Probabilistic Non-Repudiation without Trusted Third Party

Olivier Markowitch and Yves Roggeman

Université Libre de Bruxelles
Département d'Informatique
Boulevard du Triomphe - CP 212
1050 Bruxelles
Belgium
omarkow@ulb.ac.be - yrogge@ulb.ac.be

Abstract. We present a new theoretical generic protocol that offers non-repudiation services. We first go through the state of the art about non-repudiation. We will see that in order to achieve complete non-repudiation during an information transfer a trusted third party is often needed (even playing a low weight role), obtaining an acknowledgement from the recipient being the current major problem. We aim to show the possibility of achieving fair non-repudiation services without the need of a third party. We will propose a concrete implementation inspired by this generic protocol.

1 Introduction

Usually, when a person (a client) requests a service, he asks the salesman (the provider) for the required service. The provider supplies the service. Eventually the client pays. This protocol is as the following:

1. Client → Provider: Request for a service
2. Provider → Client: Service
3. Client → Provider: Payment (acknowledgement)

Although this kind of exchange does apply to human relationships, many problems appear when mapped to the "electronic world".

In a computer network, the payment could consist in sending an acknowledgement ("ack") considered as a proof of payment. This acknowledgement is therefore the proof, for the service provider, that the client has received the service and a guaranty that the client will pay.

Unfortunately, in such protocols, the client is not compelled to send the acknowledgement after receiving the service. The client could get a service and the provider will be unable to obtain a payment for it. It is said that the client repudiates the reception of the service.

Another problem arises when the client receives the service or when the provider receives the acknowledgement from the client. Each of them wants to be sure that the message is actually from the expected party. Moreover, each of them wants to be assured that nobody else can falsely allege having received a message from them.

Solutions have been proposed to avoid these problems. These protocols are said to offer *non-repudiation* services.

In this paper, we will use the inherent difference between the originator of a requested message and the recipient to settle a protocol that implements the non-repudiation services without the need of a third party.

The term "third party" designates an on-line third party and an in-line third party as described in [25], as well as an off-line third party as defined in [7]. However, an adjudicator might be needed as we will see later.

This paper is organized as follow. In section 2, we define the different non-repudiation services and investigate the previous results in this area. In section 3, we define our probabilistic fairness. A new generic protocol is then presented providing probabilistic non-repudiation services without the need of a third party. In the 4^{th} section we present an instance protocol based on the generic model. In section 5, we study the time performance of the protocol.

2 Non-repudiation services

As described in [24, 15, 9], repudiation is defined as "denial by one of the entities involved in a communication of having participated in all or part of the communication".

Many types of repudiations are relevant to electronic transaction protocols. At least, two services [24, 9] are needed to obtain a complete non-repudiation protocol as described below.

In the following definitions, we use originator to identify the entity sending a message to another entity named recipient.

Definition 1 *Non-repudiation of origin*: to prevent the originator of a message from denying having sent the message.

Definition 2 *Non-repudiation of receipt*: to prevent the recipient of a message from denying having received the message.

In many proposals, to achieve a complete non-repudiation, a third party is involved as a delivery authority. In this case, two other services can be supplied.

Definition 3 *Non-repudiation of submission*: to ensure that the message is submitted for delivery by the originator.

Definition 4 *Non-repudiation of delivery*: to ensure that the message has been delivered to the recipient.

According to these definitions, one can easily see that the protocol described in the introduction does not support the non-repudiation of receipt. Moreover, in that protocol, each party has no concrete proof about the originator's identity.

Among other solutions, a digital signature makes possible the non-repudiation of origin (in order to establish an association between the message and the originator's identity).

The problem of obtaining non-repudiation of receipt service is similar to the problem of fair exchange of a message (sent by the originator) for an acknowledgement (sent by the recipient).

During the protocol in addition to the message, a non-repudiation of origin token (NRO) and a non-repudiation of receipt token (NRR) has to be transmited as evidences usable to resolve possible future disputes (by presenting these evidences to an adjudicator).

The items expected by the recipient are the message and the NRO. The item expected by the originator is the NRR.

We can now define the necessary properties of a non-repudiation protocol. A non-repudiation protocol has to be:

Definition 5 *Fair*: at each step of the protocol run, either both parties receive their expected items, or none of them has received any valuable information about their expected items,

Definition 6 *Time-bounded*: if at least one party behaves correctly then the protocol will be completed before a finite amount of time.

Another property could be interesting but is not required (its absence does not interfere with the fairness): the non-repudiation protocol could be:

Definition 7 *Viable*: if both parties behave correctly and finish the protocol then they receive their expected items at the end of the protocol.

We suppose, prior to the realization of the protocol, that a key exchange has already occurred, providing also identification

The non-repudiation problem has been studied in ISO/IEC 13888, resulting in three models (M1, M2 and M3). These models make use of a trusted third party. But, as shown in [24], these different models do not implement correctly the non-repudiation of receipt.

It is possible to design protocols involving no third party in fault-less scenarios, as in [25, 3, 2, 4, 5, 7]. For example, assume the situation where the originator-recipient communications are performed without a trusted third party. After receiving the recipient's request, the originator sends the message. The recipient may not confirm the reception (i.e. does not send the NRR). In such a case, the originator asks for the assistance of a third party who will "oblige" the recipient to finalize the protocol. In [25] these methods are studied and a comparison between the works of Asokan et al. and the works of Zhou and Gollman are

made. This kind of protocols is efficient when most parties do not attempt to cheat. A third party is involved as soon as a problem occurs [25, 3, 2, 4, 5, 7].

In this paper, no assumptions are made about the behaviour of the parties. Furthermore, in any case the message is only sent once.

In [14] is proposed a protocol without a trusted third party but using specific secure public hardware (the pub). The pub records, as a notary, every party's operations when transmitting the decoding key.

Our protocol does not use such hardware (on which, as for trusted third parties, the two parties has to be blindly confident. The risk can not be parameterized).

Recently, in [20] is proposed a protocol resolving the exchange problem, and then some aspects of the non-repudiation problem, without a trusted third party. The protocol uses "weakly secret bit commitments" where every commited message is breakable within a "reasonable" quantity of time. During this iterative protocol each party sends a message which is breakable faster than the previous one. If we consider the problem of paid services, the author presents an interesting protocol for low value services or for short time value services. The differences with our protocol are that the Syverson's protocols have to be parameterized with respect to all participating parties computing power, moreover the exchange protocols do not feature fairness (when the first message is sent, the recipient can recover the original item, without emitting anything, but within an amount of time which is "higher" than the item's value).

3 Probabilistic non-repudiation

To achieve non-repudiation, a trusted third party is often needed, we will show a way to obtain non-repudiation without a trusted third party. We aim here at building a two party generic protocol where the fairness could be obtained with a certain probability.

At each step of our protocol, except the last one, no party can take advantage over the other party. As we will see later, the originator has no interest in stopping the protocol before the end. If the recipient stops the iterative protocol before the last step, his gain is null. The only way for the recipient to receive the message without sending the NRR is to guess the number of protocol steps. The number of steps of the protocol is chosen secretly by the originator. At each step the probability for the recipient to get the message without sending the NRR is less or equal to a quantity denoted by ε.

The iterative protocol we are presenting is :

Definition 8 *ε-fair*: at each step of the protocol run, either both parties receive their expected items, or the probability that a cheating party gains any valuable information of his expected items, while the other party gains nothing, is $\leq \varepsilon$ (with $\varepsilon \in [0, 1]$),

Time-bounded (as defined before),

and in some cases the protocol is :

Definition 9 *Viable*: if both parties follow properly the protocol, then at its end, the exchange of both message (and NRO) and acknowledgement (NRR) succeeds with probability equal to 1.

To ensure non-repudiation without a third party, we have to design a protocol where the recipient does not know when the last transmission happens. Otherwise, it is always conceivable that he repudiates having received the message i.e. refuses to send the NRR.

We can achieve this by using an iterative protocol. The originator (O) chooses randomly, according to a geometric distribution for example, a number n of steps for the protocol. This information is never unveiled to the recipient (R) during the protocol run, neither can it be computed by the recipient.

The generic protocol is defined as follow:

The recipient determines the date D

1. $\mathrm{R} \rightarrow \mathrm{O} : \mathrm{Sign}_{\mathrm{R}}(\text{request}, R, O, D)$

The originator: checks D
chooses n
computes the signed $f_1, \ldots, f_n$

2. $\mathrm{O} \rightarrow \mathrm{R} : \mathrm{Sign}_{\mathrm{O}}(f_n(\text{message}), O, R, D)$
3. $\mathrm{R} \rightarrow \mathrm{O} : \mathrm{Sign}_{\mathrm{R}}(\text{ack}_1)$

$\vdots$

$2n$. $\mathrm{O} \rightarrow \mathrm{R} : \mathrm{Sign}_{\mathrm{O}}(f_1(\text{message}), O, R, D)$
$2n{+}1$. $\mathrm{R} \rightarrow \mathrm{O} : \mathrm{Sign}_{\mathrm{R}}(\text{ack}_n)$

The functions f_i are parts of a function composition. The required message is that which results of this composition.

We have: $f_n(\text{message}) \circ f_{n-1}(\text{message}) \circ \ldots \circ f_1(\text{message}) = \text{message}$.

In this case it is essential to use a composition operator which is not commutative. Since the originator sends the functions in reverse order (from f_n to f_1), if the composition operator does not permit the commutativity, then the recipient is not able to use previous results in order to speed up the computations. Since f_i is applied on f_{i-1}, we need all functions to compute the whole composition.

It is also recommended that the function result does not have a size comparable to the message size. This is motivated by the fact that after the transmissions, the quantity of information sent may not be too important (with respect to the message size).

The non-repudiation evidences are:

- $\mathrm{NRO} = \{\mathrm{NRO}_i \mid i = 1, \ldots, n\}$, with $\mathrm{NRO}_i = \mathrm{Sign}_{\mathrm{O}}(f_i(\text{message}), O, R, D)$
- $\mathrm{NRR} = \mathrm{Sign}_{\mathrm{R}}(\text{ack}_n)$

In our protocol, if the recipient does not immediately send the expected acknowledgement after receiving a function, then the originator states that he is trying to cheat. The protocol is stopped by not sending the next function (therefore, the recipient does not receive the message).

We have to choose functions in such a way that the composition computation takes more time than the transmission of an acknowledgement. We have to ensure that the recipient cannot compute the composition between two transmissions.

We must determine the deadlines after which the parties decide to stop the protocol. We propose two different ways to manage this problem.

Firstly, a deadline, publicly known, can be used by the parties. At each step of the protocol, each party waits, at most, until the deadline expires. After the deadline the concerned party supposes that the other party cheats or that the network is overloaded, and stops the protocol.

The second solution could be that we assume the use of an operational channel between the parties. A channel between two correctly behaving players is *operational* if the messages inserted into it by one party are received, while respecting the sequencing, by the other party within a known and constant time interval [1]. On the one hand, if the recipient does not receive a new function within a given delay, he knows that the protocol ended. On the other hand, if the originator notices that the expected acknowledgement does not arrive after the fixed time interval, he states that the recipient tries to cheat by computing the composition of received functions instead of sending his acknowledgement. As a result, the originator stops the protocol.

The protocol ends up when the recipient sends to the originator the acknowledgement corresponding to the last function (the NRR). An "end of protocol" message is not mandatory (indeed, after not receiving further functions, the recipient is aware of the protocol state, he is then able to compute the message by the composition).

The i^{th} acknowledgement message carries the following semantics: "R acknowledges having received message i from O". In order to protect the recipient against attacks by replay, each message conveys a time stamp. The replay attack may consist in having the originator using old acknowledgements used during a previous communication with the recipient. The value of D is sent by the recipient with the first message. This time D is not a synchronization information, but can be assimilated to a random value. Then we have: $\text{ack}_i = (i, R, O, D)$.

Since the recipient does not know n, he is not able to determine when he will receive the last function. This condition can be met as long as all messages sent by the originator have the same structure. Moreover, the underlying technology used to carry all messages may not give any hints about the protocol state.

The probability that the recipient does not send the acknowledgement and tries to compute, precisely at the last transmission, the composition (without being sure it is the last transmission) is according to the geometric distribution used to

choose n: θ, and then the originator sends all the needed information to compute the message although the recipient does not send the NRR.

The range of values used by the originator has to be chosen with respect to the importance of the message. Indeed, the probability that the recipient receives the message for free depends on this range. This range can be parameterized by an adequate choice of the success parameter θ of the proposed geometric distribution.

Using this geometric distribution with a success parameter θ, the proposed protocol is:

ε**-fair**:

- if the originator behaves correctly, the recipient:
  - can decide to stop the protocol before the n^{th} step, then neither the originator nor the recipient have their expected items. The protocol remains fair,
  - can want to stop the protocol after the n^{th} step and then sends the NRR and receives the message and the NRO. The protocol is fair,
  - can stop the protocol at the n^{th} step and receives the message and the NRO but does not send the NRR. The protocol is then unfair and this situation happens with a probability $= \theta = \varepsilon$

  The protocol is θ-fair for the originator.
- if the recipient behaves correctly, the originator obtains the NRR only if he has already send the message and the NRO. The exchange is then complete and the protocol is fair for the recipient with a probability $= 1 \geq 1 - \theta$.

The protocol is ε-fair with $\varepsilon = \theta$. □

Time bounded: with an operational channel or with the use of deadlines, each transmited information will be either received before a fixed and finite amount of time and then the protocol continues, possibly towards his normal end, or if at least one party behaves correctly then when the delay is exceeded the protocol is stopped. □

Viable: with an operational channel, as no party wants to give up the protocol, the expected information are sent and received at each protocol's step until the messagse (and the NRO) and the NRR are exchanged (with a probability $= 1$). □

By using the deadlines rather than the operational channel, we do not obtain the viability property. Even if the emitting party is honest, the information sent are not assured of being received. The message are not ensured to be received within the deadlines and the protocol could be stopped because of the network (without succeeding in carrying out the exchange).

As the probability of losing the fairness is weak, but is not exponentially low, this protocol is suitable for low value messages (whatever the messages size).

Usually, adding a time constraint (deadlines) to a system implies the use of a trusted third party that records the emission of a message and/or distributes the

time. But in the present case the originator gains nothing by falsely pretending that the recipient sent the acknowledgment too late. When receiving each transmission, both the recipient and the originator check the delay elapsed to receive the message. If the originator detects a significant delay the protocol is canceled.

This process is valid because the originator has the NRR only if he posses the last recipient's acknowledgement. Since each signed acknowledgement contains a sequence number, the originator cannot use duplicated acknowledgments.

If the originator stops the protocol before the n^{th} step and claim that the protocol is completed, then he has to present the NRR to an authority (an adjudicator). Considering the number of steps implicitly announced by the originator (by the sequence number present in the NRR) and the recipient's received functions, the authority will confirm the message cannot be computed. Unlike [25, 3, 2, 4, 5], where a third party is referred to as soon a problem occurs, the adjudicator is used only in case of disputes (and never during the exchanges). The adjudicator, while he is not involved in the protocol, determines, if consulted, which entity is honest. There are no obligations to require an adjudicator if the protocol does not succeed.

4 Concrete Proposals

In this section, we propose a protocol offering non-repudiation messages based on the generic protocol presented in the previous section.

The protocol is inspired by the generic one in the following way: f_1 = ciphering function using a chosen key k, f_2 until f_n are random generators (which produce random numbers having the same size as the key k), and f_{n+1} return the key k in clear.

The concrete protocol is then:

The Recipient determines the date D

1. $\mathtt{R} \rightarrow \mathtt{O} : \mathtt{Sign_R(request,R,O,D)}$

The originator checks D, chooses n, computes $n-1$ random values, chooses a ciphering key k and computes $s = \text{ciphering}(\text{message}, k)$

2. $\mathtt{O} \rightarrow \mathtt{R} : \mathtt{Sign_O}(s, \mathtt{O,R,D})$
3. $\mathtt{R} \rightarrow \mathtt{O} : \mathtt{Sign_R}(\mathtt{ack}_1)$
4. $\mathtt{O} \rightarrow \mathtt{R} : \mathtt{Sign_O}(\mathtt{random}_1, \mathtt{O,R,D})$
5. $\mathtt{R} \rightarrow \mathtt{O} : \mathtt{Sign_R}(\mathtt{ack}_2)$

$\vdots$

$2n$. $\mathtt{O} \rightarrow \mathtt{R} : \mathtt{Sign_O}(\mathtt{random}_{n-1}, \mathtt{O,R,D})$

$2n$+1. $\mathtt{R} \rightarrow \mathtt{O} : \mathtt{Sign_R}(\mathtt{ack}_n)$

$2n$+2. $\mathtt{O} \rightarrow \mathtt{R} : \mathtt{Sign_O}(k, \mathtt{O,R,D})$

$2n$+3. $\mathtt{R} \rightarrow \mathtt{O} : \mathtt{Sign_R}(\mathtt{ack}_{n+1})$

Then the recipient computes: message = deciphering(s, k).

The non-repudiation evidences are:

- NRO = $\{\text{Sign}_{\text{O}}(s, O, R, D), \text{Sign}_{\text{O}}(k, O, R, D)\}$
- NRR = $\text{Sign}_{\text{R}}(\text{ack}_{n+1})$

The acknowledgements have the same form as those described in the section 3.

In this protocol, before the last transmission the recipient does not receive anything significant. The recipient cannot be aware he receives the actual key during the execution of the protocol, the time needed to verify a key (by deciphering all the message) is too long in comparison with the time expected to send the ack (choosing a cryptosystem suited in term of performance, with the message size).

In this concrete protocol, the recipient has to receive all messages sent by the originator in order to reconstruct the original message and to get the NRO. The originator has to collect the $n + 1$ acknowledgements to obtain the NRR..

The originator has to cipher using a mode where all the blocks are interdependent (whatever the cryptosystem used) in order to prevent that partial decipherings can contribute to the decision-making. For example we can have the message made up of t blocks m_i, and $m'_i = m_i \oplus m_t \ \forall i \in [1, t-1]$, $m'_t = m_1 \oplus \ldots \oplus m_{t-1} \oplus m_t$ if t is odd and $m'_t = m_1 \oplus \ldots \oplus m_{t-1}$ if t is even. The recipient receives e_i = ciphering of m'_i under the key $k \ \forall i \in [1, t]$. To retrieve the original message the recipient has to decipher all the e_i, with the m'_i obtained he computes $m_t = m'_1 \oplus \ldots \oplus m'_t$ and with m_t he retrieves $m_i = m'_i \oplus m_t \ \forall i \in [1, t-1]$.

5 Performances

In the proposed concrete protocol, we have to ensure that the computing time of the unique ciphering (and deciphering) is larger than the time needed to transfer an acknowledgement.

Suppose we send a 128 bytes random message in each transaction (after the first one). Assuming the deadlines are expressed in seconds or an operational channel guaranteeing a transfer of data in order of seconds, the choice of the cryptosystem has to be coherent with the message size. Suppose we have a n Kbytes message and a cryptosystem known to cipher x Kbit per second. Then if we want the computation does not take more than five minutes and less than twenty seconds (for example), we have to choose a cryptosystem where $x \in \left[\frac{2n}{75}, \frac{2n}{5}\right]$.

We can use, for example, RSA to cipher a 5 Mbytes message (the most efficient implementations of RSA propose a ciphering of 600 Kbit per second - using a 512 bit key). For a 50 Mbytes message, IDEA is adequate. Triple-DES could be used to cipher a 500 Mbytes message. DES is usable for a 1 Gbytes message, ...

In this performance evaluation, we suppose that the recipient has "precomputed" some signed acknowledgements. Therefore, the time needed to compute the acknowledgements may not have to be taken into account in the protocol run

duration. This is valid since the recipient can compute the signature on the ack and the date D as soon as D is known. This computation can be made in parallel with the first steps of the protocol.

6 Conclusion

We have presented a generic probabilistic protocol providing non-repudiation services without a trusted third party in the context of a originator-recipient dialog. This iterative protocol is based on the originator's secret knowledge of the number of steps of the protocol. Moreover, our protocol needs the use of deadlines at each transmission or the use of an operational channel between the parties. Then, we propose a concrete protocol based on the generic one. This iterative protocol realizes only one ciphering and uses a random value at each step. Finally, theorizing on the message size, on the cryptosystem and on the deadlines or channel quality of service, we have stressed the important difference that could occur between the transmission delay and the computation time. This lead to demonstrate the theorical feasibility of our system for low value messages.

This solution is proposed as an alternative to the use of a third party. Furthermore, no assumptions are made about the behaviour of the participants neither their computing power (the protocol being tuned according to the computing power of the recipient).

With this kind of methods, there are some communication and computation overheads. These overheads must be put against the potential bottleneck introduced by a third party in a framework where no assumptions are made about the behaviour of the parties. Moreover, in the presented concrete protocol, the computing overhead is reduced.

Our generic protocol features probabilistic fairness. With a probability $\geq 1 - \theta$ (where θ is the success parameter of a geometric distribution), we are sure that either the recipient receives the message and the originator is certain to obtain the NRR, or the originator will not receive the NRR hence the recipient does not receive the message.

The major goal is to be exempt from the need of a third party during the exchanges. If such a third party does not exist, the risk of bottleneck is avoided. Last but not least, assumptions about the third party reliability are not relevant. This property is highly desirable, since it is not easy to evaluate the security risk induced by a third party. In our system, the risk is known and could be parameterized by the originator (choosing θ).

7 Acknowledgements

The authors are grateful to J. Zhou, L. Franck and L. Demarque for their invaluable help.

References

1. N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, May 1998.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. Research Report RZ 2858 (#90806), IBM Research, Sept. 1996.
3. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *Proceedings of the fourh ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.
4. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. Research Report RZ 2976 (#93022), IBM Research, Nov. 1997.
5. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital SIgnatures. Research Report RZ 2973 (#93019), IBM Research, Nov. 1997.
6. A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 3–19. Internet Society, Feb. 1994.
7. F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line ttp. In *IEEE Symposium on Security and Privacy*, May 1998.
8. M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transaction on Information Theory*, 36(1):40–46, Jan. 1990.
9. A. Bosselaers and B. Preneel. *Integrity primitives for secure information systems: final RIPE report of RACE Integrity Primitives Evaluation (R1040)*, volume 1007 of *Lecture Notes in Computer Science*. Springer-Verlag, New York, NY, USA, 1995.
10. I. B. Damgard. Practical and provably secure release of a secret and exchange of signatures. In *Proceedings of Asiacrypt 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 133–144. Springer-Verlag, 1997.
11. R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic mail. *Journal of Network and System Management*, 4(3):279–297, 1996.
12. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.
13. O. Goldreich. A simple protocol for signing contracts. In D. Chaum, editor, *Advances in Cryptology: Proceedings of Crypto 83*, pages 133–136. Plenum Press, New York and London, 1984, 22–24 Aug. 1983.
14. Y. Han. Investigation of non-repudiation protocols. In *ACISP: Information Security and Privacy: Australasian Conference*, volume 1172 of *Lecture Notes in Computer Science*, pages 38–47. Springer-Verlag, 1996.
15. ISO, editor. *ISO 7498-2: Information processing system - Open Systems interconnection - Basic reference model, Part 2: Security architecture*. International Organisation for Standardization, 1989.
16. M. Jakobsson. Ripping coins for fair exchange. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology: Proceedings of Eurocrypt'95*, volume 921 of *Lecture Notes in Computer Science*, pages 220–230. Springer-Verlag, 21–25 May 1995.
17. T. Okamoto and K. Otha. How to simultaneously exchange secrets by general assumptions. In *Proceedings of the second ACM Conference on Computer and Communications Security*, pages 184–192, Zurich, Switzerland, 1994. ACM Press.
18. M. O. Rabin. Transaction protection by beacons. *Journal of Computer and System Sciences*, 27(2):256–267, Oct. 1983.

19. P. Syverson. A different look at secure distributed computation. In *Proceedings of The 10th Computer Security Foundations Workshop*, pages 109–115. IEEE Computer Society Press, June 1997.
20. P. Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *Proceedings of the 1998 IEEE Computer Security Foundations Workshop (CSFW11)*, june 1998.
21. T. Tedrick. How to exchange half a bit. In D. Chaum, editor, *Advances in Cryptology: Proceedings of Crypto 83*, pages 147–151. Plenum Press, New York and London, 1984, 22–24 Aug. 1983.
22. T. Tedrick. Fair exchange of secrets. In G. R. Blakley and D. C. Chaum, editors, *Advances in Cryptology: Proceedings of Crypto 84*, volume 196 of *Lecture Notes in Computer Science*, pages 434–438. Springer-Verlag, 1985.
23. J. Zhou. *Non-repudiation*. PhD thesis, University of London, Dec. 1996.
24. J. Zhou and D. Gollmann. Observations on non-repudiation. In *Proceedings of Asiacrypt 1996*, volume 1163 of *Lecture Notes in Computer Science*, pages 133–144. Springer-Verlag, 1996.
25. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *Proceedings of The 10th Computer Security Foundations Workshop*, pages 126–132. IEEE Computer Society Press, June 1997.
26. J. Zhou and D. Gollmann. Evidence and non-repudiation. *Journal of Network and Computer Applications*, 1998.