A new fair non-repudiation protocol for secure negotiation and contract signing

A. Ruiz-Martínez, C. I. Marín-López, L. Baño-López, A. F. Gómez Skarmeta
Department of Information and Communications Engineering.
Faculty of Computer Science. University of Murcia
{arm, inma, laurabl, skarmeta}@dif.um.es

ABSTRACT
In some scenarios, such as B2B or IPR contracting, or by legal requirement, the participation of an e-notary in the contract signing is required in many cases, that is, an on-line TTP. This e-notary gives validity to the contract or performs some tasks related to the contract, such as contract registration or fraud detection. In these B2B or DRM contracting scenarios, two important additional features are needed: the negotiation of the e-contract and confidentiality. However, until now, e-contract signing protocols have not considered both of these as an essential part of the protocol. In this paper, we present a new protocol which is designed to make negotiation and contract signing secure and in a confidential way. Moreover, the protocol, compared to other previous proposals based on an on-line TTP, reduces the e-notary's workload. Finally, we describe how the protocol is being used in conjunction with an extended version of ODRL to achieve agreements on the rights of copyright works in a DRM infrastructure.

Categories and Subject Descriptors
C.2.2 [Computer-Communications Networks]: Network protocols; K.4.4. [Computer and Society]: Electronic Commerce – Security, Distributed Commercial Transactions; K.6.5. [Management of Computing and Information Systems]: Security and Protection – Authentication.

General Terms
Algorithms, Design, Security, Standardization, Legal Aspects, Verification.

Keywords
Contract signing protocol, fair-exchange, abuse freeness, e-commerce, contract negotiation, confidentiality, security.

1. INTRODUCTION
Nowadays, part of the research in electronic commerce and business transactions is centered on the electronic signing of contracts, where there is an agreement between two or more parties. In these agreements, the parties implied will obtain profit from their relationship. These profits could be goods, money, services, collaborations, and so on. The contracts are particularly important in C2B and B2B commerce to establish long-term relationships or to offer services. They are also important in DRM systems in agreeing the rights of copyright works and the royalties to pay in order to use them.

In contract signing protocols, the Trusted Third Parties (TTPs) provide security and confidence to the system. However, this can be a serious problem if the TTP becomes a bottleneck due to the fact that it has to participate in a lot of transactions. For this reason, except in certain circumstances, it is desirable that the TTP participates the least possible in the execution of the protocol. Although the latest advances in this field occurred both in off-line protocols [6],[7] and in the multi-party contract signing protocols [9],[24], the use of a TTP online is still needed, in some cases [1],[5],[15],[21],[29], [34],[35]. In some scenarios, to obtain a valid contract it is mandatory that the contract is signed by a TTP, acting as an e-notary. For example, the Spanish or French laws establish that some kind of contracts, such as contracts related to real estate or legacies, must be signed by an e-notary [1],[21]. Thus, the e-notary gives validity to the contract and registers it. Therefore, in these scenarios, we are not able to make use of an off-line TTP that only participates when there is a problem, because, for the contract to be valid, the TTP has to sign it.

Other additional scenarios could be the negotiation and agreement signing of rights of copyright works or a B2B contract update. The e-notary will give validity to the contract and, depending on the DRM infrastructure, could additionally perform some tasks related to the infringement of copyright in association with other entities, such as a Monitoring Service Provider [15],[29].

Thus, in those scenarios where the TTP on-line is required, we will aim for that the contract signing protocol to reduce the TTP's overload to the minimum possible, i.e., that TTP participates in very few messages in the execution of the protocol and makes the minimum number of cryptographic operations.

In this paper we will analyze some key protocols based on an on-line TTP [34],[35] and we will introduce some problems they present, such as not guarantee abuse freeness or the over important participation of the TTP in the protocol. We will also analyze them from the point of view of two important requirements that have not been taken seriously into account until

PST 2006, Oct 30–Nov 1, 2006, Markham, Ontario, Canada.

now in business/DRM transactions: secure negotiation and confidentiality.

We propose a new protocol that is based on the use of a TTP on-line. However, the TTP does not participate in all the messages exchanged between the parties, only in the last phase of the protocol with the aim of signing the agreement reached. The main added value of this protocol against other proposals is the incorporation of important features such as confidentiality and a fundamental aspect prior to the contract signing: the secure negotiation of the contract. In this protocol, the information that allows the negotiation of the contract has been included as part of the protocol. At the same time, we have reduced the TTP's overload compared to other previous works. We also want to point out that we have designed the protocol to be based on any public key cryptosystem that provides signature and ciphering (RSA, ECDSA, ECDH, …). Therefore, the protocol could be used by the users with the usual keys and certificates that they own, because it is not necessary either to generate new keys or a registration process with a TTP, unlike [14], [27], [25]. Furthermore, we have made an implementation of the protocol based on Java, XML schemas and XML signatures [33] with RSA keys. This development is being used in a scenario where intellectual property rights are negotiated and purchased. For this scenario we have extended the ODRL [32] language and we have used it as a contract language.

In the following section of this paper, we will comment on the main features that we are going to require for a contract signing protocol. We also mention some previous works in this field. Section III details our proposal: main features, advantages and protocol specification. Once the details of the new protocol have been introduced, it will be analyzed from several points of view: security, requirements mentioned in section II and comparison with previous works. This will be done in section IV. In section V we will comment on the main aspects of the implementation of the protocol. In section 6, we will introduce how the protocol is used in a DRM infrastructure in order to negotiate and sign agreements based on ODRL expressions. Finally, we will present conclusions and possible future work.

2. CONTRACT SIGNING PROTOCOL REQUIREMENTS AND RELATED WORK

In this section we give the requirements that we will require for electronic contract signing protocols in order to negotiate and sign contracts in a secure way and make use of a TTP on-line for scenarios where the electronic signature of a TTP is essential, for example, in some business-to-business or DRM scenarios [5], [15], [21], [29], [34], [35] or by legal requirement [1], [21]. Then, we will analyze some previous works related to this issue [34], [35].

2.1 Requirements

The basic features we are going to require for a contract signing protocol are: non-repudiation, fairness, efficiency, completeness, viability and timelessness. These are discussed in more detail in [20, 23]. These criteria are the classical features that we should require for a protocol that guarantees fair exchange and therefore, for a contract signing protocol, as a kind of the protocols included in fair exchange. Besides, a new important feature for the contract signing protocols was introduced in [14]:

- *Abuse freeness*: "if it is impossible for a single player at any point in the protocol to be able to prove to an outside party that has the power to terminate (abort) or successfully complete the contract" [14].

Other additional requirements considered by S. Yang et al. in [34, 35] are:

- *Trust dependency on a third party*. A TTP that knows the content of the messages requires a huge grade of dependency on the confidence in a third party. On the other hand, a TTP that does not know the contents presents a lesser grade of dependency as regards the trust we have to deposit in the third party. We could also require this feature for a contract signing protocol unless, in some circumstances, the TTP needs to know the content in order to perform certain operations (for example, an e-notary who has to register the contract and assure its validity).
- *Existence dependency*. The aim is that the protocol generates enough evidences, in case there is a later dispute, for there to be no dependency on the existence or availability of the TTP in order to determine the result of the dispute.
- *Recipient role*. Normally, the protocols are sender-or-requester-oriented. They give this role more control and responsibility. However, in e-commerce applications, service provider has to take an active role in the execution of the protocol.

Finally, other additional features we consider essential and, therefore, which we will require for contract signing protocols are:

- *Confidentiality*. The information exchanged between the parties should be only known to them. Most of the protocols suppose a private communication channel. We think that supposition is not realistic enough. A user cannot suppose the underlined network provides this property because the network may not offer it. Besides, establishing this private channel requires the exchange of additional messages, and therefore, a greater overload for both the users and TTP.
- *Secure Negotiation*. The protocol should allow the secure negotiation of the terms and conditions of the contract since, in general, it is difficult to understand a contract signing without a previous negotiation phase.

In most of the cases, these additional criteria have not been considered. As for confidentiality, many of them suppose a channel where the information is exchanged in a ciphered way (e.g. a SSL/TLS channel needs five messages, at least, to establish it [7], [10]). However, confidentiality is not supposed as part of the protocol. This characteristic is important because we want to avoid that parties not involved in the protocol knowing the agreement between two parties, the conditions, their behavior, and

so on. As far as the secure negotiation of the contract terms and conditions is concerned, it is not a feature which any of them incorporates. With the existing protocols, this feature could be achieved by means of either the execution of the protocol for each offer or by using a previous specific protocol to make the negotiation. The main disadvantage of an execution of the protocol, for each offer, is that it supposes a lot of messages and is slow. However, from our point of view, we consider the secure negotiation an important feature since there is not much sense in making the negotiation independently of the contract signing. We also consider that we should require a similar security level in both the negotiation and contract signing phases. Thus, the protocol would make the whole lifecycle of the negotiation and contract signing secure. Thus, it could even be used by intelligent agents to make a contract using negotiation [4],[29] in a secure way, without running the risk of the contract information being intercepted by an outside party.

2.2 Related Work

In this section we are going to look at several previous works based on a TTP on-line. Then, we will discuss their main features and finally, we will analyze them from the point of view of the requirements explained above. Protocol 1 is one of the first serious proposals related to non-repudiation with a TTP on-line. Protocol 2 is the latest proposal in this kind of protocols and improves on protocol 1, as well as others proposed in [2], [18].

2.2.1 Protocol 1

The protocol proposed by Zhou and Gollmann in [37] is based on a TTP on-line and it can guarantee that a message exchanged between two parties is sent and received correctly. Besides, it provides evidences that confirm this (non-repudiation). However, the protocol, as it is, can not be used to sign a contract because one of the parties does not know the content of the messages and only confirms the reception of the message. When the two parties sign the contract, they should execute the protocol twice. In the first time, Alice signs the contract and sends it, with the participation of the TTP, to Bob who confirms the reception of the contract. Then, in the second execution, Bob signs and sends the signed contract to Alice by means of the TTP. Next, Alice confirms the reception of the signed contract. The main drawback of this protocol is that it does not guarantee the requirement of abuse freeness (after the first execution Bob has an advantage over Alice). It does not also provide the possibility of negotiation of the contract. Finally, as regards the confidentiality, it would be necessary to establish a secure channel between Alice and Bob; Alice and TTP; and, Bob and TTP.

2.2.2 Protocol 2

The goal of the protocol proposed by S. Yang et al. in [35] is to guarantee the non-repudiation in the delivery of the messages and it could be used for contract signing. In fact, it was proposed for collaborative e-commerce. But, since the protocol is conceived for message delivery, if we want to use it to sign contracts we have to execute it twice, in a similar way to the previous protocol. In the first execution, Alice sends the contract signed to Bob, and Bob confirms the delivery but he does not sign the contract. Then, in the second execution, Bob signs the signed contract by Alice and sends it to her so as to confirm she receives it. The protocol offers the evidences to prove if one of the parties is not behaving correctly. It also offers confidentiality, and the TTP will never know the content of the contract. However, there are two main problems. The most important is that the protocol does not guarantee the abuse freeness property because Bob obtains a signed contract before Alice gets it. Therefore, Bob has an advantage over Alice. Other important problem is that the TTP has a significant participation in the execution of the protocol because it participates in five of its six messages of each iteration. This fact could cause the TTP to become a bottleneck if there were many concurrent executions of the protocol. Besides, if we want to negotiate the terms and conditions of the contract, it would be necessary to offer an additional mechanism, since if we used this protocol for each message in the negotiation, then the complete process, that is, negotiation plus contract signing, would present a high overload in the system given the number of messages exchanged and the cryptographic operations made. Finally, a little problem that could be solved easily is that the protocol does not satisfy the timeless property.

In the next section, we introduce a new protocol that provides a common answer to the problems and needs that we have mentioned previously. It pursues several goals. First, to provide support to those electronic business environments where there are processes of negotiation and contract signing, and the participation of a TTP is required to validate a signed contract [1], [5], [15], [21], [29], [34], [35]. Secondly, the compliance with the previously mentioned requirements so as to provide a robust contract signing protocol. Thirdly, reducing the participation of the TTP to the minimum possible expression as regards the number of cryptographic operations (especially the asymmetric ones, due to their computation cost) and messages sent and received.

3. CONTRACT SIGNING PROTOCOL SPECIFICATION

In this section we present a new contract signing protocol which provides confidentiality and a secure negotiation, it guarantees abuse freeness and it is based on a TTP on-line for those situations in which an e-notary has to give validity to the contract by signing it. It is important to mention that it could be used with any existing public key cryptosystem, as long as it offers signature and ciphering. Thus, it could be used without generating new keys and certificates, unlike other protocols that require either specific keys (e.g., ElGamal keys [14]) or generating new ones with special features, or a previous registration process with a contract TTP [14],[27],[31]. We have decided to name it: SURENESS (SecURE NEgotiation and contractS Signing) protocol. In the explanation of the protocol, we consider that none of the parties is going to act against its own interests.

3.1 Notation

Here, we will describe the notation used to specify the sequence of messages in our protocol specification.

Table 1. Cryptographic notation

Symbol	Meaning
[*Data*]	It indicates that this piece of *data* is optional, and it may not be in the message.
H(*Data*)	A message digest of *Data*, obtained using a hash algorithm like SHA2.
$\|Data\|^{K}$	*Data*, encrypted by a symmetric cipher using the key *K*.
$\|Data\|_{K}$	*Data* is authenticated using an HMAC algorithm with a cryptographic key *K*. This represents a message composed by two elements: *Data* and its cryptographic checksum.
$\|Data\|_{K1,K2}$	This is equivalent to $\|\|Data\|_{K1}\|^{K2}$
$\{Data\}_{X}^{-1}$	*Data* is signed using the private key of *X*.
$\{Data\}_{X}$	*Data*, encrypted for *X* using public key cryptography (RSA, ECDH, ECMQH,…). For computational efficiency, this is implemented using either a digital envelope (RSA) or an agreement exchange (ECC) as specified in [8],[17].
$\{\{Data\}\}_{X}$	*Data*, encrypted for *X* using public key cryptography (RSA, ECDH, ECMQH,…). Here, it is not used a digital envelope.
X =>Y	It indicates that *X* sends one message to *Y*.
[*Data*]	It indicates that this piece of *data* is optional, and it may not be in the message.
H(*Data*)	A message digest of *Data*, obtained using a hash algorithm as SHA2.
$\|Data\|^{K}$	*Data*, encrypted by a symmetric cipher using the key *K*.
$\|Data\|_{K}$	*Data* is authenticated using an HMAC algorithm with a cryptographic key *K*. This represents a message composed by two elements: *Data* and its cryptographic checksum.

3.2 Definition of a contract in SURENESS

For this protocol, a contract has the following structure:

$\{NID,A,B,Timestamp,Nonce,H(ContractDoc),H(SC_A),H(SC_B)\}_{TTP}^{-1}$, SC_A,SC_B

The contract is basically a document signed by the e-notary (TTP) and has the following information:

- *NID* (Negotiation Identifier). This is a unique identifier of the contract.
- *A*, *B*. They are the identifiers of the parties (Alice and Bob) between whom the contract is signed. This kind of identifier is the digest of the party's public key. It is used to avoid impersonation attacks, as we will see in section IV.
- *Nonce*. A nonce received from Alice.
- *ContractDoc (Contract Document)*. A document that reflects the agreed contract between Alice and Bob.
- *H(ContracDoc)*. It is the hash of the document which represents the contract.
- *H(SC_A)*, *H(SC_B)*. They represent the hashes of the contract signed by Alice and Bob, respectively. The content of SC_A and SC_B will be commented later. We can advance that it is basically the electronic signature of the contract with other information necessary to the protocol.

By means of this contract, the TTP testifies that Alice and Bob have reached an agreement or contract (identified by the NID). This is shown in a document that contains some information, such as the hash of the contract document (*H(ContractDoc)*), the hashes of the signatures made by Alice and Bob, and the corresponding signatures. It also contains information about when the contract is signed by the TTP (*Timestamp*).

To sum up, our contract is an electronic signature made by the TTP that links the contract signed individually for each party. Thus, the contract contains the signatures of the TTP and the participating parties, as supposed.

From this contract, we can not deduce if the e-notary knows the contract content or not. The e-notary will own a copy of the contract if Alice and Bob consider that the e-notary has to know it. For example, when the e-notary takes the responsibility for monitoring if the conditions of the contract are being complied with or not, or whether the e-notary has to register and save a copy of the contract [5, 15, 29], as occurs in real estate contracts.

In the following section, we will describe the contract signing protocol when there is a phase of negotiation. Then, in a subsequent section, we will present the description of the protocol when it is not possible, or it is not desired to negotiate the conditions of the contract. For example, if there is a pre-established contract.

3.3 Normal Mode

In the normal mode, the protocol is composed of the messages that appear in Figure 1. Each message is described in more detail below.

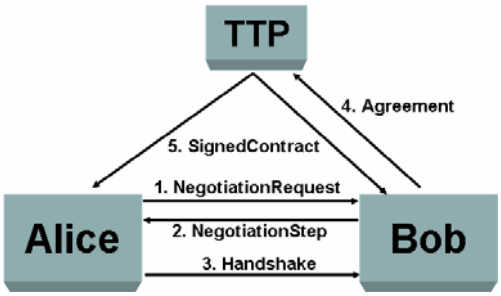

Figure 1. SURENESS messages in normal mode

STEP I. Alice=>Bob: NegotiationRequest.

$\{\{NID,Time_1,SeqN,[Cred],B,EnKey,SignKey,Flag\}_{A}^{-1},ContractDoc,\ H(ContractDoc)\}_{B}$

Where:

- *NID (Negotiation Identifier)*. This is a 4 byte value generated by Alice using a pseudo-random number generator. The *NID* identifies the negotiation being performed between Alice and Bob.

- *$Time_X$*. This is the time until which one of the parties will wait for a response from the other one. This time should include a date and hour. It will delimit the time of a possible response.
- *SeqN (Sequence Number)*. This is a value identifying the number of sequence of the message. It will be used as a nonce to avoid replay-attacks.
- *Cred (Credentials)*. This is an optional field which can be used to provide the user's credentials. For example, these could be a SAML Assertion or Artifact [22].
- *B*. Bob's identifier. It is the digest of his public key.
- *EnKey (Encryption Key)*. This is a symmetric key generated by Alice that will be used to provide confidentiality to the following messages exchanged between Alice and Bob. The default symmetric cipher to employ is AES.
- *SignKey (Signing Key)*. This is a symmetric key generated by Alice. It is used to provide integrity to the subsequent messages exchanged between Alice and Bob. The default cryptographic checksum function to employ is HMAC [19] with SHA2 [11].
- *Flag*. This is used to indicate if it is the last offer from Alice. If its value is true, it indicates that Alice will not accept counter-offers to this offer. Then, Bob can either accept the offer or finish the negotiation.
- *ContractDoc (Contract Document)*. This is a document that reflects a proposal of contract to negotiate between Alice and Bob.

The *NegotiationRequest* message indicates the beginning of the execution of the protocol. The message will be used by one of the parties when it decides to initiate the negotiation of the conditions of a contract in order to reach an agreement that will be reflected in a signed contract. The contract terms appear in the proposed contract document in the field called *ContractDoc*.

In this step, the contract document (*ContractDoc*) and *H(ContractDoc)* are not signed by Alice to avoid the abuse freeness, since if Alice signed these data, then Bob would be able to present them to another party and thus, Bob would gain an advantage over Alice.

STEP II. Bob=>Alice: NegotiationStep.

$|NID,Time_2,SeqN,[Cred],ContractDoc,Flag|_{SignKey,EnKey}$

Bob sends this negotiation message to make a new offer (counter-offer) if he does not agree with the contract terms. On the contrary, if Bob agrees to the conditions, he will send a *Handshake* message instead of this one. The *NegotiationStep* message will be used by Alice and Bob on many occasions until one of the following conditions is reached:

- One of them accepts the conditions of the other party in the last *NegotiationStep* message. In this case, a *Handshake* message will be sent.
- One of them receives a *NegotiationStep* message containing a last offer (flag is activated) that it does not agree with. In this case, the communication is closed.

STEP III. Alice=>Bob: Handshake.

$|RecKey|^{EnKey}, SC_A$

Where:

- SC_A= {NID,$Time_3$,B,TTP,{{Nonce,H(ContractDoc)}}$_{TTP}$, H(RecKey)}$_A^{-1}$
  SC_A represents the contract signing by Alice.
- *RecKey (Receipt Key)*. This is a symmetric key which will be used to receive the signed contract by the TTP. Its value comes from making the hash over the *Nonce*.
- *Nonce*. This is an array of bytes of variable length generated randomly.
- *$Time_3$*. This indicates the deadline until which Alice considers the agreement is possible with Bob. This time must be taken into account by Bob and TTP. Although Bob would sign his part of the contract, if it is signed after time *$Time_3$*, the TTP will not sign it and then there will not be a valid contract.

This is the step where Alice decides to accept the contract by sending it to Bob. However, in order to avoid Bob gaining an advantage over Alice, the signed contract is ciphered with the TTP's public key. The hash of the symmetric key, *RecKey*, is also sent signed, so Bob and the TTP could be sure the symmetric key was not changed by other party, because this key will be used to send, in a ciphered way, the signed contract in a ciphered way.

STEP IV. Bob=>TTP: Agreement.

There are two possibilities in this step:

a) $SC_A, \{SC_B\}_{TTP}$ or
b) $SC_A, \{SC_B\}_{TTP,}\ |ContractDoc|^{RecKey}$

Where:

- SC_B = {NID,$Time_4$,TTP,H(ContractDoc)}$_B^{-1}$
  This represents the contract signing by Bob.
- *$Time_4$*. This indicates the time until which Bob will wait for the contract signing. It is clear that this time should not be greater than *$Time_3$*. It has been introduced in case Bob wants to limit the deadline to sign the contract even more.

This message represents the agreement between the two parties for a contract. It contains part of the information received in the *Handshake* message (step III), i.e. the contract signing by Alice. Besides, it includes signed information by Bob that is sent ciphered to the TTP with its public key. This information reflects Bob's conformity with the contract. It is sent ciphered to the TTP in order to avoid Alice intercepting the message, which she could

show to another party to gain an advantage over Bob. There are two versions of this message. In the first, Alice and Bob decide that the TTP does not need to know the content of the e-contract (case a). But, in the second, they deem it necessary that the TTP knows the e-contract (case b). In order to avoid outside entities knowing the content of the contract, it is sent ciphered with the symmetric key (*RecKey*). When the TTP receives this message, then the TTP will check that the signatures are valid and that the contract document signed by each party is the same. In this case, the protocol continues in the following step, otherwise it finishes.

STEP V. TTP=>Alice, Bob: SignedContract.

$|\{NID,A,B,Timestamp,Nonce,H(ContractDoc),H(SC_A),H(SC_B)\}_{TTP}^{-1},\ SC_B|^{RecKey}$

Where:

- *Timestamp* indicates the date and the time when the TTP signed the contract, once Alice and Bob had reached an agreement. This time should be less than the minimum between $Time_3$ and $Time_4$.

The message represents the approval of the signed contract between Alice and Bob. It is signed by the TTP in order to prove that both entities agreed to the contract reached. Bob's signature is included so that Alice can have a copy of the signed contract by Bob. Thus, in case of dispute, the TTP will not be necessary.

3.4 Abort Protocol

We have also defined the messages needed, supposing a party that has signed a contract wants to cancel it prior to its being signed by the TTP. These messages are:

STEP I. Alice=>TTP: Abort.

$\{NID,B,TTP,\{Nonce,H(ContractDoc)\}_{TTP}\}_A^{-1}$

If the contract has not yet been signed, the step II of the abort protocol will be executed. Otherwise, the last message of the execution of the protocol will be re-sent (*SignedContract*).

STEP II. TTP =>A,B: ConfirmedAbort.

$\{NID,Timestamp,A,B,Abort\}_{TTP}^{-1}$

This message will be received as confirmation that the protocol was aborted. This is indicated by the flag *Abort*.

3.5 Resolve Protocol

It could occur when one party, after having sent its contract signature to the other one or to the TTP, after the deadline, does not receive any answer. We will comment on the sequences of messages to exchange supposing that it was Alice who did not receive the answer.

STEP I. Alice=>TTP: Resolve.

$\{\{NID,Time_X,B,TTP,Nonce,H(ContractDoc),RecKey\}_A^{-1}\}_{TTP}$

Where $Time_X$ is the time indicated in the *Handshake* or *Agreement* message. The time, when this message is sent, should be later than $Time_X$. Depending on the messages sent (or not) to the TTP, in the execution protocol, Alice could receive one of the following messages:

a) STEP II. TTP =>Alice: SignedContract.

This message is the same as we commented in step V of the protocol and it would be received if the contract was signed and sent to the TTP by Bob.

b) STEP II. TTP =>Alice: ConfirmedAbort.

$\{NID,TimeStamp,A,B,Abort\}_{TTP}^{-1}$

This message will be received if Bob cancels the signing of the contract.

c) STEP II. TTP =>Alice: NoContract.

$\{NID,A,B,Timestamp\}_{TTP}^{-1}$

If after the $Time_X$, the TTP has not received an abort or a signed contract, TTP sends a signed message to Alice indicating that there was no agreement.

3.6 Aggressive mode

The aggressive mode is useful in those cases where there is a predefined contract in which the parties already know its content, but some minimal details about it, such as dates, names of the parties and so on, have to be filled in. In this mode, the messages would be the following:

STEP I. Alice=>Bob: AgreementRequest.

$\{\{NID,Time_1,[Credentials],B,EnKey,\ SignKey\}_A^{-1}\}_B,ContractDoc|_{SignKey,EnKey},SC_A$

In this message, Alice sends an agreement request with the signed contract.

STEP II. Bob=>TTP: Agreement.

a)
$SC_A,\{\{NID,Time_4,TTP,H(ContractDoc)\}_B^{-1}\}_{TTP}$

or

b)
SC_A,{{NID,$Time_4$,TTP,H(ContractDoc)}$_B^{-1}$}$_{TTP}$,
|Contract|^{RecKey}

STEP III. TTP=>Alice,Bob: SignedContract.

|{NID,A,B,TimeStamp,Nonce,H(ContractDoc),H(S
C_A),H(SC_B)}$_{TTP}^{-1}$, SC_B|^{RecKey}

4. ANALYSIS AND COMPARISON WITH RELATED WORK

In this section we are going to analyze the protocol from different points of view: security, requirements mentioned in section II, and comparison with the protocols commented in the section related work.

- *Replay attacks*. The protocol has been designed to avoid the problem of replay attacks. This problem is avoided thanks to the use of the fields *NID* and *SeqN*. *NID* is the identifier of the transaction and represents an execution of the protocol, while *SeqN* is the number of negotiation messages within the execution. Each time a new message is received, this number is increased. Thus, if a message with an old number is received, it will be rejected. Similarly, if a message with a correct sequence number, but from another negotiation, is received (a message with a *NID* previously used), the message will be rejected.
- *Timeless*. The protocol has the ability to stop, in a finite amount of time, its execution while at the same time it preserves the fairness. This property is assured thanks to the $Time_X$ fields that we have included in the messages of the protocol in order to limit the reception time of a message. Thus, when an entity receives a message, it checks that the actual time is later than the time indicated in $Time_X$, then the entity discards the message. The contract signing process is also atomic, since the contract is either valid or not valid at all after the time-window $Time_X$ has expired.
- *Impersonation*. Impersonation attack tries to convince some protocol party that the communication is being performed only between Alice and Bob, although there is a third party participating in the communication (impersonating Alice or Bob). We have included the identifier of the parties in the messages, so as to avoid this problem, as is postulated in [3].
- *Confidentiality*. During the execution of the protocol, we have used symmetric and asymmetric ciphering algorithms to ensure the confidentiality of the information exchanged. In all the messages, as far as possible, we have used symmetric cryptography for the sake of efficiency, especially in the negotiation phase (steps II and V). In the cases where there was no previous contact between the entities, it was necessary to use asymmetric cryptography (steps I, III and IV).
- *Abuse freeness*. Each party sends the TTP the signed contract, ciphering it with the TTP's public key. Therefore, neither is Alice able to show to an outside party that Bob signed the terms of the contract, nor can Bob prove that Alice signed the same contract he did. In step III of the protocol, Bob receives a signed message; however, the signature can not be linked with the content of the contract since the hash of the contract is ciphered with the TTP's public key. In order to show it to an outside party, Bob could try to make the envelope with the hash of the contract that he knows, which is the same as Alice knows. However, he does not know the field *Nonce*, so the envelope would not match and he would not be able to show it to an outside party. Likewise if Alice intercepts the message between Bob and the TTP. The TTP is the only entity that can create the contract from the individual signature of each party, showing that each party really signed it. As far as the protocols presented in section II are concerned, none of them satisfy this requirement. However, the new protocol presented here guarantees that none of the parties gets a copy of the contract until both parties have signed it.
- *Non-repudiation and dependency of existence*. Once the TTP receives the signature of the contract from Alice and Bob, it generates a signature to validate the transaction and to relate the information signed by the entities. The TTP signs the transaction identifier, a timestamp indicating the moment of the registry of the contract, the nonce contained in the envelope of the signature of the contract by Alice, and the hash of the contract signature by Alice and Bob. With this information, even if the TTP were not available, a third party would be able to check the validity of the contract. The steps would be the following: firstly, it would verify the TTP's signature; secondly, it would verify the Bob's signature. Next, it would check that the hashes of the contract are the same. Then, from the nonce inserted in the TTP's signature, and from the hash of the contract, it could calculate the envelope if the TTP in which Alice's signature is. Finally, it would check Alice's signature.
- *Recipient role*. The contract negotiation is initiated by one of the parties. However, depending on the steps followed in the negotiation, the recipient role could become the sender if the recipient accepts the contract proposal. In this case, the recipient could send the message III to the sender thus becoming a sender. The role of recipient and sender are therefore symmetric in our protocol. Both have the control of the protocol since, without both signatures, the final contract is not possible.
- *Secure Negotiation*. The protocol, unlike the ones mentioned in Section 2, incorporates the possibility of negotiating the contract in a secure way and allowing none of the parties to gain an advantage over the other one. Besides, the negotiation does not suppose an excessive overload, since it uses symmetric cryptography.
- *Efficiency*. The protocol would be more efficient if the TTP only participated when there is a problem (optimistic TTP). However, as we commented in the introduction, our goal is to provide a protocol for those scenarios where the TTP has to participate in the contract signing, for example, by legal requirement. Since the TTP has to

participate, we have to aim to make the number of cryptographic operations (especially the asymmetric ones due to their computation cost) as well as the messages to be sent and received by the TTP be the minimum possible. In order to have a reduced computational cost, given the cryptographic operations, the protocol uses mainly symmetric cryptography (cipher, hash,…), except in the operations related to non-repudiation or when it is necessary to send information to other parties that have no previous contact, where asymmetric cryptography is used instead. The protocol, as far as the messages exchanged between all the parties is concerned, is less than any of the other protocols already mentioned. In our protocol, six messages are sent and received. In comparison, protocol 1 needs ten messages and protocol 2 twelve. Besides, if we compare the number of messages where the TTP participates, we can see that in our protocol, the TTP sends two messages and receives only one message. But in protocol 1, the TTP receives six messages and sends four. In protocol 2 the TTP receives four messages and sends six messages. Therefore, our proposal is clearly better than the previous ones because it has a lower overload both the in number of messages and cryptographic operations. Even, if we compared our protocol with only one iteration of protocol 2 (which fulfils similar features to these proposed here), our protocol has the same asymmetric cryptographic operations and the number of messages is lower. In our proposal the TTP participates in three messages unlike in protocol 2, where the TTP participates in five messages.

- *Trust dependency on a third party*. Unless the signers of the contract decide that the TTP should know the contract, the TTP will only know the contract hash. Thus, in our case, the trust dependency on a third party is minimal, unless they decide to deposit more trust or it is a requirement of the application or the environment which they are working in [5], [15], [29].

Apart from these features and since we are considering cases where the presence of a TTP on-line is mandatory (most of the cases by legal requirement), we could have problems in the service either if the TTP is not available or if TTP has to support many concurrent contract signings. Thus, the TTP is a possible single-point-of-failure in a contract signing protocol. Nowadays, this problem could be solved with replication techniques, load balancing servers or solutions based on grid [28],[30],[36]. In any case, if the TTP is not available at the contract signing moment, the parties could try to send the messages later. When the deadline indicated in the $Time_X$ is arrived, then the protocol is aborted. This mechanism allows us to preserve fairness and timeless properties.

Finally, we present two tables where the cryptographic operations made for each party, for each protocol, appear. In these tables, the three first operations are according to the numeration used in the section about related work, and our protocol is shown as the third protocol.

TABLE 2. Asymmetric cryptographic operations of the protocols in a contract signig

Protocol	Entity	Signature & Verification	Dual Signature & Verification	Encryption & Decryption
1	A	6		
	B	6		
	TTP	4		
2	A	4	2	4
	B	6	2	2
	TTP	6	2	2
3	A	4		2
	B	5		2
	TTP	3		2

TABLE 3. Symmetric cryptographic operations of the protocols in a contract signig

Protocol	Entity	Encryption & Decryption	Hash
1	A	1	1
	B	1	
	TTP		
2	A		2
	B		
	TTP	2	3
3	A	2	4
	B	2	
	TTP	2	2

As a conclusion to this analysis, we can affirm that we have proposed a protocol that satisfies the requirements established in section II. Besides, our protocol is more efficient than the previous works commented on in section II. Efficiency is measured with regard to the number of cryptographic operations and the number of messages that the TTP has to take part in. From the tables and the analysis made in this section (in the efficiency part), we can conclude that we have reduced the workload of the TTP compared to the previous proposals. Additionally, we could use replication techniques, load balancing server and cluster solutions to achieve a major availability and a better service in the TTP.

5. IMPLEMENTATION

In this section we comment the most relevant details about the implementation we have made in order to make use of the protocol. The development of the implementation was made according to the definition commented in section 3. We designed and implemented the messages of the protocol using XML schemas. The signature and encryption formats chosen were XML signature and encryption, respectively, since it is more comfortable to work with XML structures than DER-format because XML is more readable and it is designed to exchange data on the Web. These formats are also fully integrated in the application development based on XML. We made use of Java as programming language, due to its oriented object programming properties and the portability of the generated code to other platforms. As for the libraries used in the project, we have worked

with Xerces for the processing of XML data, and the XML Security Suite from IBM (although at this moment we are migrating to Apache XML Security) to work with XML signature and encryption. Finally, as cryptographic provider we are using the libraries from IAIK.

In the implementation of the protocol, there are three main classes: *Fields*, *MessagesContent* and *Messages*. All classes that represent a field in the content of a message inherit from *Fields*. *MessageContent* represents a group of several fields that are within a message. Finally, *Message* is the parent class that all the messages that are part of the protocol inherit from, as we specified in section III.

The implementation of the protocol has been applied to a DRM environment which aims to negotiate and purchase author rights and multimedia content. This scenario will be dealt with in the following section.

In the implementation of the scenario that we are going to see, we have made use of intelligent agents. The agent platform used is JADE [16] since it claims to comply with the FIPA [12] specifications. In the interactions between the agents we have used several FIPA protocols such as FIPA Brokering, FIPA iterated-contract-net, FIPA query and FIPA request. These protocols allow agents to negotiate, query and request information among themselves. Apart from the behaviors offered by JADE, we needed to develop two new ones, called: *OneShotSimpleAchieveREResponder* and *CustomContract-NetResponder*. The first is similar to the behavior *SimpleAchiveREResponder* but the message does not need an answer. The second is the finite version of the *ContractNetResponder* behavior. Besides, we have designed and implemented an ontology for the agents called *SECURingIPR_Onto*. The process of design and implementation has been supported by Protégé and its *BeanGenerator*. In this ontology we have defined the concepts, the actions and the predicates that the agents work with.

6. NEGOTIATION AND SIGNING OF AUTHOR RIGHTS AND MULTIMEDIA CONTENT

Nowadays, one of the major problems for authors, publishers and distributors of digital content is the simplicity with which people can copy a work without payment of the royalties derived from the use of the rights of this work. The Digital Rights Management (DRM) was proposed with the aim of representing, negotiating and controlling the Intellectual Property Rights (IPR) associated to copyrighted material. SECURingIPR (SECure infrastructure for negotiation and purchase of Intellectual Property Rights) [29] is a DRM proposal where the business model was designed to develop an architecture to negotiate, reach agreements and purchase copyrighted content in a secure way.

Our model guarantees that the flow of information and rights from creators to end users will be transferred in a secure way, using an open DRL such as ODRL [32] to achieve the understanding among the different parties involved in content life cycle. This language allows us to automate tasks, since it will be unambiguous, precise and machine-readable. The model mentioned is shown in Figure 2.

In this model, a work that arrives to the end user from the creator has to follow the lifecycle that we will outline briefly. More details can be found in [29]. Usually, when a creator makes a new creation, he has not enough means to carry its edition and distribution, or the work could have been made on demand of an editor. This lifecycle is initiated with the contract signing between the creator (or a party, who represents him, called rights holder) and the editor. This contract is called author's contract. With this contract, the creator negotiates the conditions and rights under which its work will be edited and distributed as well as the royalties he will receive. On the other hand, the editor gets the enough rights to initiate the production of the work. Later, the editor will reach an agreement with one or more distributors in order to make the work purchasable to the end users. This distribution agreement is reflected in a contract called distribution contract. In this contract the rights in the work are distributed and the royalties each party receives are specified.

In this lifecycle, the negotiation and the contract signing among the different parties is done using the protocol presented in this paper (see Figure 1). In this model, the e-notary is in charge of giving validity to the e-contracts. One of its services, the Monitoring Service Provider will monitor that the specified conditions and rights in the contract are being accomplished and that none of the parties is breaking the established agreement. In this model, the e-notary will act as a TTP in the contract signing protocol. Although the protocol is designed in such a way that every party could be the recipient role, in this scenario the owner's rights will act as the recipient role. Thus, in the author's contracts, Alice will be the Creator/Rights Holder and Bob will be the Editor/Publisher. In distribution contracts, Alice will be the distributor and Bob will be the Editor/Publisher.

In the execution of the protocol, the Contract Document (*ContracDoc*) will be an ODRL expression. The current version of ODRL only supports *Offer* and *Agreement* expressions. The future version of this language will also include a *Request* expression in order to allow the complete negotiation of rights as we conceive in this protocol, where the two parties can propose offers and counter-offers to a proposal. Meanwhile, we decide to extend ODRL with such expressions as to be able to negotiate the rights. We have extended it taking into account the compatibility with the new possible extension of ODRL. Thus, in the *NegotiationRequest* message or in the *NegotiationStep* sent by Alice, the *ContractDoc* field will be the new ODRL *Request* expression. On the other hand, Bob in the *NegotiationStep* will put an ODRL *Offer*.

Furthermore, in the infrastructure mentioned here, we have incorporated the use of intelligent agents in such a way that they will act on behalf of the different users in the system. The agent platform used is JADE since it claims to comply with the FIPA specifications, as we commented in the previous section. There is

an agent for each role in the system with the implementation of the behaviors they need: search of agents, contract signing protocol with ODRL and/or payment protocol with ODRL.

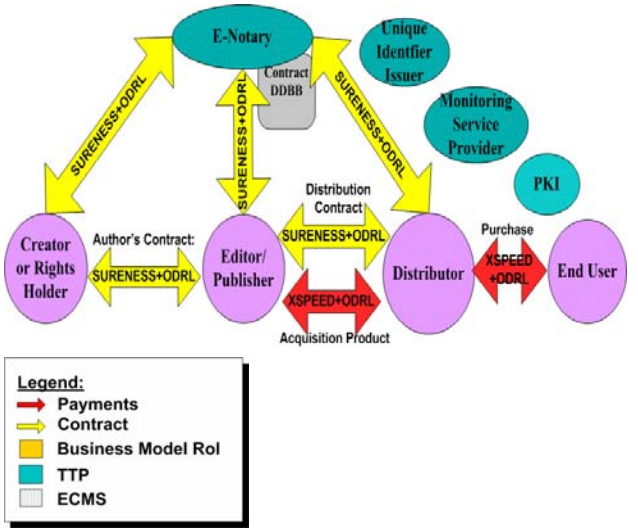

Figure 2. SECURingIPR Business Model

7. CONCLUSION AND FUTURE WORK

In some B2B and DRM systems, or by legal requirement, the use of contract signing protocols based on a TTP is required in order to give validity to e-contracts. After studying this kind of protocols, and having found some deficiencies, we decided to propose a new contract signing protocol, which we have presented here. In section I, we introduced the problem in a detailed way. Later in section II, we mentioned the main properties required to this kind of protocols such as the traditional ones: non-repudiation, fairness, efficiency, completion. Other additional requirements are abuse freeness, trust dependency on a third party, existence dependency and recipient role. We have also introduced two new requirements: confidentiality and secure negotiation.

As we have said throughout this paper, there are tow main reasons for proposing a new contract signing protocol based on a TTP online: on the one hand, the incorporation of some fundamental aspects for B2B and DRM that we have already mentioned: confidentiality and the secure negotiation of the contract conditions. On the other hand, as in the scenarios commented the participation of the TTP is mandatory (mainly by legal requirement), we have seek to offer a protocol that complies with the main security features related to non-repudiation as well as reducing the TTP's workload. Confidentiality has been added since it is important to achieve the privacy of the information exchanged between the different parties that sign a contract. Thus, only the interested parties will know the information about the protocol. We also consider that the negotiation phase is intrinsic to the contract signing and this phase should be made in a secure way, guaranteeing aspects such as confidentiality and integrity and avoiding replay-attacks. It is also important to note that the protocol could be used without generating new keys and certificates, unlike other protocols that require either specific keys or generating new ones with special features, or a previous registration with a TTP. The only requirement is that they can make two asymmetric operations: signing and ciphering. Thus, the protocol has been conceived to make use of the keys and certificate that users have. Therefore, it could be used with RSA keys or with ECC keys, which, more and more, are being used by mobile devices.

As regards the protocol, in section III we have described the different messages that are part of it. Then, in section IV, we have analyzed its security characteristics and we have compared it with the existing protocols. This comparison has shown how our protocol improves the previous ones with the new requirements and with a good efficiency for the cryptographic operations and the number of messages. Besides, in order to avoid the TTP becoming a single-point-of-failure, we have proposed the use of techniques such as replication, load balancing or grids. Finally, in order to use its features, we have described a scenario where it is being used in conjunction with ODRL to negotiate and purchase intellectual property rights and multimedia content.

There are a number of future research directions. One issue would be the extension of the contract negotiation in an environment where more than two parties participate, that is, the extension of the protocol in a multi-party contract environment but maintaining the idea of the involvement of the e-notary giving validity to the transaction. A possible scenario would be the buying of a real estate among several clients and several vendors, and where the e-notary validates the transaction by signing it as specified by the laws. Another possible research work would be a language to express the content of any contract in an e-commerce transaction. Finally, we are thinking about extending our infrastructure to a mobile scenario based on OMA [26].

8. ACKNOWLEDGMENTS

This work has been partially supported by PROFIT VIDIOS FIT-330220-2005-74 project.

9. REFERENCES

[1] AA.VV., "Código Civil", Boletín Oficial del Estado (26ª Ed)., 2005.
[2] Abadi, M., Glew, N., Horne, B. and Pinkas, B. "Certified email with alight on-line trusted third party: design and implementation", The Eleventh Internation World Wide Web Conference, Honoloulu, Hawaii, USA, 2002.
[3] Abadi, M., Needham,R.: Prudent engineering practice for cryptographic protocols, IEEE Transactions on Software Engineering, 22(1):6-15, January 1996.
[4] Aknine, S., Pinson, S., and Shakun, M.F.: An Extended Multi-Agent Negotiation Protocol. Journal of Autonomous Agents and Multi-Agent Systems, 8, 5– 45, Springer Science+Business Media B.V., Formerly Kluwer Academic Publishers B.V. 2004.
[5] Angelov, S., Till, S., Grefen, P.: "Dynamic and Secure B2B E-contract Update Management." Proceedings of the 6th ACM Conference on Electronic Commerce (EC'05), Vancouver, Canada. 2005.
[6] Asokan, N., Schunter, M., Waidner, M. Optimistic protocols for fair exchange. 4th ACM Conference on Computer and Communications Security, ACM Press, Zurich, Switzerland, 1997, pp. 6, 8-17.

[7] Asokan, N., Shoup, V., Waidner, M.: Asynchronous protocols for optimistic fair exchange. Proceedings of the IEEE Symposium on Research in Security and Privacy, 1998, pp. 86-99.
[8] Blake-Wilson, S., Lamber, P. “Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)”, Request for Comments 3278, April 2002.
[9] Chadha, R., Kremer, S., and Scedrov, A.: Formal analysis of multi-party fair exchange pro-tocols. 17th IEEE Computer Security Foundations Workshop, pages 266-279, Asilomar, CA, USA, June 2004. IEEE Computer Society Press.
[10] Dierks, T., Allen, C.: The TLS Protocol Version 1.0. RFC 2246. January 1999.
[11] Federal Information Processing Standards Publication (FIPS PUB) 180-2, Secure Hash Standard, 1 August 2002.
[12] Foundation for Intelligent Physical Agents (FIPA). http://www.fipa.org/
[13] Frier, A., Karlton, P., Kocher, P.: The SSL 3.0 Protocol, Netscape Communications Corp., Nov 18, 1996.
[14] Garay, J., Jakobsson, M., and P. MacKenzie. Abuse-free optimistic contract signing. In proceedings of CRYPTO'99, Lecture Notes in Computer Science 1666, pp. 449-466. Springer-Verlag, 1999.
[15] Jalali, M., Hachez, G., Vasserot, C.: FILIGRANE: a security framework for trading of mobile code in Internet, Autonomous Agents 2000 Workshop: Agents in Industry, Barcelona, Spain, 2000.
[16] Java Agent DEvelopment Framework (JADE). http://sharon.cselt.it/projects/jade/
[17] Housley, R. “Cryptographic Message Syntax (CMS)”, Request for Comments 3852. July 2004.
[18] Kim, K. Park, S. and Baek. J. “Improving fairness and privacy of Zhou-Gollmann’s fair non-repudiation protocol”, Proceedings of 1999 ICPP Workshops on Security (IWSEC), IEEE Computer Society, Sep. 21-22, pp.140-145. 1999.
[19] Krawczyk, Bellare, H. M., Canetti, R.: HMAC: Keyed-Hashing for Message Authentication. RFC 2104. February 1997.
[20] Kremer, S., Markowitch, O., and Zhou, J. An intensive survey of fair non-repudiation protocols. Computer Communications, 25(17): 1606-1621. Elsevier, Nov. 2002.
[21] Kötz, H., European Contract Law: Formation, Validity, and Content of Contracts; Contract and Third Parties 78 (Tony Weir trans., Hein Kötz & Axel Flessner eds., 1997).
[22] Maler, E., Mishra, P., Philpott, R.: Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1, September 2003. OASIS Standard.
[23] Markowitch, O., Gollmann, D., and Kremer, S. On fairness in exchange protocols. In Pil Joong Lee and Chae Hoon Lim, editors, 5th International Conference on Information Security and Cryptology (ICISC 2002), volume 2587 of Lecture Notes in Computer Science, pages 451-464, Seoul, Korea, November 2002. Springer-Verlag.
[24] Markowitch, O., Kremer, S. Fair Multi-Party Non-Repudiation Protocols. International Journal on Information Security 1(4), pages 223-235, 2003.
[25] National Institute of Standards and Technology, "Specification for the Advanced Encryption Standard (AES)" FIPS 197. November 26, 2001.
[26] Open Mobile Alliance: DRM Architecture Draft Version 2.0. http://www.openmobilealliance.org/release_program/drm_v20.html
[27] Park, J. M., Chong, E., Siegel, H. J., Ray, I.: Constructing fair exchange protocols for e-commerce via distributed computation of RSA signatures. In Proc. of 22th Annual ACM Symp. on Principles of Distributed Computing (PODC’03), pp. 172-181. ACM Press, 2003.
[28] Rabinovick, M. and Spatscheck “Web caching and replication”, Addison Wesley, 2002.
[29] Ruiz, A., Baño, L., Marín, C.I., Cánovas, O., and Gómez, A.F.: A Secure Infrastructure for Negotiation and Purchase of Intellectual Property Rights (SECURingIPR). Proceedings International Conference on Communication, Network, and Information Security, New York USA, 2003.
[30] Schroeder, T., Goddard, S., Ramamurthy, B. “Scalable Web Server Clustering Technologies,” IEEE Network, May/June 2000, pp. 38-45.
[31] Wang, G. An abuse-free fair contract signing protocol based on the RSA signature. Proceedings of the 14th international conference on World Wide Web, 412 – 421, Chiba, Japan. 2005.
[32] Word Wide Web Consortium (W3C), ODRL, W3C Note, 2002. http://www.w3.org/TR/odrl/
[33] Word Wide Web Consortium (W3C). XML-Signature Syntax and Processing. W3C Recommendation. 12 February 2002.
[34] Yang, S., Su, S. Y.W., and Lam, H. A Non-Repudiation Message Transfer Protocol for E-commerce, p. 320. IEEE International Conference on E-Commerce Technology (CEC'03), 2003.
[35] Yang, S., Su, S. Y.W., and Lam, H. A non-repudiation message transfer protocol for collaborative e-commerce. International Journal of Business Process Integration and Management 2005 - Vol. 1, Nº.1. pp. 34 – 42.
[36] Zegura, E., Ammar, M., Fei, Z., Bhattacharjee, S. “Application Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service,” IEEE/ACM Transactions on Networking, Vol. 8, No. 4, August 2000, pp. 455-467.
[37] Zhou, J., and Gollmann, D.: A Fair Non-repudiation Protocol, proceedings of 1996 IEEE Symposium on Security and Privacy, pp. 55-61, Oakland, CA, May 1996.
[38] Zhou, J.: On the Security of a Multi-Party Certified Email Protocol, Lecture Notes in Computer Science 3269, Proceedings of 2004 International Conference on Information and Communications Security, pages 40--52, Malaga, Spain, October 2004, Springer.