2010 2nd International Conference on Education Technology and Computer (ICETC)

Achieving Fair Non-Repudiation for Web Services Transaction

Su Rui-dan, Fu Shao-feng, Zhou Li-hua
Key Laboratory of Computer Network and Information Security of the Ministry of Education
Xidian University, 710071
Xi'an, Shaanxi, China
rdsu@xidian.edu.cn

***Abstract*—For safeguarding the interests of web services transaction parties, non-repudiation mechanism need assure fairness and timeliness. The non-repudiation service implemented currently mostly 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 complexity of 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 which provides a fair non-repudiation solution to secure web services transactions and can be embedded into single web service call. Evidence chaining is exploited to decrease the overhead of evidence verification and management and alleviates the overhead of certificate revocation checking and time-stamp generation for signatures. The protocol owns properties of strong fairness, timeliness, efficiency and practicability.**

Keywords-fair non-repudiation; web services transaction; evidence chaining; timeliness

I. INTRODUCTION

Organizations are adopting SOA to support their mission critical business applications. SOA is a computing paradigm emphasizing dynamic service discovery, composition, and interoperability. Web services are the technologies that can be used to implement SOA and are increasingly becoming the SOA implementation of choice. Web services without effective security aren't very useful. The W3C Web Services Architecture Requirements [1] outlines 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, for that matter, any business transaction. 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 to Bob, and non-repudiation of receipt evidences destined to Alice. In case of a dispute (e.g. Alice denying having sent a given message or Bob denying having received it), an adjudicator can evaluate these evidences and take a decision in favor of one of the parties without any ambiguity. Strictly, non-repudiation mechanisms must 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 literature.

The draft [3] 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 when used systematically provides cryptographic proof of both parties participation in a transaction. The protocol does not consider the property of fairness although since the receiver could ignore the request of message disposition notifications.

The paper [4] introduces a flexible framework to support fair non-repudiable B2B interactions based on a trusted delivery agent. A Web services implementation is also presented. The scheme depends on an inline TTP which aggravates 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 implemented currently mostly does not consider the requirement of fairness, only adopting certain signing method to sign the critical data. The fair non-repudiation protocols to date [2][5][6] can not be applied in real web services environment due to its complexity of interaction although they own 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 single web service call. The protocol adopts evidence chaining [7] to decrease the overhead of evidence verification and management and owns properties of strong fairness, timeliness, efficiency and practicability.

Organization. The paper is organized as follows. In Section II, we describe requirements and notions for fair non-repudiation protocols. In section III, we discuss the transaction-oriented non-repudiation requirement for web services transaction and analyze the constraints of the

978-1-4244-6370-1/$26.00 © 2010 IEEE

traditional model for the available fair non-repudiation protocols. In Section IV, we give our protocol and analyze its characteristics. Section V concludes.

II. REQUIREMENTS AND NOTIONS

A practical non-repudiation protocol must own fairness, timeliness, simple interaction and etc, as shown in Table I. We assume that audiences are familiar with those concepts related with fair non-repudiation, such as fairness, timeliness, evidence of origin, evidence of receipt, TTP involvement and etc. [2] Table II. shows the notations that will be used to describe the protocols.

TABLE I. REQUIREMENTS FOR PRACTICAL NON-REPUDIATION PROTOCOL

Strong fairness	A non-repudiation protocol provides strong fairness if and only if at the end of a protocol execution either Alice got the non-repudiation of receipt evidence for the message m, and Bob got the corresponding message m as well as the non-repudiation of origin evidence for this message, or none of them got 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.

TABLE II. NOTIONS FOR NON-REPUDIATION PROTOCOL

A	Sender.
B	Receiver.
TTP	The trusted third party.
TSA	The Time-stamping Authority.
$X \rightarrow Y$	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.
$S_X()$	the signature function of entity X.
m	the message sent from A to B.
k	the session key A uses to cipher m.
$c = E_k(m)$	the 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.

III. NON-REPUDIATION FOR WEB SERVICES

A. Fair Non-repudiation for Web service transaction

This article assumes that you are familiar with existing Web services-enabling technologies: 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. Intuitively, we can describe the requirement of fair non-repudiation for web service transaction as follows. Service requestor can not deny having initiated web service 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 available [2][5][6] 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: $A \rightarrow B : m$. Thus, typical disputes that may arise in a basic transaction could be:

- A claims that it has sent m to B, while B denies having received it.
- B claims that it received m from A, while A denies 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.

- A sends a service request to B.
- 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 received request from A, while A denies sending it.
- A claims that it has sent request to B, while B denies having received it.
- A claims that it received response from B, while B denies sending it.
- B claims that it has sent response to A, while A denies having received it.

B. Analysis

Many e-government and e-commerce applications built with Web services currently claim that trustable non-repudiation service has been implemented, but the approaches mostly utilize XML-Signature [8] to sign the SOAP message or part of it. Zhou. [9] has concluded that the mechanism above is problematic and only satisfies the requirement of audit trail to some extent. Especially, fairness can not be achieved. There exists a vulnerability that consumer could deny having received the response and provider could deny having received the transaction request.

While SOAP is independent of transport protocol, we'll consider SOAP over HTTP in this article, since this is the most commonly used transport. HTTP is based on request/response mode. Most nodes on the internet are thin-client and can not be accessed passively. Furthermore, users usually hope to finish his operation as easy as possible and avoid complexity brought by security mechanism. The fair non-repudiation protocols available are not suitable for implementing fair non-repudiation for in web services environment due to its interaction complexity.

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

IV. FAIR NON-REPUDIATION PROTOCOL FOR WEB SERVICES TRANSACTION

A. 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. 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 as Table III.

TABLE III. NOTIONS FOR WEB SERVICES ORIENTED FAIR NON-REPUDIATION PROTOCOL

evidence of origin for the request cipher	$EOO_{req} = S_A(f_{EOO_{req}}, B, TTP, l, T, h(c_{req}), h(k_{req}), E_{TTP}(k_{req}))$
evidence of receipt for the request cipher	$EOR_{req} = S_B(f_{EOR_{req}}, A, TTP, l, T, h(c_{req}), E_{TTP}(k_{req}), EOO_{req})$
Confirmation evidence for request key k	$Con_{k_{req}} = S_{TTP}(f_{Con_{k_{req}}}, A, B, l, k_{req}, EOR_{req}, T, T_0)$
evidence of origin for the response cipher	$EOO_{resp} = S_B(f_{EOO_{resp}}, B, TTP, l, T, h(c_{resp}), h(k_{resp}), E_{TTP}(k_{resp}), EOR_{req})$
evidence of receipt for the response cipher	$EOR_{resp} = S_A(f_{EOR_{resp}}, A, TTP, l, T, h(c_{resp}), E_{TTP}(k_{resp}), EOO_{resp})$
query request	$Query = S_A(f_{Abort}, A, B, l)$
confirmation evidence for query	$Con_q = S_{TTP}(f_{Con_q}, A, B, l, Con_{k_{req}}, EOO_{req}, EOR_{req}, T_{0_{req}}, Con_{k_{resp}}, EOO_{resp}, EOR_{resp}, T_{0_{resp}}, T)$ or $Con_q = S_{TTP}(f_{Con_q}, A, B, l, Con_{k_{req}}, EOO_{req}, EOR_{req}, T_{0_{req}}, T)$ or $Con_q = S_{TTP}(f_{Con_q}, A, B, l, none)$
Confirmation evidence for response key k	$Con_{k_{resp}} = S_{TTP}(f_{Con_{k_{resp}}}, A, B, l, k_{resp}, EOR_{resp}, T, T_0)$

1) Main Protocol:

M1） $A \rightarrow B$：

$f_{EOO_{req}}, B, TTP, l, T, c_{req}, h(k_{req}), E_{TTP}(k_{req}), EOO_{req}, Cert_A$

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

M2） $B \rightarrow TTP$：

$f_{Sub}, A, B, l, T, h(c_{req}), h(k_{req}), E_{TTP}(k_{req}), EOO_{req}, EOR_{req}, Cert_A, Cert_B$

Clock is before T;

Use $Cert_A$ and $Cert_B$ to verify the signature of EOR_{req}

Check the revocation status of $Cert_A$ and $Cert_B$

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

If any checks pass, TTP generates Con_k ; otherwise, TTP reports error to B.

M3) $TTP \rightarrow B$:

$f_{Con_{k_{req}}}, A, B, l, k_{req}, Con_{k_{req}}, T_{0_{req}}$

TTP records $A, B, l, k_{req}, Con_{k_{req}}, EOO_{req}, EOR_{req}, T, T_0$ into evidence database while sending session key and Con_k to B; At this step, TTP need communicate with TSA to get $Con_{k_{req}}, EOO_{req}, EOR_{req}$ time-stamped.

M4) $B \rightarrow A$:

$f_{EOO_{resp}}, A, TTP, l, T, c_{resp}, h(k_{resp}), E_{TTP}(k_{resp}), EOO_{resp}, EOR_{req}, Con_{k_{req}}, Cert_B$

A verify the signatures of EOR_{req} , $Con_{k_{req}}$ and EOO_{resp} , but need not check whether $Cert_B$ has been revoked. The valid EOR_{req} and $Con_{k_{req}}$ serve as receipt evidence for SOAP request. If EOO_{req} is valid, A generates EOR_{resp} ; otherwise, B responds error to A.

M5) $A \rightarrow TTP$:

$f_{Sub}, A, B, l, T, h(c_{resp}), h(k_{resp}), E_{TTP}(k_{resp}), EOO_{resp}, EOR_{resp}, Cert_A, Cert_B$

Clock is before T;

Use $Cert_A$ and $Cert_B$ to verify the signature of EOR_{resp}

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

If any checks fail, TTP reports error to B.

M6) $TTP \rightarrow A$:

$f_{Con_{k_{req}}}, A, B, l, k_{resp}, Con_{k_{resp}}, T_{0_{resp}}$

TTP record $k_{resp}, Con_{k_{resp}}, EOO_{resp}, EOR_{resp}, T, T_{0_{resp}}$ into evidence database while sending session key and $Con_{k_{resp}}$ to A; At this step, TTP need communicate with TSA to get $Con_{k_{resp}}, EOO_{resp}, EOR_{resp}$ time-stamped.

2) Query Protocol:

Q1) $X -> TTP$: $f_{Query}, A, B, l, Query$

Q2) $TTP -> X$:

$f_{Resp}, A, B, l, Con_{k_{req}}, EOO_{req}, EOR_{req}, Con_{k_{resp}}, EOO_{resp}, EOR_{resp}, T, T_{0_{req}}, T_{0_{resp}}, Con_q$

or $f_{Resp}, A, B, l, Con_{k_{req}}, EOO_{req}, EOR_{req}, T, T_{0_{req}}, Con_q$ or

$f_{Resp}, none, Con_q$

the step in which A or B asks for session key for request or response is very important. For request phase, TTP generates $Con_{k_{req}}$ and sends session key k_{req} to B which can be used by B to decrypt the encrypted request. $EOO_{req}, EOR_{req}, Con_{k_{req}}$ are chained by order. The validity checking of certificates for A and B is accomplished by TTP centrally. If the signature $Con_{k_{req}}$ by TTP is valid, we can ensure the evidences chained are believed to be trusted. Only TTP is needed to communicate with TSA to time-stamp EOO_{req} , EOR_{req} and Con_k relieving operation overhead for A and B. As far as response phase is concerned, the idea also applies. The protocol make 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).

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

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

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

$T_{0_{req}}$ is the time stamp at which TTP delivers the session key k_{req} to B and records the evidences into database. $T_{0_{req}}$ 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 guarantee the validity of $Cert_A$ 和 $Cert_B$ at $T_{0_{req}}$.

The explanation above applies to $T_{0_{resp}}$ in the same way.

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

B. 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.

Arbitration step 1: If A and B can provide $m_{req}, c_{req}, k_{req}, l, T_{0_{req}}, Cert_A, Cert_B, EOO_{req}, EOR_{req}, Con_{k_{req}}$ and all the following checks are successful, then B can not deny the receipt of request m_{req}, and A can not deny the origin of m_{req}.

- checks the TTP's signature Con_k and its time stamp;
- checks the validity of $Cert_A$ and $Cert_B$ of $T_{0_{req}}$;
- uses $Cert_A$ to check the signature of EOO_{req};
- uses $Cert_B$ to check the signature of EOR_{req};
- checks $l = H(m_{req}, k_{req})$
- checks $m_{req} = D_{k_{req}}(c_{req})$

Arbitration step 2: If A, B or TTP can provide $m_{resp}, c_{resp}, k_{resp}, l, T_{0_{resp}}, Cert_A, Cert_B, EOO_{resp}, EOR_{resp}, Con_{k_{resp}}, EOR_{req}$ and all the following checks are successful, then A can not deny the receipt of response m_{resp}, and B can not deny the origin of response m_{resp}.

- checks the TTP's signature $Con_{k_{resp}}$ and its time stamp;
- checks the validity of $Cert_A$ and $Cert_B$ of $T_{0_{resp}}$;
- uses $Cert_A$ to check the signature of EOR_{resp};
- uses $Cert_B$ to check the signature of EOO_{resp};
- checks $m_{resp} = D_{k_{resp}}(c_{resp})$

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.

V. SUMMARY AND CONCLUSIONS

Web services have been one of the fundamental technologies for deploying e-commerce transaction on internet. These transactions need ensure fair non-repudiation. 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 single web service call. The protocol adopts evidence chaining to decrease the overhead of evidence verification and management and owns properties of 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] http://www.w3.org/TR/2004/NOTE-wsa-reqs-20040211

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

[3] http://schemas.reactivity.com/2003/04/web-services-non-repudiation-05.pdf

[4] Paul Robinson, Nick Cook, and Santosh Shrivastava. Implementing Fair Non-repudiable Interactions with Web Services. Technical Report CSTR, School of Computing Science, Univ. Newcastle, 2005.

[5] J. Zhou, D. Gollmann, A fair non-repudiation protocol, in: IEEE Symposium on Security and Privacy, Research in Security and Privacy, IEEE Computer Society,Technical Committee on Security and Privacy, IEEE Computer Security Press, Oakland, CA, 1996, pp. 55~61.

[6] S. Kremer, O. Markowitch, Optimistic non-repudiable information exchange, in: J. Biemond (Ed.), 21st Symp. on Information Theory in the Benelux, Werkgemeenschap Informatie- en Communicatietheori, Enschede (NL), Wassenaar (NL), 2000, pp. 139~146.

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

[8] http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[9] ZHOU, J. 2001. Non-Repudiation in Electronic Commerce. Computer Security Series. Artech House.