BPEL Processes for Non-Repudiation Protocols in Web Services

M Bilal, J P Thomas, P Harrington
Department of Computer Science
Oklahoma State University
USA
jpt@cs.okstate.edu

Ajith Abraham
School of Computer Science and Engineering
Chung-Ang University
Korea
ajith.abraham@ieee.org

Abstract

BPEL provides a language for the formal specification of business processes and business interaction protocols. In business transactions non-repudiation is a serious security issue in which any involved party denies having participated in a transaction. In this paper we propose and verify novel non-repudiation protocols for business transactions in a number of scenarios and specify them in BPEL. Our proposed protocols fulfill the requirements of security, fairness, protection and timeliness.

1. Introduction

Business Process Execution Language (BPEL) for Web Services is a language for the formal specification of business processes and business interaction protocols [1]. BPEL provides an environment to describe business processes that include multiple Web services and standardize message exchange internally and among partners. Linking those web services together into a one large business process introduces a number of security problems. One of these problems is non-repudiation which means denial of having participated in a message exchange [2]. A numbers of protocols have been developed to solve non-repudiation. In general, the messages are encrypted with a secret key and sent to the receiver.

Fairness of a protocol depends on who is controlling the execution of the protocol. It may be inclined either toward the sender or receiver, or may be fair to both. For example receiver repudiation can be avoided by designing a protocol such that the sender sends the encrypted message and does not release the encryption key until he gets a receipt acknowledgment from the receiver. Such a protocol favors the sender because he may not send key after receiving acknowledgement and claims that he did. On the other hand if a Trusted Third Party (TTP) releases the key to the receiver, this makes the protocol fair. To eliminate the presence of TTP at the time of a dispute between the sender and receiver, the protocol needs to generate enough digital evidences for both the sender and the receiver. In this paper, we propose a number of non-repudiation protocols for two-party and chain-linked business transactions involving web services. The proposed protocols are specified in BPEL because they provide security, accountability, fairness, timeliness and confidentiality. We use Petri Net theory to analyze the proposed non-repudiation protocols.

2. Related Work

A detailed description of BPEL can be found in [1]. As for non-repudiation, there are two approaches. In general these protocols encrypt the message with a secret key and send it to the receiver and then the two parties exchange a delivery receipt and the message key to get the original message. An alternate approach is to involve a trusted third party that acts as a notary. Zhou [2] describes non-repudiation protocols in a number of scenarios between two entities, for example, where communicating channel is completely reliable and where the sender and receiver don't necessarily play a fair role. In Zhou and Gollmann's fair protocol [3], the originator divides a message into two parts, a commitment C and a key K. C is a cipher text of message M and L is a unique label for the protocol run. This protocol assumes that A, B and TTP are each equipped with their own private signature key and the related public verification keys and assumes that both parties will be able to retrieve the key from TTP. The main idea of this protocol is to send C first and then key K, which unlocks the message, is released. In the non-repudiation message protocol for collaborative e-business [4], the message is encrypted with secret key, which is generated at runtime. The sender sends a message encrypted with the secret key. That key is 'double-encrypted' which means a twice-encrypted secret key that is first encrypted with the receiver's public key and then with the public key of TTP. A non-repudiation protocol for chain-linked transactions is reported in [5]. In our approach there is no need for the TTP to be available at the time of dispute. Furthermore we outline verification of the protocols using Petri Nets and we specify the protocols in BPEL to enable web services implementation.

3. Secure Model for Web Services

From a security perspective, there are two things to be considered.

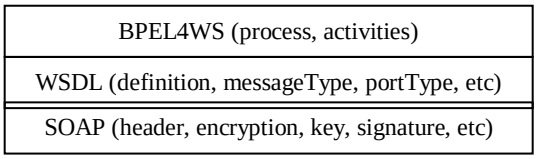
- If messages at the business level contain confidential information, it is required that no one can read the original messages except the party to which it is sent.
- Repudiation among parties may arise. Two kinds of disputes can arise [5]. Repudiation of recipient arises when originator A claims having sent a message to recipient N, who denies having received it. Repudiation of origin arises when recipient N claims having received the message from the originator A, who denies sending it.

We propose a protocol that protects the confidentiality of message contents such that no unauthorized intermediary is able to read the original message. Non-repudiation is achieved by involving a trusted third authority (TTP) but this third party is not needed at the time of dispute. Furthermore, the third party cannot access the message sent between the business entities.

We propose novel non-repudiation protocols between 2 parties as well as for a chain linked business transactions that may involve intermediate parties in different topologies. In one case of the protocol, intermediate parties are not able to access and modify the original message. In the other case intermediate parties are able to access and modify the original message depending upon the authorization granted. We also analyze the protocols for security and reliability. Furthermore, we specify these non-repudiation protocols in BPEL and Petri net models are used to verify the protocols.

We assume that web services within the organization can trust each other. A non-repudiation protocol is therefore required only when communication is between external services. Furthermore, we assume that the third party is not available at the time of dispute. Communication channels are assumed to be reliable.

BPEL is a layer on top of WSDL, i.e., it uses WSDL to specify actions that should take place in a business process, and to describe the web services provided. There are ports in WSDL that must be associated with bindings, one of which is SOAP.



We use the following notation [4] in this paper:
$X \mid Y$: concatenation of two messages X and Y.
$MD(X)$: message digest value of message X.
$eK(X)$: encryption of message X with key K.
$dK(X)$: decryption of message X with key K
$sK(X)$: digital signature of message X with the private key K
P_A, S_A : the public and private key of A.
$A \rightarrow B : X$: A sends message X to B.

3.1 Secure model of 2-party transactions

Here the transaction is between two parties, where one is a Buyer and the other is a supplier. We involve a TTP to establish Non-repudiation between parties. There are four public web services (considering BPEL process as a web service) and an internal web service. An internal web service, which is an inventory manager, sends an order to replenish inventory to the buyer request process (figure 1) which interacts with the external web service. These are the steps executed between a buyer requester and a supplier.

1. Requester sends a purchase message M, which is encrypted with a key generated by the requester, as well as a double-encrypted key (generated key is first encrypted with the public key of the recipient and then with the public key of TTP) to the seller along with the dual signature (this is a signature on the message digest of the double encrypted key and message digest of the encrypted message).
2. Supplier receives the encrypted message and sends an acknowledgement receipt back to the requester after checking the integrity of the encrypted message (eK(M)) and the double-encrypted key by comparing with the dual signature. Both eK(M) and the double-encrypted key are checked by the supplier generating the message digests and comparing with the message digests in the signature. Supplier therefore confirms that it received the correct encrypted message contents before proceeding.
3. Supplier forwards the double-encrypted key to the TTP, along with its signature1 on the message digest of encrypted message to acknowledge the correct receipt of the encrypted message. Supplier is required to send this signature1 to the TTP in order to access the key. TTP stores the signature1 temporarily for signature distribution at the end of the protocol.
4. TTP decrypts the double-encrypted key using its private key and releases the encrypted key to the Supplier. The TTP then waits for acknowledgement from the supplier. In case the TTP does not receive this acknowledgement within a certain timeout, TTP detects the supplier's misbehavior.
5. The supplier decrypts the encrypted key received from the TTP using its private key. It then sends signature2 on the message digest of the decrypted secret key to the TTP, as confirmation of receiving the key. The supplier creates the signature2 on the digested secret key so that TTP cannot access any key information from the signature.

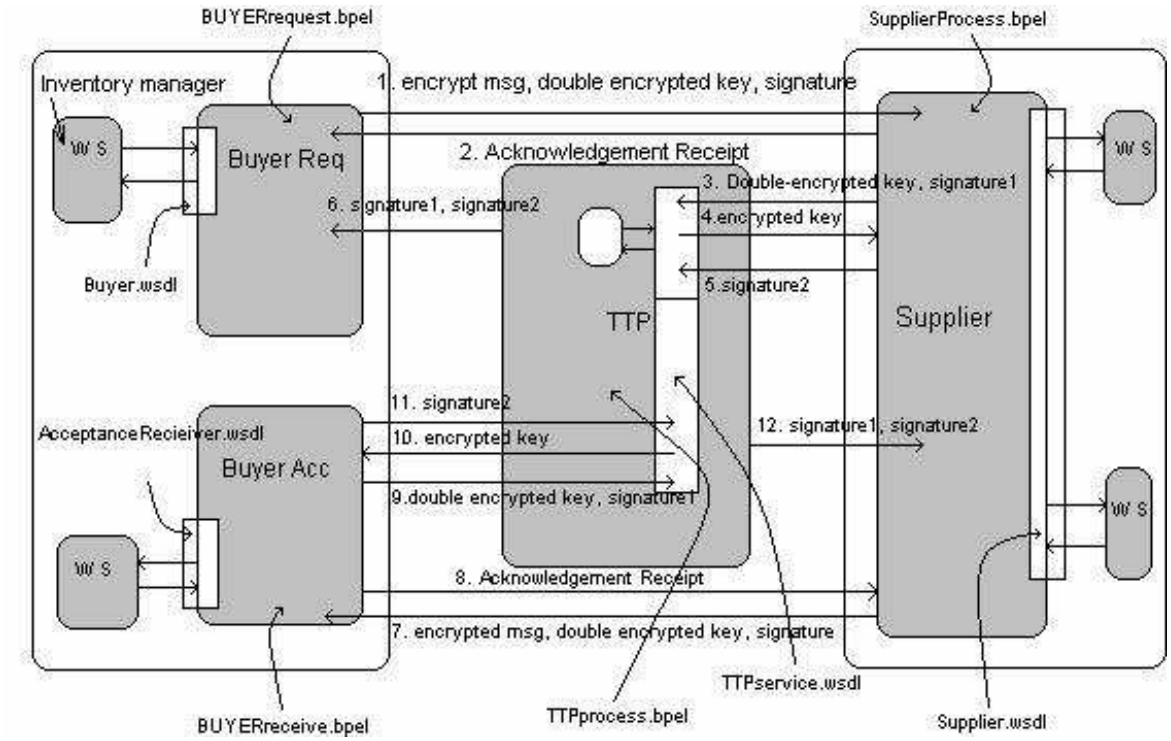


Figure 1: Non-repudiation with 2 parties

6. TTP sends the two signatures received in steps 3 and 5 to the sender. These signatures are the supplier's acknowledgement of receiving correct purchase message and secret key.
7. After processing the buyer's request, supplier sends the encrypted purchase acceptance message, along with double encrypted key and dual signature to Acceptance component of buyer.
8. Buyer Acceptance process sends an acknowledgement receipt back to supplier.
9. This is same as step 3, but instead of supplier, this message is send by Buyer acceptance component to TTP.
10. TTP decrypt the double-encrypted key and release the encrypted key to the Buyer Acceptance component process.
11. The Buyer Receiver component sends signature to the TTP, the confirmation of receiving the key same as in step 5.
12. The protocol ends with TTP forwarding both signatures to seller.

There are 12 messages. We can reduce it to 10 by removing step 2 and 8 and use step 6 and 12 as acknowledgments.

Space limitations prevent a full BPEL specification and an outline is shown below. WSDL definitions for the processes, starting with Buyer.wsdl is created. This web service allows replenishing the inventory by placing an order. There are three messages, a request for purchase order, acknowledgment receipt of the request, and signature from TTP. Service links are used to define the capabilities of partners in the BPEL process. A partner is linked to a portType and also a set of operations in the WSDL file using those service links. Similarly there is the Request Component of Buyer: This process takes the request from an inventory manager and sends request with double-encrypted key to the supplier. In the final step it receives signatures from TTP. There are three partners of the buyer request process - inventory manager, Supplier and TTP. We first define partners and containers to store data. The BPEL process at the buyer is BUYERrequest.bpel. It is layered on top of the BUYER.wsdl file. In this process, after defining the partners and containers of the process, we specify the BPEL activities of the process starting with the sequence activity.

Request process

Begin sequence

- Receive a request from Inventory Manager and deposit it in the request container.
- Assign data from request container to container to be sent to supplier.
- Invoke a "place purchase order" request with supplier based on data stored in the pervious step.
- Assign acknowledgement received from supplier to container being sent to Inventory Manager.
- Reply to Inventory Manager with the response from supplier.
- Receive signatures from trusted third party showing that supplier has accessed the original message.

End sequence
Note: Receive, Invoke, Reply, and Assign etc. are BPEL activities.
Supplier Process: The sequence of activities of BPEL process at supplier is as follows:
Begin sequence
- Receive Request from Buyer Request process.
- Assign a receipt message in the container that is used in the reply activity.
- Reply to Buyer Request process.
- Assign encrypted message and double encrypted key in the container that used in the invoke activity.
- Invoke TTP process and send Double-encrypted key and signature1.
- After receiving key from TTP again invoke TTP process to send signature2.
- After processing order send order status to Buyer Acceptance process by invoke.
- Finally, receives signatures of buyer receive process from TTP.

End sequence
Acceptance Component of Buyer. The second BPEL process at BUYER has following sequence.
Begin sequence
- Receive Acceptance from Supplier.
- Reply to supplier with receipt acknowledgement.
- Invoke TTP to decrypt double encryption key and send signature1 (on the message digest of original message and dek), and to get the encrypted key.
- Invoke the TTP to send the signature2 (on the digested key) after accessing original message.

End sequence
TTP Process
Begin sequence
- Receive Request and signature1 from supplier process to decrypt the key.
- Reply to request from supplier process.
- Receive signature2 from supplier process.
- Invoke Buyer Request process to send signatures of supplier process.
- Receive request and signature1 from Buyer Acceptance process to decrypt the key.
- Reply to request from Buyer Acceptance process.
- Receive signature2 from Buyer Acceptance process.
- Invoke Supplier process to send the signatures of the Buyer Acceptance process.

End sequence

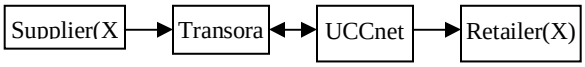
Because messages can be modified business process implementation should use WS-Security (web service security). It provides security by keeping security information in the SOAP part of the message. WS-security does not provide fairness and accountability. To fulfill such requirements there is a need to use the fair non-repudiation protocol.

3.1.1 Petri Net Model of BPEL BPEL processes consist of two types, abstract process and executable process. Both processes contain elements that can be model in Petri nets. Petri Net Model of WSDL is:
Place → PortType (Operations – input, output messages)
Transition → ServiceLinkType (Name, my role, partner role)
Token → Message (Data)
Arc → Binding
Note: The Service link type definition can be placed within the WSDL document defining the portTypes from which the different roles are defined.
Petri Net Model of Executable BPEL Process is:
Place → Containers
Transition → Invoke, Receive, Reply, Assign, Switch
Token → Message (Data)

A sequence activity is represented hierarchically and can be refined into a number of lower level activities such as invoke, receive etc. BPEL Models of each process are merged to obtain a system-wide view a complete web business transaction. Although the individual models may display the desired properties of livness, safeness and complete termination, the merged net may not display such properties. To draw the Petri net of B2B processes, global information of the processes are required. Each process is only aware of itself and other web services (or BPEL processes) it calls. The entire business transaction can be therefore modeled by merging the models of individual transactions. When a process or web service needs to be invoked, its respective WSDL file is traced. A WSDL file has all the information required to communicate. The complete Petri net represents all the possible execution paths of the whole system, in our case inventory manager's web method (web services) is followed by the WSDL and then the web service and then WSDL of process and so on. Space limitations prevent a detailed Petri Net representation.

3.2 Secure model for chain of transactions

The protocol presented in Section 3.1 established non-repudiation between two individual parties. However, business transactions are rarely so simple, and may involve more parties in many different topologies. There is a need for non-repudiation in such environments. We consider here a chain-linked business transaction. Assume a supplier (X) wants to publish details about a new product (say a bar of soap). He publishes the information to a public Market Place such as Transora. Transora gets the information from a lot of suppliers. Retailer (X) sells soap and wants to know when new soap products are available. Retailer (X) has relationship with UCCnet. UCCnet sends information to a lot of retailers. The flow is modeled as:



There are a number of security issues: How can Retailer(X) be guaranteed that the information he received is indeed from Supplier(X)? Or how can

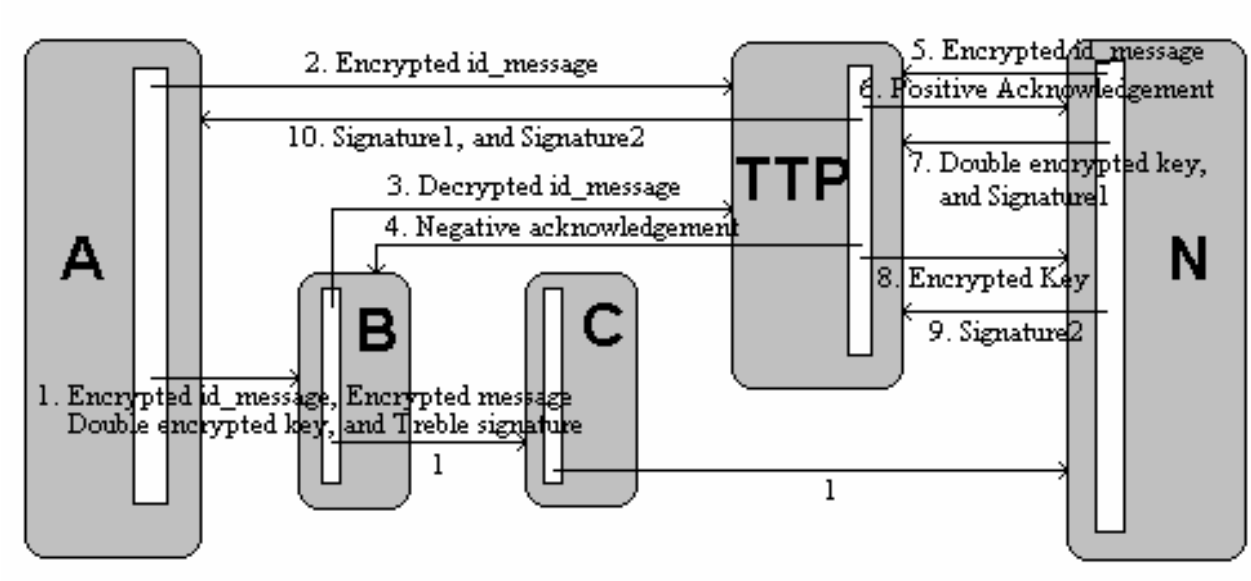
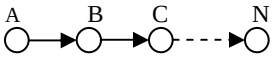


Figure 2: Non-repudiation Protocol for Chained Linked Business Transaction.

Supplier(X) be guaranteed that Retailer(X) did actually get the new product detail? We propose a novel non-repudiation protocol for chain linked business transactions. Non-repudiation in a chain linked system is modeled as follows.



There are the following cases

Case 1: A → N

Intermediate businesses (B, C …) cannot read the message or key.

Case 2: A → N

Intermediate businesses (B, C …) can read message, modify it or add their own information.

In this paper we only consider case 1.

3.2.1 Case 1 To avoid modification of the message by intermediate nodes, there must be a non-repudiation protocol down the whole business linked chain. The proposed protocol works as follows (Figure 2).

K: a symmetric secret key generated by A. The receiver N can access original message M only by using the secret key K.

t_id: transaction id,

message M encrypted using K: em = eK(M)

encrypted key from sender A: ek_from_A = $eP_N(K)$

Double-encrypted Key: dek = eP_{TTP}(ek_from_A) = $eP_{TTP}(eP_N(K))$

md1 = MD(em) md2 = MD(dek) md3 = MD(id_N)

id_N = eP_N(id_message) id-message generated by originator or sender

id_TTP = eP_{TTP}(id_message)

treble signature: ts = t_id | md1 | md2 | md3 | sS_A(t_id | md1 | md2 | md3)

Step 1: A sends the encrypted id-message, encrypted message, double encrypted key and treble signature to B, who sends it to C and so on until it reaches N.

Message from A → B : t_id | A | B | N | id_N | em | dek | ts

Message from B → C : t_id | B | C | N | id_N | em | dek | ts and so on

Step 2: A encrypts the id-message with public key of TTP and sends it to TTP.

A → TTP : t_id | eP_{TTP}(id_message)

Step 3: Now suppose an intermediate node B tries to get the key by sending it's own id message dS_B(id_N') (it is not same as id_N).

B → TTP : t_id | dS_B(id_N')

The TTP will not accept because id_message is not equal to id_message it received in step 2 from A.

Step 4: TTP → B : t_id | Negative acknowledgement

Now consider messages with the recipient N. First the recipient N uses the message digests to ensure the message has not been trampled with (using treble signature ts) and it then needs to identify itself to the TTP.

Step 5: id_N is first decrypted at N using the private key of N: dS_N(id_N) to get id_message. It is next encrypted using public key of the TTP and sent to the TTP.

N → TTP : t_id | eP_{TTP}(id_message)

Step 6: TTP → N : Positive acknowledgement

Step 7: The recipient N sends double encrypted key and signature1 to the TTP

N → TTP : t_id | A | N | md1 | md3 | dek | sS_N(t_id | md1 | md3)

Step 8: TTP decrypts the double encrypted key and sends encrypted key to the recipient N.

$$TTP \rightarrow N : t\_id \mid ek\_from\_TTP$$

Where, $ek\_from\_TTP = dS_{TTP}(dek)$: decryption of dek using private key of TTP.

Step 9: The recipient N sends his signature2 on a digested secret key to the TTP.

$$N \rightarrow TTP : t\_id \mid sS_N(MD(ek\_from\_TTP))$$

Step 10: TTP sends both signatures to the originator A.

$$TTP \rightarrow A : t\_id \mid sS_N(t\_id \mid md1 \mid md3) \mid sS_N(MD(ek\_from\_TTP))$$

We give an informal security analysis of our protocol.

- Intermediate nodes cannot get the key from the TTP because of id_N.
- Originator A knows that the recipient N gets the message because of md3 in the recipient N's signature.
- The originator A knows that message is correct because of md1 in the recipient N's signature.
- The originator A knows that the key is delivered correctly because of the signature of the recipient N on the digested secret key i.e. $sS_N(MD(ek\_from\_TTP))$.
- N know that this message is from the originator A by checking the integrity of the message using treble signature. It is the only sender that can generate that signature.

3.2.1.1 Dispute Resolution

Repudiation of Recipient If the recipient N denies receiving message 'M', the originator A can present evidence in the form of signatures of N plus (t_id, em, dek, id_message, md1, md2, md3, K, M, P_{TTP}, P_N, ek_from_TTP) to arbitrator. The arbitrator will compare the t_id and check the different encrypted messages, message digests and signatures. The log of the TTP may also be checked. The originator A wins the dispute if all the checks are positive. Originator A will win even if he is unable to provide log information of the TTP as in the last check. The presence of the TTP is therefore not required at the time of dispute.

Repudiation of Origin If originator A denies sending the message 'M', the recipient N can present evidence in the form of treble signature of A plus the different encrypted messages, message digests and signatures. Recipient N will win the dispute if all the checks are positive.

3.2.1.2 Security Protocol Properties: The properties of our non-repudiation services are:

Fairness - neither party can gain an advantage by quitting prematurely or misbehaving during the execution of the protocol. Detailed verification of fairness is not provided due to lack of space. For example, if the protocol terminates at step 1 because of communication problems or misbehavior of an intermediate node, the originator loss nothing. At this time, intermediate nodes or the recipient N may have an encrypted message em = eK(M) and double encrypted key dek = eP_{TTP}(ek_from_S) but they cannot access the message until the TTP decrypts the key. If any intermediate node tries to access the secret key, first it needs to identify itself by decrypting id_N and this is not possible because id_message is encrypted with the public key of recipient N. The recipient N gets access to entire original message only after step 8.

Protection and integrity of Message: - This protocol protects the involved parties from common message protection threats such as message interception and modification, and replay attacks. We used message digest and encryption techniques to protect the message from interception and modification. The integrity of the message can be verified by comparing with the message digest values in the treble signature. Protection is provided by encrypting a message with a key which is double encrypted so that no one but recipient can access the message content. The protocol generates a new transaction id (t_id) every time to protect from replay attacks.

Confidentiality of transaction - The protocol provides confidentiality such that only the recipient can access the original contents of the message. The TTP or intermediate parties cannot read the message. The recipient needs to identify itself by sending id_message to TTP. Although the intermediate nodes are involved in the communication, they cannot access the message. The only way to read the message is through the secret key that encrypts the message. The secret key is double encrypted to prevent the intermediaries and TTP from getting access to the key and hence the original message.

Timeliness - The Protocol achieves timeliness as each involved party can terminate the protocol at any time at their own judgment while maintaining fairness. For example, if the protocol terminates after step 1, the recipient N cannot take advantage because it cannot access the message even it gets the treble signature.

3.2.1.3 Building the BPEL Processes

For the BPEL specification we consider only one intermediate node in the above protocol. For more intermediate nodes, the BEPL specification for the intermediate nodes can be simply replicated for each intermediate node. We therefore need four processes, supplier A, buyer N, intermediate party B and TTP (figure 2).

Supplier Process A: This process takes request from process B and sends request with double-encrypted key to the intermediate process. Process A also sends id_message to process TTP. Finally it receives signatures from TTP process. First we define partners and containers to store data. After defining the partners and containers of the process, we define activities of the process starting from sequence of process.

Begin sequence

- Receive a request from the intermediate node.

- Invoke a process to produce encrypted id_message, encrypted message and double encrypted key.
- Assign the data to the container to be sent to intermediate node.
- Reply the intermediate node to send message.
- Assign encrypted id_message to a container.
- Invoke the process TTP and send encrypted id_message.
- Receive the signatures from the trusted third party that buyer access the original message.

End sequence

Intermediate Process B:

The sequence of activities at intermediate process is::

Begin sequence

- Invoke the buyer process to send request of purchase.
- Receive the information from supplier process.
- Invoke the buyer process to send that information.

End sequence

Buyer Process N

Begin sequence

- Receive the information from intermediate node.
- Assign the id_message to the container to be use in the invoke activity.
- Invoke the process TTP and send the id_message.
- Receive the acknowledgement from the process TTP.
- Assign double encrypted key and signature1 to the container.
- Reply to the process TTP and send double encrypted key and signature1.
- Receive encrypted key from the process TTP.
- Reply the process TTP and send the signature2.

End sequence

Process TTP

TTP sequence of activities is as follow:

Begin sequence

- Receive id_message from the supplier process A.
- Receive id_message from the buyer process N.
- Invoke internal process to compare id_message for identification.
- Reply with the acknowledgement to the buyer process N.
- Receive double encrypted key and signature1 from buyer process.
- Reply with encrypted key to the buyer process.
- Receive the signature2 from the buyer process.
- Reply the supplier process to send the signatures of the buyer process.

End sequence

3.2.2.1 Colored Petri Net Model for case 1 We model the above protocol using colored Petri Nets. Such modeling allows us to verify and reason about the protocol. It is beyond the scope of this paper to provide a detailed verification.

Definition 1: A colored Petri Net (CPN) is a tuple CPN = (PN, $\sum$, CR, E) where [6]

- PN = (p, n, f, m) in an ordinary Petri net,
  - p is a set of places $\{p_1,p_2,\ldots p_n\}$
  - n is a set of transitions $\{t_1,t_2,\ldots t_m\}$
  - f is set of functions from places to transitions and from transitions to places
  - m is the initial marking of the net
- $\sum = \{\sigma_1, \sigma_2, \ldots\}$ is a finite set of colors,
- CR is color factor such that $CR(p) \subseteq \sum$, and $CR(m(p)) \subseteq CR(P)$
- E, the arc function such that: $\forall f\ (p, t), f(t, p) \in F, E_f \subseteq CR(p)_{MS}$
- m(p): denotes distinct color at a place p, e.g. m(p) = g + r represents place p containing a token of color g and a token of color r, i.e., CR(m(p)) = {g, r}.
- $CR(p)_{MS}$: Represents the set of multi set or bags over CR(p) e.g. given a set CR(p) = {a, b, ….}, the multi sets a, a+b, a+2b are members of CR(p).

Petri Nets allow verification of many properties of the protocol including liveness and deadlock properties, of the protocol. Each token color represents a web services transaction. We outline a verification of the reliability of the non-repudiation protocol for chain-linked Transactions. We show that if any transaction does not take place due to communication failures or node misbehavior, the protocol will terminate.

Definition 2: Given a CPN, we define the number of distinct colors associated with a place p_i as $u_i = | C(p_i) |$.

Definition 3: Given a CPN, we define the number of ways in which a transition t_i can fire as v_i = the number of consistent substitutions of each arc function $f(p_j, t_i)$ (the condition to be satisfied for the transition to fire) with the elements in $C(p_j)$, where $p_j \in {}^\bullet t_i$. (${}^\bullet t_i$. is the set of input places of t_i.) We regard a colored Petri net as continuous time homogeneous Markov process [7] and we can analyze the system reliability..

Definition 4: System is reliable if and only if each input and output function of all transitions are reliable. Where, reliability of the system is denoted by R(system)

$R(\text{system}) = R(I(t_j)) \text{ AND } R(O(t_j))$

$R(\text{system}) = R(f(p_i, t_j)) \text{ AND } R(f(t_j, p_k))$

Now first consider $R(f(p_i, tj))$, where $p_i \in {}^\bullet t_j$

We unfold the CPN as follows: For each place p_i in CPN, create as many places as u_i and label them with color $\sigma_1, \sigma_2, \sigma_3 \ldots\ldots, \sigma_u$ and for each transition t_j in CPN create as many transitions as v_j and give them distinct label to each.

Now draw the edges from every place derived from p_i to every transition t_j with arc function $E_{f\ (pi,\ tj)}$ and substitute σ_k in $E_{f\ (pi,\ tj)}$ with logical 1 which ensure a correct execution of t_j, so

$$R(f(p_i, t_j)) = \prod_{i=1}^{u} E_{f(pi,tj)}$$

Now consider $R(f(t_j, p_k))$, where $p_k \in t_j{}^\bullet$ (set of output places of t_j). For each place p_k in CPN, create as many places as u_k and label them with color $\sigma_1, \sigma_2, \sigma_3 \ldots\ldots, \sigma_k$ and for each transition t_j in CPN create as many

transitions as v_j and give them distinct label. Now draw the edges from every transition derived from t_j to every transition p_k with arc function $E_{f\,(tj,\,pk)}$ and substitute σk in $E_{f\,(tj,\,pk)}$ with logical 1 which ensure a correct execution of t_j, so

$$R(f(t_i, p_k)) = \prod_{i=1}^{u} E_{f(pi,tj)}$$

u

Hence; R(system) = R ($f(p_i, t_j)$) AND R($f(t_j, p_k)$)
This shows that in the colored Petri net a transition may not fire properly (due to communication failure or misbehaving nodes). We assume that the Petri Net is live. If a transition does not fire, then the liveness property is no longer true and this will terminate the system.

4. Conclusions

In this paper we propose non-repudiation protocols for 2-party and chain linked web services transactions where the trusted third party signature is not considered as evidence; therefore TTP availability is not required at the time of dispute resolution. Protocols were analyzed so that they fulfill the security and non-repudiation requirements in efficient manner. We proposed Petri nets to validate the flow of protocols. The secure web services flow is modeled using BPEL.

In multiple entity non-repudiation protocols the number of originators and recipients may vary. As the number increase this can affect performance and availability. Performance improvement is one area for further work. Modeling of attacks on web services is another area for further research. Since the proposed protocols are based on the assumption of reliable communication channel, protocol independent of reliable communication channels is needed.

References

[1] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. "*Business Process Execution Language for Web Services*", Version 1.1, Pages 8-111, 2003.
http://www-128.ibm.com/developerworks/library/ws-bpel/

[2] J. Zhou, *Non-repudiation in Electronic Commerce*, Artech House, Computer Security Series, 2001.

[3] J. Zhou and D. Gollmann. "A Fair Non-repudiation protocol". *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pp. 55-61, 1996.

[4] S. Yang, Stanley Y. W. Su, and H. Lam, "A Non-Repudiation Message Transfer Protocol for E-commerce". *Proceedings of IEEE International Conference on E-commerce*, 2003.

[5] J. Onieva, J. Zhou, J. Lopez. "Non-repudiation protocols for multiple entities", *Computer Communications*, 27(16):pp. 1608-1616, October 2004

[6] V. Atluri, W. Huang, "A Petri net Based Safety Analysis of Workflow Authorization", *Journal of Computer Security*, Vol. 8*y* 8, Pages 209-240, 2000.

[7] S. Hong, K. Kim. "A Reliability Analysis of Distributed Programs with Colored Petri Nets" *ETRI Journal*, Vol. 16 No. 1, 1997.