SECURING SOAP MESSAGES WITH A GLOBAL MESSAGE HANDLER AND A STANDARDIZED ENVELOPE

M.Pather[a] and LM Venter[b]

[a]Faculty of Engineering (Information Technology Unit), Nelson Mandela Metropolitan University
[b]School of Computer Science, University of South Africa

[a] maree@nmmu.ac.za
[b] ventelm@unisa.ac.za

ABSTRACT

This paper argues that, in a collaboration context, instead of Web services requiring client applications to comply with individual permutations of security configurations, a standardized mechanism should be established to ensure global security-interoperability. Such a solution would facilitate providing Web services in Grid Services contexts as well.

A framework is proposed which comprises, inter alia, a standardized SOAP envelope and a standardized message-handling service. The standardized message-handling service receives and generates standardized SOAP envelopes at both the consumer and provider sides. The SOAP envelopes contain standardized security headers based on WS-* standards and standard security technologies. The message-handler is a Web service that acts as a relay to the actual service being called, ensuring standardized interoperability features, which includes standardized security.

KEY WORDS

SOAP security, security interoperability

SECURING SOAP MESSAGES WITH A GLOBAL MESSAGE HANDLER AND A STANDARDIZED ENVELOPE

1 INTRODUCTION

"Global Electronic Market-Space" (GEM), as used in this paper, refers to a hypothetical standardized collaboration platform in which organizations of any ilk, location and culture can participate. However, the framework proposed is also applicable to a smaller collaboration-context, in which the Web services composition is pre-specified and a standardized security model is required. The proposed logical infrastructure of the GEM is based on a Web services-Services-Oriented Architecture (WS-SOA) (W3C, 2004[2]) in which (XML-based) SOAP (W3C, 2000, 2002[1,2], 2003[1,2,3,4], 2004[1,3,4]) provides the globally-interoperable messaging-interface and XML provides the globally-process-able data format. Specific business services can be architected according to individual business requirements by extending a (hierarchical) system of specifications comprised, inter alia, of: a generic business pattern; generic business processes and atomic activities; generic XML Schema Documents (XSDs); and generic common business classes (headers/programming-interfaces and method-signatures/virtual methods). The Global Message-Handler (GMH) module at both ends subscribes to a global specification and ultimately provides the crucial "first-line" global-interoperability application-interface. The GMH is implemented as a Web service, for which the interface contract is described by a readily-available GEM-standardized Web Services Description Language (WSDL) specification (W3C, 2004[2]).

The main thread of this paper is the argument for a SOAP-level security model to promote interoperability in a GEM context. Thus, an exposition of a proposed GEM logical-infrastructure is first provided; this will be followed by a suggested framework for providing security within this infrastructure.

2 THE GEM LOGICAL-INFRASTRUCTURE

In essence, it is suggested here that *the common (virtual) collaboration-platform* should comprise the following parts:

a) An interoperable *standardized message format* and *an interoperable standard message-delivery infrastructure* encompassing *an application-level message-handler;* these are entirely platform-independent and constitute the *technical (standardized) GEM interoperability-interface*.

b) A *hierarchically-standardized business infrastructure*, pre-specified by experts in the business domain. The top-most tier of the business infrastructure should include:

   i. A minimalist, generic, *standardized business-pattern* based on the fundamental buying-selling commercial business-pattern, comprising

      1) a *pre-specification for standardized generic business processes* (for example, placing an order);
      2) the *specific atomic activities* (for example, submitting an order-form, acknowledgement of receipt of an order-form, error-handling messages, response messages, and so forth);
      3) *a standardized generic document-set* (including, for example, a standardized order-form and a standardized invoice) based on

4) *standardized metadata* (common nomenclature and common semantics used in describing data-fields, for example, elements and attributes in the order-form XSD) and the corresponding
5) *standard choreography* for atomic activities (for example, acknowledgement of receipt of a form, error-handling messages, response messages, and so forth, within a particular transactional-conversation);

ii. *appropriate data-processing applications*, on the consumer and provider sides – inheriting from standardised programming-classes - designed to function in accordance with 1) to 5) above.

c) A *standardised security model* for securing the messages, the message-delivery and the application endpoints.
d) A *Registry/Repository mechanism* for the discovery of participants and for downloading specifications, to expedite the dissemination of the framework specifications. Registries would be arranged in *a hierarchy of related servers*, bearing content in a hierarchical order. At the higher levels, more-generic GEM specifications would occur from which specifications may be "inherited" by lower, specific business domains and sub-domains. This is essential in order to maintain uniformity for integration.

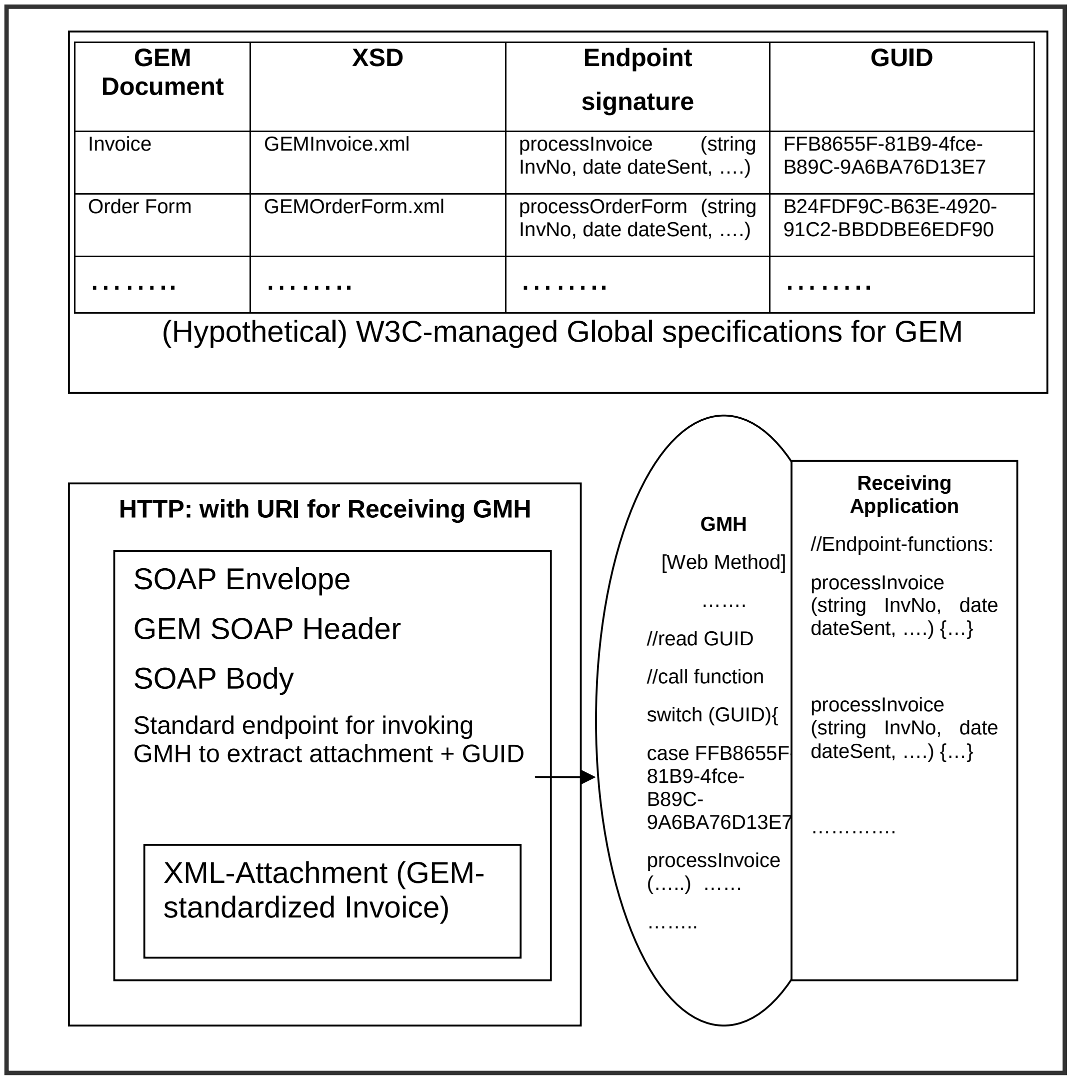
Figure 1 below illustrates the position of the GMH in an "anatomical" view of the Provider-side application.

The rationale for this structure is as follows: the common application-interface between provider and consumer, in a GEM, is a Web service, of which the Listener component can be elaborated into a Global Message Handler (GMH) with functions beyond simply acting as a listener-daemon. The GMH both receives and generates standardized SOAP-envelopes, which it processes in a specific manner. The standardized SOAP-envelope, in fact, has to primary functions: 1. to invoke, by SOAP-RPC, the GMH-service on the recipient side; and 2. to transfer XML-binary Optimised Packages (XOP) (W3C 2005[1]) between GMHs.

Data is carried in SOAP-attachments (rather than the SOAP body), using the Message Transmission Optimisation Mechanism (MTOM) standard (W3C 2004[1]). The attachment could itself be a SOAP-RPC (encoded) document, but Web-Friendly (Chatterjee and Webber, 2004:87-97) documents (XML or non-XML) are preferred. All GEM-standardized (XML) attachments would have a corresponding globally-unique identifier (GUID) (or equivalent) for a corresponding standardized endpoint-function header. "Header" refers to the function identifier; the function-name, arguments and corresponding data-types are provided, but the implementation is left to the developer. GEM endpoint-functions would be implemented as virtual functions, which can be overridden by the user; provision should obviously be made for overloading functions (allowing for variations in type and number of parameters). Thus, as illustrated in Figure 1, if a company should send a particular SOAP-attachment (based on a standardized XSD) for, say, an invoice, the receiving-GMH would read the corresponding GUID and call the (standardized) endpoint-function (header) that can process it (in a company-specific implementation); the recipient-application may even simply be designed to place the document in a repository for manual processing.

The table in Figure 1 illustrates the relationship between the GEM documents, the corresponding GUIDs, the corresponding XML schema documents and the corresponding endpoint-function headers; all of these are intended to be standardized. The illustration indicates that the GMH is invoked by SOAP-RPC; the business data is contained in an attachment (8-bit binary MIME-serialized XOP package). The external SOAP-envelope is also indicated as being carried by HTTP; whether these are the obvious choices is considered in the next section. An XOP package is created by placing a serialization of an XML Infoset in an extensible packaging format such as a MIME Multipart/Related package. Then, selected portions of the data that are base64Binary-encoded are extracted and re-encoded (that is, the data is decoded from base64) and placed into the

XOP package. The locations of those selected portions are marked with a special element (xop:Include) that links to the packaged data using URIs.



GEM Document	XSD	Endpoint signature	GUID
Invoice	GEMInvoice.xml	processInvoice (string InvNo, date dateSent, ….)	FFB8655F-81B9-4fce-B89C-9A6BA76D13E7
Order Form	GEMOrderForm.xml	processOrderForm (string InvNo, date dateSent, ….)	B24FDF9C-B63E-4920-91C2-BBDDBE6EDF90
……..	……..	……..	……..

(Hypothetical) W3C-managed Global specifications for GEM
HTTP: with URI for Receiving GMH
SOAP Envelope
GEM SOAP Header
SOAP Body
Standard endpoint for invoking GMH to extract attachment + GUID
XML-Attachment (GEM-standardized Invoice)
GMH
Receiving Application

```
[Web Method]
…….
//read GUID
//call function
switch (GUID){
case FFB8655F-81B9-4fce-B89C-9A6BA76D13E7
processInvoice (…..)  ……
……..
```

```
//Endpoint-functions:
processInvoice (string InvNo, date dateSent, ….) {…}

processInvoice (string InvNo, date dateSent, ….) {…}

………….
```

Figure 1. How the hypothetical GEM standard would operate

The precise standardized specification for the minimalist-business-pattern set of endpoint collaboration-functions, the corresponding business processes (atomic transactions and business activities), and the corresponding document schemas (XSDs) could be inferred from existing B2B-interaction models, such as BizTalk (www.BizTalk.org), RosettaNet (RosettaNet Consortium, 2002), and ebXML (www.ebXML.org) and standards such as BPEL (Chatterjee and Webber, 2004: 125-248).

2.1 The Message Format and Message-Delivery Infrastructure

SOAP-Web services provide a synchronous means for sending parameter-values to a remote method and receiving real-time responses. However, synchronous processing in a GEM-context has obvious ramifications for complexity (not-to-mention bandwidth) considerations in delayed business processes; further, maintaining state is considered not "Web friendly" in terms of REpresentational State Transfer (REST) architecture (Chatterjee and Webber, 2004: 96).

Synchronous processing involves multiple exchanges of business-process data in real-time. The messages would have to be packaged in a standardized manner for SOAP-handling and for security, on both sides, over a persistent request-response MEP. SOAP does not specify algorithms for the use of optimistic concurrency, roll back, or other transaction-processing techniques. However, WS-AtomicTransaction (W3C, 2004[6]) makes provision for such features; additional elements have to be added to the SOAP-envelope to enable this functionality. The more complex the standardized business processes are specified (in terms of processing and number of exchanges, for instance), the greater the potential for problems in synchronous transmission. Processes could require time for validation and ratification of the input-data and the response, for instance.

In synchronous messaging, only the Web server need have a valid Internet-Assigned Numbers Authority (IANA) Internet address (IP address and host name); the response messages are simply returned via the open link. In asynchronous messaging, the response messages have to be sent to a server with a valid Internet address, to be accessible via the Internet Domain Name System (DNS); this would require every consumer-participant to have either an HTTP-server available on the Internet. Client systems commonly act from behind a firewall, on which Network Address Translation (NAT) is typically configured and where IP-addresses are allocated dynamically.

Thus, each SOAP-request sent to the Provider-GMH would require a synchronous SOAP-response (a transaction-limited request-response MEP) or each Requestor would have to make provision for an Internet server for asynchronous processing (Server-to-Server). For (Client-to-Server) asynchronous activities, it is incumbent on the client-side to establish the connection, in each atomic activity. Further, each exchange must result in pre-determined, orchestrated closure (in terms of choreography) and be able to link to the next exchange with the same client, involving the same transaction; each exchange must be "conversation-bound".

The synchronous-versus-asynchronous (processing) dilemma is heightened by the following polemic. The proposed framework suggests that XML business process data be incorporated within a pre-specified SOAP-envelope, as an MTOM/XOP *attachment* to the SOAP-envelope (W3C, 2002[2,1]), rather than for it to comprise part of the actual body of the SOAP-envelope itself. In synchronous processing, there could be a performance drop in using SOAP-attachments as opposed to XML-data embedded in the SOAP-body. If the data were carried in the SOAP-body, the RPC-method for processing the data could have direct access to the data. In either case, the SOAP-header would need to be standardized for GEM purposes, for example, by adding features such as WS-Security.

The advantages of using attachments would include separating the (standardised) specifications for the SOAP-envelope and the business documents. An added advantage is that the entire attachment could then be encrypted with XML-Encryption (Siddiqui, 2002) and be signed with XML-Digital Signature, as a standardized GEM security mechanism.

Further, this mechanism would accommodate strategic 1:n and 1:1 dyadic relationships beyond the GEM-context, by allowing non-GEM-standardized document-formats to be attached as well. The choice-matrix may be represented as in Table 1 below:

	SYNCHRONOUS PROCESSING	ASYNCHRONOUS PROCESSING
XML-data contained in external SOAP-body	1. Over HTTP: Client has to initiate each session; many sessions may be required, which complicates orchestration. Idempotence and	3. Over HTTP: Requires queuing by every sending and receiving application. Requires each participant to have a Web server

	persistence issues.	(static IP address)
XML-data in attachment	2. Over HTTP: Requires attachment to be persisted on receiving end (non-repudiation). Possible performance issues. Idempotence issues.	4. Over SMTP: Standardized SOAP-Envelope (containing XML-document attachment) is attached to e-mail message.

Table 1: The choice-matrix for carrying data in the GEM

The following is offered as a possible means to resolve the available choices. If the advantages of asynchronous processing are to be sought, then an SMTP model would provide the better infrastructure. Its store-and-forwarding mechanism makes it a better candidate than HTTP in this regard; for asynchronous SOAP-over-HTTP, queuing would have to be implemented and idempotence (same-message) be provided for. The GMH for the SMTP model would be similar in function to that of the HTTP model (a standardized SOAP-envelope with SOAP-attachments); the GMH would simply process/generate SMTP messages rather than HTTP ones.

For synchronous processing, XML-data in an attachment is preferred. Using a standardized SOAP-envelope, as the direct means for invoking the GMH by RPC, devolves the first-line interoperability-interface function to the SOAP-envelope.

The Provider-side GMH, therefore, would provide for connecting to specific endpoint-functions, which handle standardized client-side requests. The Client-side GMH, on the other hand, would provide for connecting to specific endpoint-functions, which handle standardized server-side responses and standardized server-side SOAP-documents (for example, an invoice). The GMH on both sides would be capable of generating standardized SOAP messages – some with user-selected standardized attachments; others simply as an acknowledgement or a response to an error – for which only the receiving host URI (Uniform Resource Identifier) is required to be entered by the user; the attachments could be chosen and completed via user interface forms. Only relevant endpoint-functions need be available (depending on the user-community and the hierarchy-level).

As mentioned above, placing the data for processing in attached SOAP-documents promotes scalability and upgradeability of the standardized SOAP-envelope. This is in accordance with current modular, incremental development-methodologies. Using this (SOAP-attachment) message format, the message-handler (Listener) could even poll both the SMTP server and the HTTP server for standardized attachments.

This (SOAP-attachment) mechanism also allows a pre-specified security model to be imposed. A GEM-standardized SOAP-envelope could be used with a pre-specified standardized set of features, including WS-Security, XML digital signatures and that XML Encryption. The TCP connection could be secured by SSL/TLS (simply set on the Web server providing the service).

Most authentication mechanisms, including client certificates, rely on HTTP transfer, whereas SOAP is transfer-independent. Therefore, to create a custom authentication mechanism in order to decouple authentication from the transfer protocol, one would pass authentication credentials (e.g. X509 digital certificates) in the SOAP-header rather than at the HTTP/SMTP level. At the minimalist-pattern generic level, fine-grained access control and differential encryption would not be required. Whichever security mechanisms are used, they must obviously become part of the GEM standard.

In the minimalist-pattern proposed framework, *business documents* would, typically, be XML-documents, for example, order-forms and invoices, based on GEM-standardized generic XML-Schema documents (XSD's). The generic XSD's would not be comprehensive enough to accommodate all nuances required, but XSD's standardized at lower levels would extend them. The

GEM-standardized *orchestration* (pre-specified messaging sequence) for the exchange of GEM-standardized XML business documents (request messages, such as order-forms; acknowledgement messages; error messages; and response messages) would also be based on the minimalist-pattern, at the generic level. Figure 2 illustrates a typical choreography for a general atomic business activity.

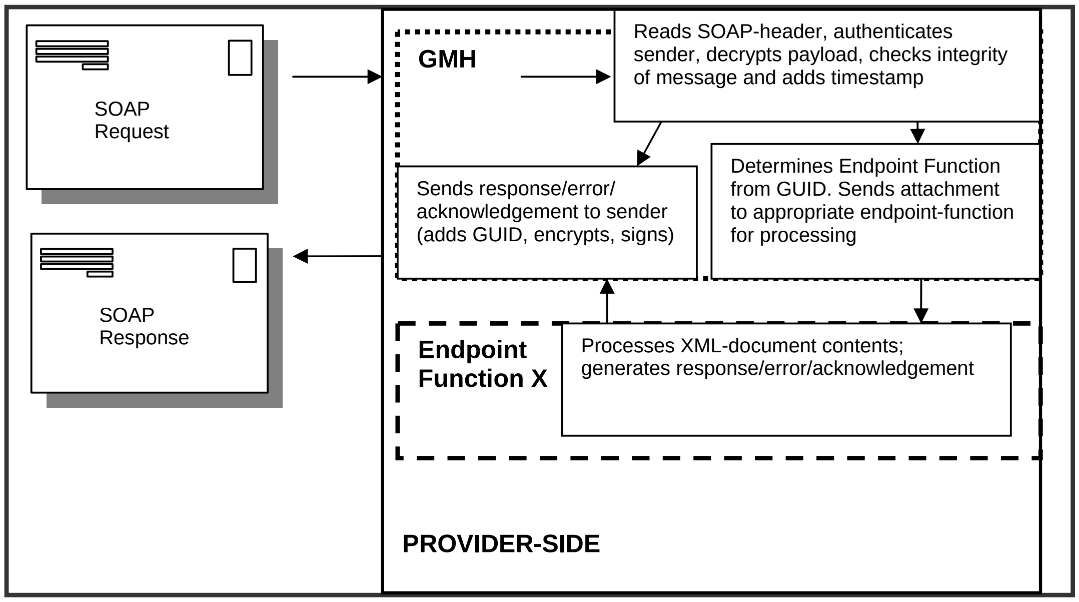

SOAP Request
SOAP Response
GMH
Reads SOAP-header, authenticates sender, decrypts payload, checks integrity of message and adds timestamp
Sends response/error/ acknowledgement to sender (adds GUID, encrypts, signs)
Determines Endpoint Function from GUID. Sends attachment to appropriate endpoint-function for processing
Endpoint Function X
Processes XML-document contents; generates response/error/acknowledgement
PROVIDER-SIDE

Figure 2. Provider-end of the GEM Message Pathway

The GMH could even be used in a non-commercial context – as, for instance, in the education system – by simply hosting the appropriate list of applicable GUIDs and the corresponding endpoint function libraries.

The Sending application for invoking the GMH-Service over HTTP would typically send a Request message such as the following:

```
POST /GMHInvokeService HTTP/1.1
Content-Type:text/xml
Content-Length:nnnn
SOAPAction:”urn:GMHInvokeService#GMHInvoke”
<soap:Envelope xmlns:soap='http://www.w3.org/2003/05/soap-envelope' >
 <soap:Body>
<GMHInvoke xmlns=” urn:GMHInvoke”>
  <x:Data xmlns:x='http://GEM.org/data'>
IK44HhIvWXSX2NIeoJyjiUfI5+ynntOwSmsYyf29ks0NuVSwaHWQedq6kn/qDql6Rmnu5W2
a44HaiNSnF5B22g==
</x:Data>
 </soap:Body>
</soap:Envelope>
```

Typically, the GMH Web service would be invoked on the Receiver's end and a standardized XML document (based on the arbitrarily-defined namespace, "http://GEM.org/docs") would be "attached" using Message Transmission Optimisation Mechanism (MTOM) (W3C, 2004)[1]. In the listing above, the attached XML file occurs within the <x:Data></x:Data> elements. Elements with the namespace name "http://GEM.org/data" and a local name of "Data" will be of a type derived from xs:base64Binary (as defined in that namespace). Such elements will have an xop:Include element child in the MTOM messages and base64 text as children in the case of XML-data. After XML-binary Optimisation Packaging (XOP) (W3C, 2004[4]; W3C, 2005), the MIME part of the message will appear as follows:

```
MIME-Version: 1.0
Content-Type: Multipart/Related;boundary=MIME_boundary;
  type="application/xop+xml";
  start="<GMHInvoke.xml@GEM.org>";
  startinfo="application/soap+xml; action=\"ProcessData\""
Content-Description: A SOAP message with an XML part/attachment
--MIME_boundary
Content-Type: application/xop+xml;
  charset=UTF-8;
  type="application/soap+xml; action=\"ProcessData\""
Content-Transfer-Encoding: 8bit
Content-ID: <GMHInvoke.xml@GEM.org>
<soap:Envelope
  xmlns:soap='http://www.w3.org/2003/05/soap-envelope'
  xmlns:xmlmime='http://www.w3.org/2004/11/xmlmime'>
 <soap:Body>
      <gem:data xmlns='http://GEM.org/docTypes'
      <gem:doc xmlmime:contentType='application/soap+xml'
      <xop:Include xmlns:xop='http://www.w3.org/2004/08/xop/include'
      href='cid:http://Sender.org/enclosedFile.xml'/><gem:doc>
      </gem:data>
  </soap:Body>
</soap:Envelope>
--MIME_boundary
Content-Type: image/png'application/soap+xml'
Content-Transfer-Encoding: binary
Content-ID: < http://Sender.org/enclosedFile.xml >
// binary octets for xml attachment
--MIME_boundary--
```

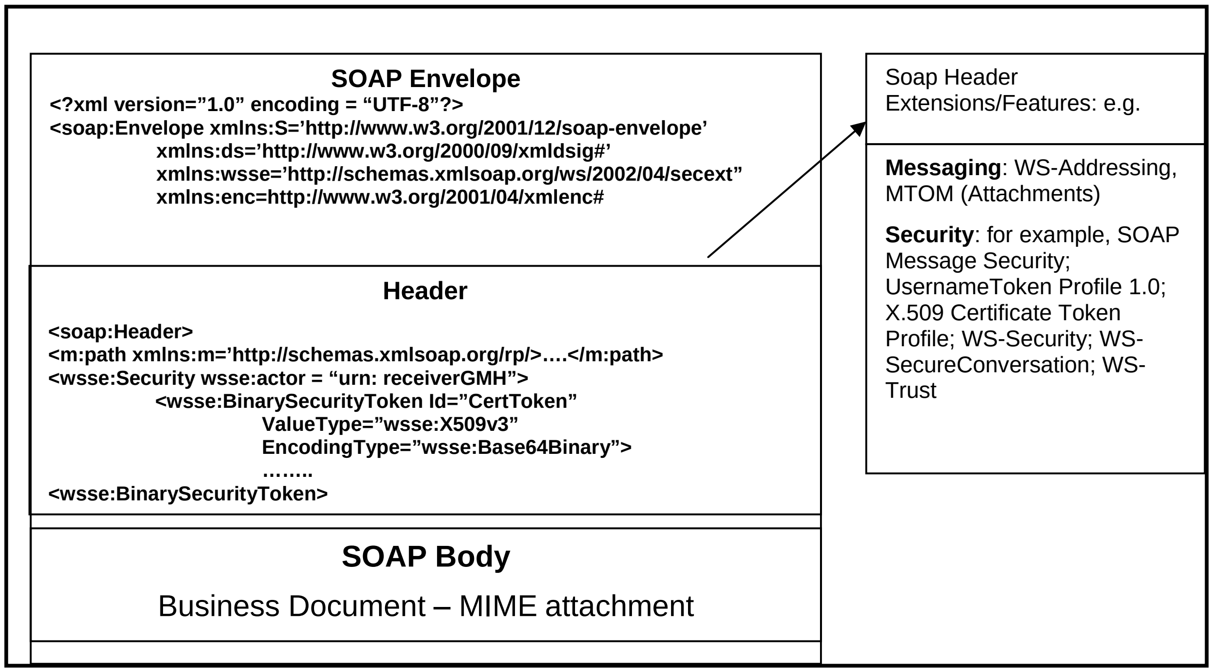
If the XML attachment were a SOAP-RPC document, the contentType would be of the format: **"application/soap+xml;action=\"http://www.ServiceLocation.net/Method\"".**

The GMH would need to provide security services, including digital signature creation and verification, encryption, authentication and authorization. A Header-Processing component would implement security services at the SOAP-envelope level. The encrypted payload (the SOAP-attachment) would be decrypted, and the digital signature verified, before being passed to the appropriate endpoint-function. The WS-ReliableMessaging feature, placed in the standardized SOAP-header, could be used as a standard means for providing reliable messaging (for the delivery and acknowledgment of SOAP Messages) in the GMH. The service would include persistence, retry, error notification and acknowledgment of messages.

2.1.1 The GEM SOAP Envelope

Decoupling business information (the SOAP-attachment) from messaging information (the external SOAP-envelope) reduces the structural complexity of the GEM. Thus, to reiterate, a standardized GEM-standardized SOAP-envelope would be used, with appropriate SOAP-headers to include features (as defined, but not included, in the SOAP version 1.2 specification) such as "reliability", "security", "correlation", "routing", and "Message Exchange Patterns" (MEPs). This is illustratred in Figure 4 below. As two major design goals for SOAP are simplicity and extensibility, the standardized SOAP specification (version 1.2) omits these features from the messaging framework. SOAP Version 1.2 provides specifics only for two MEPs. Other features are defined as extensions by other specifications, such as WS-Security and WS-Addressing.

In a GEM, SOAP-headers could be used to pass all out-of-band (not pre-negotiated or related to the semantics of the business process) information. Unlike the *Body* element of a SOAP-message, which generally includes the *in* and *out* parameters for the XML Web service method, the *Header* element is optional and can thus be processed by the message-handling infrastructure. However, by providing a standardized (GEM) infrastructure, most out-of-band message-handling requirements (*a la* ebXML Collaboration Partner Agreements) become redundant. A SOAP-header would provide the transfer protocol binding.


SOAP Envelope
Header
SOAP Body
Business Document – MIME attachment
Soap Header Extensions/Features: e.g.
Messaging: WS-Addressing, MTOM (Attachments)
Security: for example, SOAP Message Security; UsernameToken Profile 1.0; X.509 Certificate Token Profile; WS-Security; WS-SecureConversation; WS-Trust

```
<?xml version="1.0" encoding = "UTF-8"?>
<soap:Envelope xmlns:S='http://www.w3.org/2001/12/soap-envelope'
              xmlns:ds='http://www.w3.org/2000/09/xmldsig#'
              xmlns:wsse='http://schemas.xmlsoap.org/ws/2002/04/secext"
              xmlns:enc=http://www.w3.org/2001/04/xmlenc#
```

```
<soap:Header>
<m:path xmlns:m='http://schemas.xmlsoap.org/rp/>….</m:path>
<wsse:Security wsse:actor = "urn: receiverGMH">
              <wsse:BinarySecurityToken Id="CertToken"
                            ValueType="wsse:X509v3"
                            EncodingType="wsse:Base64Binary">
                            ……..
<wsse:BinarySecurityToken>
```

Figure 4. The SOAP-envelope header can be extended with standard features

The functionality of the SOAP-Header can be extended with other XML-Infosets (W3C, 2004[3]), such as those defined for the WS-* (Web Services standard) features.

2.2 GEM Registry/Repository

Similar to the ebXML Registry/Repository, the GEM Registry/Repository would provide UDDI services for GEM-standardized generic specifications, XSD's, document templates, and core components (object libraries). Users would be required to register for services. The GEM Registry/Repository, the contents and the corresponding services, would constitute the *business information processing infrastructure of the proposed framework.*

The Registry/Repository would need to be secured from denial-of-service and integrity attacks. Only members authorized by the standards body (W3C) would be able to add or edit the contents of the GEM. As with all open standards, copyright would exist, but free dissemination of documents would be allowed.

In practice, a hierarchy of Registry servers would probably exist, allowing for systematic "inheritance" of generic standardized documents, generic business processes, generic common components, and so forth. The hierarchy would mimic the refined requirements – from the more generic to the more specific – in business patterns (and related documents, businesses processes and so forth), from cross-industry (m:n) collaboration to monopolistic dyadic collaboration. The authority for managing each level of the hierarchy would logically devolve to major standards bodies at each level, provided that original standardisation is maintained (similar to PKI); the most granular-defined business activities/documents should still be interoperable with the most generally-defined ones.

In subsequent paragraphs, user-groups occupying a particular level are referred to as a "*user-community*".

3 A SECURITY MODEL FOR THE GEM

In general, a Web service typically works via an RPC between a SOAP-based Service client and a SOAP-based Service, on the consumer and provider sides, respectively. The calling application makes an HTTP request containing a SOAP-based Service URI. If the SOAP-based Service is for use by a closed user group (CUG) or bears security differentials defined by the identity or role of the principal, then the principal (application or user) normally needs to first be authenticated and then be granted the allowed permission-set (granted authorization). This authentication-authorization mechanism could be application-level verification of identity and permissions (code-based) or operating system-level verification of identity and permissions (role-based). The final run-time permission-set of the principal is resolved from the permissions granted to the principal by the application (sand-boxing) and the permissions granted by the environment in which the application is executed.

Under typical circumstances, the identity credentials of the principal could be obtained during the HTTP-Request (using request variables such as the logged-in user identity or the IP-address of the calling principle) or by explicitly demanding authentication from the user role-based or from the calling application (evidence-based). If the user is interacting with the provider-service via a Web browser, data can be input as required. However, if the interaction is to be automated, the consumer-application needs to be able to interact with the provider-application in a pre-defined manner.

It is suggested here that a standardised mechanism for (a) separating document payloads from the actual soap envelope (b) securing the documents uniformly (despite additional intra-document security measures) and (c) securing transport of SOAP-messages between applications, are essential in a GEM context. GMH applications would service the SOAP-messages moving between consumer- and provider-applications and thus would have to be congruent in respect of protocols used; the protocols for transport, messaging (packaging and encapsulation of data) and security on both sides would have to be interoperable. Conversely, protocol-neutrality would stultify the effectiveness of GMHs in a global context; a standardized GMH security model must be used uniformly throughout.

In addition, the general Client-Server security paradigm – that is, securing the client, the server and the communication line between them – needs to be considered for general specification. As Web servers in a GEM-context would generally be made available in a demilitarized zone (DMZ), general security measures apply. It is essential that the general measures do not conflict with GEM interoperability. Thus, firewalls and protocol filters must not be set to reject SOAP attachments, for instance.

The security model proposed here recommends that WS-SecureConversation (O' Neill, 2003) be used for communication between client and server for HTTP connections. This will ensure that a Server-determined encrypted tunnel ensues between client and server. To ensure identification and authentication of the sender, a GEM-standardized token (for example, X509 certificates) must be decided upon by the standards body. This will be reflected in the standard WS-Services security elements used in the standardized SOAP-envelope header.

In the minimalist approach, standardized roles could be specified for the set of business processes. Security assertions could be enforced using Security Assertion Markup Language (SAML), which could also be specified in the SOAP header. This could then be used for authorization (access-control) to specific endpoint-functions. Membership of roles is determined by the Provider, with members being added as per Collaboration Partner Agreement. Each Provider is therefore responsible for maintaining its own directory services for pre-negotiated partnering arrangements.

Secrecy and integrity can be implemented using XML-Encryption and XML-Digital Signatures, respectively. A GEM-standardized encryption algorithm – Rijndael (NIST, 2001) is recommended; the WS-Security EncryptedKey element could be used for encrypting a key with the receiver's public key – and a GEM-standardized encryption mechanism must be employed uniformly. First, a GEM-standardized hash-algorithm could be used to create a hash of the XML-attachment before XOP packaging. This is to ensure integrity of the attachment. The hash could then be encrypted using the sending application's private key. (This would be used to authenticate the sender and to ensure privacy of the hash). The attachment and the hash could then be converted to Binary64 and added to a MIME package with MTOM.

Public keys could be distributed by XKMS (XML Key Management System). Once the XOP:Include element for each attachment has been added, a hash value could then be created of the entire SOAP-body, using the GEM-standardized hash-algorithm, to ensure that no further additions are made to the envelope. The SOAP-body (including the encrypted attachment) and the hash value could then be encrypted with the receiver's public key. (This would ensure confidentiality between the sender and the receiver; only the receiving application can use its private key for decryption).

Upon receipt of the SOAP-message, the receiver would have to be authenticated by its security token (as per the SOAP-header specification). Thereafter, it would be able to decrypt the body of the SOAP-message using its private key (again, as per the SOAP-header specification). It would then create a hash from the entire SOAP-body, using the GEM-standardized hash algorithm. If the two hash values are the same, the integrity of the SOAP-body is deemed to have been preserved. The SOAP-attachment would then be decrypted and a hash created, using the GEM-standardized hash-algorithm. If the two hash values are the same, the integrity of the attachment is deemed to have been preserved. The GUID in the SOAP-body would be used to determine which endpoint-function to pass the SOAP-attachment to for processing. Standardized error messages, as per WS-Security and SOAP 1.2, will be generated as appropriate.

The formalisation of specific standards to be used in the WS-Security specification could be described using WS-Policy.

The implementation of services and the related security features is application-specific, but following standard guidelines is recommended. The following are typical examples. To prevent luring attacks, security assertions should be checked at the method-level. To prevent code-injection attacks, all input should be validated comprehensively. Errors should not be allowed to disclose information that might be usable by an attacker; provide appropriate custom errors. Spoofing and masquerading attacks could be eliminated through the use of digital certificates, provided that private keys are kept secret and rigorous authentication is applied.

4 CONCLUSION

The notion of a GEM is a hypothetical construct. However, similar to many less-conceivable frameworks (such as alleviating poverty world-wide or achieving global peace), one often arrives at theoretical antecedents for achieving the desired ideal (or near-ideal). This, in itself, often provides guidelines for smaller-scoped endeavours. Thus, one might, quite conceivably, apply the framework discussed in this paper within a smaller context, within a particular collaboration focus area.

5 REFERENCES

Apshankar K. (2002, July 24). WS-Security: Security for Web Services. Retrieved February 18, 2003 from http://www.webservicesarchitect.com/content/articles/apshankar04.asp

Chatterjee S and Webber J. (2004). Developing Enterprise Web Services - An Architect's Guide. Prentice-Hall PTR.

Kay R. (2003), XACML, Retrieved August 12, 2003 from http://www.computerworld.com/securitytopics/security/story/0,10801,81295,00.html, 19 May, 2003)

Loeb, L (2002). Donald Eastlake on XML digital signatures. Retrieved September 7, 2003 from http://www-106.ibm.com/developerworks/xml/library/s-east.html?dwzone=xml

O' Neill et al. (2003). Web Services Security. McGraw-Hill/Osborne.

NIST (2001). AES Home Page. Available [online] at http://www.nist.gov/aes/. Accessed: 31/08/01

Rosencrance, L. (2002). SAML Secures Web Services. Retrieved July 20, 2003, from http://www.computerworld.com/securitytopics/security/story/0, 10801,73712,00.html

RosettaNet Consortium. (2002). RosettaNet Implementation Framework: Core Specification, Standards Specification Version: V02.00.01, RosettaNet.

Siddiqui, B. (2002). Exploring XML Encryption. Retrieved September 7, 2003 from http://www-106.ibm.com/developerworks/xml/library/x-encrypt/

W3C (2000). SOAP (Simple Object Access Protocol) 1.1 Specification. Retrieved May 15, 2003 from http://www.w3.org/TR/2000/NOTE-SOAP-20000508

W3C (2002) [1]. SOAP Version 1.2 Email Binding. W3C Note 26 June 2002; Retrieved July 2002 from http://www.w3.org/TR/2002/NOTE-soap12-email-20020626

W3C (2002) [2]. SOAP 1.2 Attachment Feature. W3C Working Draft 24 September 2002 http://www.w3.org/TR/2002/WD-soap12-af-20020924

W3C (2003)[1]. SOAP ver 1.2 Part 0: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

W3C (2003)[2]. SOAP ver 1.2 Part 1: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

W3C (2003)[3.] SOAP ver 1.2 Part 2: Primer. Retrieved December 15, 2003 from http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

W3C (2004)[1]. MTOM (SOAP Message Transmission Optimization Mechanism). Retrieved August 15, 2004, from http://www.w3.org/TR/2004/WD-soap12-mtom-20040608/

W3C (2004)[2]. Web Services Architecture. Working Group Note 11 February 2004. Retrieved 20 June 2004 from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

W3C (2004)[3]. XML Information Set. W3C Recommendation, 4 February 2004. Retrieved June 2, 2004 from http://www.w3.org/TR/2004/REC-xml-infoset-20040204

W3C (2004)[4]. XML-binary Optimized Packaging. W3C Candidate Recommendation $Date: 2004/09/15 12:33:28 $ Retrieved December 17, 2004 from http://www.w3.org/TR/xop10/

W3C. (1997). HTTP (Hypertext Transfer Protocol) 1.1. Retrieved August 5, 2003.

W3C (2004)[5]. Extensible Markup Language (XML) 1.1. W3C Recommendation 04 February 2004, edited in place 15 April 2004. Retrieved: 30 October 2004 from http://www.w3.org/TR/2004/REC-xml11-20040204/ Editors: Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan

W3C (2004)[6]. Web Services Choreography Requirements. W3C Working Draft, 11 March 2004. Editors: Austin, D., Barbir, A., Peters, W. and Ross-Talbot, S. Retrieved 25 October, 2004 from http://www.w3c.org/TR/2004/WD-ws-chor-reqs-20040311/

W3C (2004)[7]. XML Information Set (2004). W3C Recommendation, 4 February 2004. Retrieved June 2, 2004 from http://www.w3.org/TR/2004/REC-xml-infoset-20040204

W3C (2005). XML-binary Optimized Packaging Recommendation. 25 January 2005. Retrieved from http://www.w3.org/TR/2005/REC-xop10-20050125/ 30 January 2005.

The financial assistance of National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the authors and are not necessarily to be attributed to the National Research Foundation.