6-2009

Applying Sanitizable Signature to Web-Service-Enabled Business Processes: Going Beyond Integrity Protection

Kar Way TAN
Singapore Management University, karway.tan.2007@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Applying Sanitizable Signature to Web-Service-Enabled Business Processes: Going Beyond Integrity Protection

Kar Way Tan
School of Information Systems
Singapore Management University
80 Stamford Road Singapore 178902
karway.tan.2007@smu.edu.sg

Robert H. Deng
School of Information Systems
Singapore Management University
80 Stamford Road Singapore 178902
robertdeng@smu.edu.sg

Abstract

This paper studies the scenario where data in business documents is aggregated by different entities via the use of web services in streamlined business processes. The documents are transported within the Simple Object Access Protocol (SOAP) messages and travel through multiple intermediary entities, each potentially makes changes to the data in the documents. The WS-Security provides integrity protection by allowing portions of a SOAP message to be signed using eXtensible Markup Language (XML) signature scheme. This method however, has not considered the situation where a portion of data may be modified by another entity, therefore a need to allow the originating system to control which intermediary entity is authorized to change which portion of the data. The XML signature scheme also does not provide the final recipient the trust for the intermediary entity that makes the changes. In our paper, we study the security requirements for a streamlined business process, and proposes a novel scheme using sanitizable signature on SOAP messages to complement the XML signature to address not only integrity protection but also control of change as well as establishment of trust for intermediary entities. We show how the proposed scheme can be incorporated into the existing standards and be customizable to achieve flexible use of both the vanilla and sanitizable signatures as required in a business scenario. With the proposed technique, IT systems can be more loosely coupled and reap the benefits of distributed systems, such as delegation of work and encapsulation of business logic.

1. Introduction

System integration has been a challenge since organizations started deploying loosely-coupled distributed systems that are adaptable and capable of handling complexities. The challenges include the need to support cross-platform and cross-programming-language interactions. Such challenges are in part addressed by web services, a cross-platform, cross-programming-language technology that allows a set of operations to be exposed over the Internet using standard communication protocols (e.g. SOAP [11] over HTTP) and data representations (e.g. XML). In general, web services are used for point-to-point integrations, and for enabling business process automation that may have information flows spanning across organizations. We term the latter as web-service-enabled business process in this paper.

An automated web-service-enabled business process is able to reap the benefits of decoupled systems by having work delegation. Each application within the business process is entrusted to perform a particular set of functions without having to interact repeatedly with a centralized controller. In such a process, information in a business document may be encapsulated in a SOAP envelope, and traverse to multiple systems before reaching the final recipient. For example, a business document such as purchase order may traverse from an ordering system to fulfillment system and to billing system, each may need to modify or add data to the purchase order. In practice, such a business process is deployable only if the message can be trusted, proven free from unauthorized changes, authenticated, changes be controlled and accounted for by the respective entities.

To address message integrity, WS-Security [7] defines how to attach signatures to SOAP messages using XML signature [3]. XML signatures provide the ability to selectively sign the pieces of XML data that each entity may update in a business process. Although the WS-Security signature scheme is able to ensure integrity of the data, authenticate the message originator and make intermediaries accountable for the changes, it has overlooked a few important issues. Firstly, it is not able to handle data that requires changes by another entity. Secondly, it does not allow the originator of the business document to control which portion of the data that can be modified by which intermediary. Thirdly, it does not establish trust of the intermediary for the final recipient without using communication-expensive trust assertions. Finally, it is not clear to the intermediary, which portion of the data it is allowed to change without having extensive encryption or access control configurations within the document.

Although some of the issues mentioned above, such as

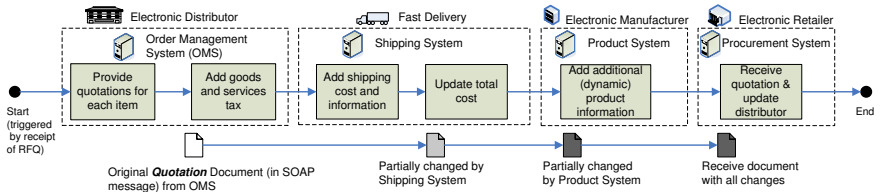

Figure 1. *Quotation Response* Process: Running example of a web-service-enabled business process

trust, may be partially addressed with extensive use of WS-*[1] policy-based configurations, we are motivated to find a succinct solution to all the issues above by seamlessly incorporating sanitizable signature [1] in SOAP messages. We illustrate the problem by first applying the XML signature to an example web-service-enabled business process. Through the example, we elicit the capabilities and limitations of the XML signature scheme. We then propose and apply a scheme that incorporates sanitizable signature to the example business process. In our analysis for the proposed scheme, we demonstrate that the new scheme not only provides data integrity, authentication and accountability as with the XML signature, it can also overcome the limitations of the XML signatures by achieving control of source and extent of information change and establishment of trust for intermediary. Finally, we present a preliminary implementation design for integrating sanitizable signature with the existing WS-Security standards.

In the related works, several efforts [5][6][8][9] focus on issues of unauthorized modifications of SOAP messages by intermediaries through XML rewriting. Their proposals use either policy-based configurations or the concept of SOAP account to detect these modifications. In contrast, we focus on authorized modifications to SOAP messages. The work in [12] uses extensive policy-based configurations to control and authorize changes in an XML document. This approach requires the entities to understand the specific proposed policy language. In our work, we focus on using a concise method of using sanitizable signature scheme for control and authorization and adhere closely to standards governed in [3][7]. Another application of sanitizable signature is proposed for end-to-end authentication in intermediary-enabled multimedia delivery systems [2], where intermediary proxies are allowed to make modifications such as transcoding of the multimedia artifacts, yet the validity of the originator's signature is preserved. Our work here differs from that of [2] in several important aspects. We propose a model and study the implications of applying both the XML signature and the sanitizable signature, respectively, to SOAP messages in web-service-based streamlined business processes. We give detailed design specifications on how to incorporate both signature schemes seamlessly in WS-Security standards in order to satisfy security needs as per required by different business processes. We illustrate, using an example business process, the need for such a hybrid scheme.

1. WS-* is used to denote other web service security standards such as WS-Policy and WS-Trust.

2. The Application Scenario

We illustrate the problem and application scenario through an example business process involving distribution of electronic appliances such as television sets. This shall be used as a running example throughout the paper.

2.1. Quotation Response Process: A Web-Service-Enabled Business Process

Consider a company called *Electronic Distributor* that distributes electronic products from a manufacturer *Electronic Manufacturer* to a large number of retailers. *Electronic Distributor* manages its own warehouse operations but has outsourced delivery services to a transportation company *Fast Delivery*. *Electronic Distributor* and *Fast Delivery* have a pre-negotiated agreement on the delivery charges for its retailers and *Fast Delivery* is trusted to provide accurate delivery charges for the retailer's purchases based on the ship-to address. The *Electronic Manufacturer* is responsible to provide additional information about the products based on the retailer's information, e.g., regulatory test results or warranty terms that could vary from country to country, depending on the location of the retailer.

With reference to Figure 1, we consider a simple business process – *Quotation Response* Process, involving *Electronic Distributor, Fast Delivery and Electronic Manufacturer*. We assume that a business document *Quotation* (in XML format as shown in Figure 2) is transferred between various entities with use of document-styled web services.

The *Quotation Response* process begins when *Electronic Distributor* receives a *Request for Quotation (RFQ)* from a retailer, *Electronic Retailer*. *Electronic Distributor* generates the *Quotation* using the *Order Management System (OMS)* by providing quotes for each item and the taxes associated. Thereafter, the *OMS* forwards the *Quotation* document to the *Fast Delivery's Shipping System* via a SOAP message. Upon receipt of document, the *Shipping System* inputs the delivery cost, delivery information and updates the total cost. After which, it forwards the document to the *Electronic Manufacturer's Product System*. Similarly, the *Product System* inputs the additional product information based on the retailer's information and finally forwards the *Quotation* to *Electronic*

Structure of Quotation Document

```
01. <Quotation>
02.    <Customer>
03.       ...customer information...
04.    </Customer>
05.    <Items>
06.      (<Item>...item information...</Item>)+
07.    </Items>
08.    <Shipping>
09.       ...shipping information & cost...
10.    </Shipping>
11.    <TotalCost>
12.      (<Tax>...tax information...</Tax>)*
13.       <Total>...total cost...</Total>
14.    </TotalCost>
15.    <PdtInfo>
16.       ...Addition Product information...
17.    </PdtInfo>
18. </Quotation>
```

Figure 2. Structure of the *Quotation* document

Retailer. Upon receipt of the document, *Electronic Retailer* updates *Electronic Distributor* to inform the latter that the quotation has been received. Table 1 shows the portion of the data within the *Quotation* that are updated or inserted by each system.

XML Element	Updated by
Customer/Items/Tax	OMS
Shipping	Shipping System
Total	OMS & Shipping System
PdtInfo	Product System

Table 1. XML elements & updating systems

2.2. Security Requirement Analysis for the Process

As seen in the process above, a business document *Quotation* is transferred from system to system crossing organizational network boundaries via SOAP. As with any business data, such a document raises basic security concerns such as revelations and integrity of data. Focusing only on integrity-related aspects in this paper, we list in the following common security requirements for such a business process. We use the term *entity* to refer to a *system* or a party in the process.

We begin the analysis from the perspective of *Electronic Distributor*. The security requirements contains two aspects, i) *Integrity*: The need to prevent unauthorized modifications between the originating entity (*OMS*) and the final receiving entity (*Procurement System*) as the *Quotation* passes through two intermediaries. The *OMS* must ensure that appropriate parts of the *Quotation* are protected such that the data it places in the document does not get changed while another part of the document may be updated by intermediaries; ii) *Control*: The originating entity (*OMS*) needs to control which intermediary has the right to modify a specific portion of data. For example, only the *Shipping System* has the right to insert shipping information and only the *Product System* has the right to insert additional product information. In addition, if a particular piece of data needs to be updated by the originator and another intermediary, then the originator can authorize the intermediary to perform the update, preferably without having the need to re-endorse the changes in order to minimize communications between entities and promote loosely-coupled systems.

From the viewpoint of *Electronic Retailers*, the security requirements are four-folded: i) *Integrity*: The *Procurement System* must be able to verify that there is no unauthorized changes to the data in *Quotation* before it reaches itself; ii) *Authenticity*: When *Procurement System* receives the *Quotation*, it needs to verify that the information comes from the right sources, i.e., prices are quoted by *Electronic Distributor* and product specifications are endorsed by *Electronic Manufacturer*; iii) *Accountability*: Since *Quotation* is modified along the process, it must ensure that each entity be held accountable for the changes. This is also to ensure non-repudiation; finally iv) *Trust for Intermediaries*: As outsourcing, decentralization and specialization have become prevalent in businesses today, it is inevitable to increasingly involve more business partners within a business process. In our example, *Electronic Distributor* has outsourced the delivery services to *Fast Delivery*. However, to a customer like *Electronic Retailer*, its main concern is likely to be about striking a business transaction and may not be concerned about whether the transportation services has been outsourced. Therefore, *Electronic Retailer* may not be aware of the delivery agreement between *Electronic Distributor* and *Fast Delivery* and may not trust the information (e.g., cost and shipment terms) entered by *Fast Delivery*. As such, there is a need to provide the final recipient an evidence of trust for the information changed or inserted by intermediaries.

Finally, from the viewpoint of the intermediary entities, there are two security requirements, they are i) *integrity*: The intermediary entities need to ensure that the information they added or modified is not changed by other entities and no other entity can make the changes on behalf of them; and ii) *Accountability*: The intermediary entities need to know if they are accountable for the changes.

Our objective in this paper is to find a solution that could satisfy all the requirements in a concise manner.

3. Application of Signature Schemes to Quotation Response Process

This section evaluates how the two signature schemes, the standard XML signature scheme and sanitizable signature scheme can be applied to streamlined business processes such as the *Quotation Response* process. For easy reference, we refer the existing standard XML signature in WS-Security as the *vanilla XML signature*.

To ensure consistency, we first explain some of the standard notations used in the paper. Let m be a SOAP message and s_i be the set of transformed XML element references (ref_1, ref_2, ...,ref_n) within m that are protected

using a single signature, i.e., $ref_1 \cup ref_2 \cup ... \cup ref_n = s_i$ and $s_i \subseteq m$. Since there could exist more than one signature, each protecting a set of references within the same SOAP message, the index i denotes the i^{th} signature within m. Digital signatures are performed on hashes of the data. Let H denote a secure collision-resistant hash function such as *SHA1* and let $H(ref_j) = h_j$, i. e., h_j is the hash value of the j^{th} reference ref_j in s_i. The collection of hashes(h_1, h_2,...,h_j) are then put into a single $SignedInfo$ element, SI_i, in the SOAP header. A signature σ_i is then generated over SI_i. Each entity in our model has a pair of private-public keys,(PK_{sys},SK_{sys}), where PK denotes the public key used for verification, SK the private key used for signing or modification and the subscript sys denotes the owner (entity) of the key pair.

3.1. Using Existing Vanilla XML Signature Scheme

For consistency and ease of explanation later, we provide a formal model of the vanilla XML signature scheme.

3.1.1. The Model. The vanilla XML signature scheme consists of three algorithms: key generation, signing and verification.

Signature Key Generation: The key used for signature can be of any digital signature scheme. Certification and revocation of the keys are usually performed by a Certification Authority. The key generation algorithm *KeyGen* is a probabilistic polynomial-time algorithm that takes an input security parameter 1^k. For each entity in the business process,

$$(PK_{sys},\, SK_{sys}) \longleftarrow KeyGen(1^k)$$

Sign: The signing operation involves the canonicalization [10] on each SI_i, transformation for each ref_j, hash and the signature operation. The signature may use an optional random number z. For each s_i, we form the $SignedInfo$ element, SI_i in the header by including one or more hashes, i.e. h_1, h_2,..., h_n. The signature is generated by

$$\sigma_i \longleftarrow \text{Sign}(SK_{sys},\, [z],\, SI_i)$$

Verify: The verification algorithm is a deterministic algorithm involving canonicalization on each SI_i, transformation for each ref_j, hashing and verification of each references and usual signature verification operation over SI_i. Given σ_i, SI_i, h_j, and the newly computed hash h'_j of ref_j, for j = 1, 2, ..., n,

$$(1, 0) \longleftarrow \text{VerifyHash}(h_j, h'_j)$$
$$(1, 0) \longleftarrow \text{Verify}(PK_{sys},\, [z],\, \sigma_i,\, SI_i)$$

3.1.2. Application of Vanilla XML Signature to Quotation Response Process. The application of vanilla XML signature is straightforward. Based on the *Quotation* document shown in Figure 2, a total of three signatures are generated. The first signature is generated and signed by the originating system *OMS* on elements <*Customer*>, <*Items*> and <*Tax*>. The second signature is generated and signed by the *Shipping System* on the element <*Shipping*>. Recall that once an element is signed, changes made to it by any intermediary would invalidate the signature. Therefore, the signatures must be scoped such that each entity signs only the portion where it is responsible and leaves the modifiable parts open. With this restriction, we note that the <*Total*>, an element that needs to be updated by both the *OMS* and the *Shipping system*, cannot be handled easily. One possible way to overcome the problem is having the *Shipping System* signs on behalf of *OMS*. Another possibility is to leave the element unsigned but to ensure that the field can be re-computed by the final recipient. For the purpose of comparison with our proposed scheme, we shall adopt the former approach. Finally, the third signature is generated by the *Product System* on element <*PdtInfo*>. In Figure 3, the portion labeled with "using vanilla XML signature" provides the pictorial view of the application of vanilla XML signature to *Quotation Response* process.

3.1.3. Evaluation of the Scheme. We evaluate the scheme according to the security requirements listed in Section 2.2.

From the viewpoint of the originating entity, i.e., *Electronic Distributor*, the *integrity* is partially achieved. It is achieved if the data is solely modifiable by a single entity. However, if there are data items such as <*Total*>, that are modifiable by another entity, then there is no assurance that the data is not changed before the intermediary makes the modification. This signature scheme does not allow the message originator to *control* which intermediary has the ability to change which portion of the SOAP message. Although control may be made possible by combination of other WS-* specifications and encryption, they introduce additional complexity and overheads into the SOAP message, above and beyond the overhead introduced by XML signature.

From the viewpoint of the final recipient, *Electronic Retailer*, *integrity* is also only partially achievable since some data can be modified by another entity. As for *authenticity*, since all modifications by intermediary entities are signed, final recipient can authenticate each part of the data as identified by the signer of each signature. Each signer is also *accountable* for the respective changes. However, there is no single entity whom can be held accountable for the entire document. As such, the final recipient *Electronic Retailer* needs to establish separately, the *trust relationships* with all the entities in order to verify integrity and authenticity of the document. This in practice can be a complicated task to establish the business relationships between the originator and all its business partners.

From the viewpoint of the intermediary entities, *Fast Delivery* and *Electronic Manufacturer*, similarly, the *integrity* is partially addressed. If the information is signed, integrity can be preserved. However, for parts of information that have yet to have a signature (such as the <*Total*> element), the

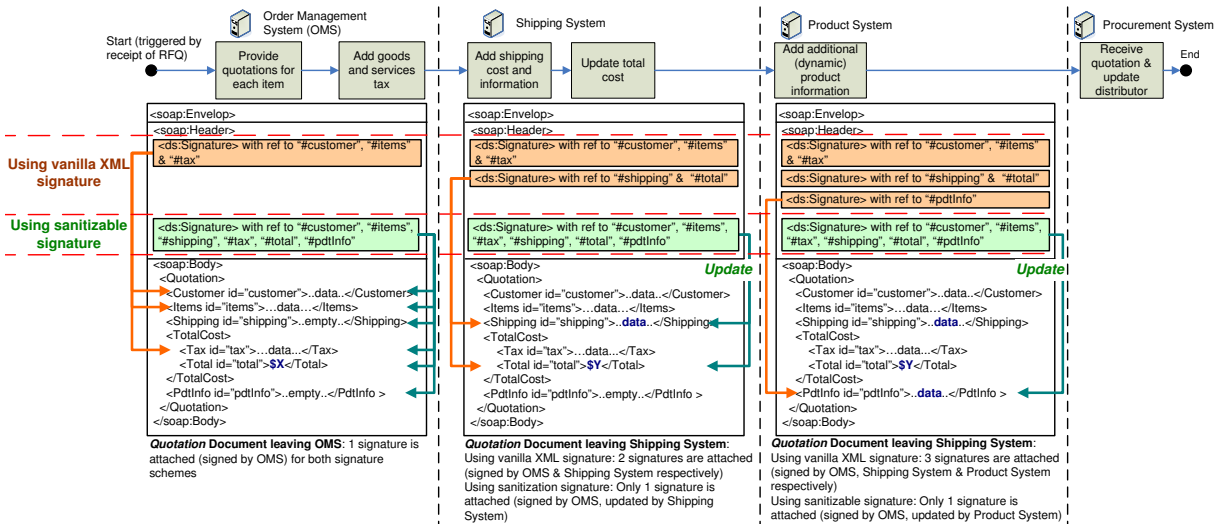

Figure 3. Application of both signature schemes to *Quotation Response* process

intermediary entities cannot verify that they are not changed. Since all changes are signed, the intermediary entities are *accountable* for the changes.

With *Quotation Response* process, we can see that there are some limitations in using the vanilla XML signature.

3.2. Using Sanitizable Signature

To overcome the limitations of the vanilla XML signature in meeting the security requirements for our example business process, we propose a new approach which employs both sanitizable signature [1] and the vanilla XML signature.

3.2.1. Sanitizable Signature Overview. A sanitizable signature allows authorized intermediaries to modify parts of a signed message without interacting with the original signer, and without invalidating the original signature. Its main benefit is that modifications are enabled in a "limited and controlled fashion"[1], and allowing recipient to verify only a single signature with the modifications. In the context of the *Quotation Response* process, it allows an intermediary such as *Shipping System* to add shipping cost to the *Quotation* while keeping the original signature by *Electronic Distributor* valid. The basic notion of this scheme makes use of a *trapdoor* hash function that is first applied to the message and then the resulting hash value is signed by the digital signature algorithm such as *RSA*. Thereafter, an intermediary, who knows the secret key to the trapdoor, is able to find a collision to the hash function such that amendment can be made to the message and yet producing the same digest value, keeping the signature intact. The basic operations of sanitizable signature scheme [1] is as follows:

The private-public key pair for an intermediary: A pair of public-private keys is required for an intermediary. An example of the key pair is the following. Let p and q be prime numbers such that $p = 2q + 1$. Obtain g, a generator of the subgroup of squares of order q. The private key x is selected at random in $[1, q-1]$. The public and private key pair is defined as $(y = g^x, x)$.

The trapdoor hash function: The hash function is based on the chameleon hash [4] on (ref_j, r), where r is a random number pair, $r = (\rho, \delta) \in Z_q \times Z_q$ and ref_j is the j^{th} reference within the XML message m that will be protected by a signature σ_i. Let H be a regular collision-resistant hash function such as *SHA1*. The *trapdoor* hash function is in the form $TH_y(ref_j, \rho, \delta) = \rho - (y^e g^\delta \mod p) \mod q$ where $e = H(ref_j, \rho)$. Note that the trapdoor is created using the public key y.

Collision computation: Given a hash value $\nu = TH_y(ref_j, \rho, \delta)$, the intermediary with the private key x can compute a collision for the modified ${ref_j}'$ as follows. First choose a random value $k' \in [1, q-1]$. Compute three values, (1) $\rho' = \nu + (g^{k'} \mod p) \mod q$, (2) $e' = H({ref_j}', \rho')$ and (3) $\delta' = k' - e'x \mod q$. Putting the newly computed ρ' and e' into the *trapdoor* hash function, we have $\nu' = \nu + (g^{k'} \mod p) \mod q - (g^{xe'} g^{\delta'} \mod p) \mod q$. Substituting value of δ' as in (3), we get $\nu' = \nu + (g^{k'} \mod p) \mod q - (g^{xe'+k'-e'x} \mod p) \mod q = \nu$. Therefore, $({ref_j}', \rho', \delta')$ is the collision of the chameleon hash function that would result in the same signature for the referenced element within the XML message.

3.2.2. General Approach for Using Sanitizable Signature. The general approach to apply sanitizable signature to SOAP message is: i) the portions of data that do not require modification by intermediary entity uses the usual hash function; ii) the portion of data that requires changes by an intermediary uses a *trapdoor* hash function; iii) the signature generation remains the same, i.e., generated over the entire *<SignedInfo> element* which may contain one or

more hash values.

3.2.3. The Model. The use of sanitizable signature consists of four algorithms: key generation, signing, modification and verification.

Signature Key Generation: Similar to vanilla XML signature, key generation uses a probabilistic polynomial-time algorithm. A pair of private-public keys is given to each entity. Let (PK_{sys0}, SK_{sys0}) denote the key pair for the originator and (PK_{int}, SK_{int}) the key pair for an intermediary. Note that the private key and public key for the originator is for standard signature generation and verification respectively, while the public key and private key for the intermediary is for computing a trapdoor hash function and for collision computation respectively. Without loss of generality, assuming that there is only one intermediary, we have

$$(PK_{sys0}, SK_{sys0}, PK_{int}, SK_{int}) \longleftarrow KeyGen(1^k)$$

Sign: The signing operation involves hash on each reference section and the signature operation. For each reference ref_j, the hash value h_j is produced either using H or TH depending on whether it is modifiable. If TH is used, random coins r_j and the public key of the intermediary PK_{int} are used in generating the hash value with the *trapdoor*. The signature σ_i, is then generated using the private key of the originating system SK_{sys0} over the SI_i, where SI_i contains a collection of hashes h_1,...,h_n. Let z be an optional random number for signature generation. Then

$$h_j \longleftarrow \text{either } TH(PK_{int}, r_j, ref_j) \text{ or } H(ref_j)$$
$$\sigma_i \longleftarrow \text{Sign}(SK_{sys0}, [z], SI_i)$$

Modify: The intermediary can make modification to the specified reference ref_j authorized by the originator. The intermediary uses its private key SK_{int} to recompute new random coins $r_j{}'$ required to generate a collision for the *trapdoor* hash h_j. For each modified reference element ref'_j,

$$h_j \longleftarrow TH(SK_{int}, r_j{}', ref_j{}')$$
$$SI_i \text{ and } \sigma_i \text{ remains the same}$$

Verify: The verification operation involves the hash and the signature verification operation. For each referenced section ref_j, the hash value h_j is reproduced either using H or TH depending on whether it is modifiable. If TH is used, the new random coins $r_j{}'$ and the corresponding public key of the intermediary PK_{int} are used to regenerate the hash. The signature σ_i, is then verified using the public key of the originator SK_{sys0} over the collection of hashes h_j.

$$h'_j \longleftarrow \text{either } TH(PK_{int}, r_j{}', ref_j{}') \text{ or } H(ref'_j)$$
$$(1,0) \longleftarrow \text{VerifyHash}(h_j, h'_j)$$
$$(1,0) \longleftarrow \text{Verify}(PK_{sys0}, [z], \sigma_i, SI_i)$$

3.2.4. Application of Sanitizable Signature to Quotation Response Process. Consider the same *Quotation Response* process as shown in Figure 1, we show that a single signature is sufficient to sign all the necessary data with use of sanitizable signature. The originating entity can now control which reference element in the XML document contains a *trapdoor* such that it can be modified by an intermediary.

In our business process example, the *OMS* creates a sanitizable signature σ_1 with (1) regular hash function H over the elements <*Customer*>, <*Items*> and <*Tax*>, (2) a trapdoor hash function over elements <*Shipping*> and <*Total*> using $PK_{shipping}$ and (3) a second trapdoor hash function over element <*PdtInfo*> using $PK_{product}$. The signature is signed by OMS using SK_{oms}. With this signature, *Shipping System* can modify elements <*Shipping*> and <*Total*> using $SK_{shipping}$ and *Product System* can modify <*PdtInfo*> using $SK_{product}$. In Figure 3, the portion labeled with "using sanitizable signature" provides the pictorial view of the application of sanitizable signature to *Quotation Response* process.

3.2.5. Evaluation of the Scheme. Similarly, we evaluate the new scheme according to the security requirements defined in Section 2.2. From the viewpoint of the originator *Electronic Distributor*, the data *integrity* protection can now be extended to data that are modifiable by another entity, hence satisfying the integrity requirement. This scheme allows the originating entity to *control* which intermediary has the ability to change a specified portion of the SOAP message. In our example, only the *Shipping System* (belonging to *Fast Delivery*) can modify data in <*Shipping*> and <*Total*>, and only the *Product System* (belonging to *Electronic Manufacturer*) can modify data in <*PdtInfo*>. This is enforced because only the systems with the corresponding private key can recompute the *trapdoor* hash function without invalidating the original signature. In addition, there is an added advantage of providing the flexibility to the originating entity to decide which business partner to work with at the start of the business process. For example, in our scenario, *Electronic Distributor* may work with different transportation companies either based on the location of the retailer or based on load-balancing distribution of work. In such situation, the *Electronic Distributor* could then control to which transportation company it assigns the task by creating the *trapdoor* hash using the public key of the assigned company.

From the viewpoint of the final recipient *Electronic Retailer*, the *integrity* protection is achieved as data that is modifiable by both originating and intermediary entities can now be handled. As a result of the use of sanitizable signature, the *Electronic Retailer* receives only a single signature signed by the originator *Electronic Distributor*, hence *authenticating* the business document as a whole as though the information comes entirely from the originator. In addition, our proposed scheme allows orginator to flexibly decide which portion of the data to be authenticated by which entity. For example, since the additional product information is updated only by the *Electronic Manufacturer's Product*

System, and it is a type of information that the end recipient could trust (i.e., it is commonly required that regulatory information or warranty information come directly from the manufacturer), the product information can be signed by the *Electronic Manufacturer* using the vanilla XML signature. Each change to the document is also *accounted* for as only the entity that is authorized to make the changes can successfully recompute the random coins required by the *trapdoor* hash function. We will show in Section 4 that the retailer can verify that the random coins are changed and use the new coins to verify the hashes. If *trapdoor* hashes are verified, we can deduce that modifications is performed by the intermediary entity holding the secret key. Finally, the *trust for intermediaries* can be established using the proposed scheme. Although the intermediaries are not entirely transparent to the recipient as it still requires the intermediaries' public keys to recompute the hashes, the recipient is aware that only intermediaries specified by the originating entity can successfully recompute the random coins for the *trapdoor* hash that matches the modified data. Therefore, they are authorized and trusted intermediaries.

From the viewpoint of the intermediary entities, *Fast Delivery* and *Electronic Manufacturer*, the *integrity* is satisfied since information such as <*Total*> element can be first signed by originator and be modified by an authorized intermediary. As per the discussion for final recipient, we will illustrate in Section 4 that intermediary entities are *accountable* for the changes. Therefore, we can see that all the security requirements can be satisfied by combined use of sanitizable signature and vanilla XML signature. In the next section, we will show how we can allow co-existence of both schemes.

4. Implementation Design

We propose in the following, a preliminary design (with references to specification of XML Signature [3]) to incorporate the use of sanitizable signature in actual implementations. The main difference between the vanilla XML signature and sanitizable signature is the hash function algorithms, i.e., the *DigestMethod* for each reference. The *trapdoor* hash function used in sanitizable signature requires additional information such as *KeyInfo* (of the intermediary who is allowed to modify the reference section), the original random coins and the new random coins computed by the authorized intermediary. Figure 4 shows the schema for the standard *DigestMethod* as defined in WS-Security. The provision of the <*any*> element at Line 04 enables us to extend the XML document with elements not specified by the schema. Therefore, we propose to place the additional information required by the *trapdoor* hash function as part of child element within *DigestMethod*. As the *DigestMethod* lies within the *SignedInfo* element, we also need to make provision to allow new random coins to be placed by intermediaries without invalidating the signature. The *Signature* element has a provision to include <*Object*> element (schema as shown in Figure 5) as an extension to include more information for the signature. We use this extension to place the new random coins required to verify the *trapdoor* hash function. In short, we propose to place (1) the *KeyInfo* of the intermediary, original random coins and a reference to the new random coins within the *DigestMethod* and (2) the new random coins in the extension *Object* element.

Schema of Digest Method

```
01. <complexType name="DigestMethodType"
02.  mixed="true">
03.   <sequence>
04.     <any namespace="##other" processContents="lax"
05.      minOccurs="0" maxOccurs="unbounded"/>
06.   </sequence>
07.   <attribute name="Algorithm" type="anyURI"
08.    use="required"/>
09. </complexType>
```

Figure 4. Schema of *DigestMethod* in *SignedInfo*

Schema of Object Element

```
01. <element name="Object" type="ds:ObjectType"/>
02. <complexType name="ObjectType" mixed="true">
03.   <sequence minOccurs="0" maxOccurs="unbounded">
04.     <any namespace="##any" processContents="lax"/>
05.   </sequence>
06.   <attribute name="Id" type="ID" use="optional"/>
07.   <attribute name="MimeType" type="string" use="optional"/>
08.   <attribute name="Encoding" type="anyURI" use="optional"/>
09. </complexType>
```

Figure 5. Schema of *Object* extension

A Sample Implementation of Sanitizable Signature

```
01.<Signature>
02. <SignedInfo>
03.   <CanonicalizationMethod Algorithm="http://
        www.w3.org/TR/2000/CR-xml-c14n-20001026"/>
04.   <SignatureMethod Algorithm="...#rsa-sha1"/>
05.   <Reference URI="#Ref_1">
06.     <Transforms/>
07.     <DigestMethod Algorithm="...URI to trapdoor
          hash specifications...">
08.       <Random-rho>384474..</Random-rho>
09.       <Random-delta>98327..</Random-delta>
10.       <Digest-KeyInfo>
11.         ...Key Info of Intermediary...
12.       </Digest-KeyInfo>
13.       <New-Random-Ref>#Ref_new</New-Random-Ref>
14.     </DigestMethod>
15.     <DigestValue>j6x3rvEPO0vK...</DigestValue>
16.   </Reference>
17. </SignedInfo>
18. <Object>
19.   <New-Random Id="Ref_new">
20.     <New-Random-rho>384474..</New-Random-rho>
21.     <New-Random-delta>98327..</New-Random-delta>
22.   </New-Random>
23.   ...other info e.g., SignatureProperty...
24. </Object>
25. <SignatureValue>MCrVLtRlk=...</SignatureValue>
26. <KeyInfo>...Key Info of Originator...</KeyInfo>
27.</Signature>
```

Figure 6. A sample implementation of sanitizable signature

Figure 6 shows a sample implementation of a sanitizable signature (namespace prefixes are omitted). Line 07 specifies the hash function used for the referenced element. Lines

$08-09$ provide and commit the original random coins used in the *trapdoor* hash function. Line $10-12$ specifies which intermediary is allowed to change the element referenced and also provides the final recipient with the key information required to retrieve the public key for re-generation of the hash value. Line 13 specifies the location where the new random coins (computed by intermediary) could be found. Lines $18-24$ show the placement of the new random coins. The new random-coin elements would be empty initially and only be filled by the intermediary. Finally, lines 25 and 26 provide the signature value and the key information of the original signer respectively.

In this design, there is an implicit verification that the intermediary has made changes to the new random coins. The original random coins and the reference to the new coins are included in the signature by the originator and hence remained unchanged if it passes signature verification. Recall that only the entity with the private key can recompute the right new random coins that correspond to the new data. The new random coins in lines 20 and 21, although not signed, if they could produce the correct *trapdoor* hash, we can deduce that the new coins are issued by the authorized intermediary. This is to ensure accountability of the corresponding data received by the final recipient.

We can see that with this design, sanitizable signature can be incorporated into the existing XML signature standards by effective use of extensions provided in the schema. This design also allows co-existence of (1) both regular hash and *trapdoor* hash in a single signature and (2) both signature schemes in a SOAP message. Such a capability provides backward compatibility and flexibility for mix-and-match depending on the business requirements.

5. Conclusion and Future Work

In this paper, we studied the practical message integrity, trust, change control, authenticity and accountability requirements of business documents (wrapped in SOAP message) that undergo modifications in web-service-enabled business processes. We presented with an example, a business process where use of the vanilla XML signature scheme as specified by WS-Security, is not sufficient in fully satisfying the security requirements. This is because the XML signature scheme is unable to handle data that requires modification by another entity, it does not allow originating entity to control the information that can be changed and does not establish trust for the intermediary entities. We then introduced the use of sanitizable signature as a concise solution to complement and overcome the limitations of the vanilla XML signature scheme. We illustrated using the example, that when used together with vanilla XML signature, we can now satisfy the security needs of the process. As the security requirements of each process and each pair of cross-organizational integration are different, we would envision that both XML signature and sanitizable signature schemes be flexibly deployed and co-exist to suit the needs of a particular business process. The proposed scheme is specifically suitable for situations in which the entities involved in changing the SOAP messages are decided by the originating entity. The proposed scheme strongly encourages decoupled systems and delegation of tasks to trusted entities in the business process, hence avoiding being overwhelmed with the work of coordination and consolidation of information from various entities by a single centralized entity.

In summary, the main benefit of use of sanitizable signature is to empower the message originator to delegate the intermediary entities to make changes in the "limited and controlled" [1] manner while protecting the data integrity with a single signature. However, the benefits of such a scheme would be diluted in the case of a dynamic situation involving dynamic composition of web services and routing. Going forward, a promising direction for future work is to design a similar scheme that is suited for an environment with dynamic composition of web services.

References

[1] G. Ateniese, D. H. Chou, B. de Medeiros, and G. Tsudik, *Sanitizable signatures*, in European Symposium on Research in Computer Security, 2005.

[2] R. H. Deng and Y. Yang, *Achieving end-to-end authentication in intermediary-enabled multimedia delivery systems*, in Information Security Practice and Experience Conference, 2007.

[3] W3C, *XML-signature syntax and processing*, http://www.w3.org/TR/xmldsig-core/.

[4] H. Krawczyk and T. Rabin, *Chameleon signatures*, in Network and Distributed System Security Symposium, 2000.

[5] M. McIntosh and P. Austel, *XML signature element wrapping attacks and countermeasures*, in ACM Workshop on Secure Web Services, 2005.

[6] M. R. Mohammad Ashiqur Rahaman and A. Schaad, *An inline approach for secure SOAP requests and early validation*, in Open Web Application Security Project, AppSec, 2006.

[7] OASIS, *Web services security: SOAP message security 1.1 (ws-security 2004)*, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf.

[8] M. A. Rahaman and A. Schaad, *SOAP-based secure conversation and collaboration*, in IEEE International Conference on Web Services, 2007.

[9] M. A. Rahaman, A. Schaad, and M. Rits, *Towards secure SOAP message exchange in a SOA*, in ACM Workshop on Secure Web Services, 2006.

[10] W3C, *Canonical XML version 1.0*, http://www.w3.org/TR/xml-c14n.

[11] W3C, *SOAP specifications*, http://www.w3.org/TR/soap.

[12] G. Mella, E. Ferrari, E. Bertino, Y. Koglin: *Controlled and cooperative updates of XML documents in byzantine and failure-prone distributed systems*, in ACM Transactions on Information and System Security, 2006.