OPTIMISTIC NON-REPUDIABLE INFORMATION EXCHANGE

Steve Kremer and Olivier Markowitch

Université Libre de Bruxelles, Computer Science Dept.
Bld du Triomphe C.P.212, 1050 Bruxelles, Belgium
skremer@ulb.ac.be, omarkow@ulb.ac.be

In this paper we consider the optimistic approach of the non-repudiation protocols. We study the most complete non-repudiation protocol with off-line trusted third party and we put forward some weaknesses. To fix these weaknesses we present two solutions: one keeping a passive TTP, a second one, more applicable, with an active TTP [1]*. We show that our protocol respects the fairness and timeliness properties.*

INTRODUCTION

The impressive growth of open networks during the last decade has created several new security related problems. The non-repudiation problem is one of them. 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. In comparison to other security issues, such as privacy or authenticity of communications, non-repudiation has not been studied intensively. However many applications such as electronic commerce, fair exchange, certified electronic mail, etc. are related to non-repudiation. Non-repudiation of origin can easily be provided by signing the sent information. A digital signature provides an irrefutable non-repudiation of origin evidence. Non-repudiation of receipt is more difficult to provide: therefore Alice and Bob have to follow a protocol that assures both services.

The, until recently, most complete protocols providing non-repudiation have been presented by J. Zhou [4]. His first proposal relies on a trusted third party (TTP) that has to intervene during each protocol run. This TTP plays the role of a low-weight notary. The TTP only has to publish an evidence in a read-only public

[1] A similar protocol has been published by Zhou et al. in [3]. We independently found equivalent results.

directory, accessible to both Alice and Bob. Although the TTP is low-weight it may create a communication bottleneck. Therefore Zhou presented a second protocol based on the optimistic idea [1]: he assumes that in general Alice and Bob are honest, i.e. they correctly follow the protocol, and that the TTP only intervenes, by the mean of a recovery protocol, when a problem arises.

We shall start defining the properties of the different communication channels and the requirements that have to be provided by non-repudiation protocols. Then we go on describing the Zhou-Gollman protocol and put forward some of its weaknesses. To fix these weaknesses we present two solutions: one keeping a passive TTP, but not entirely satisfying, a second one with an active TTP. We show that our protocol respects the fairness and timeliness properties.

PROPERTIES

Communication channels can be divided into three classes:

- unreliable: no assumptions are made on these channels and data may be lost;
- resilient: data may be delayed but always arrives after a finite amount of time;
- operational: data always arrives after a constant known amount of time.

Note that operational channels are rather unrealistic in heterogenous networks.

Now we will have a look on the requirements a non-repudiation protocol must fulfill. A first requirement is fairness: a non-repudiation protocol is said to be *fair* if at the end of the protocol Alice has got a complete non-repudiation of receipt evidence if and only if Bob has got the message with a complete corresponding non-repudiation of origin evidence. A second property we require is timeliness: a protocol must be finished after a finite amount of time for each participating entity that is behaving correctly with respect to the protocol.

THE ZHOU-GOLLMAN OPTIMISTIC NON-REPUDIATION PROTOCOL

We use the following notation to describe the protocol:

- $X \to Y$: transmission from entity X to entity Y
- $X \leftrightarrow Y$: ftp get operation performed by X at Y
- $h()$: 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
- $S_X()$: the signature function of entity X
- m: the message sent from A to B
- k: the message key A uses to cipher m
- $c = E_k(m)$: the cipher of m under the key k
- $l = h(m, k)$: a label to identify a protocol run
- f: a flag indicating the purpose of a message
- $EOO = S_A(f_{EOO}, B, l, t, c)$: the evidence of origin of c

- $EOR = S_B(f_{EOR}, A, l, t, c)$: the evidence of receipt of c
- $EOO_k = S_A(f_{EOO_k}, B, l, t, k)$: the evidence of origin of k
- $EOR_k = S_B(f_{EOR_k}, A, l, t, k)$: the evidence of receipt of k
- $Sub_k = S_A(f_{Sub_k}, B, l, k)$: the evidence of submission of k
- $Con_k = S_{TTP}(f_{Con_k}, A, B, l, t, k)$: the evidence of confirmation of k issued by the TTP

The protocol is divided into two subprotocols: the main protocol and a recovery protocol. The trusted third party (TTP) does only intervene in the recovery protocol. We shall first have a look at the main protocol.

1. $A \rightarrow B : f_{EOO}, B, l, t, c, EOO$
2. $B \rightarrow A : f_{EOR}, A, l, EOR$
3. $A \rightarrow B : f_{EOO_k}, B, l, k, EOO_k$
4. $B \rightarrow A : f_{EOR_k}, A, l, EOR_k$

Alice starts by sending the digitally signed cipher $c = E_k(m)$ to Bob. In the second message Bob responds with the evidence of receipt for this cipher (EOR). If Alice does not receive the second transmission she stops the protocol, otherwise she sends the signed decryption key k to Bob. Bob answers by sending the receipt EOR_k for the key. The label l is present in each transmission in order to identify the protocol run. The time-out t specified in message 1 is used in the recovery protocol. Alice may initiate the recovery protocol if Bob does not send the receipt for the key. The steps of the recovery protocol are the following:

1. $A \rightarrow TTP : f_{Sub_k}, l, t, k, Sub_k$
2. $B \leftrightarrow TTP : f_{Con_k}, A, B, l, t, k, Con_k$
3. $A \leftrightarrow TTP : f_{Con_k}, A, B, l, t, k, Con_k$

Alice sends the signed key, together with the deadline t to the TTP. If the key arrives after t, the TTP does not accept the recovery. Otherwise the TTP publishes the key together with a confirmation Con_k for the key in a read-only accessible directory, where both Alice and Bob can fetch the key as well as Con_k. Con_k serves to Bob as the evidence of origin of the key, and to Alice as the evidence of receipt of the key as it is accessible to Bob. The deadline t is necessary for Bob to know the moment when either the key is published or will not be published anymore. A more detailed description can be found in [4].

Zhou makes the assumption that the channels between Alice and Bob are unreliable and that the channels between the TTP and both Alice and Bob are resilient.

There is however one scenario where fairness is lost. Imagine that Alice sends message 3 of the main protocol. Bob now possesses a complete evidence of origin and may stop the protocol. In that case Alice has to initiate the recovery protocol. However with resilient channels, Alice does not know the time it will take for message 1 of the recovery protocol to arrive at the TTP. As messages on a resilient channel can be delayed, message 1 may arrive after time t and hence be refused by the TTP. At the end of this scenario Bob possesses a complete non-repudiation of origin evidence, while Alice does not have the corresponding non-repudiation of receipt evidence: fairness is broken. Zhou [4] proposes to choose a big enough t to react to this problem. However on a resilient channel it is not possible to estimate the transmission time of a message. So it is impossible to set a lower bound for t.

AN OPTIMISTIC NON-REPUDIATION PROTOCOL

We shall now have a look at the modifications we have to insert in order to keep a low weight TTP (a TTP that only needs to maintain a public read accessible directory), without the need of an operational channel. The channels between the TTP and respectively Alice or Bob only need to be resilient and the channel between Alice and Bob may be unreliable.

Only the recovery protocol needs to be changed. If, for any reason, the main protocol of Zhou fails, Alice initiates the recovery protocol. The TTP verifies the request for recovery and the validity of the signature, as well as the time limit t. If the recovery request arrives before t (with respect to the clock of the TTP), the TTP continues the protocol as it has been proposed by Zhou. However if the recovery request arrives after t, the TTP revokes the public signature keys of both Alice and Bob. Therefore the keys are inserted in a certificate revocation list (CRL). Thus, each time the non-repudiation evidences are used, the CRL has to be checked to be sure that the keys are still valid.

Now, if Bob does not reply after having received the third message or if his reply is lost, it is not anymore crucial for Alice to contact the TTP before time t. If the message arrives at the TTP before time t the recovery is performed and both Alice and Bob receive their evidences. Otherwise, if the recovery request arrives after t, the evidences are revoked by revoking the public signature keys.

This solution does only need a passive TTP (the TTP does not send messages but only maintains a public read-only accessible directory). However, we remark the following weaknesses. In the case Bob does not send a receipt for the key, he can use his non-repudiation of origin evidence during the time interval between t and the key revocation. This results in a temporary unfair situation. On the other hand Alice has an advantageous position with respect to Bob: if the main protocol is executed without intervention of the TTP, Alice may revoke the evidences later by the mean of the recovery protocol.

We shall now propose a protocol that does not suffer from the here outlined drawbacks. Our work independently ends up to a similar protocol presented by Zhou et al. in [3]. The protocol is based on an active TTP (the TTP needs to send messages to Alice and Bob via resilient channels, but does not need to maintain a public directory accessible via ftp).

The following notation is used to describe the protocol:

- $l = h(m, k)$
- $EOO = S_A(f_{EOO}, B, l, h(c))$
- $EOR = S_B(f_{EOR}, A, l, h(c))$
- $Sub = S_A(f_{Sub}, B, l, E_{TTP}(k))$
- $EOO_k = S_A(f_{EOO_k}, B, l, k)$
- $EOR_k = S_B(f_{EOR_k}, A, l, k)$
- $Rec_X = S_X(f_{Rec_X}, Y, l)$
- $Con_k = S_{TTP}(f_{Con_k}, A, B, l, k)$
- $Abort = S_A(f_{Abort}, B, l)$
- $Con_a = S_{TTP}(f_{Con_a}, A, B, l)$

Main protocol

1. $A \rightarrow B$: $f_{EOO}, f_{Sub}, B, l, c, E_{TTP}(k), EOO, Sub$
2. $B \rightarrow A$: f_{EOR}, A, l, EOR (time-out: abort)
3. $A \rightarrow B$: $f_{EOO_k}, B, l, k, EOO_k$ (time-out: recovery$[X := B, Y := A]$)
4. $B \rightarrow A$: f_{EOR_k}, A, l, EOR_k (time-out: recovery$[X := A, Y := B]$)

Alice starts the protocol by sending the cipher of the message, as well as the decryption key, ciphered under the public key of the TTP, to Bob. The message does also contain Alice's signature on the encrypted key and the hash of the cipher. This signatures serves as a non-repudiation of origin evidence of these ciphers.

If Bob receives the first message he replies with a receipt to confirm that he got the first message. This receipt contains Bob's signature on the hash of the cipher c and serves to Alice as a non-repudiation of receipt evidence of the cipher.

In the case that Alice does not receive message 2 from Bob before a given time-out, she initiates the abort protocol. Note that Alice cannot perform a recovery at this moment, as the recovery protocol requires EOR, the evidence of receipt for the cipher. If message 2 arrives to Alice before the time-out, she sends to Bob the decryption key k, as well as a signature on this key. This signature is used as the non-repudiation of origin evidence of the key. The non-repudiation of origin message of the cipher c, together with the non-repudiation of origin evidence of the key k, form together the non-repudiation of origin evidence of the message m.

Message 3 has to arrive to Bob before a given time-out. Otherwise Bob initiates the recovery protocol with the TTP. If message 3 arrives in time, Bob sends a receipt for the key to Alice: his signature on the key k. The signature serves as the evidence of receipt of the key. Together with the evidence of receipt of the cipher c, they form the non-repudiation of receipt evidence of the message m. Alice may also initiate the recovery protocol with the TTP if this last message does not arrive in time.

Abort protocol

Alice has the possibility to run an abort protocol, if she does not receive message 2. If she decides to do so she sends a signed abort request, including label l, to the TTP. If the TTP accepts the request (neither a recovery nor an abort has yet been initiated), the TTP sends to both Alice and Bob a signed abort confirmation.

```
if recovery or abort then stop
  abort=true
  1. $A \rightarrow TTP : f_{Abort}, l, B, Abort$
  2. $TTP \rightarrow A : f_{Con_a}, A, B, l, Con_a$
  3. $TTP \rightarrow B : f_{Con_a}, A, B, l, Con_a$
```

Recovery protocol

To launch the recovery protocol Alice or Bob has to send to the TTP the hash of c, the key k ciphered for the TTP, the non-repudiation of origin evidence for the cipher c EOO, the non-repudiation of origin evidence for the encrypted key Sub, the non-repudiation of receipt evidence for the cipher c EOR, as well as the non-repudiation of origin evidence of the recovery request Rec_X (where X may take the values A or B). Note that the recovery protocol can only be executed once per protocol run.

By the mean of these evidences the TTP can be sure that Alice sent the cipher c to Bob and that Bob really received it. Moreover the party X, that initiates the protocol, cannot indicate a wrong identity for Y, the second participating entity, as the TTP verifies signatures of both entities. Otherwise, X could give any wrong identity making it impossible to Y to get the evidences as the recovery may not be run a second time.

```
  1. $X \rightarrow TTP :$   $f_{Rec_X}, f_{Sub}, Y, l, h(c), E_{TTP}(k), Rec_X, Sub, EOR, EOO$
if abort then
  2a. $TTP \rightarrow X : f_{Con_a}, A, B, l, Con_a$
```

else if recovery then stop
else recovery=true

2b. $TTP \rightarrow A: \quad f_{Con_k}, A, B, l, k, Con_k, EOR$

3. $TTP \rightarrow B: \quad f_{Con_k}, A, B, l, k, Con_k$

If the first message arrives and the TTP accepts to perform a recovery protocol, the TTP sends to Alice the confirmation of submission of the key, as well as the non-repudiation of receipt evidence for the cipher EOR. It is important to include EOR, as Bob can initiate the recovery protocol after having received the cipher, without having sent a receipt for it. The TTP sends to Bob the key k, as well as the confirmation of the submission of the key, serving to Bob as an evidence of origin for k.

If the recovery protocol is executed, the key confirmation evidence Con_k will make part of the non-repudiation evidences for the message m. It is used to replace both the non-repudiation of origin evidence for the key as well as the non-repudiation of receipt evidence for the key.

Dispute resolution

When Alice denies the origin of the message, Bob has to present to the judge EOO, EOO_k or Con_k, l, c, m and k. The judge verifies that

- $EOO = S_A(f_{EOO}, B, l, c)$,
- $EOO_k = S_A(f_{EOO_k}, B, l, k)$ or $Con_k = S_{TTP}(f_{Con_k}, A, B, l, k)$,
- $l = h(m, k)$,
- $c = E_k(m)$.

If Bob can provide all the required items and all the checks hold, the adjudicator claims that Alice is at the origin of the message. When Bob denies receipt of m, Alice can prove his receipt of the message by presenting EOR, EOR_k or Con_k, l, c, m and k to a judge. The judge verifies that

- $EOR = S_B(f_{EOR}, A, l)$,
- $EOR_k = S_B(f_{EOR_k}, A, l, k)$ or $Con_k = S_{TTP}(f_{Con_k}, A, B, l, k)$,
- $l = h(m, k)$,
- $c = E_k(m)$.

If Alice can present all of the items and all the checks hold, the adjudicator concludes that Bob received the message.

Fairness and timeliness

If Bob stops the protocol after having received the first message, Alice may perform the abort protocol, in order to avoid Bob to initiate a recovery later. As neither Bob nor Alice received complete evidences the protocol remains fair. If Bob had already initiated the recovery protocol, the TTP sends all the missing evidences to Alice and Bob. Note that the TTP also sends the EOR to Alice, as she has not received it yet. Thus the protocol stays fair.

If Alice does perform step 3, Bob receives a complete non-repudiation of origin evidence. There are two ways to finish the protocol: Bob sends message 4 of the main protocol and Alice receives a complete non-repudiation of receipt evidence or Alice performs the recovery protocol. As the channels between the TTP and both Alice and Bob are resilient, all data sent by the TTP to Alice and Bob eventually arrive. In both cases all entities receive valid evidences and the protocol finishes providing fairness. If Alice does not send message 3 during the main protocol, Alice and Bob may initiate the recovery protocol. Fairness is still guaranteed, as during the recovery protocol, Alice and Bob receive all expected evidences.

When looking at the timeliness, three situations may arrive: the main protocol ends up successfully (without any time-out); Alice aborts the protocol and the abort confirmation signed by the TTP arrives at Alice and Bob after a finite amount of time, as the channels between the TTP and both Alice and Bob are resilient; a recovery protocol is performed and Alice and Bob receive the evidences after a finite amount of time because of the resilience of the channels.

CONCLUSION

In this paper, we point out some weaknesses of the optimistic version of the Zhou-Gollman protocol. We then show the problems that are remaining when trying to keep a low-weight offline TTP. Therefore we present a protocol, that combines an active, offline TTP with a resilient channel. Note that this is the first optimistic non-repudiation protocol that succeeds to guarantee fairness and timeliness assuming that channels are resilient.

REFERENCES

[1] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.

[2] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography.* CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 0-8493-8523-7.

[3] J. Zhou, R. Deng, and F. Bao. Evolution of fair non-repudiation with TTP. In *ACISP: Information Security and Privacy: Australasian Conference*, 1999.

[4] J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop.* IEEE Computer Society Press, 1997.