Fair and Abuse-free Contract Signing Protocol
Supporting Fair License Reselling

Tarek Gaber
School of Computer Science
University of Manchester
Manchester, UK
gabert@cs.man.ac.uk

Ning Zhang
School of Computer Science
University of Manchester
Manchester, UK
nzhang@cs.man.ac.uk

Abstract—Most of the fair contract signing protocols published to date make use of a Trusted Third Party (TTP) to achieve fairness. In this paper, we have designed a fair contract signing protocol to support fair reselling of a DRM license without using a dedicated TTP. This protocol makes use of the concurrent signature (CS) and the existing license distribution infrastructure. By making use of the CS scheme, and integrating it into the existing license distribution infrastructure, we avoid the use of a dedicated TTP, thus introducing no additional communication overhead in providing fair license reselling. Also, the protocol is designed such that none of the two signers can prove to an outside entity that he is in control of the outcome of the protocol, thus achieving abuse-freeness.

Index Terms—Contract Signing, Concurrent Signatures, Fairness, Abuse-freeness, DRM, Reselling Deal, Non-repudiation

I. INTRODUCTION

Digital Rights Management (DRM) is a technology that allows content owners to manage usage rights over their digital contents. Examples of these rights include copy permit, pay-per-view, one-week rental, etc. With the DRM technology, consumers can only access a digital content if they have purchased a digital license corresponding to this digital content from a License Issuer (LI). However, with the current DRM technology, consumers are not allowed to resell their licenses. Reselling something that a consumer rightfully owns (including digital licenses) is a legitimate right under the first sale doctrine [4]. To support this legitimate right, a number of license reselling proposals [5], [6], [7] have been published in the literature. Nonetheless, these proposals have not addressed the property of fairness in a license reselling process. Fairness here means that, at the end of a license reselling process, either a buyer obtains a license which he has paid for and a reseller receives a payment, or neither party gets anything useful.

In [8], we have proposed a method that allows a reseller to fairly exchange his DRM license for a payment from a buyer. The method comprises four phases: Reselling Deal (RD) creation, RD signing, RD activation-request, and RD activation. In the first two phases, a reseller (Alice) and a buyer (Bob) negotiate and electronically sign a contract known as RD. In the last two phases, Bob then sends an RD activation-request along with a payment to LI who activates the signed RD in the activation phase. In this phase, LI sends Bob an activated license and sends Alice the payment. To fairly sign a given RD, Alice and Bob have to execute a fair and abuse-free contract signing protocol. The abuse-freeness property [9] is required in our case to prevent Bob (the second signer) from using the RD only signed by Alice (the second signer) to gain any advantage by showing this RD to another reseller.

Fair contract signing protocols typically use one of two approaches: gradual release of secrets and optimistic exchange. The protocols in [10], [11] use the former approach to achieve fair exchange. These protocols typically work as follows. Two entities first engage in dividing their signatures into N parts, each of which being a verifiable component. They then exchange their respective signatures part-by-part. To achieve fairness, it is assumed that the entities involved have equal computational power.

The gradual secret release approach is not suited to sign a RD contract in a reselling process described in [8]. The reason for this is twofold. Firstly, it is not practical to assume that both a reseller and a buyer have equivalent computational power. If one of them has a superior computing capability may terminate the protocol execution prematurely and use his resources to compute the remainder of the other entity’s signature. As a result, the other entity could be left in a disadvantageous position. Secondly, this approach requires: (1) N message flows to exchange the N parts of the entities’ signatures, and (2) a high computational cost to compute and to verify each of the exchanged parts. Thus, it is inefficient in terms of communication overheads and computational cost.

The optimistic exchange protocols in [12], [13] use a dedicated TTP to achieve fairness. The main idea behind this class of protocols is that a dedicated TTP is invoked to provide fairness if the two communicating entities failed to do so. (For more detail about fairness protocols, see [14]). Although the idea is simple, these protocols are subject to a number of problems. Firstly, the TTP may become a performance and security bottleneck leading to the denial of service if the TTP is congested or attacked. Secondly, when the TTP is involved in a protocol run, it decreases its efficiency and increases the cost of a transaction. Last but not least, it may be difficult to find a third party which is trustworthy and could serve as TTP.
More recent optimistic exchange protocols [15], [9], [16] make use of bilinear pairing primitives to achieve fairness. As reported in [15], [9], these protocols are efficient in terms of communication. However, they still make use of a dedicated TTP which could lead to the above problems.

In a DRM infrastructure, there is already a trustworthy entity, i.e. LI, to support a license reselling process [8]. Introducing another TTP in order to support fairness in this infrastructure would add more cost into a transaction. One could say that LI itself could be used to serve as a TTP in the signing process to help to achieve fairness. Yes, this should be planned out carefully. Otherwise, the following scenario may occur. For example, if the TTP-based protocol proposed by Nenadic et al [17], (the authors report that this protocol is efficient in comparison with related work), is used, and LI serves as the TTP during the signing process, LI will need to perform the following additional tasks: (1) prior to executing a signing process, LI has to issue and sign a special digital certificate to certify additional public/private key pair for the initial signer; (2) in case of a dispute, LI has to perform a number of signature verifications in order to resolve the dispute [17]. These tasks would add on to the existing workload which LI is already performing in the current license distribution processes.

In addition, if Nenadic’s protocol is used to sign a given RD, the communication overhead of the reselling process would also be increased. This is because: (a) Nenadic’s protocol requires 7 messages: 4 messages for main exchange protocol and 3 for recovery protocol; (b) further messages (at least 2 messages) are required for the initiator to get a digital certificate from LI before he initiates a protocol execution with the other party. Thus, if the RD contract signing process is performed normally between a reseller and a buyer, LI must engage in the 2-message protocol to issue the reseller (the initiator) the special digital certificate. If there is any dispute, LI will need to engage in executing the recovery protocol consisted of additional 3 messages. This means that LI, in the worst case scenario, will need to send and receive 5 messages during the process of signing an RD. This would increase the communication overhead of a license reselling process.

In [15], Zhang et al have proposed an optimistic exchange protocol based on bilinear pairing. They have also proved that their protocol is more efficient than Nenadic’s one [17]. However, if Zhang’s protocol is used to sign a given RD in a reselling process, the communication overhead would be increased as well. This is because Zhang’s protocol requires three sub-protocols to achieve fairness. It requires 4 messages for the main exchange protocol, and additional 3 messages for both the abortion and dispute resolution. In other words, if this protocol is used in our case [8], LI may have to execute a protocol of 3 messages either for abortion or for dispute resolution.

The communication and the processing overheads imposed on LI can be reduced if the RD signing protocol makes use of the Concurrent Signature (CS) scheme [18]. This scheme is proposed to achieve fair signature exchange. This scheme does not require the signers to have the same level of computational power as in case of gradual release approach, nor does it require any assistance of a TTP. However, as reported in [18], the CS scheme can only provide a weak fairness. It can provide strong fairness, under two conditions: (1) the initial signer releases a secret token, called a keystone, ks, at the end of the exchange, and (2) he does not abuse a pre-binding token signed by the other signer before the completion of the exchange process (i.e. it can provide the abuse-freeness property).

This paper is a continuation of the work presented in [8]. It presents a RD-contract Signing (RDS) protocol to support fair RD signing process in reselling a license of DRM-protected content. This RDS protocol makes use of the CS scheme to achieve fairness, but overcomes the two weaknesses of the CS scheme (described above) by integrating it with the existing license distribution infrastructure (i.e. LI) that issues licenses (digital content usage permissions). LI’s role in the RD signing process is the same as the role played by LI in the license reselling process described in [8]. In other words, LI is only involved in one case which takes place if the RD has been signed; and Bob (the buyer) wants to activate the license stated in the signed RD. In this case which is called RD activation process, LI is involved to receive a payment from Bob, verifies the signatures on the RD. LI then sends Bob the license and sends Alice the payment. This means that the RD activation process enables LI to handle any misbehaving that could happen by either Alice or Bob.

The RDS protocol can also be used to enable a consumer, C1, who is subscribed to access a media website, to fairly resell her access permissions to another consumer, C2. In such scenario, C1 can use the RDS protocol to sign a deal with C2. This signed deal can then be used by C2 to claim C1’s access permissions from the owner of the website. This can only be done should C2 make a payment stated in the signed deal to the owner. Also, C1 can use this deal to claim the agreed payment paid to the owner by C2.

The main contribution of this paper is twofold. Firstly, to the best of the authors’ knowledge, this is the first paper that integrates the features of the CS scheme with what is already available in the existing license distribution infrastructure (i.e. LI) to support fair RD-contract signing process in a license reselling process. By this integration, we are able to make use of the strengths of both the CS scheme and the LI while overcoming their weaknesses, thus achieving a secure license reselling with strong fairness and abuse-freeness protections. Secondly, the RDS protocol design does not require a dedicated TTP to achieve fairness. Rather it embeds some additional functionalities required to ensure fairness and legitimacy into LI. In this way, the additional computational cost introduced as the result of introducing security can be kept at a minimum.

The remainder of this paper is structured as follows. The protocol preliminaries (requirements, notations, and assumptions) are presented in Section II. An overview of the CS scheme is given in Section III. In Section IV and V, an
overview and a description of the RDS protocol are given, respectively. In Section VI, an analysis of the protocol is presented. Finally, the paper is concluded in Section VII.

II. PRELIMINARIES

A. Design Requirements

The RDS protocol is designed to satisfy the following requirements.

(R1) Non-repudiation of signature origin - the recipient of a signature is assured that the signature is indeed generated by the claimed signer.

(R2) Non-repudiation of signature receipt - the sender of a signature is provided with evidence that the intended recipient has indeed received this signature.

(R3) Abuse-freeness: before the completion of a protocol execution, neither of the signers can prove to an outside entity that he is in control of the outcome of the protocol execution.

(R4) Strong fairness: upon the completion of a protocol execution or upon an acceptance of an RD activation request, either both the signers are committed to the contract or neither is.

B. Notations

The notations used in the RDS protocol are given below.

1) A, B, and LI: Alice, Bob and License Issuer.

2) Lic: A resalable license that Alice has bought from LI and she wants to resell it to Bob.

3) RD: A contract, called Reselling Deal (RD), that has been agreed by Alice and Bob.

4) $ks$ and $f$: Keystone and its keystone fix, respectively. They are used in the signing process.

5) $RP_{Lic}$: A reselling permission issued with Lic to prove that Lic is resalable. It is of the form $RP_{Lic} = \{Lic|f||Sign_{j1}(Lic)|f}\}.

6) $PK_i$ and $SK_i$: Public and private keys of entity $i$.


8) $E_{PK_i}$: Asymmetric encryption using $PK_i$ of entity $i$.

9) $Sign_i$ : An ambiguous signature created by entity $i$.

10) $H()$: A digital signature created by entity $i$.

11) $H()$: Cryptographic hash function.

C. Assumptions

The RDS protocol has used the following assumptions.

1) Similar to the current DRM systems, LI is assumed to be a trustworthy entity which is responsible for issuing a license and facilitating a license reselling process.

2) Alice has got Lic with $RP_{Lic}$ and $ks$ from LI. We assume that $RP_{Lic}$ and $ks$ for a given license are embedded in the original one. Thus, when Alice bought Lic from LI, she should have been issued with $ks$ and $RP_{Lic}$ that contains $f$. This means that there is no need for LI to perform any additional operation with Alice before she starts the execution of the RDS protocol.

3) Keystone, $ks$, can only be used once.

4) Hash functions are collision-free.

5) Alice and Bob have agreed on RD's terms and conditions.

6) Each entity, $E_i$, where $(i \in A,B,LI)$, has a public/private key pair, $PK_i/SK_i$. $PK_i$ has been certified by a certification authority and $SK_i$ is kept secret by their respective holders.

7) Communication channels among entities are resilient.

8) Communication channels are authenticated, confidential and integrity-protected. These channels can be established using the Secure Socket Layer (SSL) protocol.

III. CONCURRENT SIGNATURE

The concurrent signature (CS) is a digital signature scheme which consists of four algorithms: SETUP, ASIGN, AVERIFY, and VERIFY. These algorithms are briefly described below (more detail can be found in [18]).

SETUP: It is a probabilistic algorithm that sets up all parameters including keys. It sets up two large prime numbers, $p$ and $q$, such that $q|(p−1)$, a generator, $g$, of a multiplicative subgroup of order $q$ in $Z_p^*$ and two cryptographic hash functions, $H_1$ and $H_2 : \{0,1\}^* → Z_q$. $H_1$ is only used in creating $f$, i.e. the hash value of $ks$, while $H_2$ is used in computing other hash values. It also generates secret keys, $SK_i$ and $SK_j$ for involved entities. The corresponding $PK_i$ and $PK_j$ are computed respectively as follows: $PK_i=g^{KS_i}(mod\ p)$ and $PK_j=g^{KS_j}(mod\ p)$. $PK_i$ and $PK_j$ and descriptions of message space $M$, signature space $S$, keystone space $K$, and keystone fix space $F$ set as public parameters. These spaces are defined as: $S=\{F\}Z_q$ and $M=\{0,1\}^*$.

ASIGN: It is a probabilistic one that takes $(PK_i,PK_j,SK_i,f,m)$ as its input, where $PK_i$ and $PK_j$ are two public keys $PK_i≠PK_j$, $SK_i$ is the private key corresponding to $PK_i$, $f=H_1(ks)$, $ks$ is a keystone, and $RD∈M$ is the message to be signed. It then outputs an ambiguous signature $ASign_i=(s_i,h_i,f)$ on RD, where $s_i∈S$, and $h_i,f∈F$.

AVERIFY: An ambiguous signature is verified via the use of AVERIFY which is a deterministic algorithm that takes the tuple, $(ASign_i,PK_i,PK_j,RD_i)$ as its inputs, where $ASign_i=(s_i,h_i,f)$ and outputs an accept, or a reject, result.

VERIFY: It is used to determine the originator of $ASign_i$ and the recipient of a given ambiguous signature. It takes two inputs: $ks$ and $S_i$, where $ks$ is the keystone from which $f$ is computed, $S_i=\langle ASign_i,PK_i,PK_j,RD_i\rangle$, and $ASign_i=(s_i,h_i,f)$ an ambiguous signature on RD. Given the inputs $(ks,S_i)$, the VERIFY algorithm first checks whether $H_1(ks)=f$. If not, the VERIFY outputs a reject result. Otherwise, AVERIFY ($S_i$) is executed (in which case, the output of VERIFY is just the same as that of AVERIFY). As a result, the pair $(ASign_i,ks)$ is called concurrent signatures and $ASign_i$ become binding to their respective signers.

IV. RDS PROTOCOL OVERVIEW

As shown in Figure 1, the protocol consists of three messages: $Msg1$, $Msg2$, and $Msg3$. In $Msg1$ and $Msg2$, Alice and Bob exchange their respective ambiguous signatures on a given RD. Alice first creates $ASign_A$, where $ASign_A=$
ASign_A(RD), and sends it along with RP_Lic to Bob. Upon receiving and verifying both RP_Lic and ASign_A (i.e. B_V1 and B_V2), Bob creates ASign_B, where ASign_B = ASign_B(RD)|ASign_A). He then sends it to Alice. After receiving ASign_B, Alice verifies its correctness using (A_V1 and A_V2) and, then, in Msg3, sends ks to Bob. Upon releasing ks, Alice obtains Bob’s binding signature, i.e. (ks, ASign_B) and Bob gets Alice’s binding signature, i.e. (ks, ASign_A), thus achieving fair RD signing process.

V. RDS PROTOCOL DESCRIPTION

The main cryptographic primitive of designing the RDS protocol is the CS scheme outlined in Section III. So, prior to executing the protocol, Alice and Bob first run the SETUP algorithm to agree on parameter values to be used in the signing process. They then engage in executing the RDS protocol, depicted in Figure 1. As Alice holds f and ks (assumption 2), she will initiate the protocol execution.

Step 1: Alice ambiguously signs RD and sends Msg1, to Bob. Signing RD requires the following operations.

1) Choosing a random number, r ∈ Z_q ; and
2) Creating ASign_A. To do this, she runs the ASIGN algorithm, described in Section III, with the following inputs: PK_A(SK_A, PK_B, RD) and f given in RP_Lic (assumption 2). This is formally described as follows:

ASign_A = ASIGN(PK_A, PK_B, SK_A, f, RD)

where:

a) f = H_1(ks)

b) h_A = h - f(mod q); where: h = H_2(g^r PK_B^f mod p||RD) ;

c) s_A = r - h SK_A(mod q).

Step 2: Once Bob receives Msg1, he performs verification B_V1. If B_V1 is negative, Bob terminates the protocol run. If it is positive, Bob performs B_V2 i.e. checking whether equation (1) holds.

\[ h_A + f = H_2(g^h PK_A^{h_B} PK_B^f mod p||RD) mod q \]  (1)

If equation (1) does not hold, Bob aborts the protocol run. If it holds, Bob creates ASign_B on (RD)|ASign_A = RD|ASign_A. The purpose for Bob to sign RD|ASign_A, instead of signing on the RD directly, is to prevent Alice from abusing ASign_B once she receives it in Msg2 (i.e. achieving abuse-freeness). This is because if Bob signs on RD, Alice can use ks to make ASign_B binding to Bob and then show it to a third party. In this case, Alice can gain some advantage over Bob.

To create ASign_B on RD|ASign_A, Bob runs the ASIGN algorithm with the following inputs: PK_B/SK_B, PK_A, f (the same f used by Alice) and RD|ASign_A. This is formally described as follows:

ASign_B = ASIGN(PK_B, PK_A, SK_B, f, RD|ASign_A), where:

1) f = H_1(ks)
2) h_B = h - f(mod q); and h = H_2(g^r PK_B^f mod p||RD|ASign_A), and r’ is a random number chosen from Z_q.
3) s_B = r’ - h_B SK_B mod q.

Upon generating ASign_B, Bob sends Msg2 to Alice, i.e. Bob → Alice: Msg2 = {RD|ASign_A}. Step 3: On the receipt of Msg2, Alice performs A_V1. If A_V1 is negative, Alice aborts the protocol run. Otherwise, she performs A_V2, i.e. confirms that equation (2) holds.

\[ h_B + f = H_2(g^h PK_A^{h_B} PK_B^f mod p||RD|ASign_A) mod q \]  (2)

If A_V2 is negative, Alice aborts the protocol. Otherwise, if it is correct, Alice uses ks to convert ASign_A and ASign_B to signatures which are binding to their respective signers. As a result, Alice gets a signed RD which is of the form = [RD, ASign_A, ASign_B, ks]. In order for Bob to also obtain a signed RD, Alice encrypts ks using PK_B, and then sends it in Msg3 to Bob. The keystone fix, f, is included in Msg3 to enable Bob to identify that ks is the one corresponding to f used in Msg1 and Msg2.

Alice → Bob: Msg3 = \{f||E_{PK_B}(ks)\}

Once Bob receives Msg3, he obtains ks from it and performs B_V3 to confirm that the hash value of this ks is equal to f received in RP_Lic. Bob then uses this ks to convert ASign_A and ASign_B to signatures which are binding to their respective signers. As a result, Bob will obtain a signed RD which is of the form = [RD, ASign_A, ASign_B, ks]. At this stage, both Alice and Bob have obtained the same signed RD, achieving fair contract (RD) signing.

VI. PROTOCOL ANALYSIS

In this section, the RDS protocol is analysed against the requirements set in Section II-A. In a license reselling process, after RD has been signed by a reseller and a buyer, LI has to be invoked by the buyer to activate this signed RD. This activation process can deter both the reseller and the buyer to misbehave during the RD sign process, as indicated below, it
is hard for any entity to benefit from misbehaving. During the activation process, LI will perform the following verifications and will only accept a given RD if they are all positive.

(LI\textsubscript{V1}): Confirming that the payment, $P$, equal to the amount stated in RD, has been made by Bob.

(LI\textsubscript{V2}): Checking that Lic is legitimate to resell. This check consists of two further checks: LI\textsubscript{V2,1} and LI\textsubscript{V2,2}. In LI\textsubscript{V2,1}, LI verifies whether Lic has a valid RP (i.e., verifying $\text{Sign}_{LI}(\text{Lic}||f)$). If this signature is invalid, either Lic or $f$ is incorrect. Hence, Lic is deemed as non-resalable and LI will reject RD. If the signature is valid, Lic is considered as resalable. LI then proceeds to perform LI\textsubscript{V2,2} in which LI ascertains whether $ks$, corresponding to $f$, has already been released. If $ks$ has already been publicised, it means that Lic has already been resold. So, LI will reject RD. Otherwise, LI proceeds to perform LI\textsubscript{V3}.

(LI\textsubscript{V3}): Checking whether the license identity, Lic, specified in $RP_{Lic}$, is identical to the license identity, Lic, specified in RD. This check prevents Bob to replace $RP_{Lic}$ with another less valuable $RP_{Lic'}$. LI\textsubscript{V4}: Verifying $\text{ASign}_B$ to confirm Bob has signed RD. LI\textsubscript{V5}: Verifying $\text{ASign}_A$ to ensure Alice has signed RD.

A. Fairness Analysis

This section analyses the fairness property of the RDS protocol. As shown in Figure 2, we will examine every possible scenario Alice or Bob could follow when she or he, respectively, creates and/or receives a protocol message. In this analysis, it is assumed that any payment, $P$, made to LI is equal to the amount stated in the submitted RD. Otherwise, this RD will be rejected.

In our analysis, the following three scenarios will be considered. These scenarios are: (a) Alice and Bob behave properly; (b) Alice behaves improperly; (c) Bob behaves improperly.

Alice and Bob behave properly: As depicted in Figure 2, there are two scenarios for Alice and Bob to achieve fairness. These two scenarios are depicted as dashed lines in the figure. In the first scenario, Alice and Bob honestly follow the protocol described in Section V.

Secondly, like in the first scenario, Alice sends $Msg1$ to Bob who may be very happy with the RD to be signed. Thus, instead of sending $Msg2$ to Alice, Bob constructs an RD activation request and sends it to LI. This request is of the form $\{Msg1||Msg2||P\}$ or $\{Msg2||RP_{Lic}||P\}$. Although the signatures in $Msg1$ and $Msg2$ are still ambiguous, LI can use $ks$ to convert them into binding signatures. LI can then activate the RD. In this activation, LI sends Bob the activated license Lic, makes $ks$ public and sends Alice the payment, $P$. In this case, Alice will not be affected as her license has been resolved (this is the main aim behind the RD signing protocol). She can also get the signed RD with the payment. Thus, in the second scenario, both Bob and Alice can get a signed RD (i.e., achieving fairness).

Alice behaves improperly: As shown in Figure 2, Alice may misbehave in three occasions. In the first occasion, when Alice creates $Msg1$, she may take two malicious actions with LI: A2 and A3. (Malicious action here means to cause denial of service attack on LI by sending an invalid RD activation request). For both of these actions, upon performing LI\textsubscript{V1}, LI will reject the RD request as there is no payment provided in A2 and A3. Thus, Alice can neither gain anything useful nor cause any harm to Bob by taking A2 and A3.

In the second occasion, upon the receipt of $Msg2$, Alice may take one or more of five malicious actions (i.e., A7, A8, A9, A10, and A11). In all these actions, as shown in Figure 2, Alice may send an incomplete RD activation request to LI. However, as Alice did not provide a payment in these actions, she will gain nothing by doing so as upon performing LI\textsubscript{V1}, LI will reject these RD submissions. Also, since Bob has not yet made any payment for these requests, he will not experience any loss.

In the third occasion, after creating $Msg3$, Alice may perform two misbehaving actions with Bob: i.e. A5 and A6. In A6, Alice may abort the protocol run. However, Bob will not be affected as he can still be able to activate the agreed RD with LI. He can send LI an RD activation request containing $\{Msg1||Msg2||P\}$, in which case, he can get $ks$ along with the activated license from LI. In A5, Alice, in $Msg3$, may send Bob $ks_1 \neq ks$. If this happens, Bob will not be affected as he can detect this misbehaviour when he performs $B_V$. Bob then has three choices: (1) ask Alice to resend the correct $ks$; (2) terminate the protocol run as Bob may no longer be interested in completing the signing process of this RD with Alice (for example, he may have got another deal which is better than that offered by Alice; or (3) send an RD activation request to LI to activate the license and get the $ks$ with the activated license. In any of the above cases, Bob will not experience any loss as he can still able to activate RD.

Bob behaves improperly: As shown in Figure 2, Bob may misbehave in two occasions. In the first occasion, once Bob receives $Msg1$, he may take one of two malicious actions: (a) either submit B5 or B6 to LI as RD activation requests, and (b) misbehave with Alice (i.e. send Alice B10). If Bob
submits B5 or B6 to LI, LI will reject his submission as $LI_{V1}$ and $LI_{V4}$ will be failed respectively. Therefore, if Bob does any of B5 or B6, he can neither bring any harm to Alice nor gain anything useful himself. Bob can do B10 as it is his right to abort the protocol run if he does not wish to go for the deal. However, if this happens, similar to the case of B5 and B6, Alice will not be affected as, at this stage, Bob has only got $ASign_A$ which is not yet binding to Alice.

In the second occasion, upon the creation of $Msg2$, Bob may take one of four malicious actions with LI, (i.e. submit B4, B7, B8, and B9 to LI to activate the RD). In B4, although Bob has submitted an RD activation request containing all the required elements, based on the verification of $LI_{V3}$, LI will reject this request. In B7 and B8, LI will also reject such requests as $LI_{V1}$ will fail. In B9, though the payment is included, after performing $LI_{V2}$, LI will also reject Bob’s submission as $RP_{Li}$ is not provided. Thus, by doing B4, B7, B8, or B9, Bob will not gain any benefit. In addition, since Bob has got an ambiguous signature, $ASign_A$, from Alice, he can not bring any harm to Alice using this signature.

From the above discussion, three remarks can be drawn. Firstly, Alice could not bring any harm to Bob if she refused to send $ks$ to Bob in $Msg3$. This is because once Bob has received $Msg1$ and created $Msg2$, he can activate RD and this activation can only be successful if Bob sends LI $\{Msg1||Msg2||P\}$. Secondly, Bob can not gain any advantage by receiving $ASign_A$ first as $ASign_A$ is not binding to Alice before Alice releases $ks$. Thirdly, all Alice’s submissions to LI will be rejected unless she provides the payment. Of course, it does not make any sense that Alice, as the reseller, would make any payment to resell her license.

B. Non-repudiation Analysis

This section analyses the RDS protocol against the non-repudiation requirements: the non-repudiation of origin (NOO) and non-repudiation of receipt (NOR) of signatures being exchanged.

In our problem context, NOO means that upon the successful completion of a protocol execution, both Alice and Bob can not falsely deny having created their respective signatures: (ks, $ASign_A$) and (ks, $ASign_B$). When ks is released in $Msg3$, both $ASign_A$ and $ASign_B$, exchanged in $Msg1$ and $Msg2$, will become binding to their respective signers. Hence, neither entity can falsely deny having created their signatures. If ks is not released or what is released is not equal to the key used in signing process (i.e. $ks1 \neq ks$), only Alice will have (ks, $ASign_B$) and Bob will not have (ks, $ASign_A$).

In this case, Alice could deny having signed RD. However, if Bob submits $\{Msg1||Msg2||P\}$ to LI for activating RD, he will receive ks with the activated RD from LI. Bob then uses this ks to obtain (ks, $ASign_A$). Thus, once ks is publicised, neither Alice nor Bob could falsely deny having created their respective signatures: (ks, $ASign_A$) and (ks, $ASign_B$). Consequently, our protocol meets the NOO requirement.

Regarding NOR of the signatures, upon the successful completion of a protocol run, both Alice and Bob will have a proof that each one of them has received the other’s signature. For Bob, once he receives ks in $Msg3$, this ks serves as a proof that Alice has indeed received $ASign_B$ sent in $Msg2$. This is because Alice only releases ks once she receives $ASign_B$. If Bob does not receive ks, he will not be able to tell that Alice has indeed received $ASign_B$. However, if Bob sends LI an RD activation request of the form $\{Msg1||Msg2||P\}$, Alice will receive $ASign_B$ from LI during the RD activation process. In other words, a successful RD activation process ensures Bob that Alice has indeed received $ASign_B$ when LI sends Alice both the payment and the signed RD.

For Alice, once she receives $ASign_B$ in $Msg2$, she can use this $ASign_B$ as a proof that Bob has certainly received $ASign_A$ in $Msg1$. This results from the idea that $ASign_A$ is a part of $ASign_B$. If Alice does not receive $ASign_B$, she will not be able to make sure that Bob has indeed received $ASign_A$. However, when Bob sends LI an activation request of the form $\{Msg1||Msg2||P\}$, Alice will receive $ASign_B$ from LI during the RD activation process. So, this request to LI serves as proof that Bob has indeed received $ASign_A$.

From the above discussion, it can be concluded that upon a successful execution of the protocol or upon a successful RD activation, both Alice and Bob will have received the each other’s signatures. This implies that the our protocol meets NOR requirement.

C. Abuse-freeness Analysis

The section shows that the RDS protocol is abuse-free. We discuss whether Alice or Bob could gain any benefit by showing any intermediate result of the RDS protocol run to a third party. In fact, the RDS protocol produces two intermediate results: (1) RD ambiguously signed by Alice, i.e. $RDAASign_A$, and (2) RD ambiguously signed by both Alice and Bob, i.e. $[RDAASign_A||ASign_B]$. $RDAASign_A$ is received by Bob in $Msg1$. $[RDAASign_A||ASign_B]$ is received by Alice in $Msg2$.

For $RDAASign_A$, Bob can not abuse it to gain an advantage over Alice. Of course, Bob can show this result to another reseller, Eve. However, Bob can not prove to Eve that $RDAASign_A$ is indeed signed by Alice as $ASign_A$ is an ambiguous signature and it could have been produced by Bob. Thus, Bob can not abuse $RDAASign_A$.

With respect to $[RDAASign_A||ASign_B]$, although it is an ambiguous signature, Alice is able to make it binding signature to Alice and then show it to a third party. As Alice holds the ks, she can use it to prove to Carol that $[RDAASign_A||ASign_B]$ is indeed signed by Bob. However, Carol can also see that this RD has also been signed by Alice as $ASign_A$ is a part of $ASign_B$ (see Section V). Hence, it would be difficult for Alice to gain anything useful by doing so as it does not make sense for Alice to show Carol RD which is binding to both of Alice and Bob. So, we say that Alice can not abuse $[RDAASign_A||ASign_B]$.

From this analysis, we can conclude that neither Alice nor Bob can abuse each others’ signatures if the protocol is not successfully completed. Thus, the RDS protocol satisfies the abuse-freeness property.
D. Security Analysis

As it can be seen from Section V, the CS scheme is the main cryptographic primitive of the RDS protocol. So, the security of the RDS protocol lies in the security of the CS scheme [18]. In other words, the security level of the RDS protocol is the same as the security of the CS scheme. In addition, since communications between entities A and B is carried out through a confidential channel (owing to assumptions 8), the exchanged data is not exposed to any outsiders. This prevents any intruder from gaining anything useful from the messages while they pass through the network. SSL protocol also protects against any replay attack [19].

Also, it is crucial for the security of the RDS protocol to ensure that the key $k_s$ is not reused. If $k_s$ is reused, Alice would be able to resell her license, Lic, multiple times. Consequently, the security of the DRM system (i.e. the content owner’s rights) will be compromised.

The security of the RD contract is protected by Alice’s and Bob’s signatures. As Alice signs RD first, if she has modified the negotiated RD before signing and sending it to Bob, Bob can detect such modification and then he can refuse to sign this RD. Also, as Bob is responsible for sending an RD activation request to LI, Bob would be able to modify RD before he sends the request to LI. However, as Bob have to jointly sign RD, which has been signed by Alice, Bob can not modify the RD unless he can create Alice’s signature on this modified RD. In other words, in order for a given RD to be activated, the owner’s rights will be violated.

Consequently, the security of the DRM system (i.e. the content owner’s rights) will be compromised.

The security (integrity) of $RP_{Lic}$ is protected by LI’s signature. If Bob attempts to modify it, during performing $LI_{V2}$, LI can detect such modification. LI can then reject Bob’s RD activation request.

With our protocol, Bob could replace $RP_{Lic}$ with a different one, e.g. $RP_{Lic2}$. This $RP_{Lic2}$ may be with less value than $RP_{Lic}$. If this happens, with $LI_{V3}$, LI can detect that the license identity, Lic, stated in the signed RD, is different from what is provided in the $RP_{Lic}$. LI can then reject the RD activation process. Thus, Bob can gain nothing by doing such misbehaviour.

VII. Conclusion

This paper has presented a fair and abuse-free contract signing protocol which does not make use of a dedicated TTP to help with achieving fairness. The protocol is designed to allow a reseller and a buyer to fairly sign a contract (reselling deal) to support fair license reselling. The protocol design has made full use of the special features provided by two building blocks: (1) the CS scheme that provides both weak fairness and ambiguity, and (2) the existing license distribution infrastructure (i.e. LI). By combining the weak fairness property of the CS scheme and the existing LI, our protocol achieves strong fairness. Also, by utilising the ambiguity of the CS scheme and the joint signature of both Alice and Bob on the contract, the protocol provides abuse-freeness. In addition, it has been shown that our protocol has a range of potential applications, including supporting the reselling of both digital licenses and access permissions. Furthermore, the protocol analysis has shown that our protocol satisfies the design requirements. Our future work includes the formal verification of the protocol.

ACKNOWLEDGEMENT

This work is financially sponsored by the faculty of Computers and Informatics, Suez Canal University, Egypt.

REFERENCES