of the X-TNL policy template, we have discovered that the policy template was not clarified how it could support the conjunction (AND operation) and disjunction (OR operation) by using the <certificate> block. We suggest that an attribute “group” with the value as a positive integer should be added into the <certificate> element. In this way, the conjunction can be expressed when the relevant <certificate> elements hold the same value, while the disjunction can be expressed when the irrelevant <certificate> element hold a different value. For instance, a policy declares that the *Resource* (denoted as R) can be accessed if the requester can submit *credential 1* (denoted as C1) and *Credential 2* (denoted as C2) or can submit *Credential 3* (denoted as c3) instead, represented as $R \leftarrow C1 \wedge C2 \vee C3$ [24]. This policy can be implemented in the X-TNL policy template where C1, C2 and C3 are <certificate> elements as shown below:

```
<certificate group=”1”>
<certificate group=”1”>
<certificate group=”2”>
```

6. CONCLUSION AND FUTURE WORK

In this paper, an XML-based protocol has been presented for WS to support TN. By using this protocol, two WS are able to establish a trust relationship by using existing approaches for TN. In addition, before TN starts, a preparation stage will be conducted first to check whether two WS can identify a common supported file format for TN, which can successfully avoid failed TN caused by the unreadability of different file formats. However, the weaknesses of this protocol are basically three: a. only one file can be revealed in each step during TN; b. The common supported file format for credentials, declarations and policies must be the same, which neglects the scenario when a system uses different file formats for dealing with credentials, declarations and policies respectively; c. TN cannot be used if no common file format can be recognised by both WS. All these limitations will be addressed in the future work.

7. REFERENCES

[1] Bertino, E., Ferrari, E., and Squicciarini, A. C. 2003. X-TNL: An XML-based language for trust negotiations. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks.* Italy, (June. 2003), DOI= http://doi.ieeecomputersociety.org/10.1109/POLICY.2003.1206960.

[2] Bonatti, P. A., Coi, J. L. D. Olmedilla, D. and Sauro, L. 2010. A Rule-based Trust Negotiation System. *IEEE Transactions on Knowledge and Data Engineering.* Italy. 22(11), (Nov. 2010).1507-1520. DOI=10.1109/TKDE.2010.83.

[3] He, Y. and Zhu, M. 2007. A complete and efficient strategy based on petri net in automated trust negotiation. In: *Proceedings of the 2nd international conference on Scalable information systems.* [Online] Available from: http://portal.acm.org.ezproxy.lib.uts.edu.au/citation.cfm?id=1366804.1366900&coll=DL&dl=GUIDE&CFID=115729778&CFTOKEN=48955637.

[4] He, Y., Zhu, M. and Zheng, C. 2008. An Efficient and Minimum Sensitivity Cost Negotiation Strategy in Automated Trust Negotiation. In: *2008 International Conference on Computer Science and Software Engineering.* DOI=10.1109/CSSE.2008.867.

[5] Horst, T. W., Sudenlin, T., Seamons, K. E. and Knutson, C. D. 2005. Mobile Trust Negotiation Authentication and Authorization in Dynamic Mobile Networks. *IFIP International Federation for Information Processing.* 175(2005), 97-109. Available from: DOI=10.1007/0-387-24486-7_7.

[6] Koshutanski, H. M., F. 2005. Interactive Credential Negotiation for Stateful Business Processes. *Lecture Notes in Computer Science.* 3477(2005), 256-272. DOI=10.1007/11429760_18.

[7] Lee, A. J. and Winslett, M. 2008. Towards an Efficient and Language-Agnostic Compliance Checker for Trust Negotiation Systems. In: *Proceedings of the 2008 ACM symposium on Information, computer and communications security.* 228-239.DOI=10.1145/1368310.1368343.

[8] Lee, A. J. and Winslett, M. 2008. Towards Standards-Compliant Trust Negotiation for Web Services. *IFIP International Federation for Information Processing.* 311-326. DOI=10.1007/978-0-387-09428-1_20.

[9] Lee, A. J., Winslett, M. and Perano, K. J. 2009. TrustBuilder2: A Reconfigurable Framework for Trust Negotiation. *Proceedings of the Third IFIP WG 11.11 International Conference on Trust Management.* [Online] Available from: www.itr-rescue.org/pubs/upload/896_Lee2009.pdf.

[10] Liu, Z. and Xiu, D. 2005. Agent-based Automated Trust Negotiation for Pervasive Computing. In: *Proceedings of the Second International Conference on Embedded Software and Systems.* 300-309. DOI=10.1109/ICESS.2005.19.

[11] Li, J., Li, N., and Winsborough, W. 2005. Automated Trust Negotiation Using Cryptographic Credentials. In: *Proceedings of the 12th ACM conference on Computer and communications security.* 46-57. DOI=10.1145/1102120.1102129.

[12] Lu, h. and Liu, B. 2008. Improving Negotiation Efficiency and Success in Automated Trust Negotiation. In *Proceedings of the 2007 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System.* 442-449. DOI=10.1109/SITIS.2007.27.

[13] Nejdl, W., Olmedilla, D. and Winslett, M. 2004. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. *Secure Data Management.* 3178, 118-132. DOI=10.1.1.59.5844.

[14] Olmedila, D., Lara, R., Polleres, A. and Lausen, H. 2004. Trust Negotiation for Semantic Web Services. In: *1st International Workshop on Semantic Web Services and Web Process Composition in conjunction with the 2004 IEEE International Conference on Web Services.* 81-95. DOI=10.1.1.59.4059.

[15] Perera, S., Herath, C. Ekanayake, J., Chinthaka, E., Ranabahu, A., Jayasinghe, D., Weerawarana, S. and Daniels, G. 2006. Axis2, Middleware for Next Generation Web Services. In *IEEE International Conference on Web Service*. Chicago, Illinois, USA. DOI= 10.1109/ICWS.2006.36.

[16] Seamons, K. E., Winslett, M. and Yu, T. 2001. Limiting the Disclosure of Access Control Policies during Automated Trust Negotiation. In: *Proceedings of New York and Distributed System Security Symposium.* DOI=10.1.1.36.1585.

[17] Seamons, K. E., Winslett, M., Yu, T., Yu, L. and Jarvis, R. 2002. Protecting privacy during on-line trust negotiation. In: *Proceedings of the 2nd international conference on Privacy enhancing technologies.* 129-143. DOI=10.1007/3-540-36467-6_10.

[18] Skogsrud, H., Benatallah, B. and Casati, F. 2004. A trust negotiation system for digital library Web services. *International Journal on Digital Libraries.* 4(3), 185-207. DOI=10.1007/s00799-004-0083-y.

[19] Skogsrud, H., Motahari-Nezhad, H. R., Benatallah, B. and Casati, F. 2009. Modeling Trust Negotiation for Web Services. *Computer.* 42(2), 54-61. DOI=10.1109/MC.2009.56.

[20] Squicciarini, A., Bertino, E., Ferrari, E., Paci, F. and Thuraisingham, B. 2007. PP-trust-X: A system for privacy preserving trust negotiations. *ACM Transactions on Information and System Security (TISSEC),* 10(3). DOI=10.1145/1266977.1266981.

[21] Winsborough, W. H., Seamons, K. E. and Jones, V. E. 1999. Negotiating Disclosure of Sensitive Crendentials. In: *Second Conference on Security in Communication Networks.* DOI=10.1.1.34.8359.

[22] Winsborough, W. H., Seamons, K. E. and Jones, V. E. 2000. Automated Trust Negotiation. *DAPRA Information Survivability Conference and Exposition.* [Online] Available from: http://isrl.cs.byu.edu/pubs/discex2000.pdf.

[23] Ye, S., Makedon, F. and Ford, J. 2004. Collaborative Automated Trust Negotiation in Peer-to-Peer Systems. In: *Proceedings of the Fourth International Conference on Peer-to-Peer Computin.*108-115. DOI=10.1109/P2P.2004.13.

[24] Yu, T., Ma, X. and Winslett, M. 2000. PRUNES: an efficient and complete strategy for automated trust negotiation over the Internet. In: *Proceedings of the 7th ACM conference on Computer and communications security.* 210-219. DOI=10.1145/352600.352633.

[25] Yu, T., Winslett, M. and Seamons, S. E. 2001. Interoperable strategies in automated trust negotiation. In: *Proceedings of the 8th ACM conference on Computer and Communications Security.* 146-155. DOI=10.1145/501983.502004.

[26] Yu, T. and Winslett, M. 2003. A Unified Scheme for Resource Protection in Automated Trust Negotiation. In: *Proceedings of the 2003 IEEE Symposium on Security and Privacy.* 110-122. DOI=10.1109/SECPRI.2003.1199331.

[27] Winsborough, W. H. and Li, N. 2002. Towards Practical Automated Trust Negotiation. In: *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks.* DOI=10.1109/POLICY.2002.1011297.

[28] Winsborough, W. H. and Li, N. 2002. Protecting sensitive attributes in automated trust negotiation. In: *Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society.* 41-51. DOI=10.1145/644527.644532.

[29] Winslett, M., Yu, T., Seamons, K. E., Hess, A., Jacobson, J., Jarvis, R., Smith, B. and Yu, L. 2002. Negotiating Trust on the Web. *IEEE Internet Computing.* 6(6), 30-37. DOI=10.1109/MIC.2002.1067734.

[30] Zartman, W. I. and Berman, M. R., 1982. *The Practical Negotiator*,Yale University Press. New Haven, CT, USA.