Extending the Similarity-Based XML Multicast Approach with Digital Signatures

Antonia Azzini, Stefania Marrara
Dipartimento di Tecnologie dell’Informazione
Università degli Studi di Milano
Crema, Italy
antonia.azzini@unimi.it
stefania.marrara@unimi.it

Meiko Jensen, Jörg Schwenk
Horst Görtz Institute for IT-Security
Ruhr-University Bochum
Bochum, Germany
meiko.jensen@rub.de
jörg.schwenk@rub.de

ABSTRACT
This paper investigates the interplay between similarity-based SOAP message aggregation and digital signature application. An overview on the approaches resulting from the different orders for the tasks of signature application, verification, similarity aggregation and splitting is provided. Depending on the intersection between similarity-aggregated and signed SOAP message parts, the paper discusses three different cases of signature application, and sketches their applicability and performance implications.

Categories and Subject Descriptors

General Terms
Security, Standardization

Keywords
XML similarity, SMP protocol, XML Signature, similarity-based aggregation

1. INTRODUCTION
Web Services (WS) Security [14] and associated emerging standards define SOAP-level techniques to move security related information along with message content, maintaining functionality and interoperability in business processes. Since SOAP messages often carry vital business information, their integrity and confidentiality needs to be protected, and SOAP message security assurance is a challenging part of SOA integration.

SOAP security activities consist of encryption operations and XML filtering [4], with particular interest in signing and verification, that includes parsing, validation, transformation and other document-level operations.

While protecting message confidentiality is of paramount importance, it is often desirable to encrypt only parts of a message that is being sent from one entity to another, so that intermediate nodes between the two entities can process the message appropriately.

Another important issue regards the performance of SOAP processing. Indeed, like other XML-based protocols, SOAP can consume a large amount of network resources when transmitting messages over the wire. This issue has drawn great interest, and many studies have proposed techniques for enhancing SOAP’s performance. SOAP messages consume significant network bandwidth and therefore cause higher latency than other, competitive technologies.

While a great effort has been made to optimize SOAP performance in transmission, less attention has been given to the problem of reducing the overhead of applying security policy rules to a large number of messages to be transmitted. Recent studies [5] investigate the possibility of using a similarity-based approach in order to identify similar SOAP messages to be aggregated in a single message at the sender side before performing encryption and signing.

In this paper we discuss aggregation in the framework of general WS-Security performance. WS-Security utilizes existing XML digital signature and encryption models to specify how to attach security tokens to SOAP messages, together with a set of processing rules. Digital signatures play an important role, due to their non-repudiation, authentication and integrity capabilities. The paper is organized as follows: Section 2 presents a sample scenario to depict the application of the presented approach, Section 3 reports some basic foundations necessary to understand this work, Section 4 discusses in detail three possible cases of application of our message aggregation approach outlining advantages and drawbacks, and finally Section 5 depicts conclusions of this work and future possible developments.

2. SAMPLE SCENARIO
Most research on SOAP performance focused on the reduction of SOAP message size and, thereby, on decreasing the overall traffic in the network. This can be done by reducing the size of each SOAP message individually via compression or by combining SOAP messages to avoid sending duplicate parts of the messages.
A similarity-based SOAP multicast protocol has to satisfy three main aspects: first, it must be able to measure the similarity between messages to determine which messages are similar enough to be grouped together into one message. Second, a special SMP message structure based on the standard SOAP envelope must be defined to contain the data of multiple recipients in one message. Third, the SMP solution needs to deal with the processing of aggregated SOAP messages at intermediary nodes.

In this paper we consider as running example a national meteorological service authority (NMSA), which usually broadcasts weather forecasts and actual weather data to interested customers. For instance, airports may use this data to decide on their operation mode (e.g., close runways due to storm or fog). Therefore, customers have to register themselves to the NMSA, providing the location they want to receive the forecast for, and a communication endpoint for data delivery. As weather is likely to be equally bad in a broader area, the resulting weather forecasts for a certain region are probably containing similar or even identical weather data values, such as temperature, cloudiness, and wind strength and direction.

As the customers of the NMSA are spread over the whole Internet, the distribution of the weather data can not be accomplished via traditional IP-based multicast. Thus, the weather data usually has to be transmitted using single TCP connections in unicast style. This poses a severe load overhead, since SOAP is based on XML and therefore inherits all XML’s verbosity disadvantages. When there are many transactions involving similar messages, one by one differential encryption, decryption and signature checking of SOAP messages can generate a very large amount of work overload for the service requester and provider.

Figure 1 shows the XML tree structure of how a weather data broadcast SOAP message may look like. The Header of the message contains the information related to the signature, certified by the corresponding tokens indicated as AuthToken in this example, and all the security information presented in the message. Reference elements specify the resources being signed in the SOAP message, and they are included in the element SignedInfo to protect them from tampering. Our approach is based on a similarity-based SOAP multicast protocol and digital signatures work in order to reduce the size of all multicast messages, by joining their common parts in a new one defined as a new single part of the SMP body. The new message structure is represented in Figure 2 which shows an aggregated SMP message that represents five nearly-identical SOAP messages. The only data value that slightly differs across these messages is the value of wind speed, which is 3 beaufort for messages 1, 2, and 4, and 2 beaufort for messages 3 and 5. Apart from this, the original messages are completely identical. In the aggregate message the signed body is described by a SMP header, that specifies all the information regarding how the message has been re-organized, and a SMP body, divided into the common element, that contains all the information that remain unchanged in the messages, and the distinctive element, that explains all the data that cannot be aggregated, divided into single parts. In such a scenario the overall computation time is reduced since the hash function related to the common parts of all the SOAP messages is computed only the first time, and it is presented for all the others. Different hash functions are then used for the distinctive remaining parts of the aggregated messages.

3. FOUNDATIONS AND OBJECTIVES

In this section we present some notions and approaches that are preliminary to the idea presented in this work.

3.1 Similarity-Based Aggregation in XML

In the past few years, there have been an increased interest in developing efficient techniques for comparing XML-based documents both in the field of information retrieval (IR) and database retrieval. Before discussing the possibility of applying an XML Signature to an aggregate SOAP message, we briefly review the proposals available in literature on similarity computation that could provide the foundation for message aggregation. All available algorithms differ in the efficiency of the aggregation proposed and in the computational costs required. Application to the SOAP aggregation poses strict requirements on both accounts so an efficient trade-off becomes necessary.

3.1.1 XML Structural Similarity

In the literature it is possible to retrieve various approaches for determining structural similarities between XML documents. Most of them derive from the techniques for finding edit distance between strings (e.g., [11]). In
essence, all these approaches find the cheapest sequence of edit operations that can transform one tree into another. Early approaches as [3] allow insertion, deletion and relabeling of nodes anywhere in the tree. Some works as, for instance, [2] restrict insertion and deletion operations to leaf nodes and add a move operator that can relocate a sub-tree, as a single edit operation, from one parent to another. The approach presented in [1] allows insertion and deletion operations of leaf nodes, and allows the relabeling of nodes anywhere in the tree. The paper [15] extends a previous work by adding the operations insert tree and delete tree to allow insertion and deletion of whole sub-trees. The last structural similarity approach we review is [7], which uses the Fast Fourier Transform to compute similarity between XML documents.

3.1.2 Semantic Similarity Measures

Semantic Similarity Measures are used in evaluating the effectiveness of web search mechanisms in finding and ranking results [12]. In the field of Information Retrieval (IR), knowledge bases (thesauri, taxonomies and/or ontologies) provide a framework for organizing words (expressions) into a semantic space [9]. Therefore, several methods have proposed to determine semantic similarity between concepts in a knowledge base. They can be classified as edge-based approaches and node-based approaches. The edge-based approach is used to evaluate semantic similarity in a knowledge base. As instance, [10] estimate the distance between nodes corresponding to the concepts being compared: the shorter the path from one node to another, the more similar they are. Nevertheless, a widely known problem with the edge-based approach is that it often relies on the notion that links in the knowledge base represent uniform distances [9]. In real knowledge bases, the distance covered by a single link can vary with regard to network density, node depth, link type and information content of corresponding nodes [9].

3.2 Similarity-Based Aggregation and Broadcasting of XML messages

The basis of our approach, initially presented by Damiani and Marrara in [5], is [16] proposing a SOAP multicast technique, called Similarity-based Multicast Protocol (SMP), which takes into account the similarity of SOAP messages. SMP was designed to deal with SOAP performance issues by exploiting the similar structure of SOAP messages. The goal is to reduce the total traffic generated over a network when sending SOAP responses from servers to clients. SMP allows similar SOAP messages that share some parts of the SOAP template to be sent as a single customized SMP message instead of being sent as multiple copies. Clients’ addresses are represented as strings and stored in the SMP header, which is encapsulated inside the SOAP message body. The SMP body is also embedded inside the SOAP message body. There are two sections in the SMP body: the Common section containing common values and structures of all messages addressed to clients encoded in the SMP header; and the Distinctive section containing individual different parts for each response message.

The outermost envelope is referred to as an SMP message. The destination of an SMP message, which is specified in the SOAP header, is the next router in a network when the message is forwarded to all clients given in its SMP header. Despite its advantage of saving network traffic, SMP has a remarkable disadvantage: it uses a conventional routing protocol (OSPF) to deliver messages to clients. Since OSPF uses Dijkstra’s algorithm, SMP messages are routed along their shortest paths to destinations. Two nodes of a network are often connected with multiple paths. Therefore, sending messages just along least hop paths does not maximize the saving of traffic resulted from the similarity of messages. In addition, SMP has a user-configured time frame. During this time period, outgoing SOAP response messages will be lined in a queue if their similarity level falls within a threshold limit. When a new request message arrives at the server, the server generates its corresponding SOAP response message and computes its similarity against existing on-queue messages. The algorithm used for computing similarity relies on structural and content comparisons (see Section 3.1). If the computed similarity satisfies the threshold then it is inserted into the queue. If not, the messages that already reside in the queue are sent out as an aggregated SMP message. As a result, the queue is empty for new requests and the above aggregation steps can be repeatedly carried out again. Messages in the queue can also be dispatched automatically after the defined time period expires. It is important to note that to deploy SMP in a real network, all routers in the network need to be SMP-compatible. This can be done by installing an SMP software, which is an implementation of the proposed SMP on each router, to enable it to interpret SMP messages. The SOAP header in an SMP message specifies the next hop router as the message’s destination. Therefore, when an intermediary router receives an SMP message, it processes the message as if it is the final destination of the message. Since an SMP-compatible router operates on the application layer, it has full access to the message’s envelope and parses the SOAP body to get the list of clients encoded in the SMP header and the actual payload in the SMP body.

3.3 XML Signature and WS-Security

Along with the benefits that web services provide for network’s transactions, also considerations about data integrity and security should be posed, together with the rights and access permissions of the message’s users. Since web services communications regard XML formatted messages a satisfactory solution, also considered in this approach, uses XML Signature for identifying the requester web service, validating message integrity, conforming non-repudiation and ensuring proper security.

As reported in the literature [17] XML Signature defines a set of rules and syntax that are used to handle digital signatures of data. Different from a traditional digital signature, that is calculated over a complete message, generally an XML signature considers a message or a document as consisting of many elements, and it signs one or more (i.e. the aggregation) of such elements, making the sign process more flexible and practical. As also reported in [18] XML Signature Working Group have been creating a specification for defining digital signatures in an XML format. The author also reported the main reasons for an XML signature standard when there are alternative mechanisms in order to maintain data security in transit. The first corresponds to the portability of an XML signature, entwined within the XML data; the second is flexibility, due to their capability to refer to many documents or parts of a single document,
as required, also due to the use of XML standards. Furthermore, XML signature is optimized for XML documents, but it can also be used to sign non-XML documents.

The XML signature components are shown in Figure 3.

XML digital signatures can be divided into three main classes, depending on where the signature is applied: indeed, if used to sign a resource outside its containing XML document it is called a detached signature; if used to sign some part of its containing document it is called enveloped signature, and finally, if it contains the signed data within itself it is called enveloping signature.

The main steps carried when a digital signature is implemented are as follows: 1) create a SignedInfo element with SignatureMethod, CanonicalisationMethod and References; 2) canonicalize the XML document; 3) calculate the SignatureValue, depending on the algorithms specified in the SignedInfo element; 4) create the digital signature, which also includes the SignedInfo, KeyInfo and SignatureValue elements.

4. DIGITAL SIGNATURES AND XCAST

Whatever kind of similarity-based aggregation is used, if the data to be delivered using XCast has to be integrity-protected, it must be digitally signed. As digital signatures are invalidated on any character modification within the signed block, this poses some problems to the use of similarity-based aggregation. These are to be investigated next. Please note that the scenarios and requirements discussed below must be considered regardless of the actual similarity approach used, as they apply to any kind of similarity-based aggregation. However, the optimization potential of the signature application strongly depends on the optimization level of the similarity approach used.

Regarding the application of digital signatures to XCast style broadcast messages, it is necessary to consider four different tasks in a SOAP message’s lifecycle: 1) The task of applying a digital signature to the SOAP message’s contents, 2) the aggregation of several single SOAP messages to a single, similarity-based broadcast message, 3) the splitting of a single broadcast message into several new messages (either smaller broadcasts or single SOAP messages), 4) the verification of the digital signatures applied to the document.

In the following, we are discussing the different scenarios that result from performing these tasks in different orders.

4.1 Sign-Join-Split-Verify: Naïve Approach

Obviously, as SOAP messages with digital signatures still remain SOAP messages, they can be used as-is for similarity-based aggregation. According to the WS-Security and XML Signature specifications, the application of a digital signature to SOAP messages does not require many changes to the document’s contents. It causes the addition of a new SOAP header value (Security), and—depending on the XML referencing approach taken—it might require an additional ID attribute at the root element of the signed subtree. Either way, most of the document’s contents remain identical, thus the similarity-based aggregation most likely will perform its benefits even in this kind of scenario.

Once the message’s aggregation and re-separation tasks have been completed, the resulting single SOAP messages are identical to those messages used in the signature-application step. Thus, a verification of the contained digital signatures will result in the same hash values, and signature verification succeeds.

Nevertheless, though this approach can benefit from the performance boost effects described in [16], it does not provide any valuable improvements regarding digital signature application in detail.

4.2 Join-Sign-Split-Verify: Broadcast Approach

Considering the original SMP-based XCast approach [16], the use of this approach would require re-application and re-verification of digital signatures on every router from the sender to all recipients. Obviously, this approach has severe flaws. Thus, we investigate the scenario considered for XCast, and provide some details on how to apply digital signatures appropriately.

Figure 6 shows an example routing path for an XCast-
Figure 6: XCast scenario with digital signatures using the Join-Sign-Split-Verify approach

Figure 7: Examples of semantically equivalent XML fragments

Based on message broadcast. Starting at the message creator’s side, it shows how three different (but similar) SOAP messages are to be transmitted to three different recipients. As the recipients are not located within the same network, there are two XCast routers located at the network edge nodes.

The XCast approach now consists in aggregating the three SOAP messages at the sender side to become a single new message (compare Figure 2). This single message is then delivered to the first XCast router (first *split* block in the figure). Here, the message is processed and split into two new messages ($C_1$ and $C_4$). $C_1$ only contains those parts of the SMP message that are addressed to $R_1$ or $R_2$, while $C_4$ contains only the single SOAP message addressed to $R_3$. This processing mode is repeated for the SMP message $C_1$ accordingly. In the end, the final XCast routers ($C_2$, $C_3$, and $C_4$) transform the last SMP messages to the original single SOAP messages.

Before going into the details of the approach, we discuss here the benefits of using a similarity approach to aggregate the messages *before* applying the signature. Indeed the similarity approach, applied *before* computing the canonicalization of the original messages, can identify parts of messages that are identical in a semantics point of view but with different structure. These are parts that *applying the canonicalization algorithm* will result *identical*. Some examples of such documents are shown in Figure 7. Using a traditional approach, we should first canonicalize each original message and then recognize that these parts are *identical* and can be aggregates. With our similarity-based approach we can first recognize and aggregate these *semantically but not structurally identical* parts and then apply the canonicalization procedure only once on the resulting aggregate saving time and computational costs. Indeed it is a well known that it is the canonicalization process the cost bottleneck of the entire XML signature algorithm.

After this short discussion about the potentiality of the similarity approach, in order to enable digital signatures in a most efficient way, it is necessary to distinguish three different cases in terms of the relation of SMP messages and signed XML subtrees. These are to be discussed next.

4.2.1 Case 1: Signed Subtree is Common

The first case (illustrated in Figure 8) is that a digital signature must be applied to an XML subtree that is semantically identical among all aggregated SOAP messages. An example would be a PKI certificate of the sender, which may be contained in all outgoing SOAP message headers for authentication purposes. As such a certificate is a static set of string values, it will result in having the same, or at most a very similar, representation in XML for all outgoing messages.

As the signed subtree (the certificate data) is common to all SOAP messages considered in the aggregation step, it will completely be contained in the `<common>` part of the resulting SMP message. Thus, instead of having the signature calculated for every single SOAP message, a better approach consists in performing the aggregation step first. Then, the signature can be calculated on the appropriate XML subtree from the `<common>` part of the SMP message, and the resulting `<Security>` header can be added to the SOAP header within the common part of the SMP message.

Once the so-created SMP message is split at an XCast router, the common part of the SMP message will be copied into all outgoing SMP messages. Thus, also the signature header is copied.

At the final recipient, the last XCast router re-instantiates the original SOAP message, but it now includes all digital signature metadata in the new `<Security>` header. Additionally, as the signed contents did not change since their
4.2.2 Case 2: Signed Subtree is Completely Distinctive

Depending on the degree of similarity aggregation, it is possible to have signed contents to be completely distinctive for all SOAP messages considered (cf. Figure 9). An example may be a signed WS-Addressing message ID [8], which is intended to be different for all outgoing SOAP messages of a certain service.

In this case, the resulting hash values and thus the signature values definitely will be different for different SOAP messages (for else the signature would be breakable). Thus, there is not much opportunity here to gain the same performance optimization effects as in the previous case. It is necessary and inevitable to calculate every single digital signature itself, and have the resulting <Security> header placed in the SMP message appropriately.

Using this approach, the signed contents and also the <Security> header are placed in appropriate <distinctive> parts of the SMP message. Then, the XCast routers will split the SMP message as described above. The final XCast router will place the contents of the distinctive part at the corresponding position in the resulting SOAP message. Thus, the signed contents at the final recipient remain unmodified, and the signature verification will succeed.

The only potential performance gain consists in exploiting the fact that most of the <Security> header remains static, and thus will be common among all considered SOAP messages. Thus, it is possible to have most of the <Security> header placed in the <common> part of the SMP message. Nevertheless, for the <DigestValue> and <SignatureValue> elements it would be necessary to add appropriate new <distinctive> parts for all considered SOAP messages to the SMP message. Thus, the real performance impacts of this kind of optimization can be doubted.

4.2.3 Case 3: Signed Subtree is Partially Distinctive

Considerably the most common case for the relation of signed contents and distinctive subtrees consists in signed contents that are common to all SOAP messages, but contain some distinctive subtrees (cf. Figure 11). For example, SOAP messages invoking the same WSDL operation most likely have a very similar message structure, but differ in the text node values of their contents.

On the other hand, this case also implies the most difficulties in applying digital signatures, as the message fragments to be signed are no longer represented in a single block, but may be spread over a bunch of common and distinctive parts within the aggregated message. Thus, applying a digital signature in such a scenario causes some severe troubles, and may raise doubts on whether this is a viable approach at all.

Nevertheless, in the following we outline an approach that enables applying a digital signature to aggregated SOAP messages while keeping them valid on each XCast split. Thus, a signature is applied once for the aggregated message, and on each split is automatically copied into each of the resulting messages. This way, the end, each recipient is provided with a single SOAP message that still contains a valid digital signature that protects the very same message fragments as a common XML signature—applied prior to similarity-based aggregation—would provide.

Approach Description.

The main idea of this approach is to apply a signature for each part of the aggregated message that is to be signed (cmp. Figure 10). Thus, instead of performing a single signature on the whole message fragment, we suggest to apply n signatures on all common and distinctive parts that contain contents to be signed. Then, once the aggregated message is split at any intermediate XCast router, these signatures are attached to all resulting messages (either SMP or SOAP) that still contain contents covered by the signature. In the end, the final recipient has to verify the signature for each single part he receives in order to determine the validity of the signature as a whole.

Before coping with the security considerations for this approach, it is necessary to dive into some details of XML Signature for the realization. As explained in Section 3.3, XML Signature enables the application developer to instantiate one signature with a list of Reference elements, resulting in that each of these references is resolved, hashed, and verified as part of the overall signature verification task. This concept can be adapted easily to be used for the approach to be presented. Thus, each signed part of the aggregated message is referenced by its very own Reference element within the (single) SignedInfo block. Then, once the aggregated message is split at any XCast router, the SignedInfo block is added to all messages that still contain signed contents. This implies that some of the fragments referenced in the SignedInfo block are no longer present in the resulting (SMP or SOAP) message, but for those that remain, the referenced fragment’s hash values and thus the DigestValues are still valid. In the end, the only adaptation required for proper signature validation is that the final re-
recipient ignores all references that are not pointing to a valid location in the message document.

Security Considerations.

Investigating the security properties provided by the presented approach of fragmentized signature contents, an important precondition is to evaluate the reasonable requirements posed on a digital signature to be fulfilled. This strongly depends on the intended semantics behind the signature application.

The major purpose a digital signature is used for is to ensure data integrity. As every modification to the signed data contents would instantly invalidate the signature, a common digital signature perfectly fulfills the requirements of this task. Let us investigate this property regarding the approach presented above. As can be seen, every content that is referenced within the signature is also protected by the corresponding DigestValue, thus every modification to any of these signed message parts also instantly invalidates the aggregated signature as a whole. In this sense, given that the union of all message fragments results in the complete original SOAP message, the aggregated signature provides the same level of data integrity assurance as a single signature would. The only property an attacker could exploit here is to add or remove some of the parts of the aggregated SMP message in order to have these be present or absent intentionally in the recipient’s SOAP message. Thus, an attacker can only add or remove message fragments at those fragment cut-off points that were used during the similarity-based aggregation (cmp. Figure 12). Nevertheless, he can not add his very own contents, nor modify any of the existing parts without causing a signature invalidation, thus he can only “play with the aggregation-dependent message fragments”, so to speak.

Obviously, this attack vector still provides some opportunities—depending on the actual SMP message and scenario—for some sophisticated XML rewriting attacks (cf. [13, 6]), but it can be shown that all of these issues can be addressed by embedding appropriate message structure metadata along with each part of the aggregated message (see also Section 5). Thus, these issues can be coped with, and in the end, the data integrity provided by the aggregated signature approach provides the same level of data integrity as a common signature applied prior to similarity-based aggregation would.

4.3 Join-Sign-Verify-Split: Singlecast Approach

The Join-Sign-Verify-Split is the case in which we obviously expect the greatest performance gains. In this case we suppose that the messages are first aggregated and then signed. At the receiver side we suppose to have a trusted
entity which is in charge to verify the signature directly on the aggregate message (hence the verification process is performed only once for all the recipients) and then the trusted un-signed messages are delivered to the final Recipients. In this case, at the signing side, all consideration carried out for the Join-Sign-Verify-Case still hold. The only difference consists in the presence of this trusted entity that performs the verification process directly on the aggregate message, before the splitting phase, saving time and computational costs also in this process.

4.4 Other Application Schemes

The other potential permutations of the four tasks do not result in useful scenarios. Either do they require nonsense activities (e.g. verifying digital signatures that have not yet been created) or they fail to be of interest for real-world scenarios (e.g. the Sign-Verify-Join-Split approach).

5. CONCLUSION AND FUTURE WORK

In this paper, we investigated the interplay between similarity-based SOAP message aggregation and digital signature application. We provided an overview on the approaches resulting from the different orders for the tasks of signature application, verification, similarity aggregation and splitting. Depending on the intersection between similarity-aggregated and signed SOAP message parts, we discussed three different cases of signature application, and sketched their applicability and performance implications.

Concluding on the results, we found out that the approach of interweaving similarity-based aggregation and signature application has a lot of potential for providing real performance gains, compared to each of these tasks being performed separately. Thus, an obvious future work consists in finishing the ongoing prototype implementation of the approaches, and providing a full evaluation on the different performance and network bandwidth impacts. Furthermore, the investigation of using a similarity-aware canonicalization approach prior to signature value calculation seems promising, as it may provide further optimization potentials. Finally, we intend to perform an in-depth analysis on the provided level of security, and to derive a formal security proof on the approach of aggregated signatures as described in Section 4.2.3.

6. REFERENCES