Wuhan University Journal of Natural Sciences

2010, Vol.15 No.5, 385-392

Article ID 1007-1202(2010)05-0385-08

DOI 10.1007/s11859-010-0671-1

Fair Non-Repudiation for Web Services Transactions

□ **SU Ruidan, FU Shaofeng, ZHOU Lihua**†

Key Laboratory of Computer Network and Information Security of the Ministry of Education, Xidian University, Xi'an 710071, Shaanxi, China

Abstract: To safeguard the interests of transacting parties, non-repudiation mechanisms need to assure fairness and timeliness. The non-repudiation service currently implemented usually does not consider the requirement of fairness and the fair non-repudiation protocols to date can not be suitably applied in real environment due to its complex interaction. This paper discusses the transaction-oriented non-repudiation requirement for Web services transaction, analyzes the constraints of the traditional model for the available fair non-repudiation protocols and designs a new Online-TTP fair non-repudiation protocol. The new protocol provides a fair non-repudiation solution to secure Web services transactions and can be embedded into a single Web service call. The protocol adopts evidence chained to decreasing the overhead of evidence verification and management and alleviates the overhead of certificate revocation checking and time-stamp generation for signatures. The protocol has strong fairness, timeliness, efficiency and practicability.

Key words: fair non-repudiation; Web services transaction; evidence chaining; timeliness

CLC number: TP 309

Received date: 2010-02-02

Foundation item: Supported by the National High Technology Research and Development Program of China (863 Program) (2006AA01Z405)

Biography: SU Ruidan, male, Ph.D. candidate, research direction: networks and information security. E-mail: rdsu@xidian.edu.cn

† To whom correspondence should be addressed. E-mail: zhlh@xidian.edu.cn

0 Introduction

Web services without effective security are not very useful. The W3C Web Services Architecture Requirements[1] outline the following six important security considerations for a comprehensive security framework: authentication, authorization, confidentiality, integrity, non-repudiation and accessibility. For legal reasons, non-repudiation is critical for business Web services and any business transaction, for that matter. Non-repudiation means that a communicating partner can prove that the other party has performed a particular transaction[2]. Non-repudiation services must ensure that when Alice sends some information to Bob over a network, neither Alice nor Bob can deny having participated in a part or the whole of this communication. Therefore a non-repudiation protocol has to generate non-repudiation of origin evidences intended for Bob, and non-repudiation of receipt evidences destined to Alice. In case of a dispute (e.g. Alice's denying having sent a given message or Bob's denying having received it), an adjudicator can evaluate these evidences and make a decision in favor of one of the parties without any ambiguity.

Non-repudiation is one of the essential security services in computer networks[3,4]. Zhou[5] analyzes the non-repudiation mechanism and shows that it does not satisfy fairness. We say that a non-repudiation protocol is fair if, at the end of the protocol, either Alice receives a non-repudiation of receipt evidence and Bob receives the message and the corresponding non-repudiation of origin evidence or none of them obtains any valid evidence. Any party should not be in an advantageous position[2]. Fair non-repudiation protocols are the ones traditionally studied in literatures.

The fundamental goal of designing fair non-repudiation protocol is to resolve dispute of transactions in e-government or e-commerce applications. From the view of practicability, we prefer the ease of implementing fair non-repudiation during the phase of system development.

WS-Security[6] was developed to provide Simple Object Access Protocol (SOAP) extensions that define mechanisms using XML Encryption and XML Signature to secure SOAP messages. XML digital signature, like any other digital signing technology, provides authentication, data integrity (tamper-proofing), and non- repudiation. XML digital signature provides a flexible means of signing and supports diverse sets of Internet transaction models. For example, you can sign individual items or multiple items of an XML document. XML Encryption develops XML syntax for representing encrypted data and establishes procedures for encrypting and decrypting such data. Unlike SSL (Secure Socket Layer), with XML Encryption, you can encrypt only the data that needs to be encrypted. In essence, WS-Security provides the support for message-level encryption and signature, but is not a solution to fair non-repudiation unless certain fair non-repudiation protocol is designed.

The draft[7] defines a method to allow senders of SOAP messages to request message disposition notifications that may optionally be signed to prove that the receiver received the SOAP message without modification. The specification makes use of the Web Services Security and a protocol is presented for voluntary non-repudiation of receipt that systematically provides cryptographic proof of both parties' participation in a transaction. The protocol does not consider the property of fairness since the receiver could ignore the request of message disposition notifications.

Ref. [8] introduces a flexible framework to support fair non-repudiable B2B interactions based on a trusted delivery agent. An implementation of Web services is also presented. The scheme depends on an inline TTP which increases the overhead and complexity of message exchange. Its design is established on the basis of B2B environment and is not suitable for application cases in which one party may be thin-client and can not act as server.

The non-repudiation service currently implemented usually does not consider the requirement of fairness, only adopting a certain signing method to sign the critical data. The fair non-repudiation protocols to date[2,9,10] can not be applied in real Web services environment due to its complex interaction although they have perfect fairness. Aiming to provide a fair non-repudiation solution to secure Web services transactions, this paper discusses the transaction-oriented non-repudiation requirement for Web services transaction, analyzes the constraints of the traditional model for the available fair non-repudiation protocols and designs a new Online-TTP fair non-repudiation protocol which provides a fair non-repudiation solution to secure Web services transactions and can be embedded into a single Web service call. The protocol adopts evidence chaining[11] to decrease the overhead of evidence verification and management and has strong fairness, timeliness, efficiency and practicability.

1 Review Two Fair Non-Repudiation Protocols

This section reviews two classical fair non-repudiation protocols. A practical non-repudiation protocol must have fairness, timeliness, simple interaction and etc, as shown in Table 1. We assume that audiences are familiar

Table 1 Requirements for practical non-repudiation protocol

Protocol property	Explanation
Strong fairness	A non-repudiation protocol provides strong fairness if and only if at the end of a protocol execution either Alice gets the non-repudiation of receipt evidence for the message *m*, and Bob gets the corresponding message *m* as well as the non-repudiation of origin evidence for this message, or none of them gets any valuable information.
Timeliness	A non-repudiation protocol provides timeliness if and only if all honest parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness.
Confidentiality	Confidentiality can be provided by non-repudiation protocol itself or other security components, such as SSL and etc.
Communication channel	Internet provides a communication channel like unreliable channels to a great extent.
Protocol interaction	Due to the influences of bandwidth, online time, implementation complexity, it is advisable to make interactions of protocol as simple as possible.
Evidence management	The verified non-repudiation evidences must be managed securely in order to guard against their loss which causes the impossibility of dispute resolution. The lifetime of evidences is determined by system policy for non-repudiation.

with those concepts related with fair non-repudiation, such as fairness, timeliness, evidence of origin, evidence of receipt, TTP involvement etc[2]. Table 2 shows the notations that will be used to describe the protocols.

Table 2 Notions for fair non-repudiation protocol

Notion	Definition
A	Sender
B	Receiver
TTP	The trusted third party
TSA	The Time-stamping Authority
$X \to Y$	Transmission from entity X to entity Y
$X \leftrightarrow Y$	GET operation made by entity X. The underlying protocol can be HTTP or FTP
$h()$	A collision resistant one-way hash function
$E_k()$	A symmetric-key encryption function under key k
$D_k()$	A symmetric-key decryption function under key k
$E_X()$	A public-key encryption function under X's public key
$D_X()$	A public-key decryption function under X's private key
$S_X()$	Signature function of entity X
m	Message sent from A to B
k	Session key used by A to cipher m
$c = E_k(m)$	Cipher of m under the session key k
$l = h(m,k)$	A label that in conjunction with (A, B) uniquely identifies a protocol run
f	A flag indicating the purpose of a message

1.1 Fair Non-Repudiation Protocol with Online TTP

Zhou and Gollmann[9] presented a practical fair non- repudiation protocol which explores an Online TTP. Table 3 shows the notations used to formalize the protocol. The protocol works as follows:

Table 3 Notions for Zhou's protocol

Item	Notion
Evidence of origin for the cipher	$\text{EOO} = S_A(f_{\text{EOO}}, B, l, T, c)$
Evidence of receipt for the cipher	$\text{EOR} = S_B(f_{\text{EOR}}, A, l, T, c)$
Evidence of submission for session key	$\text{Sub} = S_A(f_{\text{Sub}}, B, l, T, k)$
Evidence of confirmation for session key	$\text{Con}_k = S_{\text{TTP}}(f_{\text{Con}_k}, A, B, l, k)$

1) $A \to B:\ f_{\text{EOO}}, B, l, T, c, \text{EOO}$;
2) $B \to A:\ f_{\text{EOR}}, A, l, T, \text{EOR}$;
3) $A \to \text{TTP}:\ f_{\text{Sub}}, B, L, T, k, \text{Sub}$;
4) $B \leftrightarrow \text{TTP}:\ f_{\text{Con}_k}, A, B, L, T_0, k, \text{Con}_k$;
5) $A \leftrightarrow \text{TTP}:\ f_{\text{Con}_k}, A, B, L, T_0, k, \text{Con}_k$.

If the communication channel between TTP and *A/B* is resilient, the protocol has strong fairness and timeliness. The evidences obtained by the receiver are EOO and Con_k, while the evidences obtained by the sender are EOR and Con_k. T is the deadline to limit the time Con_k and k can be accessed by the public TTP. T_0 is a time stamp to indicate when the confirmed key has actually been made available to the public. T and T_0 are introduced to guarantee timeliness. Obviously, the protocol does not support confidentiality and is suitable for application in which real-time communication is not a necessity.

1.2 Fair Non-Repudiation Protocol with Offline TTP

Ref. [10] designs a fair non-repudiation protocol which involves TTP only in case of exception. Table 4 shows the notations used to formalize the protocol. The protocol consists of three sub-protocols: main protocol, abort protocol and recovery protocol.

Table 4 Notions for fair non-repudiation protocol with offline TTP

Item	Notion
Evidence of origin for the cipher	$\text{EOO} = S_A(f_{\text{EOO}}, B, \text{TTP}, l, h(c))$
Evidence of receipt for the cipher	$\text{EOR} = S_B(f_{\text{EOR}}, A, \text{TTP}, l, h(c))$
Submission evidence for key k	$\text{Sub} = S_A(f_{\text{Sub}}, B, l, E_{\text{TTP}}(k))$
Evidence of origin for key k	$\text{EOO}_k = S_A(f_{\text{EOO}_K}, B, l, k)$
Evidence of receipt for key k	$\text{EOR}_k = S_B(f_{\text{EOR}_K}, A, l, k)$
Recovery request	$\text{Re}c_X = S_X(f_{\text{Re}c_X}, Y, l)$
Confirmation evidence for key k	$\text{Con}_k = S_{\text{TTP}}(f_{\text{Con}_k}, A, B, l, k)$
Abort request	$\text{Abort} = S_A(f_{\text{Abort}}, B, l)$
Abort confirmation evidence	$\text{Con}_a = S_{\text{TTP}}(f_{\text{Con}_a}, A, B, l)$

Main protocol:

1) $A \to B$:
$f_{\text{EOO}}, f_{\text{Sub}}, B, \text{TTP}, l, c, E_{\text{TTP}}(k), \text{EOO}, \text{Sub}$;
2) $B \to A: f_{\text{EOR}}, A, \text{TTP}, l, \text{EOR}$;
If A times out, then abort;
3) $A \to B: f_{\text{EOO}_k}, B, l, k, \text{EOO}_k$;
If B times out, then recover;
4) $B \to A:\ f_{\text{EOR}_k}, A, l, k, \text{EOR}_k$;
If A times out, then recover.

Abort protocol:

1) $A \to \text{TTP}: f_{\text{Abort}}, l, B, \text{Abort}$;
If aborted or recovered, then stop else aborted=true;
2) $\text{TTP} \to A:\ f_{\text{Con}_a}, A, B, l, \text{Con}_a$;
3) $\text{TTP} \to B: f_{\text{Con}_a}, A, B, l, \text{Con}_a$.

Recovery protocol:

1) $X \to \text{TTP}$:
$f_{\text{Re}c_X}, f_{\text{Sub}}, Y, l, h(c), E_{\text{TTP}}(k),\ \text{Re}c_X,\ \text{Sub, EOR, EOO}$;

If aborted or recovered, then stop else recovered = true;

2) $\text{TTP} \to A : f_{\text{Con}_k}, A, B, l, k, \text{Con}_k, \text{EOR}$;

3) $\text{TTP} \to B$: $f_{\text{Con}_k}, A, B, l, k, \text{Con}_k$.

If the communication channel between TTP and *A*/*B* is resilient, the protocol has strong fairness and timeliness. In normal cases, *A* and *B* follow the main protocol. Eventually, *A* gets the evidences of EOR and EOR_k , while *B* gets the evidences of EOO and EOO_k. In cases of exception, such as network failure or dishonest party, abort or recovery protocol will be used; non-repudiation evidences for origin are EOO and Con_k while non-repudiation evidences for receipt are EOR and Con_k.

2 Non-Repudiation for Web Services Transaction

2.1 Web Services

This article assumes that you are familiar with existing Web services-enabling technologies: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI)[12]. SOAP is an extensible XML messaging protocol. It defines an XML encoding for function calls and their parameters, allowing applications to communicate with one another regardless of their environments. WSDL is an XML encoding for the description of Web services, specifying properties of services, such as what a service can do, how to invoke it, and where it resides. UDDI is a repository-based registry service for finding and publishing Web services.

A Web service can be easily located by making a request to a Web service registry, which acts as the centralized system to connect Web service requestors with Web service providers. Initially, a service provider will register its Web services at UDDI registry. UDDI will process the Web services and publish them into a searchable registry system. When a request to discover and describe a Web service is made by a user, UDDI will locate the named service in its database. If the service is found, UDDI will post a list of available service providers who may satisfy the user's request. At this point of the request, the return message to the user is wrapped in a WSDL to help the user locate the required service provider.

The user will continue the Web services interaction workflow by preparing a SOAP request and sending it to the service provider. The service provider is designed to process this request upon receiving it and return a SOAP response to the user. The user receives the SOAP response, extracts the response, and performs post processing to convert the response into a format appropriate for consumption, thus ending the Web service interaction.

A Web service interaction behaves as one round exchange of XML messages which are SOAP request and response across the network. In many implementations, SOAP requests are similar to function calls with SOAP responses returning the results of the function call. For example, an E-Trade service provider receives a stock transaction order from one of its clients and performs the transaction on behalf of that client. E-Trade wants to ensure that it can prove the completeness of that transaction to an arbitration committee, for example, if a dispute arises. We need some level of non-repudiation for Web services transactions.

2.2 Fair Non-Repudiation for Web Services Transaction

Intuitively, we can describe the requirement of fair non-repudiation for Web services transaction as follows. Service requestor can not deny having initiated Web services request while service provider can not deny having made a certain response to requestor according to the request. Especially for fairness, any party should not be in an advantageous position.

We find that the fair non-repudiation protocols in Section 1 do not adapt to the requirement of Web services transaction very well. All the studies about fair non-repudiation surround the transaction model in which we define a basic transaction as the transferring of a message *m* from entity *A* to entity *B*, and represent this event with the following message flow: $A \to B : m$. Thus, typical disputes that may arise in a basic transaction could be:

1) *A* claims that it has sent *m* to *B*, while *B* denies having received it;

2) *B* claims that it has received *m* from *A*, while *A* denies having sending it.

The basic transaction model can not describe typical Web services transaction accurately. In our opinion, a typical Web services transaction should cover both the request phase and response phase which are associated with each other.

1) *A* sends a service request to *B*;

2) *B* sends a service response to *A* after processing the request.

Briefly speaking, a business transaction consists of two traditional transactions for request and response respectively which are combined together. So the possible disputes could be:

● *B* claims that it has received a request from *A*, while *A* denies having sent it;

● *A* claims that it has sent a request to *B*, while *B* denies having received it;

● *A* claims that it has received response from *B*, while *B* denies having sent it;

● *B* claims that it has sent response to *A*, while *A* denies having received it.

2.3 Analysis

Many e-government and e-commerce applications built with Web services currently claim that trusted non-repudiation services have been implemented, but the approaches mostly work as follows:

1) Requestor signs some fields in the SOAP request and embeds the signature and digital certificate in the request which would be sent to the service provider;

2) Upon receiving the SOAP request, the provider verifies the certificate and the signature, including certificate revocation checking. If both are valid, the provider makes a time stamp for the signature from Time-Stamping Authority (TSA) and records the request data, signature and time stamping into the audit database;

3) After validating the SOAP request successfully, the provider extracts the parameters and invokes the service method. When making SOAP response, the provider accomplishes the signature using its private key just as the consumer does and returns the SOAP response to the requestor;

4) Upon receiving the SOAP response, the requestor validates the certificate and signature. If valid, the requestor assures the result is made by the trusted provider. Similarly, it is maintained that time stamping for the signature is done and record the response data, the signature and time stamping are recorded into audit database on the requestor side.

Zhou[5] has concluded that the mechanism above is problematic and only satisfies the requirement of audit trail to some extent. Specifically, fairness can not be achieved. There exists the vulnerability that the provider could deny having received the service request and the requestor could deny having received the response. Furthermore, costs of certificate revocation checking and time stamping generation for signature can not be ignored when implementing a practical system. Certificate revocation checking imposes another notorious cost on PKI-Based applications although the standard approaches, such as Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP), are available[13]. The conventional approach for maintaining the validity of digital signatures requires that either the evidence sender or the evidence receiver interacts with an on-line trusted time-stamping authority to get each newly generated digital signature time-stamped so that there is extra evidence to prove whether the signature was generated before the corresponding public key certificate was revoked and thus is deemed valid[14]. Obviously, such approaches are not cost-effective in ordinary on-line transactions.

While SOAP is independent of transport protocol, we'll consider SOAP over HTTP in this article, since this is the most commonly used transport method. HTTP is based on request/response mode. Many nodes on the internet are thin-client and can not be accessed passively. Additionally, users usually hope to finish his operation as easy as possible and avoid complexity brought by security mechanism. We analyze the challenges for applying the protocols mentioned in Section 1 to Web services transaction and conclude that they are not suitable for implementing fair non-repudiation in Web services environment.

1) We apply the protocol in Section 1.1 to SOAP request and response transmission independently. For the request phase, requestor needs to interact with TTP to perform session key submission and obtain Con_k. Interaction with TSA to make a time stamping for EOR of the transaction request is needed. The provider needs to interact with TTP to get session key and Con_k, and communicate with TSA to make a time stamping for EOO of transaction request. This applies to the response phase in the same way;

2) Considering the protocol in Section 1.2, a successful run involves two round message exchange (four messages). Thus, a total of four round message exchanges (eight messages) are needed for an entire Web services transaction. Both parties still need to make time stamping with the help of TSA. Multiple rounds of message exchange will increase the complexity of session management. Combined with user session management, it is complicated to implement in practice;

3) Because the client-side computers are not protected adequately due to weak security consciousness, non-repudiation evidences should not only be stored on the user's side. Too many factors may lead to the loss of these important evidences, such as hard disk damage, deletion by malicious software and etc. So it is advisable to make TTP to manage all the evidences intensively even if users may choose to store evidence locally.

The fundamental goal of fair non-repudiation is the ease of integration with application. The solution meets the security requirements (fairness and timeliness) and avoids the complexity of implementation of security mechanism. Ideally, we expect to embed the support of fair non-repudiation into the exchange of a single Web services transaction to decrease the overhead of message exchange. Further, it is significant to lessen the overhead of certificate validation, evidence verification and management.

3 Fair Non-Repudiation Protocol for Web Services Transaction

3.1 Protocol Design

We design a new fair non-repudiation protocol to support fair non-repudiation of Web services transaction. The scheme introduces the idea of evidence chaining to simplify evidence verification and management. Certificate revocation checking is performed by TTP intensively. The scheme consists of main protocol and query protocol. For Web services transaction, A represents consumer while B provider. Notions used in protocol are shown in Table 5.

Main Protocol:

M1) $A \rightarrow B$:

$f_{\mathrm{EOO}_{\mathrm{req}}}, B, \mathrm{TTP}, l, T, c_{\mathrm{req}}, h(k_{\mathrm{req}}), E_{\mathrm{TTP}}(k_{\mathrm{req}}), \mathrm{EOO}_{\mathrm{req}}, \mathrm{Cert}_A$;

B uses Cert_A to verify the signature of $\mathrm{EOO}_{\mathrm{req}}$, but need not check whether Cert_A has been revoked. If $\mathrm{EOO}_{\mathrm{req}}$ is valid, B generates $\mathrm{EOR}_{\mathrm{req}}$; otherwise, B responds error to A.

Table 5 Notions of fair non-repudiation protocol for Web services transaction

Item	Notion
Evidence of origin for the request cipher	$\mathrm{EOO}_{\mathrm{req}} = S_A(f_{\mathrm{EOO}_{\mathrm{req}}}, B, \mathrm{TTP}, l, T, h(c_{\mathrm{req}}), h(k_{\mathrm{req}}), E_{\mathrm{TTP}}(k_{\mathrm{req}}))$
Evidence of receipt for the request cipher	$\mathrm{EOR}_{\mathrm{req}} = S_B(f_{\mathrm{EOR}_{\mathrm{req}}}, A, \mathrm{TTP}, l, T, h(c_{\mathrm{req}}), E_{\mathrm{TTP}}(k_{\mathrm{req}}), \mathrm{EOO}_{\mathrm{req}})$
Confirmation evidence for request session key	$\mathrm{Con}_{k_{\mathrm{req}}} = S_{\mathrm{TTP}}(f_{\mathrm{Con}_{k_{\mathrm{req}}}}, A, B, l, k_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}}, T, T_0)$
Evidence of origin for the response cipher	$\mathrm{EOO}_{\mathrm{resp}} = S_B(f_{\mathrm{EOO}_{\mathrm{resp}}}, B, \mathrm{TTP}, l, T, h(c_{\mathrm{resp}}), h(k_{\mathrm{resp}}), E_{\mathrm{TTP}}(k_{\mathrm{resp}}), \mathrm{EOR}_{\mathrm{req}})$
Evidence of receipt for the response cipher	$\mathrm{EOR}_{\mathrm{resp}} = S_A(f_{\mathrm{EOR}_{\mathrm{resp}}}, A, \mathrm{TTP}, l, T, h(c_{\mathrm{resp}}), E_{\mathrm{TTP}}(k_{\mathrm{resp}}), \mathrm{EOO}_{\mathrm{resp}})$
Confirmation evidence for response session key	$\mathrm{Con}_{k_{\mathrm{resp}}} = S_{\mathrm{TTP}}(f_{\mathrm{Con}_{k_{\mathrm{resp}}}}, A, B, l, k_{\mathrm{resp}}, \mathrm{EOR}_{\mathrm{resp}}, T, T_0)$
Query request	$\mathrm{Query} = S_A(f_{\mathrm{Abort}}, A, B, l)$
Confirmation evidence for query	$\mathrm{Con}_q = S_{\mathrm{TTP}}(f_{\mathrm{Con}_q}, A, B, l, \mathrm{Con}_{k_{\mathrm{req}}}, \mathrm{EOO}_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}}, T_{0_{\mathrm{req}}}, \mathrm{Con}_{k_{\mathrm{resp}}}, \mathrm{EOO}_{\mathrm{resp}}, \mathrm{EOR}_{\mathrm{resp}}, T_{0_{\mathrm{resp}}}, T)$ or $\mathrm{Con}_q = S_{\mathrm{TTP}}(f_{\mathrm{Con}_q}, A, B, l, \mathrm{Con}_{k_{\mathrm{req}}}, \mathrm{EOO}_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}}, T_{0_{\mathrm{req}}}, T)$ or $\mathrm{Con}_q = S_{\mathrm{TTP}}(f_{\mathrm{Con}_q}, A, B, l, \mathrm{none})$

M2) $B \rightarrow \mathrm{TTP}$:

$f_{\mathrm{Sub}}, A, B, l, T, h(c_{\mathrm{req}}), h(k_{\mathrm{req}}), E_{\mathrm{TTP}}(k_{\mathrm{req}}), \mathrm{EOO}_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}},$ $\mathrm{Cert}_A, \mathrm{Cert}_B$;

If any of the following checks fails, TTP reports error to B.

① Check clock is before T;

② Use Cert_A and Cert_B to verify the signature of $\mathrm{EOR}_{\mathrm{req}}$;

③ Check the revocation status of Cert_A and Cert_B ;

④Decrypt the session key k_{req} and verify its digest.

M3) $\mathrm{TTP} \rightarrow B : f_{\mathrm{Con}_{k_{\mathrm{req}}}}, A, B, l, k_{\mathrm{req}}, \mathrm{Con}_{k_{\mathrm{req}}}, T_{0_{\mathrm{req}}}$;

TTP records $A, B, l, k_{\mathrm{req}}, \mathrm{Con}_{k_{\mathrm{req}}}, \mathrm{EOO}_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}},$ $T, T_{0_{\mathrm{req}}}$ into evidence database while sending session key and Con_k to B. $T_{0_{\mathrm{req}}}$ is the time TTP records evidence for the request phase. At this step, TTP needs to communicate with TSA to get $\mathrm{Con}_{k_{\mathrm{req}}}, \mathrm{EOO}_{\mathrm{req}}, \mathrm{EOR}_{\mathrm{req}}$ time-stamped.

M4) $B \rightarrow A$:

$f_{\mathrm{EOO}_{\mathrm{resp}}}, A, \mathrm{TTP}, l, T, c_{\mathrm{resp}}, h(k_{\mathrm{resp}}), E_{\mathrm{TTP}}(k_{\mathrm{resp}}), \mathrm{EOO}_{\mathrm{resp}},$ $\mathrm{EOR}_{\mathrm{req}}, \mathrm{Con}_{k_{\mathrm{req}}}, \mathrm{Cert}_B$;

A verifies the signatures of $\mathrm{EOR}_{\mathrm{req}}$, $\mathrm{Con}_{k_{\mathrm{req}}}$ and $\mathrm{EOO}_{\mathrm{resp}}$, but need not check whether Cert_B has been revoked. The valid $\mathrm{EOR}_{\mathrm{req}}$ and $\mathrm{Con}_{k_{\mathrm{req}}}$ serve as receipt evidence for SOAP request. If $\mathrm{EOO}_{\mathrm{req}}$ is valid, A generates $\mathrm{EOR}_{\mathrm{resp}}$; otherwise, B responds error to A.

M5) $A \rightarrow \mathrm{TTP}$:

$f_{\mathrm{Sub}}, A, B, l, T, h(c_{\mathrm{resp}}), h(k_{\mathrm{resp}}), E_{\mathrm{TTP}}(k_{\mathrm{resp}}), \mathrm{EOO}_{\mathrm{resp}},$ $\mathrm{EOR}_{\mathrm{resp}}, \mathrm{Cert}_A, \mathrm{Cert}_B$;

If any of the following checks fails, TTP reports error to A.

① Clock is before T;

② Use Cert_A and Cert_B to verify the signature of $\mathrm{EOR}_{\mathrm{resp}}$;

③ Decrypt the session key k_{resp} and verify its digest.

M6) $\text{TTP} \rightarrow A : f_{\text{Con}_{k_{\text{req}}}}, A, B, l, k_{\text{resp}}, \text{Con}_{k_{\text{resp}}}, T_{0_{\text{resp}}}$.

TTP records $k_{\text{resp}}, \text{Con}_{k_{\text{resp}}}, \text{EOO}_{\text{resp}}, \text{EOR}_{\text{resp}}, T, T_{0_{\text{resp}}}$ into evidence database while sending session key and $\text{Con}_{k_{\text{resp}}}$ to A. At this step, TTP needs to communicate with TSA to get $\text{Con}_{k_{\text{resp}}}, \text{EOO}_{\text{resp}}, \text{EOR}_{\text{resp}}$ time-stamped.

Query Protocol:

Q1) $X \rightarrow \text{TTP} : f_{\text{Query}}, A, B, l, \text{Query}$;

Q2) $\text{TTP} \rightarrow X$:

$f_{\text{Resp}}, A, B, l, \text{Con}_{k_{\text{req}}}, \text{EOO}_{\text{req}}, \text{EOR}_{\text{req}}, \text{Con}_{k_{\text{resp}}}, \text{EOO}_{\text{resp}}, \text{EOR}_{\text{resp}}, T, T_{0_{\text{req}}}, T_{0_{\text{resp}}}, \text{Con}_q$ or $f_{\text{Resp}}, A, B, l, \text{Con}_{k_{\text{req}}}, \text{EOO}_{\text{req}}, \text{EOR}_{\text{req}}, T, T_{0_{\text{req}}}, \text{Con}_q$ or $f_{\text{Resp}}, \text{none}, \text{Con}_q$.

The step in which A or B asks for session key for request or response is very important. For request phase, TTP generates $\text{Con}_{k_{\text{req}}}$ and sends session key k_{req} to B which can be used by B to decrypt the encrypted request. $\text{EOO}_{\text{req}}, \text{EOR}_{\text{req}}$ and $\text{Con}_{k_{\text{req}}}$ are chained by order. The validity checking of certificates for A and B is accomplished by TTP. If the signature $\text{Con}_{k_{\text{req}}}$ by TTP is valid, we can ensure that the chained evidences are believed to be trusted. Only TTP is needed to communicate with TSA to time-stamp $\text{EOO}_{\text{req}}, \text{EOR}_{\text{req}}$ and $\text{Con}_{k_{\text{req}}}$, relieving operation overhead for A and B. As far as response phase is concerned, the idea also applies. The protocol makes a chain for EOR_{req} and EOO_{resp} in order to link the request and response of the same transaction which can be identified by the tuple (A, B, l).

For timeliness, A assigns a time stamp T to give a timeout limitation before which A expects to receive B's response. In normal cases, A can receive the response from B and get all the needed evidences. B can initiate the query protocol to obtain A's response receipt evidence. Therefore, both parties own all the evidences for the transaction.

If A times out, A can start query protocol with TTP to know the status of the transaction. The result can be divided into two cases:

① B has not asked for request session key from TTP. Obviously, this case is fair.

② B has obtained request session key from TTP, but does not generate a response or send back a response which is not received by A. This case is fair for the request phase.

$T_{0_{\text{req}}}$ is the time stamp at which TTP delivers the session key k_{req} to B and records the evidences into database. $T_{0_{\text{req}}}$ plays two important roles in protocol run. First, the session key confirmed by TTP is available to B so that B can decrypt the encrypted request; Second, we can convince that the chained EOO_{req} and EOR_{req} are valid because TTP is responsible for checking the validity of Cert_A and Cert_B at $T_{0_{\text{req}}}$. The explanation above applies to $T_{0_{\text{resp}}}$ in the same way.

Only one interaction between A and B is needed so that the protocol implementation can be embedded in the original SOAP message exchange run. With TTP archiving all the evidences centrally, the risk of evidence loss is avoided.

3.2 Dispute Resolution

Settling dispute is a critical operation for fair non-repudiation service. In case of dispute, the judge will start arbitration program, evaluate the available non-repudiation evidences and make a decision about which party is honest. The arbitration flow for the protocol above is as follows:

Arbitration step 1: If A, B or TTP can provide $m_{\text{req}}, c_{\text{req}}, k_{\text{req}}, l, T_{0_{\text{req}}}, \text{Cert}_A, \text{Cert}_B, \text{EOO}_{\text{req}}, \text{EOR}_{\text{req}}, \text{Con}_{k_{\text{req}}}$ and all the following checks are successful, B can not deny the receipt of request m_{req}, and A cannot deny the origin of m_{req} .

1) check the TTP's signature $\text{Con}_{k_{\text{req}}}$ and its time stamp;

2) check the validity of Cert_A and Cert_B of $T_{0_{\text{req}}}$;

3) use Cert_A to check the signature of EOO_{req} ;

4) use Cert_B to check the signature of EOR_{req} ;

5) check $l = H(m_{\text{req}}, k_{\text{req}})$;

6) check $m_{\text{req}} = D_{k_{\text{req}}}(c_{\text{req}})$.

Arbitration step 2: If A, B or TTP can provide $m_{\text{resp}}, c_{\text{resp}}, k_{\text{resp}}, l, T_{0_{\text{resp}}}, \text{Cert}_A, \text{Cert}_B, \text{EOO}_{\text{resp}}, \text{EOR}_{\text{resp}}, \text{Con}_{k_{\text{resp}}}$, EOR_{req} and all the following checks are successful, A can not deny the receipt of response m_{resp}, and B cannot deny the origin of response m_{resp}.

1) check the TTP's signature $\text{Con}_{k_{\text{resp}}}$ and its time stamp;

2) check the validity of Cert_A and Cert_B of $T_{0_{\text{resp}}}$;

3) use Cert_A to check the signature of EOR_{resp} ;

4) use Cert_B to check the signature of EOO_{resp} ;

5) check $m_{\text{resp}} = D_{k_{\text{resp}}}(c_{\text{resp}})$.

If both of these two arbitration steps pass, we can ensure fair non-repudiation of the entire transaction; otherwise, we can identify non-repudiation for the request and response respectively with the support of TTP.

4 Conclusion

This paper discusses the transaction-oriented non-repudiation requirement for Web services transaction, analyzes the constraints of the traditional model for the available fair non-repudiation protocols and designs a new Online-TTP fair non-repudiation protocol which can be embedded into a single Web service call. The protocol adopts evidence chaining to decrease the overhead of evidence verification and management and has strong fairness, timeliness, efficiency and practicability. In practical deployment, much more attention should be paid to TTP for its high performance, high availability and system security itself since TTP is online and involved in every protocol run.

References

[1] Austin D, Barbin A , Ferris C, *et al*. Web Service Architecture Requirements, W3C[EB/OL]. [2010-02-15]. *http://www.w*3.*org/TR/*2004/*NOTE-wsa-reqs*-20040211.

[2] Kremer S, Markowitch O, Zhou J. An intensive survey of fair non-repudiation protocols[J]. *Computer Communications*, 2002, **25**(17): 1606-1621.

[3] Kremer S, Markowitch O, Zhou J. An intensive survey of fair non-repudiation protocols[J]. *Computer Communications*, 2002, **25**(17): 1606-1621.

[4] Onieva J A, Zhou Jianying, Lopez J. Multiparty non-repudiation: A survey[J]. *IEEE Communications Magazine*, 2008, **46**(4): 102-107.

[5] Zhou J. Non-Repudiation in Electronic Commerce[C]// *Computer Security Series*. London: Artech House, 2001.

[6] Nadalin A, Kaler C, Phillip Hallam-Baker. OASIS Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)[EB/OL].*http://docs.oasis-open.org/wss*/2004/01/*oasis*-200401-*wss-soap-message-security*-1.0.*pdf.*

[7] Eric L, Gravengaard. Web services security: Non–repudiation proposal draft[EB/OL]. [2010-02-15]. *http://schemas. reactivity.com*/2003/04/*web-services-non-repudiation*-05.*pdf.*

[8] Robinson P, Cook N, Shrivastava S. Implementing Fair Non-repudiable Interactions with Web Services[R]. Tyne: School of Computing Science, Univ Newcastle, 2005.

[9] Zhou J, Gollmann D. A fair non-repudiation protocol[C]// *IEEE Symposium on Security and Privacy, Research in Security and Privacy*. Oakland, CA: IEEE Computer Security Press, 1996: 55-61.

[10] Kremer S, Markowitch O. Optimistic non-repudiable information exchange[C]// 21*st Symp on Information Theory in the Benelux*, *Werkgemeenschap Informatieen Communicatietheori*. Enschede (NL): Wassenaar Press, 2000: 139-146.

[11] You C, Zhou J, Lam K. On the efficient implementation of fair non-repudiation[J]. *Computer Communication Review* 1998, **28**(5): 50-60.

[12] David B, Hugo H, Francis M C, *et al*. Web service architecture[EB/OL]. [2010-02-15]. *http://www.w3c.org/TR/ws-arch/*. 2004.

[13] Naor M, Nissim K. Certificate revocation and certificate update[C]// *Proceedings of the* 7*th USENIX Security Symposium*. Texas: USENIX Association, 1998: 217-228.

[14] Haber S, Stornetta W S. How to time-stamp a digital document[J]. *Journal of Cryptology*, 1991, **3**(2): 99-111.

□