Has WS-I's Work Resulted in WS-* Interoperability?

Andreas Schönberger
Distributed and Mobile Systems Group
University of Bamberg
Bamberg, Germany
andreas.schoenberger@uni-bamberg.de

Johannes Schwalb
Senacor Technologies AG
Schwaig b. Nürnberg, Germany
johannes.schwalb@senacor.com

Guido Wirtz
Distributed and Mobile Systems Group
University of Bamberg
Bamberg, Germany
guido.wirtz@uni-bamberg.de

***Abstract*—Recently, the Web Services Interoperability Organization (WS-I) has announced to have completed its interoperability standards work. The latest deliverables include the so-called "Basic Security Profile" and the "Reliable Secure Profile". This gives rise to the question whether or not Web Services adopters can rely on interoperability of Web Services stacks, in particular in terms of security and reliability features. To answer this question, we thoroughly analyze two important Web Services stacks for interoperability of WS-Security and WS-ReliableMessaging features. Our analysis shows that security and reliability features are far from being implemented in an interoperable manner. Additionally, we reveal that some of those interoperability problems are not even covered by WS-I profiles and therefore conclude that WS-I's work has not yet resulted in Web Services interoperability.**

***Keywords*-Web Services Interoperability; Interoperability Testing; WS-Security; WS-ReliableMessaging**

I. Introduction

Quality-of-Service (QoS) features such as security and reliability are brought to the Web Services world by the so-called WS-* standards. WS-Security 1.1 (WS-Sec, [1]) and WS-ReliableMessaging 1.2 (WS-RM, [2]) are prominent representatives of WS-* standards that define data formats and processing instructions for extending the SOAP [3] messages that implement Web Services exchanges. For example, an XML `Signature` tag together with `SignedInfo`, `SignatureValue` and `KeyInfo` tags would have to be inserted into the SOAP `Header` tag to provide integrity protection. For convenience, a Web Services developer is not supposed to '*manually*' insert all that information into SOAP messages. Instead, *Web Services Security Policy 1.3* (WS-Sec-Pol, [4]) and *Web Services Reliable Messaging Policy Assertion 1.2* (WS-RM-Pol, [5]) can be used to extend the WSDL definition of a Web service with assertions that instruct the Web Services stack implementations (WS stack in the following) in use to apply WS-Sec and WS-RM features to the SOAP message exchanges. This policy-based realization of QoS features for Web services has been postulated in several publications [6], [7], [8], [9], [10], [11], [12] and promises better interoperability than letting all Web Services developers implement QoS features *their own way*.

On November 10th 2010, the *Web Services Interoperability Organization* (WS-I) released a press announcement with the following title

"WS-I Completes Web Services Interoperability Standards Work
Industry Collaboration Enables Interoperability in the Cloud"[1]

This announcement followed the release of new versions of WS-I's core deliverables, most notably the *Basic Security Profile* (BSP, [13]) and *Reliable Secure Profile* (RSP, [14]) in 2010. WS-I *profiles* are the core deliverables of WS-I and document *"[..]clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability"* [14]. The main target of BSP and RSP are WS-Sec and WS-RM. As WS-I promotes interoperability and has declared its standardization work to be completed, this means that Web Services adopters should be able to rely on the interoperable implementation of Web Services security and reliability features which are pivotal for Web Services according to [11], [15]. However, WS-Sec and WS-RM as well as WS-Sec-Pol and WS-RM-Pol are highly complex specifications so that interoperability across WS stacks does not come easy. Consistently, [6] stress that *"although one of the main purposes of the standard [i.e., a WS security standard] is to guarantee the interoperability between different platforms, it might be necessary to test it on the field."*

This paper investigates in how far WS-I's work has led to interoperability across WS stacks regarding the implementation of WS-Sec(-Pol) and WS-RM(-Pol). In order to operationalize the question whether or not a Web Services adopter can rely on interoperability between different WS stacks, we use two 'optimistic' hypotheses:

H1: *The overwhelming majority of WS-Sec-Pol/WS-RM-Pol features are implemented by Web Services stacks.*

H2: *Out of those WS-Sec-Pol/WS-RM-Pol features implemented by two platforms, the overwhelming majority is implemented in an interoperable manner.*

The contribution of this paper is threefold. First, we assess the coverage of WS-Sec-Pol/WS-RM-Pol specifications by two major Java-based WS stacks. Second, we assess the interoperability of implemented features. Based on these results, both hypotheses will have to be rejected. Third, we

[1] http://ws-i.org/docs/press/pr_101110.pdf, 02/09/2011

analyze in how far the detected interoperability issues could have been avoided by strict compliance to the BSP and RSP. In section II, the notion of interoperability is operationalized for the work at hand and the approach towards interoperability testing WS-* standards is described. Section III presents the results of our interoperability and coverage investigation while section IV analyzes in how far strict compliance to WS-I's BSP/RSP could have been of help. Section V discusses related work and section VI concludes and points out directions for future work.

II. Test Approach

In order to provide a sound foundation for this work, we sketch our approach for testing WS-* interoperability [16], [17] by operationalizing the notion of interoperability, describing a concept for executing test cases and outlining the systematic derivation of test cases.

[18] defines '*interoperability*' as the *"the ability of two or more software components to cooperate despite differences in language, interface, and execution platform."* While this definition is good enough for an abstract characterization of interoperability in arbitrary systems, it has to be refined for the purpose of WS-* interoperability testing. Remember that we require the use of WS-Policy for asserting QoS properties of Web Services interactions. Hence, the following sources of interoperability issues between two WS stacks have to be considered. First, one of the WS stacks under test may not know/refuse a particular WS-Policy assertion that specifies a particular communication feature. Second, one of the WS stacks may accept a WS-Policy assertion, but ignore it. Third, a WS stack may deviate from one or more of the processing instructions that are specified by a WS-* standard for the implementation of a particular WS-Policy assertion. Considering these sources of interoperability issues and taking into account that a Web service interaction typically takes place between a client role and a server role, 12 interoperability levels can be identified that range from a policy being refused/ignored by one of the roles over abrupt termination of communication to full protocol success (for details, please see [16]). So, for the purpose of this work, interoperability is defined as follows:

Definition 2.1 (Interoperability):
Two WS stacks are interoperable with respect to a WS- policy assertion if client and server process the assertion such that the exchange of corresponding SOAP messages succeeds without errors and such that WS-* processing rules are applied.*

For determining interoperability as defined above, the approach visualized in figure 1 is applied. WS-Sec-Pol/WS-RM-Pol definitions are used to specify concrete test cases. For each test case, the WSDL of a sample Web service is extended with such a definition to be used by the WS stacks of the Web service client and provider for determining the number, sequence and contents of the SOAP messages to be exchanged (upper part of figure 1). The test case is then performed for four different WS stack configurations. Assume that two WS stacks A and B are to be tested for interoperability and that a configuration 'X-Y' expresses that WS stack X takes the client role and WS stack Y takes the server role. Then, the test case is first performed in homogeneous environments (A-A, B-B) for checking whether or not the functionality is implemented and afterwards in heterogeneous environments (A-B, B-A) for checking interoperability. Note that if only one of the homogeneous environments does not work, then the heterogeneous environments still are worth testing. Our practical tests show (cf. [17]) that some features do not work in a homogeneous environment (A-A or B-B), but in a heterogeneous one (B-A or A-B).

The interoperability levels are to be examined for each test case and WS stack configuration. Some of the interoperability levels can be verified without investigating the SOAP messages exchanged, e.g., refusal of the policy by the server. The analysis and determination of other interoperability levels require the use of network analysis tools like Wireshark[2] that enable capturing the SOAP messages exchanged (lower part of figure 1). However, we do not check the strict conformance of SOAP messages to WS-* standards in our interoperability testing approach. Instead, SOAP messages are only analyzed for the existence of WS-* headers as well as for unexpected errors and premature termination. Not checking conformance allows for the possibility of 'interoperable' communication that violates WS-* standards. Consistently, definition 2.1 deliberately does not require that WS-* processing rules are applied correctly. From our experience, this is a purely theoretical limitation for heterogeneous environments.

For deriving the configuration options for a single policy assertion, we propose to make use of the assertion structure definitions that are published in the WS-Policy extension standards. Listing 1 shows the structure definition of the WS-RM-Pol standard's `RMAssertion` assertion (note that usual regular expression operators are used to define structural constraints on the assertion). This assertion basically says that delivery semantics options `ExactlyOnce`, `AtLeastOnce` and `AtMostOnce` of WS-RM must be combined with either `InOrder` delivery or not. Checking these features in isolation frequently is not possible because a deployable WS-Policy configuration may require additional assertions, e.g., testing a WS-Sec-Pol *protection* assertion requires declaring assertions for a valid security binding (options for so-called Asymmetric-/Symmetric-/TransportBindings). To solve this issue, we propose to start with sample policy configurations that ship with WS stacks or are retrievable from the web and then to permute the options of the assertion under test only. By using the WS-Policy structure definitions and sample policy configurations as proposed, it is possible to identify

[2] www.wireshark.org, 02/09/2011

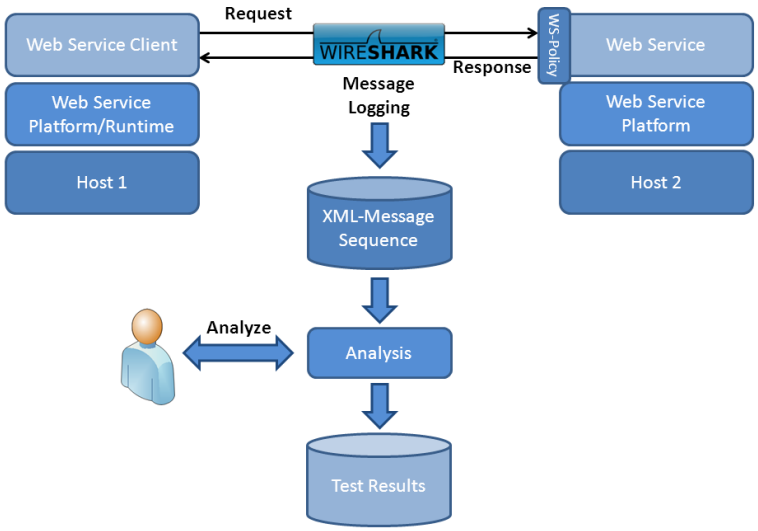

Figure 1. Setup of Test Environment

test cases that test a WS-* feature almost in isolation and to achieve reasonable coverage of the WS-Policy standards. Based on the results of these *"isolated"* test cases, *"combined"* test cases that cover the interplay of several WS-* features can be derived.

Listing 1. Structure definition of `RMAssertion` (cf. [5])

```
<wsrmp:RMAssertion (wsp:Optional="true")?
    ... >
   <wsp:Policy>
      (<wsrmp:SequenceSTR /> |
       <wsrmp:SequenceTransportSecurity /> )
            ?

      <wsrmp:DeliveryAssurance>
         <wsp:Policy>
            (<wsrmp:ExactlyOnce /> |
             <wsrmp:AtLeastOnce /> |
             <wsrmp:AtMostOnce />  )
            <wsrmp:InOrder /> ?
         </wsp:Policy>
      </wsrmp:DeliveryAssurance> ?
   </wsp:Policy>
   ...
</wsrmp:RMAssertion>
```

III. Interoperability Assessment

For evaluating WS-Sec-(Pol)/WS-RM-(Pol), we have chosen two of the most reputable JAVA-based WS stacks, namely Oracle's (Sun's) *Metro* WS-stack that comes with the GlassFish Application Server[3] and Apache's *Axis2* WS stack that is reused in IBM's WebSphere Application Server[4]. Below, we show that considerable interoperability problems between these WS stacks exist which justify rejecting both hypotheses H1 and H2.

The following groups of *isolated* test cases have been identified for WS-Sec-(Pol) and WS-RM-(Pol). In anticipation of the results, *combined* test cases are left out:

1) WS-RM-Pol Assertions

This test group essentially comprises the test cases derivable from listing 1 and cover the delivery semantics of reliable messaging.

2) WS-Sec-Pol Protection Assertions

This test group covers the various ways of asserting the need for signing or encrypting SOAP messages or parts of SOAP messages.

3) WS-Sec-Pol Tokens

This test group covers assertions for configuring how security tokens are processed, e.g., whether or not tokens have to be included in every SOAP message, and the assertions for declaring the various token types themselves such as `UsernameTokens` or `X509Tokens`.

4) WS-Sec-Pol Security Bindings

Security binding assertions configure the algorithms to be used for signing/encrypting as well as options for configuring the basic security mechanisms *transport level security*, *symmetric key security* as well as *asymmetric key security*.

5) WS-Sec-Pol Supporting Tokens

This test group covers assertions that are used to augment the claims provided by the token of the basic Security Binding. For example, an `EndorsingSupportingTokens` assertion may be used to require a signature of the signature of a SOAP message (cf. [4], section 8.3).

6) WS-Sec-Pol WS-Sec and WS-Trust Options

This test group covers general WS-Sec and WS-Trust assertions such as what kind of token references must be supported, whether client or server challenges must supported, or whether client or server entropy is required.

[3]http://glassfish.dev.java.net, 02/09/2011

[4]http://www-01.ibm.com/software/webservers/appserv/was/, 02/09/2011

All in all, the number of test cases derived amounted to 169. This number proved to still be manageable. For the majority of test cases, it was possible to retrieve executable sample configurations for at least one of the WS stacks from the web. Executable sample configurations for the remaining test cases then could be derived by just replacing or reconfiguring an assertion, e.g., using a 'WssX509V3Token11' instead of a 'WssX509V3Token10'. 109 of the test cases could successfully be performed in at least one of the homogeneous environments. This fact taken together with the exception messages of the WS stacks under test about not supporting particular features indicates that our policy configurations in itself were correct (in the sense of complying to WS-Sec-Pol/WS-RM-Pol) for most test cases and therefore not the source of the detected interoperability problems. In the following, section III-A describes the core issues detected and section III-B summarizes the interoperability results per group of test cases.

A. Core Interoperability Issues

In order to protect solution provider interests we have made the following interoperability issues anonymous (more detailed test results are available as a technical report [17]) and stick to the A,B-notation of section II:

1) *No WS-ReliableMessaging Policy Support*
   Platform A uses a proprietary API for configuring reliable messaging features that is accessible via a GUI and does not accept WS-RM-Pol for configuration. So, if platform A is used for the client, then no interaction is possible. However, if platform A is used for the server then platform B can be configured using WS-RM-Pol such that interaction is possible for some test cases.
2) *No `TransportBinding` Support*
   Platform A does not support the `TransportBinding` assertion as defined in WS-Sec-Pol ([4], section 7.3). This assertion allows for configuring the use of transport protocol features for securing messages, in particular using HTTPS. Note that this does not mean that platform A does not support HTTPS at all, it just means that the `TransportBinding` assertion cannot be used.
3) *No `XPath` Support for Element Identification*
   The `EncryptedElements` assertion ([4], section 4.2.2) and the `SignedElements` assertion ([4], section 4.1.2) of WS-Sec-Pol define an XML tag named `XPath` for specifying the elements of a SOAP message to be encrypted/signed. However, platform B does not support this tag.
4) *No `OnlySignEntireHeadersAndBody` Support*
   This optional WS-Sec-Pol element ([4], section 7.4, 7.5) enforces that *"[..]digests over the SOAP body and SOAP headers MUST only cover the entire body and entire header elements"*. ([4], section 6.6). Although the WS-Sec-Pol standard explicitly recommends to use this element in order to *"[..]combat certain XML substitution attacks"* ([4], section 12), platform A does not support it.
5) *No `EncryptBeforeSigning` Support*
   This optional WS-Sec-Pol element ([4], section 7.4, 7.5) can be used to override the default value `SignBeforeEncrypting` of the protection order property ([4], section 6.3). However, only platform B supports this element.
6) *Deviating Processing of `UsernameToken`*
   The WS-Sec-Pol `UsernameToken` assertion can be used to leverage user name/password authentication for interactions and version 1.0 of the so-called UsernameToken profile [19] is accepted by both platforms. However, platform A leaves the `Username` and `Password` elements empty whereas platform B by default encrypts the whole `UsernameToken`. Platform A essentially does not allow for configuring `UsernameTokens` using WS-Sec-Pol whereas platform B disregards the following WS-Sec-Pol recommendation by applying encryption by **default**: *"When the UsernameToken is to be encrypted it SHOULD be listed as a SignedEncryptedSupportingToken (Section 8.5), EndorsingEncryptedSupportingToken (Section 8.6) or SignedEndorsingEncryptedSupportingToken (Section 8.7)"* ([4], section 5.4.1).
7) *Deviating Signing Strategy for Timestamp*
   The optional WS-Sec-Pol `IncludeTimestamp` element ([4], section 7.3,7.4,7.5) can be used to require the inclusion of a `Timestamp` element in the SOAP headers of an interaction and is supported by both platforms. Additionally, WS-Sec-Pol requires that if `IncludeTimestamp` is specified and if there is no transport layer encryption specified then the `Timestamp` has to be integrity protected at the message level, i.e., signed ([4], section 6.2). However, platform A does not directly implement this rule but requires the Web Services developer to add an according `SignedElements/XPath` expression to sign the timestamp.
8) *Ignored `IncludeToken` Values*
   The optional attribute `IncludeToken` allows for specifying in which SOAP messages of an interaction a corresponding token, e.g., a `UsernameToken`, should be present. For example, the `IncludeToken` value `AlwaysToRecipient` specifies that a token should be present in all messages from the initiator of the interaction to the recipient, but not vice versa. Alternative values are `Never`, `Once`, `AlwaysToInitiator` and `Always` ([4], section 5.1.1). Both platforms under test accept all `IncludeToken` values, but platform A simply ignores the actual value and always includes the

corresponding token in the SOAP messages. This leads to an interoperability error with platform B in case `Never` is configured even in the A-B configuration because platform B rejects SOAP messages that carry a token if `Never` is specified. Inconsistently, platform B does not stop communication if SOAP messages carry a token that is not permitted due to the `AlwaysToRecipient`/`AlwaysToInitiator` values.

9) *Deviating Processing of `SignedParts/Body`*
The WS-Sec-Pol `SignedParts` assertion ([4], section 4.1.1) can be used to specify the integrity protection of a SOAP message's `Header`, `Body` or `Attachment` parts. Both platforms under test support the optional element `Body` which requires that *"[..] the `soap:Body` element, it's attributes and content, of the message needs to be integrity protected"* ([4], section 4.1.1). However, platform B signs the first element of the SOAP message `Body` whereas platform A signs the `Body` element itself.
10) *Deviating `EncryptedParts` Implementation*
Both platforms support this WS-Sec-Pol assertion ([4], section 4.2.1) that can be used to specify the encryption of a SOAP message's `Header`, `Body` or `Attachment` parts. However, platform B only supports the use of `EncrpytedParts` for the `IncludeToken` value `Never`. So, if platform B is used as a server, an interoperability issue arises because platform A *always* (cf. issue 8) includes a token which is rejected by the first platform.

B. Overall results

Due to space limitations, we only present the most interesting figures of the interoperability tests. The interoperability levels detected for all test cases are available as a technical report [17]. In table I, the column headers provide the following information:

a) # counts the number of test cases per test group
b) A-A (B-B) counts the number of test cases per test group for which full interoperability could be detected with platform A (B) as both, client and server.
c) A-A $\wedge$ B-B counts the number of test cases per test group for which full interoperability could be detected for both homogeneous environments.
d) A-A $\vee$ B-B counts the number of test cases per test group for which full interoperability could be detected for at least one of the homogeneous environments.
e) A-B $\vee$ B-A counts the number of test cases per test group for which full interoperability could be detected for at least one of the heterogeneous environments.
f) A-B $\wedge$ B-A counts the number of test cases per test group for which full interoperability could be detected for both homogeneous environments.
g) (A-A $\wedge$ B-B) $\wedge \neg$(A-B $\wedge$ B-A) counts the number of test cases per test group for which full interoperability could be detected for both homogeneous environments, but where an interoperability problem was detected for at least one of the heterogeneous environments.

In turn, the row headers simple distinguish the different test groups. The overall figures reveal that platform A implements only 30.2% (51 test cases) and platform B only 58.6% (99 test cases) of the WS-Sec-Pol/WS-RM-Pol functionality. Based on this data, hypothesis H1 must be rejected. Luckily, the functionalities implemented by the two platforms largely overlap which can be inferred by observing that (A-A $\wedge$ B-B) is close to Min(A-A,B-B) for all test groups. Therefore, there are at least 41 test cases that could be performed successfully on both homogeneous platforms which allows for testing hypothesis H2. Out of those 41 test cases there were 13 (31.7%) test cases that could not be performed successfully in both heterogeneous environments. Considering the complexity of WS-RM-Pol and especially WS-Sec-Pol this number does not seem to be too high, but for practical purposes an error rate of about one third is not acceptable and therefore H2 must be rejected as well.
Taking the effects of low coverage and bad interoperability together results in very low WS-Sec-Pol/WS-RM-Pol functionality that is supported in an interoperable manner. Only 47 (27.8%) out of 169 test cases can be performed successfully in at least one of the heterogeneous environments and only 28 (16.6%) test cases for both heterogeneous environments. This means that implementing security and reliability for Web Services based on WS-RM-Pol and WS-Sec-Pol for the heterogeneous platform configurations investigated here is at least a challenge. In particular, it is not possible to exchange a SOAP message between the two platforms that is both, confidentiality and integrity protected. Platform B does not support XPath for identifying the elements to be encrypted. Instead, it relies on using the `EncryptedParts` assertion and assumes an `InlcudeToken` value of `Never` for using X509 tokens (which is the only basic token type supported for the heterogeneous environments). Platform A ignores any `IncludeToken` value an always inserts the token into the SOAP messages which is then rejected by platform B. Additionally, platform A does not support the `TransportBinding` assertion so that SSL encryption cannot be asserted either. In consequence, deriving *combined* test cases from isolated test cases (cf. section II) essentially is senseless.
At least, there is an integrity and confidentiality protected interaction that comes close to WS-Sec-Pol/WS-RM-Pol based QoS implementation. For confidentiality protection, SSL is used which is configured for platform B using a standard `TransportBinding` assertion and for platform A using proprietary configuration. For integrity protection, an `AsymmetricBinding` together with a `X509Token` is

Test group	#	A-A	B-B	A-A ∧ B-B	A-A ∨ B-B	A-B ∨ B-A	A-B ∧ B-A	(A-A ∧ B-B) ∧ ¬(A-B ∧ B-A)
Basic	2	1	1	1	1	1	1	0
WS-RM	9	2	8	2	8	2	0	2
Protection Assertions	19	8	4	4	8	2	1	3
Token Assertions	65	10	37	10	37	8	5	5
Binding Assertions	46	22	38	18	42	27	15	3
Supporting Token Assertions	8	0	4	0	4	0	0	0
WS-Sec and WS-Trust Options	20	8	7	6	9	7	6	0
Overall	169	51	99	41	109	47	28	13

Table I
INTEROPERABILITY RESULTS PER TEST GROUPS

specified and the elements to be signed are identified using the `SignedParts` assertion. However, apart from not being fully standards based, only using platform A as server and platform B as client can be performed successfully because the other way round a platform A client tries to retrieve proprietary configuration information in vain.

IV. WS-I TO THE RESCUE?

The two main deliverables of the WS-I that cover the application of WS-Sec and WS-RM are the BSP and the RSP (cf. section I). Those profiles are complemented by test tools and sample applications, but the profiles are authoritative. In the standard document itself, the purpose of the BSP is described as follows:

> "*This document defines the WS-I Basic Security Profile 1.1, based on a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications which promote interoperability.*" [13]

Those clarifications and amendments become manifest in so-called requirements together with some explaining text and, in case of the RSP, test expressions for evaluating SOAP messages. In [20], one of the BSP editors explains that by using the requirements *"the BSP limits the set of common functionality that vendors must implement and thus enhances the chances for interoperability. This in return reduces the complexities for the testing of Web Services security."* Moreover, she explains that *"the security consideration statements provide guidance that is not strictly interoperability related but are testable best practices for security."*

For example, requirement *R2001* of the BSP says that *"A SENDER MUST NOT use SSL 2.0 as the underlying protocol for HTTP/S."*. The explanatory text justifies the requirement by pointing out that *"SSL 2.0 has known security issues and all current implementations of HTTP/S support more recent protocols"*.

A common characteristic of those requirements is that they define constraints on the level of SOAP messages, i.e., the existence, order and content of XML elements within SOAP message or the actual exchange of SOAP messages is described. Consequently, the testing tools of the WS-I take SOAP messages as input and check them for compliance to the BSP and RSP requirements. From the perspective of facilitating interoperability, this amounts to replacing actual interoperability testing as described in section II by checking standard compliance of SOAP messages. However, checking standard compliance itself is subject to errors and therefore merely an add-on to true interoperability testing but not a replacement. Even worse, the relation between WS-Sec-Pol/WS-RM-Pol assertions and the corresponding SOAP messages exchanged is not described in BSP and RSP at all. In section 5.1.1, the BSP explicitly allows for out of band agreement for specifying the use of WS-Sec. Moreover, it states in several sections (9, 10, 13.1) that "*[..]no security policy description language or negotiation mechanism is in scope for the Basic Security Profile[..]*". The RSP recommends (though not requires) the use of WS-RM-Pol for configuring the use of WS-RM in its section 2.4, but it does not define the relation between WS-RM-Pol assertions and SOAP messages either.

In so far, the interoperability issues 2, 3, 4, 6, 7, 8 and 10 of section III-A are not covered by the BSP/RSP at all. For issue 1 (no WS-RM-Pol support), platform A can be considered to ignore a WS-I recommendation. But as the RSP does not explicitly require the use of WS-RM-Pol, platform A nonetheless cannot be said to violate the RSP. For issue 5 (no `EncryptBeforeSigning` support), the BSP explicitly states in its section 6.1 ('Processing Order'), that both, encryption before signing as well as signing before encryption, may be appropriate depending on the application scenario. In so far, both protection orders must be supported by a WS-I compliant stack. However, that actually has got nothing to do with supporting the WS-Sec-Pol `EncryptBeforeSigning` assertion. As the BSP explicitly allows for out of band agreement for specifying the use of WS-Sec, not supporting `EncryptBeforeSigning` can be considered to be WS-I compliant. Finally, for issue 9 (deviating Processing of SignedParts/Body), the BSP states in its section 19.4 that *"it is RECOMMENDED that applications signing any part of the SOAP body sign the entire body."* However, the WS-Sec-Pol specification is absolutely clear at

this point itself (cf. [4], section 4.1.1).

All in all, none of the core interoperability problems detected is due to violating WS-I profiles. Only 1 issue out of 10 is reinforced by the BSP. This, taken together with the test results of the previous section, seems to imply that the approach of replacing true interoperability testing by WS-I compliance checking and neglecting the relation between WS-Sec-Pol/WS-RM-Pol assertions and SOAP messages is not sufficient for ensuring interoperability between WS stacks. Note that WS-I does not question the use of WS-Policy standards. The RSP recommends using WS-RM-Pol, the BSP states that "*strict policy specification and enforcement regarding which message parts are to be signed*" ([13], section 19.4) is a countermeasure against attacks and the so-called *delivery package* of the RSP ships with a few WS-Policy definitions (though without defining the effect on SOAP messages in detail). However, it leaves out a detailed treatment of WS-RM-Pol or WS-Sec-Pol.

V. Related Work

In SOA research, considerable efforts have been undertaken towards testing QoS, for example [21], [22], in the traditional sense of measurable network qualities like throughput or latency. [23] provides an evaluation of the WS-Sec implementation of the Axis2 WS stack, but only considers the processing time and message size when using different WS-Sec features. In contrast, we focus on testing the interoperability of implementations of QoS features as provided by WS-* standards and security and reliability in particular.

[20] discusses challenges of testing Web Services and security in SOA environments. Interoperability is identified as a core requirement for testing service compositions, but an interoperability assessment of different WS-* implementations is not provided. Several approaches such as [24], [25] target at testing interoperability of Web Services, but do not consider WS-* based QoS features.

In the area of actually testing interoperability of WS-* based QoS features, the majority of approaches follows [26] in defining interoperability testing as *"Conformance testing to ensure compliance with SOA protocols and standards"*. [27] describe a tool for statically validating WS-Sec-Pol configurations and [28] demonstrate how to use predicate logic to statically validate security policies. [7] present an approach for checking SOAP messages for WS-Sec-Pol conformance at run-time. [29] check conformance to the BSP statically and dynamically by inspecting SOAP messages. All these approaches analyze the configurations and message traffic of a single WS stack instance whereas we focus on the interaction of two WS stack instances of different solution providers. While we investigate whether or not communication can be completed successfully these approaches are able to check conformance to the WS-* specifications. In so far, these kind of approaches and our approach can be considered to complement each other.

[30] presents an interoperability assessment of SOA products with respect to WS-* standards for the Hungarian e-Government infrastructure. They evaluate only two security related test cases, but six SOA products and conclude that some products (including Metro/GlassFish) are mature for WS-* interoperability. The problems identified in the work at hand, however, reveal that much more thorough testing is needed.

[31] provide a mature framework for testing, monitoring and analyzing Web Services that are secured via WS-Sec-Pol. While interoperability assessment across different stack vendors is out of scope, it may be useful for streamlining the interoperability assessment of more WS stacks as described in this work. Moreover, the pattern-based approach for deriving WS-Sec-Pol configurations described in [12] may be used for deriving test cases that go beyond this work's test cases in providing application level features such as mutual authentication.

VI. Conclusion and Future Work

In this paper, we have thoroughly analyzed the interoperable implementation of WS-Sec-Pol/WS-RM-Pol by two major Java-based WS stacks. Our results show that Web Services developers cannot rely on interoperability of such WS-* features as provided by arbitrary WS stacks. While WS-I's work indisputably contributed to better Web Services interoperability, it apparently is not sufficient for guaranteeing interoperability of WS-Policy-driven QoS implementation. Using WS-Sec/WS-RM without WS-Policy however places unacceptable burdens on Web Services developers in requiring them to manually manipulate SOAP messages and agreeing on security mechanisms without predefined format. Future work will focus on streamlining interoperability testing across platforms by tool support and on deriving a comprehensive set of WS-Policy-based test cases that can be used among WS stack vendors to assess cross-stack interoperability.

References

[1] OASIS, *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)*, OASIS, February 2006.

[2] OASIS, *Web Services Reliable Messaging (WS-ReliableMessaging) Version 1.2*, OASIS, February 2009.

[3] W3C, *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, W3C, April 2007.

[4] OASIS, *WS-SecurityPolicy 1.3*, OASIS, February 2009.

[5] ——, *Web Services Reliable Messaging Policy Assertion (WS-RM Policy) Version 1.2*, OASIS, February 2009.

[6] L. Martino and E. Bertino, "Security for web services: Standards and research issues," *Int. J. Web Services Res.*, vol. 6, no. 4, pp. 48–74, 2009.

[7] N. Gruschka, N. Luttenberger, and R. Herkenhöner, "Event-based SOAP message validation for WS-SecurityPolicy-enriched web services," in *Proceedings of the 2006 International Conference on Semantic Web & Web Services, SWWS 2006, Las Vegas, Nevada, USA, June 26-29, 2006*. CSREA Press, 2006, pp. 80–86.

[8] N. Gruschka, M. Jensen, and T. Dziuk, "Event-based application of WS-security policy on SOAP messages," in *Proceedings of the 2007 ACM workshop on Secure web services*, ser. SWS '07. New York, NY, USA: ACM, 2007, pp. 1–8.

[9] F. Curbera, R. Khalaf, and N. Mukhi, "Quality of service in SOA environments. an overview and research agenda (quality of service in soa-umgebungen)," *it - Information Technology*, vol. 50, no. 2, pp. 99–107, 2008.

[10] R. Khalaf, A. Keller, and F. Leymann, "Business processes for web services: principles and applications," *IBM Syst. J.*, vol. 45, pp. 425–446, January 2006.

[11] H. Nezhad, B. Benatallah, F. Casati, and F. Toumani, "Web services interoperability specifications," *Computer*, vol. 39, no. 5, pp. 24–32, May 2006.

[12] M. Menzel, R. Warschofsky, and C. Meinel, "A pattern-driven generation of security policies for service-oriented architectures," in *Proceedings of the 2010 IEEE International Conference on Web Services, Miami, Florida, USA*, ser. ICWS '10. IEEE Computer Society, 2010, pp. 243–250.

[13] WS-I, *Basic Security Profile Version 1.1*, WS-I, January 2010.

[14] ——, *Reliable Secure Profile Version 1.0*, WS-I, November 2010.

[15] L. E. Moser, P. M. Melliar-Smith, and W. Zhao, "Building dependable and secure web services," *Journal of Software*, vol. 2, no. 1, pp. 14–26, 2007.

[16] J. Schwalb, A. Schönberger, and G. Wirtz, "Approaching interoperability testing of QoS based on WS-* standards implementations," in *The 4th Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing (NFPSLAM-SOC'10), co-located with 8th IEEE European Conference on Web Services (ECOWS 2010), Ayia Napa, Cyprus*. IEEE, December 2010.

[17] J. Schwalb and A. Schönberger, "Analyzing the Interoperability of WS-Security and WS-ReliableMessaging Implementations," Otto-Friedrich-Universität Bamberg, Technical Report: Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik 87, 09 2010.

[18] P. Wegner, "Interoperability," *ACM Comput. Surv.*, vol. 28, no. 1, pp. 285–287, 1996.

[19] *Web Services Security UsernameToken Profile 1.0*, OASIS, March 2004.

[20] A. Barbir, C. Hobbs, E. Bertino, F. Hirsch, and L. Martino, "Challenges of testing web services and security in SOA implementations," in *Test and Analysis of Web Services*, L. Baresi and E. D. Nitto, Eds. Springer Berlin Heidelberg, 2007, pp. 395–440.

[21] Gerardo Canfora and Massimiliano Di Penta, "Service-Oriented Architecture Testing: A Survey," in *Revised Tutorial Lectures of the International Summer Schools for Software Engineering (ISSSE 2006 - 2008), Salerno, Italy*, ser. Lecture Notes in Computer Science, A. D. Lucia and F. Ferrucci, Eds., vol. 5413, 2006 - 2008, pp. 78–105.

[22] A. Bertolino, G. D. Angelis, and A. Polini, "A QoS Test-Bed Generator for Web Services," in *Proceedings of the 7th International Conference on Web Engineering (ICWE 2007), Como, Italy, July 2007*, ser. Lecture Notes in Computer Science, L. Baresi, P. Fraternali, and G.-J. Houben, Eds., vol. 4607, July 2007, pp. 17–31.

[23] M. Shopov and N. Kakanakov, "Evaluation of a single WS-Security implementation," in *Proceedings International Conference on Automatics and Informatics, Sofia, Bulgaria*, October 2007, pp. 39–42.

[24] S. Shetty and S. Vadivel, "Interoperability issues seen in Web Services," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 9, no. 8, pp. Seoul, Republic of Korea, pp. 160–169, August 2009.

[25] A. Bertolino and A. Polini, "The audition framework for testing web services interoperability," in *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, ser. EUROMICRO '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 134–142.

[26] W.-T. Tsai, X. Zhou, Y. Chen, and X. Bai, "On testing and evaluating service-oriented software," *Computer*, vol. 41, pp. 40–46, August 2008.

[27] K. Bhargavan, C. Fournet, A. D. Gordon, and G. O'Shea, "An advisor for web services security policies," in *Proceedings of the 2005 workshop on Secure web services*, ser. SWS '05. New York, NY, USA: ACM, 2005, pp. 1–9.

[28] Y. Nakamura, F. Sato, and H.-V. Chung, "Syntactic validation of web services security policies," in *Proceedings of the 5th international conference on Service-Oriented Computing*, ser. ICSOC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 319–329.

[29] S. Prennschütz-Schützenau, N. K. Mukhi, S. Hada, N. Sato, F. Satoh, and N. Uramoto, "Static vs. dynamic validation of BSP conformance," in *Proceedings of the 2009 IEEE International Conference on Web Services, Los Angeles, CA, USA*, ser. ICWS '09. IEEE Computer Society, 2009, pp. 919–927.

[30] B. Simon, Z. László, B. Goldschmidt, K. Kondorosi, and P. Risztics, "Evaluation of WS-* standards based interoperability of SOA products for the hungarian e-government infrastructure," in *Proceedings of the 2010 Fourth International Conference on Digital Society*, ser. ICDS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 118–123.

[31] M. Menzel, R. Warschofsky, I. Thomas, C. Willems, and C. Meinel, "The service security lab: A model-driven platform to compose and explore service security in the cloud," in *Proceedings of the 2010 6th World Congress on Services, Miami, Florida, USA*, ser. SERVICES '10. IEEE Computer Society, 2010, pp. 115–122.