

On the Security of Fair Non-repudiation Protocols

Information Security Conference 2003

Sigrig Gürgens¹⁾, Carsten Rudolph¹⁾, and Holger Vogt²⁾

1) Fraunhofer – Institute for Secure Telecooperation SIT

2) SRC Security Research & Consulting GmbH

02.10.2003

Outline

- Fair non-repudiation protocols
- New attacks on non-repudiation protocols
- Design principles for secure fair exchange protocols
- Improved asynchronous optimistic fair non-repudiation protocol

Non-repudiation

- Party A wants to transfer a message M to party B in a verifiable way
- A and B want non-repudiation tokens to prove certain facts about the message transfer
- Non-repudiation protocols provide:

EOO – Evidence of origin for B:

Proves that A has sent message M

EOR – Evidence of receipt for A:

Proves that B has received message M

Fair Non-repudiation

Party A

Party B

Message M,
Evidence of origin EOO



Evidence of receipt EOR



Fairness: Either A and B get the other one's items or none of them gets anything

Security Goals of a Fair Non-repudiation Protocol

Fair non-repudiation of receipt:

When A completes the protocol,

- either A owns a valid EOR created by B for M
- or B has neither M nor a valid EOO.

Fair non-repudiation of origin:

When B completes the protocol,

- either B owns M and a valid EOO created by A
- or A has no valid EOR.

Timeliness for A and B:

Each party can complete the protocol.

Asynchronous Optimistic Fair Non-repudiation Protocols

trustee participation in every exchange?

yes

no

problems with
scalability and availability

optimistic protocols

time limit for message delays required?
(= synchronous communication)

yes

no

difficult to guarantee
this assumption in practice

Asynchronous
optimistic protocols

Current Solutions for Asynchronous Optimistic Fair Non-repudiation

- Efficient fair non-repudiation-protocols:
 - KM by Kremer, Markowitch (2000)
 - ZDB by Zhou, Deng, Bao (1999) with an enhancement by Boyd, Kearney (2000)

Attacks on fairness and timeliness are possible!

- Finite state security analysis using the SH Verification Tool

Implementation of Current Fair Non-repudiation Protocols

- Party A encrypts
 - M with random symmetric key K resulting in $C=e_K(M)$
 - K for the TTP resulting in $EK=E_{TTP}(K)$
- Chiphertexts are sent first, then follows the exchange of M and EOO for EOR
 - All messages contain a label L chosen by A
- Conflict resolution with the TTP
 - Abort sub-protocol for A (exchange will not be finished)
 - Resolve sub-protocols for A and B (TTP provides missing information)

Attacks on Fair Non-repudiation Protocols

Attack 1: Reuse of encrypted key $EK = E_{TTP}(K)$

- Attack on the fairness of the KM protocol
- B can request decryption from the TTP in a different context
- B gets the message, but A does not receive EOR
- Label L does not link all messages in KM

Attacks on Fair Non-repudiation Protocols

Attack 2: Sending wrong encrypted key $EK \neq E_{TTP}(K)$

- Attack on the timeliness of the ZDB protocol
- A sends wrong EK to B
- B cannot prove this misbehavior of A
- B cannot complete the protocol
- A can complete the protocol using the correct EK

Attacks on Fair Non-repudiation Protocols

Attack 3: Restarting a protocol run

- Attack on the fairness of the KM and ZDB protocol
- Evidence of receipt EOR of B consists of two parts
 - In a first unfinished protocol run A receives one part of EOR
 - Later A restarts the protocol with B to receive the second part of EOR
 - A gains valid EOR
- A misbehaves to prevent any conflict resolution for B

Reasons for Attacks

- Protocols have several small design weaknesses
 - A single weakness may not be exploitable
 - Considering several weaknesses can lead to successful attacks

Enhance the security of all protocol parts!

- Specific design principles as guide to a development of secure fair exchange protocols

Unique Labels

- Labels identify messages that belong to the same exchange transaction
 - choosing a random number is not sufficient ...

Verifiability: Anyone can check label creation

Uniqueness: Same label implies same exchange

Secrecy: Label reveals nothing about the exchanged information

Example: $L = H(A, B, TTP, H(C), H(K))$ with $C = e_K(M)$

Construction of Messages

- Manipulation of messages must be detectable or even provable

Authenticity: Complete message should be signed

Verifiability: Receiver can check the validity of every message

Context: Message can only be accepted in one context

Example: $\text{message} = [\text{flag}, L, EK, \text{sig}_A(\text{flag}, L, EK)]$

TTP Actions

- TTP behavior has to be fully specified

Meaningful TTP decisions:

Resolve/abort decisions should be followed by according actions

Reply every request:

TTP answers every request and thus informs about its current state

Example: Before a resolve the TTP checks all items and then remembers its answer

Improved Asynchronous Optimistic Fair Non-repudiation Protocol

- Guidelines are applied to KM and ZDB
 - Improve the security of many protocol parts
- Resulting protocol is immune to known attacks
 - No attacks found with finite state security analysis using the SH Verification Tool

Improved Definition of Messages

- some unique flags $f_K, \dots, f_{\text{aborted}}$
 - K: symmetric key
 - $C = e_K(M)$ encrypted message
 - $L = H(A, B, TTP, H(C), H(K))$
 - $EK = E_{TTP}(f_K, L, K)$ encrypted key
 - $EOO_C = \text{sig}_A(f_{EOO_C}, L, EK)$
 - $EOR_C = \text{sig}_B(f_{EOR_C}, L, EK)$
 - $EOO = \text{sig}_A(f_{EOO}, L, K)$
 - $EOR = \text{sig}_B(f_{EOR}, L, K)$
 - $\text{con} = \text{sig}_{TTP}(f_{\text{con}}, L, K)$
 - $\text{sig}_A(f_{\text{abort}}, L)$ abort request
 - $\text{aborted} = \text{sig}_{TTP}(f_{\text{aborted}}, L)$
- Evidence for a successful exchange

The Fair Non-repudiation Protocol

Party A

wants to send message M to B

Party B

$A, B, TTP, C, H(K), EK, EOO_C$

Computes L and
checks EEO_C

(if not OK, quit protocol)

EOR_C

Verify EOR_C

(if not OK, start abort)

K, EOO

Evidence of origin valid
for message $M = d_K(C)$?

(if no, start resolve)

EOR

Evidence of receipt valid
for message M?

(if no, start resolve)

Abort Sub-Protocol

Party A

TTP

wants to abort the exchange

$A, B, H(C), H(K), \text{sig}_A(f_{\text{abort}}, L)$



```
if request is invalid
  response="Error"
else if exchange resolved
  response=con
else
  response=aborted
```

response



Resolve Sub-Protocol

Party P

wants to finish the exchange

$A, B, H(C), H(K), EK, EOO_C, EOR_C$



TTP

if request is invalid
response="Error"

else if exchange aborted by A
response=aborted

else if already resolved
or decrypting EK succeeds
response=K,con

else decrypting EK fails [A cheated]
response=aborted

response



Properties of the Protocol

- Fairness for both parties
 - either both get non-repudiation tokens for the message M or none of them receives anything
- Timeliness for both parties
 - Conflict resolution with the TTP ensures protocol completion
- Efficient
 - Only one signature verification to check the evidence
 - One signature per message in the main protocol
- Simple protocol, clear semantics, complete specification
- Extensible (e.g. message confidentiality)

Conclusion

- 1) Analysed two asynchronous optimistic fair non-repudiation protocols
 - 3 different attacks found
 - Exploiting several small weaknesses
- 2) Proposed guidelines for improving the security of certain protocol parts
 - Improved construction of labels, messages and the TTP
- 3) Developed new protocol for fair non-repudiation
 - Overcomes weaknesses of previous protocols
 - Finite state security analysis using the SH Verification Tool