Some Guidelines for Non-repudiation Protocols

Panagiotis Louridas
38 Domboli St.
116 36 Athens
Greece
louridas@acm.org

ABSTRACT
Non-repudiation protocols aim at preventing parties in a communication from falsely denying having taken part in that communication; for example, a non-repudiation protocol for digital certified mail should ensure that neither the sender can deny sending the message, nor the receiver can deny receiving it. We identify some guidelines for non-repudiation protocols. The guidelines are derived by examining a series of non-repudiation protocols that descend from a single ancestor.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General—*security and protection (e.g., firewalls)*; K.4.4 [**Computers and Society**]: Electronic Commerce—*security*; K.5.m [**Legal Aspects of Computing**]: Miscellaneous; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms
Legal Aspects, Security, Verification

Keywords
Non-repudiation, Fair Exchange, Formal Verification

1. INTRODUCTION
When public key cryptography was first presented, it was pointed out that its techniques "also protect against the *threat of dispute*. That is, a message may be sent but later repudiated by either the transmitter or the receiver. Or, it may be alleged by either party that a message was sent when in fact none was" [21]. When the first practical implementation of public key cryptography was presented, it was observed that the capability to produce unforgeable digital signatures offers a step beyond privacy and authentication: "the recipient can convince a 'judge' that the signer sent the message" [28].

This is but an instance of a more general problem, that of *non-repudiation*: a party that has taken part in a communication should not be able to falsely deny the truth or validity of the communication or its parts. Unforgeable digital signatures are part of the solution; however, the problem is complicated when non-repudiation is, additionally, required to be *fair*.

Intuitively, a protocol is fair if it does not allow a party to gain an undue advantage. A simple exchange of signed items is not fair: it does not guard against threat of disputes, unless the signed items are exchanged simultaneously; otherwise the first recipient may never send her own item. Simultaneity, however, is hard to achieve when the parties do not interact face to face.

If fairness is ensured completely within the system specified by the protocol it is *strong*; if fairness is ensured within the system but with the proviso that additional assumptions about the participating parties are made it is *eventually strong*; and if fairness is ensured by external means it is *weak* [24]. Strong fairness ensures that it is never the case that only one party gets what she expects from the other. Weak fairness ensures that it is possible for a slighted party to prove her case to a judge (e.g., there is all the necessary evidence for the slighted party to convince the judge to strike down on the miscreant). In eventual strong fairness, recourse to external means is avoided at the cost of extra assumptions, such as that the parties follow a certain behaviour (e.g., eventually cooperate).

A number of non-repudiation protocols purporting to be fair, in general, have been proposed. In § 2 we introduce the main ideas. Then, in § 3, we list some guidelines to the practitioner who wants to implement and deploy a fair non-repudiation protocol. The guidelines are illustrated by way of a family of such protocols, descendants of a single ancestor. Some conclusions are offered in § 4.

The present paper follows previous work presenting informal guidelines for the design of cryptographic protocols [35, 5, 1]. It differs from the existing corpus in two ways: it deals specifically with non-repudiation (although its findings may be of wider interest); and it dons two hats: that of the protocol designer, and that of the implementer, as it is mindful of the ramifications of putting a protocol to practice.

As an aside, the author had no part in the development of any of these protocols; and the author's purpose is not to highlight problems and propose solutions, but rather to elicit guidelines that may be of a more general relevance.

2. BACKGROUND

One approach to achieving fair non-repudiation uses cryptography for the gradual simultaneous exchange of secrets. In a typical instance [22], the message is first encrypted and sent; then, the two parties exchange a delivery receipt and the message key in a lockstep way so that none can derive the full receipt or the full key before the other.

Two weaknesses of this kind of protocols are that, first, the two principals must have equal computing power—or one of them can stop the protocol and use her computational prowess to derive the rest of the exchange; and second, there is no near-term deadline by which the process clearly terminates, either with the exchange successfully completed or with the exchange cancelled [14]. In addition, it has been argued, for example by proponents of an alternative approach [37, 25], that they are cumbersome and tend to be impractical.

The alternative approach is to employ an auxiliary party (a third party when the exchange involves two principals). The auxiliary party can be *on-line*, when she participates actively in the protocol run [19, 18, 20, 31]. Alternatively, she can be *off-line*, in the so-called *optimistic protocols*, when her services are required only for dispute resolution. Some such protocols achieve only weak fairness [8, 9], but others are not thus restrained [13, 12, 11, 10]. Depending on the degree of fairness that is desired, it has even been proposed that protocols can be synthesised using appropriate modules [34]. The auxiliary party is usually assumed to be *trusted*, i.e., that she does not behave maliciously; but she may be *semi-trusted*, i.e., she may misbehave, but only by herself, without colluding with any of the other parties [23].

The protocols examined here employ an on-line trusted third party that acts as a notary. They provide non-repudiation evidence through a number of tokens, specifically [40]:

- Non-repudiation of Origin (NRO) or Evidence of Origin (EOO), supplied by the originator, which provides the recipient with proof of origin and guards against the originator of a message falsely denying having sent the message.
- Non-repudiation of Receipt (NRR) or Evidence of Receipt (EOR), supplied by the recipient, which guards against the recipient of a message falsely denying having received the message.
- Non-repudiation of Delivery (NRD), supplied by the delivery agent, which provides the originator of the message with evidence that the message has been delivered to the recipient.
- Non-repudiation of submission (NRS), supplied by the delivery agent, which provides the originator of the message with evidence that the message has been submitted for delivery to the recipient.

3. GUIDELINES

3.1 Match Protocols and Requirements

A protocol makes some assumptions on the domain of its use. At the same time, each application domain demands that specific requirements be met. When putting a protocol to practice, care should be taken to ensure that protocols and requirements match. This might require work both from the application's point of view, to uncover the requirements, and from the protocol's point of view, to flesh out the assumptions it makes—not all of them may be obvious.

Let us take the protocol's point of view, i.e., fleshing out its assumptions, first. An example will help illustrate the issue. Fairness is too broad a term and, as we have seen, it is possible to be more precise by adopting more specific terminology. The type of fairness that a given protocol achieves (e.g., strong, eventually strong, and weak fairness) should be indicated by the developers and taken into account by the implementers depending on the requirements of the application where the protocol is to be implemented.

We can see this by way of a specific protocol. In the following protocol (which we shall call ZG-1), C is the ciphertext for message M, i.e., M encrypted under K. L is a unique label for the protocol run, and $sS_A(X)$ is message X signed by principal A. $A \rightarrow B : X$ means that A sends X to B, while $A \leftrightarrow TTP : X$ means that A fetches X from B using "ftp get" operations or some web browser operations. The TTP is a trusted third party. The various f_{NRO}, f_{NRR}, and so on, are flags denoting the purpose of each step in the protocol.

The idea is that C, which serves as a commitment, is sent first, and then K, which unlocks the message proper, is released (the other approaches that employ an auxiliary party use similar ideas) [37]:

$$\begin{aligned} NRO &= sS_A(f_{NRO}, B, L, C) \\ NRR &= sS_B(f_{NRR}, A, L, C) \\ sub\_K &= sS_A(f_{SUB}, B, L, K) \\ con\_K &= sS_T(f_{CON}, A, B, L, K) \end{aligned}$$

$$\begin{array}{lll} 1. & A \rightarrow B: & f_{NRO}, B, L, C, NRO \\ 2. & B \rightarrow A: & f_{NRR}, A, L, NRR \\ 3. & A \rightarrow TTP: & f_{SUB}, B, L, K, sub\_K \\ 4. & B \leftrightarrow TTP: & f_{CON}, A, B, L, K, con\_K \\ 5. & A \leftrightarrow TTP: & f_{CON}, A, B, L, K, con\_K \end{array}$$

In step 1, A contacts B and sends the encrypted message. In step 2, B confirms receipt, but cannot read the message. In step 3, A submits the message key to a trusted third party; sub_K is the proof of submission of K. The trusted third party stores the tuple (A, B, L, K, con_K) in some read-only directory accessible to the public; con_K is the confirmation of K issued by the TTP. Then, in step 4, B gets the key while, in step 5, principal A confirms that B can indeed get the key. The last two steps can be performed in any order.

The protocol is called "fair". Actually, though, it offers eventually strong fairness, with two assumptions (in what follows we are quoting verbatim from the original publication [37]). First, "even in the case of network failures, both parties will eventually be able to retrieve the key from TTP". Second, no untimely adjudication process should be initiated. This follows from the process of dispute resolution for non-repudiation of receipt. "If A claims that B had received M, the judge will require A to provide M, C, K, L, and the non-repudiation evidence NRR and con_K". The judge will check con_K, and "assume that B is able to retrieve the key K from TTP". The judge will then check NRR and "assume that B had received C and is committed to retrieving K from TTP. The judge will finally check that C decrypted with K produces M. If this is so, "the judge will uphold A's claim", i.e., "that B had received M". This, however, may very well not be true—yet—although it will eventually be true. The protocol equates in this way accessibility and commitment with possession. The implementer should check that this does not contradict the application's requirements.

Before we proceed we have to note that, to all fairness, it is only recently that the analysis of the concept of fairness began [24]. It is obvious that protocols developed earlier cannot be damned for not being able to foresee. But protocols developed in the future should take heed. Moreover, implementers who wish to use a protocol which does not specify the type of fairness it offers should be careful.

Leaving fairness aside, a question arises, concerning at what stage of the protocol is A committed to M. "On its own, the commitment [C] need not restrict the content of the message in any way" [37]. On the one hand, "if we choose labels which are independent of the messages, then the message M may not be defined until Step 3 of the protocol". If, on the other hand, "the label [L] is a function of the message, then the message is already defined in Step 1". In other words, depending on the implementation, A may be committed to M at step 1 or at step 3; it is the requirements of the application that should decide the features of the specific protocol implementation.

Returning now to the application's point of view, understanding the application and uncovering its requirements is no minor undertaking. It is a real challenge; it is the subject of a whole field, that of Requirements Engineering, complete with its own conferences, books, and a journal by that name. Cryptosystems can be fiendishly difficult to specify correctly: "There are ... some particular domains in which specification is well known to be hard. Security is one example; the literature has many examples of systems which protected the wrong thing, or protected the right thing but using the wrong mechanisms" [3]. This is not the place to embark on a review of developments in Requirements Engineering; protocol implementers, however, should be aware that establishing the right requirements and ensuring they mesh with the application is an important and indispensable task. The situation is perhaps exacerbated by the fact that the qualities that make a good cryptographer (e.g., a flair for certain kinds of mathematics) are not necessarily those that make a good Requirements Engineer (e.g., good communication skills), and vice versa.

> Match protocols and requirements. This is a feat that requires working both on the protocols and on the application in order to establish the assumptions and the requirements they make and to make sure they mesh with each other.

3.2 When Does a Protocol Terminate?

The question would arise in any protocol in which the principals are allowed to fetch items at their convenience. A protocol that has no time limits can be awkward to use in a practical setting. For example, if the exchange involves purchase or sale of equity, it is important that it be completed or aborted by some moment in time. The same problem arises in the approaches that use cryptography for the simultaneous exchange of secrets. In that case it is argued that it is a fundamental problem that cannot be mitigated: "We believe that this deficiency is not a coincidence, as there are no reasonable alternatives" [14]. Moreover, analogous situations have been identified in the context of optimistic fair exchange protocols [10]; the problem is similar to the "half-sold house" problem of two-party contract signing protocols [27]. Caution is needed with a protocol that makes no provisions for its termination, as the midst of "half-sent" messages can be a blurry area.

The ZG-1 protocol has no deadline by which all transactions must be complete. It is not clear when the protocol can be considered terminated, aborted, or simply in progress, so that there is no indication when an adjudication process should be initiated. The absence of any deadline cannot be avoided in practice since fixing a time-out makes the protocol insecure. Suppose A starts by performing steps 3 and 5 (there is nothing providing against this in the protocol). She then waits long enough to know that step 4 cannot be performed; upon which she carries out steps 1 and 2. Principal A now holds both con_K and NRR, while B cannot have the message. The TTP cannot revoke con_K because she cannot distinguish between A's maliciousness and B's indifference in retrieving K.

Moreover, if a protocol run extending (for ever) into the future is not acceptable, one must be careful how a time limit is to be incorporated. It is probable that simply grafting some note of expiry in an existing protocol will not automatically solve all problems. See, for example, the following protocol (which we shall call ZG-2) [37]:

$$\begin{aligned}
&NRO = sS_A(f_{NRO}, B, L, T, C)\\
&NRR = sS_B(f_{NRR}, A, L, T, C)\\
&sub\_K = sS_A(f_{SUB}, B, L, T, K)\\
&con\_K = sS_T(f_{CON}, A, B, L, T, T_0, K)
\end{aligned}$$

1. $A \rightarrow B:$ f_{NRO}, B, L, T, C, NRO
2. $B \rightarrow A:$ f_{NRR}, A, L, NRR
3. $A \rightarrow TTP:$ $f_{SUB}, B, L, T, K, sub\_K$
4. $B \leftrightarrow TTP:$ $f_{CON}, A, B, L, T_0, K, con\_K$
5. $A \leftrightarrow TTP:$ $F_{CON}, A, B, L, T_0, K, con\_K$

T is the time limit on the TTP's clock and T_0 is the time that the confirmed key has been made available to the public; it remains so until T. However, A might delay step 3 up to the last moment before T, so that she can perform step 5 while standing a good chance that B might subsequently miss step 4.

> Be careful with the termination problem. If termination is needed and a protocol cannot offer that, be careful with grafting time-outs on it.

3.3 Be Careful with Implementation Details

Most cryptosystems fail not because of flaws in cryptography; their failure is due to other causes, such as organisational and implementation blunders [7, 30]. It is prudent to minimise such risks and to be explicit about the implementation assumptions built in a protocol; these might, again, not be very obvious.

ZG-1 uses symmetric-key cryptography. B receives the key K from A. When a session key is used for privacy, trusting the principals to choose it may be problematic. This is the case in the Wide-mouthed-frog protocol: "This kind of trust is often thought unacceptable because of the quality requirements placed on key generation such as secrecy, nonrepetition, unpredictability, and doubtless more" [1].

Should this caution apply here? The protocol's developers note that encryption is not used for confidentiality here: "the cryptographic key is used only for defining [a communication channel], not for any other purpose, and in particular not for confidentiality" [37]. This must be taken to mean that the protocol is not designed to provide confidentiality against an outside interceptor, i.e., that it is not designed to provide privacy; because confidentiality against B *is* essential. It is clear that if K is poor, B will just break C, drop the protocol, get M and repudiate having done so, getting around non-repudiation.

Of course, if A chooses a bad key, it is A that suffers the consequences, not both parties, as in the Wide-mouthed-frog, or indeed any other protocol that uses a session key for ensuring privacy between the communicating parties. Hence, choosing a good key is in A's best interest. Unfortunately, this does not ensure that a good key will be chosen; for instance, users have a poor record in choosing passwords [2].

Furthermore, in a more recent presentation of the same protocol, K is used with M to derive L [36]; in that case, care should be taken to guarantee L's uniqueness. All in all, the choice of K is important, and the question of whether A can be trusted to issue a good key is pertinent. In a practical setting, this is a detail, not very obvious, that should be taken into account by the implementers.

Another protocol [38], which we shall call ZG-3, tries to avoid the situation in ZG-1 where B follows dilatory tactics and delays in replying to A's initial request. Specifically, the protocol's developers argue (but see § 3.6 below) that as A can submit the message key to the TTP only following B's reply, B can impose a delay that can attribute to A (who cannot prove that it was owing to B). The protocol proceeds thus:

$$\begin{aligned}NRO &= sS_A(f_{NRO}, TTP, B, M)\\ NRS &= sS_{TTP}(f_{NRS}, A, B, T_S, L, NRO)\\ NRR &= sS_B(f_{NRR}, TTP, A, L, NRO)\\ NRD &= sS_{TTP}(f_{NRD}, A, B, T_D, L, NRR)\end{aligned}$$

1. $A \rightarrow TTP: \quad f_{NRO}, TTP, B, M, NRO$
2. $A \leftrightarrow TTP: \quad f_{NRS}, A, B, T_S, L, NRS$
3. $TTP \rightarrow B: \quad A, L, NRO$
4. $B \rightarrow TTP: \quad f_{NRR}, L, NRR$
5. $B \leftrightarrow TTP: \quad L, M$
6. $A \leftrightarrow TTP: \quad f_{NRD}, T_D, L, NRR, NRD$

T_S is the time the TTP received A's submission, while T_D is the time M is made available to the public, and hence to B.

The protocol may fail in step 3. At that step B receives NRO. The protocol defines $sK(X)$ simply as "digital signature of message X with the private key K". If the signature allows message recovery then B can get M, which is the plaintext, and then abort the protocol without committing to anything. The signature algorithm should not allow that.

It is possible that these points would not escape the implementers' notice and appropriate care would be taken. Unfortunately, as the development of cryptosystems abounds with organisational and implementation blunders this cannot be taken for granted. It is best to lean on the cautious side and identify such implementation details so as to minimise any related risks.

Be careful with implementation details. This is nothing new in fact. It is known that in cryptosystems, as elsewhere, the devil is often in the details. Non-repudiation is no exception.

3.4 Formal Verifications Highlight Assumptions

On the one hand, "quite a few protocols which have been 'proved' secure have been successfully attacked" [5], and informal design principles for protocol design have been enunciated [35, 5, 1]. On the other hand, informal principles, like the ones presented here, are no silver bullet [32]. The situation need not be antagonistic. It is possible that formal verifications can highlight assumptions that merit our attention. Indeed, "perhaps the benefit which [logic] brings is as much from forcing us to think clearly about what is going on than from any intrinsic mathematical leverage" [4].

ZG-1 has been formally analysed using two different approaches. Both analyses have verified the protocol. Their proofs are not oriented towards the principals' beliefs (as, for example, in proofs of authentication protocols), but of a judge's beliefs regarding what the principals have or have not done.

One analysis was performed using CSP, i.e., "an abstract language designed specifically for the description of communication patterns of concurrent systems components that interact through message passing" [29]. The other analysis employed the SVO logic [33], an offshoot of BAN logic [16, 17], itself a many-sorted modal logic. It was carried out by the protocol's original developers [41].

In a given system, safety properties specify what the system is allowed to do, or equivalently, what is not allowed to do; liveness properties specify what the system must (eventually) do [26]. With this in mind, "the CSP modelling reveals an aspect of non-repudiation unusual for a security property. Most security properties are safety (trace) properties, essentially that nothing bad (a breach of security) should happen at any stage. In the case of the protocol described in this paper, some of the aspects of non-repudiation involve liveness as well as safety. For example, the evidence that A collects does not guarantee that B has in fact received the message, but it does guarantee that the message must be *available* to B" [29].

The notion of availability also emerges when using SVO. It is asserted as a premise that TTP *said* $(A, B, L, K) \supset$ A *said* $(A, B, L, K) \wedge B$ *received* (A, B, L, K) which means that if the TTP has made K available, it follows that this K had come from A and that B has received it.

A similar statement is made in the CSP model:

$$\begin{aligned}&NRR(tr, X) = \\ &\quad evidence.A.s_B(f_{NRR}.A.L.C) \text{ in } tr \\ &\quad \wedge\ evidence.A.s_T(f_{CON}.A.B.L.K) \text{ in } tr \\ &\qquad \Rightarrow B \text{ sent } s_B(f_{NRR}.A.L.C) \\ &\qquad \wedge\ ftp.B.TTP.(s_T(f_{CON}.A.B.L.K)) \notin X\end{aligned}$$

This means that the evidence of B's receiving C and T's making K public implies that B has sent the former and that the ftp operation to get K does not fail, i.e., K is available to B. The implication is *defined* to be the NRR.

Regarding the SVO premise, in the proof it is noted that it, along with some other premises made there, "are closely related to the protocol features and not so intuitive in the framework of the SVO logic. We should be prudent when producing such kind of premises since the soundness of the reasoning will be in doubt if their truth is based on improper assumptions" [41]. It is clear that this premise does *not* always hold. Specifically, the second conjunct does not hold automatically; it will hold eventually, when (if) the protocol terminates. The fact that it is asserted as a premise means that the soundness of reasoning is based on the assumption that K's availability implies receipt. Along the same path, the CSP proof, while it notes that "there is no guarantee that all of the messages have actually been received by B by the time A presents the evidence" [29], defines non-repudiation of receipt through availability.

In both approaches, therefore, the proof is carried through by assuming that availability can be used to establish receipt. In practice, of course, there is a difference between something being available, or something being possible, and something having actually been acquired, or having actually been performed. Still, if the assumption is acceptable, the protocol can be accepted as well. If not, perhaps a protocol with different assumptions would be more appropriate.

Formal verifications highlight assumptions. By casting a protocol in a formal framework they state explicitly premises that may be noteworthy.

3.5 Be Careful with Practical Improvements

Protocols may be modified in order to make them more amenable to practical implementation. Such optimisation, however, can be attended by untoward consequences. We give some examples below.

3.5.1 Practical Improvements: ZG-3

We have already noted, in § 3.3, that ZG-3 was proposed to deal with some perceived practical deficiencies in ZG-1; and that it may fail depending on the signature algorithm used. There is yet another hitch in this protocol.

In ZG-3, the unique label L is generated by the TTP, not by A, to identify the message M, and it is not contained in NRO. There is no way then to distinguish between copies of NRO for the same recipient, message and third party. The adjudication process only states that "B can use NRO to verify the message M obtained at Step 5. With NRO, B can also prove that M originated from A" [38]. This leaves room for two interpretations.

First, that if B can present M, L_1, NRO and M, L_2, NRO, where $L_1 \neq L_2$, it will be held that A sent M twice. Unfortunately, A might have sent it only once. B could present as much evidence as wished and prove that A has sent it as many times as wished. Even if the adjudication involved a TTP which could show otherwise—provided it had kept a record of all past transactions—if B can impersonate A in the first two steps, she could carry out all the protocol by herself by using a previously acquired NRO (principal A signs nothing else). In short, ZG-3 confers an undue advantage to B, allowing her to falsely ascribe a communication to principal A.

Second, that if B can present M, L_1, NRO and M, L_2, NRO, where $L_1 \neq L_2$, it will be twice held that A sent M. That is, two judgements on the same fact. Unfortunately, A might have actually sent M twice. In short, ZG-3 confers an undue advantage to B, allowing her to falsely repudiate a communication from A. (Note that in both cases, and in fact always, $L_1 \neq L_2$: Ls are unique in order to prevent selective receipt by B.)

To be fair, the protocols developers do not assume that ZG-3 is immune to such abuse. But they do not mention anything relevant in their assumptions. Moreover, ZG-1 and ZG-2 are immune.

3.5.2 Practical Improvements: ZG-4

The involvement of the third party in ZG-1 can lead to bottlenecks. A variant of the protocol (which we shall call ZG-4) involves the third party only in dispute resolution [39]:

$$\begin{aligned}EOO &= sS_A(f_{EOO}, B, L, C)\\ EOR &= sS_B(f_{EOR}, A, L, C)\\ EOO\_K &= sS_A(f_{EOO\_K}, B, L, K)\\ EOR\_K &= sS_B(f_{EOR\_K}, A, L, K)\end{aligned}$$

1. $A \rightarrow B:\ f_{EOO}, B, L, C, EOO$
2. $B \rightarrow A:\ f_{EOR}, A, L, EOR$
3. $A \rightarrow B:\ f_{EOO\_K}, B, L, K, EOO\_K$
4. $B \rightarrow A:\ f_{EOR\_K}, A, L, EOR\_K$

If A does not get message 4 from B then steps 3–5 of ZG-1 are performed—ZG-4 reverts to ZG-1. In essence, A sends K directly to B in step 3. If confirmation fails to arrive it is acquired by way of the TTP.

The protocol's developers show that adopting a deadline on the time that the TTP offers K to the public (as in ZG-2) breaks fairness [39]. Suppose that B receives K in step 3, but does not perform step 4. Then A must submit K to the TTP before the deadline, but if there are communication problems she cannot be sure that it will be done on time; in which case B gets the better of her.

3.5.3 *Practical Improvements: ZG-5*

ZG-1 rests on the validity of digital signatures. It is important to ensure that digital signatures are not compromised, or the protocol will be breached. One way to guarantee the validity of digital signatures at a given point in time is by employing a trusted time-stamping authority to validate them; this, however, increases the number of messages that have to be exchanged. ZG-1 has been modified to demonstrate another, more efficient way (we shall call the resulting protocol ZG-5) [36]:

$$\begin{aligned} EOO &= sS_A(f_{EOO}, B, L, C) \\ EOR &= sS_B(f_{EOR}, A, L, C, EOO) \\ sub\_K &= sS_A(f_{SUB}, B, L, K, EOR, Cert_B) \\ con\_K &= sS_{TTP}(f_{CON}, A, B, L, K, EOR, T) \end{aligned}$$

1. $A \rightarrow B$: f_{EOO}, B, L, C, EOO
2. $B \rightarrow A$: f_{EOR}, A, L, EOR
3. $A \rightarrow TTP$: $f_{SUB}, B, L, eV_{TTP}(K), EOR, Cert_B, sub\_K$
4. $B \leftrightarrow TTP$: $f_{CON}, A, B, L, K, T, con\_K$
5. $A \leftrightarrow TTP$: $f_{CON}, A, B, L, K, T, con\_K$

Here all evidence is chained, and the two public key certificates for the signatures of A and B, $Cert_A$ and $Cert_B$ respectively, are checked only by the TTP before issuing con_K. T is a time stamp for the time that the TTP checks the message key and makes it available in the read-only directory. $eV_{TTP}(K)$ is K is encrypted with the TTP's public encryption key to prevent eavesdropping: B could eavesdrop in step 3 and then revoke her signature having obtained both K, which would have been in plaintext, and C.

Suppose that A signs EOO with the key of a revoked certificate $Cert'_A$ and then proceeds to sign sub_K with a valid certificate $Cert_A$. Later in the protocol the TTP will check $Cert_A$—but not EOR or EOO—before issuing con_K. Principal A cannot deny sending the message, since she has checked EOR, which contains EOO, before step 3: an adjudicator will rule that A has sent it. But A can turn this to her advantage.

In the adjudication process [36], A or B will present the adjudicator with M, C, K, L, T, $Cert'_A$, $Cert_B$, EOO, EOR, and con_K. The adjudicator will use $Cert'_A$ to check EOO, and will accept it—if she cannot check, from the material provided, the validity of $Cert'_A$ at time T; in fact no such check is envisaged in the adjudication process, and $Cert_A$ is not present in any message in the protocol. That $Cert'_A$ is not valid at the time of the dispute is irrelevant, as the dispute may arise after the certificates have expired. In this way, though, the lifetime of $Cert'_A$ is surreptitiously extended, and that can lead to problems. If $Cert'_A$ had been granted to A for a limited period, e.g., A can sidestep the expiry.

3.5.4 *Practical Improvements: ZG-6*

Another protocol has been proposed as an improved version of ZG-1. The protocol, which we shall call ZG-6, adds explicit time limits and encrypts K for increased privacy [25]:

$$\begin{aligned} NRO &= sS_A(f_{NRO}, B, L, T, C) \\ EOR &= sS_B(f_{NRR}, A, L, T, T_1, C) \\ sub\_K &= sS_A(f_{SUB}, B, L, T, K_{sub}) \\ con\_K &= sS_{TTP}(f_{CON}, A, B, L, T, T_0, K_{sub}) \end{aligned}$$

1. $A \rightarrow B$: f_{NRO}, B, L, T, C, NRO
2. $B \rightarrow A$: f_{NRR}, A, L, T_1, NRR
3. $A \rightarrow TTP$: $f_{SUB}, B, L, T, K_{sub}, sub\_K$
4. $B \leftrightarrow TTP$: $f_{CON}, A, B, L, T_0, K_{sub}, con\_K$
5. $A \leftrightarrow TTP$: $f_{CON}, A, B, L, T_0, K_{sub}, con\_K$

To ensure privacy, A, instead of sending K, sends K_{sub}, from which B, using her private key, can retrieve K—that is, a Diffie-Hellman public key distribution scheme is introduced [21]. To get around timing problems, A, in step 1, suggests a time limit T. Then B, in step 2, suggests another time limit $T_1 < T$. The idea is that A must perform step 3 before T_1, and B can perform step 5 at any time between T_1 and T. T_0 is the time K_sub is made available by the TTP; during the adjudication process the adjudicator checks that $T_0 < T_1 < T$.

Suppose B suggests a $T_1 < T_0$. This can happen because the TTP has no knowledge of T_1. B is still able to perform step 4, retrieve K_{sub}, and then argue that the protocol run was invalid. The adjudicator will concur, since $T_0 < T_1 < T$ is a requirement in the adjudication process. And it cannot be otherwise: a protocol cannot be valid with $T_1 < T_0 < T$ and T_1 meaning that B can get K_{sub} at any time between T_0 and T.

The evolution from ZG-1 to ZG-6 is a series of improvements on previous protocols, improvements such as adding time information, adding privacy, and creating evidence more effectively through chaining. We saw, however, that introducing new features or mitigating previous snags can create new, unanticipated difficulties. There is a well-known dictum in Software Engineering that smoking out bugs often introduces new, nastier ones so that it is one step forward and one step back [15]. That dictum could inform research on cryptographic protocols as well.

Be careful with practical improvements. It is tempting to develop a theoretically sound protocol and then try to make it more practical for use in specific settings. It is possible that, in the process, problems will be introduced.

3.6 Beware of Model Twisting

It is important to keep in mind during protocol development that model twisting is not equivalent to flaws fixing. A protocol is acceptable or unacceptable on its own terms (i.e., the assumptions and model it proposes); if the terms don't do for a specific situation, the protocol is incompatible with that situation; but it is neither wrong, nor flawed. The choice between different terms is evident in protocols ZG-4 and ZG-5.

In ZG-4, it is noted that ZG-1 "is suitable for situations where two parties want to involve a trusted third party in every protocol run", while ZG-4 "is suitable for environments where the two parties normally will resolve any communication problems between themselves and rely on a trusted third party only as a last recourse" [39]. This is a matter of choice between competing alternatives based on different models, not something that can be decided on correctness.

In ZG-5 it is argued that the problem of establishing and maintaining signature validity in ZG-1 hampers efficient implementation [36]. This may well be so; but it is not a problem with ZG-1 per se, which simply assumes valid signatures, but with the means whereby signature validity can be guaranteed.

No pitfalls as of yet. But come ZG-3, it is noted that in ZG-1 "the originator can only submit the message key to the *TTP* after obtaining the reply to its commitment from the recipient. The originator would not like to take the responsibility of late submission caused by the the recipient's late reply"; hence ZG-3 is offered [38]. We have again a choice between different models. The recipient's delay is not a problem for ZG-1, as ZG-1 will work correctly no matter how much time the recipient takes to respond. Whether this will do for the originator, is a different matter—indeed, a matter for choosing another protocol.

Unfortunately, the reasoning in the previous paragraph is wrong. ZG-1 has been surreptitiously twisted. There is nothing in ZG-1 to inhibit A from performing step 3 without waiting for step 2 (cf. § 3.2). It is true that A would normally wait for B's response; nevertheless it by no means follows that A must wait for B's response.

Consider, finally, ZG-6. It is argued that ZG-1 is unfair, and that keeping K unencrypted can cause privacy problems [25]. The argument goes against the assumptions made in ZG-1. ZG-1 is made unfair by twisting its model in various ways: a time limit is introduced to guard against the pernicious effects of a combined network failure and a principal bungling her part; then the protocol is shown to be unfair to B if A can disrupt communications between B and the *TTP*. None of this can happen if the original assumptions made in ZG-1 hold and the two principals do not play against themselves. Moreover, as we saw in § 3.3, cryptography in ZG-1 is not used for privacy, but for establishing a communication channel. The criticism on privacy problems is therefore irrelevant.

The reasoning followed in ZG-3 and ZG-5 is scientifically unsound. Protocols can be twisted to exhibit undesirable features, and other protocols can be introduced to alleviate them; yet this does not do justice to the original protocol and it provides a dubious service to the new one.

> Beware of model twisting. Twisting a protocol in order to build a case for a new one is not sound, scientifically or logically—and is no fair play. A protocol can be judged on its assumptions, compared to other protocols for its assumptions, but it cannot be analysed on assumptions different than the ones it actually pledges for.

3.7 Treat Adjudication as an Integral Part of the Protocol

The interaction between cryptography and judicature may be discordant, and we should not expect engineering to solve legal problems [6]. If this is true for cryptography in general, it is particularly pertinent to non-repudiation. Even the terminology employed in non-repudiation points to the relevance of legal matters to the issue: principals collect *evidence* that can be presented to a *judge*. It is implied that engineering will be called to solve a legal problem, and the case will be decided according to the adjudication process described.

The adjudication process, therefore, should be conceived as forming part of the protocol. In view of the relation between engineering and judicature, problems or vagueness in the adjudication process should not be taken any more lightly than problems in the protocols themselves. For a given protocol, the adjudication process proposed should be deterministic.

All this is reason for examining it in detail. Whenever we made references to it, we referred to the exact procedure described in the original publication, accepting that protocol analysis should include adjudication analysis. If problems during the adjudication process are identified, the problem is not solely with the protocol; a change in the adjudication may instead be warranted. For example, the adjudication process of ZG-5 might be made to include a check of $Cert_A$ at time T (cf. § 3.5.3); in general, in a protocol that uses certificates, those certificates are part of the evidence and certificate evaluation has to be covered explicitly by the adjudication rules. Also, the adjudicator of ZG-1 might require from *TTP* to confirm that the network connection with the two principals would not have hindered their performance of steps 4 and 5 (cf. especially the possible problems of B in the discussion on the untimely adjudication process in § 3.1).

Unfortunately, even a well-designed, deterministic adjudication may run into problems as what is technically sound may not necessarily curry much favour in a legal process [6]:

> Lawyers are well aware that the use of technical evidence, and in particular computer evidence, is fraught with difficulty. Most judges have a background in the humanities rather than the sciences, and may be more than normally technophobic; even where these feelings are dutifully suppressed, experienced and otherwise intelligent men can find it impossible to understand simple evidence. The author has observed this phenomenon at a number of computer trials from 1986 down to the present, and has often felt that no-one in court had any idea what was going on. Specialist computer lawyers confirm that this feeling is not uncommon in their practice.

Worse, there may be specific points in a protocol that can render themselves as conduits for deliberate obfuscation. Let us loosen the discussion a bit and speculate on possible candidates.

Is it possible for A to be committed to a message she does not know? The issue can be followed through the protocol trail. In ZG-1, A never signs the original message. She signs the *encrypted* message; it is in step 3 that A commits to a message she knows (this is in general; the situation may be different, depending on how L is chosen—recall § 3.1). Then, in ZG-5, A signs neither the original message, nor the actual message key, which is now encrypted; it is only in step 5, when she retrieves the confirmed key from the TTP, that A commits to M. Finally, in ZG-6, A can commit to a message she has no means of retrieving, i.e., knowing. According to the protocol, if s_A and s_B are the secret keys of A and B, their public keys are $p_A = g^{s_A} \bmod p$ and $p_B = g^{s_B} \bmod p$, where p and q are large primes with $p = 2q + 1$ and g a primitive element over $GF(p)$. For the key distribution, A generates a random number r $(0 < r < p-1)$ and computes $K = p_B^r \bmod p$ and $K_{sub} = g^r \bmod p$; B can then retrieve $K = K_{sub}^{s_B} = g^{rs_B} \bmod p$. It follows that if A does not know the discrete logarithm of K_{sub}, she commits to a message she cannot know. This can easily happen: A might just pick, or be passed, a value for K_{sub} and send it.

If a principal can commit to a message she has no means of knowing, the question becomes whether this can be acceptable in practice. This will depend on the particulars of the specific application—and it will, ultimately, be a matter of matching protocol features, which must be understood thoroughly, with application requirements, which must be established carefully.

As a last point, controversy can result from the combination of the two protocols in ZG-4. It is mentioned that "if A does not send message 3, the protocol ends without disputes. If A cannot get message 4 from B after sending message 3 (either because B did not receive message 3 or because B does not want to acknowledge it), A may initiate ... the recovery phase". There is a contradiction here. If B does not receive message 3 she can rightfully assume that the protocol has ended without disputes; while A, not receiving message 4, can rightfully assume that the protocol has not terminated, but that the recovery phase is to be initiated. Imagine that A does initiate it, B gets the key and then, following the above, argues that no protocol run has taken place.

Although this is admittedly pure speculation, one might take advantage of exactly this sort of niceties to cause a judicial process to flounder. It might be prudent to guard against such bones of contention. For example, if proof of dispatch is equated with proof of origin, or if knowledge of the message contents is taken for granted, this should be set out in the protocol's assumptions. In this way, if two principals embark on a protocol run accepting its assumptions, the above problems cannot stand.

> Treat adjudication as an integral part of the protocol. Adjudication should be well defined and deterministic, and guarantee a single yes or no answer for each set of input. Protocol analysis should include adjudication analysis. Unfortunately, one should be prepared for when even a correct adjudication process fails in practice, as technology and judicature do not always make a perfect couple—fleshing out possible bones of contention might help.

4. CONCLUSIONS

Non-repudiation is special in that the scope of the system is particularly wide, as it encompasses agents (e.g., judges) outside the communication exchange. Getting a protocol right involves taking account of a great many potential loopholes, not solely technical.

If there is a common thread running through guidelines proposed here—a meta-guideline, as it were—it is the value placed on making assumptions explicit. *Assumptions should always be explicit*, and in non-repudiation all the more so. Non-repudiation can be seen as a contract, whereby the parties engage verifiably in some exchange: in this metaphor, when contract signing is complete, a principal has completed her part for the other principal's acknowledgement. As in all contracts, everything rests on the terms; and these should be clearly and openly set.

> Assumptions should always be explicit. The terms of a non-repudiation protocol should be set out clearly in the open.

5. ACKNOWLEDGEMENTS

This paper was spurred by discussions with I. S. Iliadis and T. Kosmopoulos, both with the University of the Aegean, to whom thanks are owed. The anonymous reviewers provided valuable feedback that greatly benefited this work.

6. REFERENCES

[1] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[2] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):41–46, December 1999.

[3] R. Anderson. How to cheat at the lottery (or, massively parallel requirements engineering). In *Proceedings of the 15th Annual Security Applications Conference (ACSAC '99)*, Phoenix, AZ, 6–10 December 1999. IEEE Computer Society Press, Los Alamitos, CA.

[4] R. Anderson and R. Needham. Programming satan's computer. In J. van Leeuven, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of *Lecture Notes in Computer Science Series*. Springer-Verlag, Berlin, 1995.

[5] R. Anderson and R. Needham. Robustness principles for public key protocols. In P. Coppersmith, editor, *Advances in Cryptology—CRYPTO '95: 15th Annual International Cryptology Conference*, volume 963 of *Lecture Notes in Computer Science Series*, pages 236–247, Santa Barbara, CA, 27–31 August 1995. Springer-Verlag, Berlin.

[6] R. J. Anderson. Liability and computer security: Nine principles. In D. Gollmann, editor, *ESORICS 94: Third European Symposium on Research in Computer Security*, volume 875 of *Lecture Notes in Computer*

Science Series, Brighton, UK, 7–9 November 1994. Springer-Verlag, Berlin.

[7] R. J. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40, November 1994.

[8] N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS '97)*, pages 6, 8–17, Zurich, 1–4 April 1997. ACM Press, New York, NY.

[9] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy (S & P '98)*, Oakland, CA, 3–6 May 1998. IEEE Computer Society Press, Los Alamitos, CA.

[10] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal of Selected Areas in Communications*, 18(4):593–611, April 2000.

[11] G. Ateniese. Efficient verifiable encryption (and fair exchange) of digital signatures. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99)*, pages 138–146, Singapore, 1–4 November 1999. ACM Press, New York, NY.

[12] F. Bao, R. Deng, K. Q. Nguyen, and V. Varadharajan. Multi-party fair exchange with an off-line trusted neutral party. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications (DEXA '99)*, Florence, 1–3 September 1999. IEEE Computer Society Press, Los Alamitos, CA.

[13] F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy (S & P '98)*, Oakland, CA, 3–6 May 1998. IEEE Computer Society Press, Los Alamitos, CA.

[14] M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, January 1989.

[15] F. P. Brooks, Jr. *The Mythical Man-Month: Essays on Software Engineering.* Addison-Wesley, Reading, MA, anniversary edition, 1995.

[16] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. SRC Research Report 39, Digital Equipment Corporation Systems Research Center, Palo Alto, CA, 28 February 1989. Revised on February 22, 1990.

[17] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[18] T. Coffey and P. Saidha. Non-repudiation with mandatory proof of receipt. *Computer Communications Review*, 26(1), January 1996.

[19] B. Cox, J. D. Tygar, and M. Sirbu. NetBill security and transaction protocol. In *Proceedings of the First USENIX Workshop on Electronic Commerce*, New York, NY, 11–12 July 1995.

[20] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang. Practical protocols for certified electronic mail. *Journal of Network and Systems Management*, 4(3):279–297, September 1996.

[21] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[22] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, June 1985.

[23] M. K. Franklin and M. K. Reiter. Fair exchange with a semi-trusted third party. In *Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS '97)*, pages 1–5, 7, Zurich, 1–4 April 1997. ACM Press, New York, NY.

[24] F. C. Gärtner, H. Pagnia, and H. Vogt. Approaching a formal definition of fairness in electronic commerce. In *Proceedings of the 18th IEEE Symposium of Reliable Distributed Systems (SRDS '99)*, Lausanne, 18–21 October 1999. IEEE Computer Society Press, Los Alamitos, CA.

[25] K. Kim, S. Park, and J. Baek. Improving fairness and privacy of Zhou-Gollmann's fair non-repudiation protocol. In *Proceedings of the 1999 29th International Conference on Parallel Processing (ICPP '99): Workshop on Security (IWSEC)*, pages 140–145, Wakamatzu, Japan, 21–22 September 1999. IEEE Computer Society Press, Los Alamitos, CA.

[26] L. Lamport. A simple approach to specifying concurrent systems. *Communications of the ACM*, 32(1):32–45, January 1989.

[27] B. Pfitzmann, M. Schunter, and M. Waidner. Optimal efficiency of optimistic contract signing. In *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing (PODC '97)*, pages 113–122, Puerto Vallarta, Mexico, 28 June–2 July 1998. ACM Press, New York, NY.

[28] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[29] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 54–65, Rockport, MA, 9–11 June 1998. IEEE Computer Society Press, Los Alamitos, CA.

[30] B. Schneier. Cryptographic design vulnerabilities. *Computer*, 31(9):29–33, September 1998.

[31] B. Schneier and J. Riordan. A certified e-mail protocol. In *Proceedings of the 14th Annual Security Applications Conference (ACSAC '98)*, Scottsdate, AZ, 7–11 December 1998. IEEE Computer Society Press, Los Alamitos, CA.

[32] P. Syverson. Limitations on design principles for public key protocols. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy (S & P '96)*, Oakland, CA, 6–8 May 1996. IEEE Computer Society Press, Los Alamitos, CA.

[33] P. F. Syverson and P. C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy (S & P '96)*, pages 62–72, Oakland, CA, May 1996. IEEE Computer Society Press, Los Alamitos, CA.

[34] H. Vogt, H. Pagnia, and F. C. Gärtner. Modular fair exchange protocols for electronic commernce. In *Proceedings of the 15th Annual Security Applications Conference (ACSAC '99)*, Phoenix, AZ, 6–10 December 1999. IEEE Computer Society Press, Los Alamitos, CA.

[35] T. Y. C. Woo and S. S. Lam. A lesson on authentication protocol design. *Operating Systems Review*, 28(3), July 1994.

[36] C.-H. You, J. Zhou, and K.-Y. Lam. On the efficient implementation of fair non-reputation. *Computer Communications Review*, 28(5):50–60, October 1998.

[37] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy (S & P '96)*, pages 55–61, Oakland, CA, May 1996. IEEE Computer Society Press, Los Alamitos, CA.

[38] J. Zhou and D. Gollmann. Observations on non-repudiation. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology—ASIACRYPT '96: International Conference on the Theory and Applications of Cryptology and Information Security*, volume 1163 of *Lecture Notes in Computer Science Series*, pages 133–144, Kyonkgju, Korea, 3–7 November 1996. Springer-Verlag, Berlin.

[39] J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *Proceedings of the 10th Computer Security Foundations Workshop (CSFW '97)*, Rockport, MA, 10–12 June 1997. IEEE Computer Society Press, Los Alamitos, CA.

[40] J. Zhou and D. Gollmann. Evidence and non-repudiation. *Journal of Network and Computer Applications*, 20(3):267–281, July 1997.

[41] J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In J. Grundy, M. Schwenke, and T. Vickers, editors, *International Refinement Workshop and Formal Methods Pacific '98: Proceedings of IRW/FMP '98*, Discrete Mathematics and Theoretical Computer Science Series, pages 370–380, Canberra, 29 September–2 October 1998. Springer-Verlag, Berlin.