Nuovo DRM Paradiso: Towards a Verified Fair DRM Scheme

M. Torabi Dashti[1], S. Krishnan Nair[2], and H. L. Jonker[3]

[1] CWI Amsterdam
[2] Vrije Universiteit Amsterdam
[3] Technische Universiteit Eindhoven

Abstract. We formally specify the recent DRM scheme of Nair et al. in the μCRL process algebraic language. The security requirements of the scheme are formalized and using them as the basis, the scheme is verified. The verification shows the presence of security weaknesses in the original protocols, which are then addressed in our proposed extension to the scheme. A finite model of the extended scheme is subsequently model checked and shown to satisfy its design requirements, including secrecy, fairness and resisting content masquerading. Our analysis was distributed over a cluster of machines, allowing us to check the whole extended scheme despite its complexity and high non-determinacy.

1 Introduction

Recent years have seen a rapid increase in the popularity of personal devices capable of rendering digital contents. Large content providers as well as independent artists are looking forward to these new opportunities for selling their copyrighted materials, necessitating the development of systems to protect digital contents from illegal access and unauthorized distribution. Technologies used to enforce policies controlling usage of digital contents are referred to as Digital Rights Management (DRM) techniques. A major challenge in DRM is enforcing the policies after contents have been distributed to consumers. This problem is currently addressed by limiting the distribution of protected contents only to the so-called *compliant* devices (e.g. iPods), that by construction are guaranteed to always enforce the DRM policies associated with the contents they render.

A unique concept of DRM-preserving content *redistribution* was proposed in [30], hereafter called NPGCT scheme, where users act also as content redistributors. This potentially allows consumers to not only buy the rights to use a content, but also to redistribute the content in a controlled manner. From a security point of view, this is technically challenging, since the resulting system forms a peer-to-peer network of independent devices, each of them a consumer, an authorized distributor, and also a potential attacker. Recent sobering experience [22] has shown that DRM techniques are inherently complicated and if carelessly enforced can infringe on customers', as well as vendors', rights. These serve as motivation for using formal methods to verify the NPGCT scheme to provide both content vendors and customers a certain degree of confidence in the security and fairness of the system.

Contributions Our contribution in this paper is twofold. First, on the security side, we formally specify the NPGCT protocols and analyze them. Our analysis reveals two security flaws in the scheme. We then propose an extended scheme, dubbed *Nuovo DRM*, to address these issues. A formal specification and verification of Nuovo DRM is subsequently presented and (a finite model of) the scheme is shown to indeed achieve its design goals.

Second, we use state-of-the-art formal tools and techniques to handle the verification problem of DRM schemes. We use the μCRL process algebraic language [20] and toolset [9] to specify the protocol participants and the intruder model. Due to the complexity and sheer size of the schemes, we resorted to a distributed instantiation of the toolset [8] to generate and minimize the corresponding state spaces. In particular, since the Nuovo DRM scheme is highly non-deterministic due to the presence of several fall-back scenarios, with the inclusion of an intruder model to the system, it easily runs into the limits of single-machine state space generation. To the best of our knowledge, we are the first to formally verify a whole DRM scheme. Moreover, we adapt the standard formal model of intruder, namely the Dolev-Yao model [14], to reflect the restricted behavior of compliant devices in DRM systems.

Related work Nuovo DRM contains an optimistic fair exchange protocol. Optimistic fair exchange protocols have been introduced in [4] and since then have attracted much attention. The closest fair exchange protocol to our scheme is perhaps the probabilistic synchronous protocol of [5], as it relies on trusted computing devices in exchange. In contrast to [5], we present a deterministic asynchronous protocol that achieves strong (as opposed to probabilistic) fairness, but, as a drawback, relies on impartial agents to secure unsupervised exchanges.

In this paper we do not address modeling semantics and derivations of rights associated with DRM-protected contents, which constitutes a whole separate body of research, e.g. see [32]. We focus on formal analysis of transactional properties of DRM schemes. Related to this, there are several papers on model checking (usually small instances of) optimistic fair exchange protocols, e.g. [21, 27, 33]. What makes our study unique is the size of the system that is automatically analyzed here, as well as, capturing some DRM-specific features of the system, e.g. compliant devices, in the model. Constraint solving for checking fair exchange protocols has been proposed in [25]. This can detect type-flaw attacks, but is restricted to checking safety properties. Theorem-proving approaches to checking fairness of protocols [2, 7, 15] can provide a complete security proof at the cost of heavy human intervention, and thus cannot be easily integrated in the protocol design phase.

Structure of the paper We start by explaining the notations and (cryptographic) assumptions used in the paper, in Section 2. Section 3 summarizes the NPGCT scheme, which provides the basis for our refined scheme. Section 4 presents the Nuovo DRM scheme, its assumptions and its goals. Some details of Nuovo DRM's payment mechanism, that fall outside the scope of our formalization, are described in Appendix A. Nuovo DRM is then formally analyzed in

Section 5 and shown to achieve its goals. Finally, Section 6 concludes the paper with some possible future research directions.

2 Notations and assumptions

Trusted devices assumptions Compliant devices are tamper-proof hardware, possibly operated by malicious owners, that follow only their certified software. We assume that compliant devices are able to locally perform *atomic* actions: multiple actions can be logically linked in these devices, such that either all or none of them are executed. They also contain a limited amount of secure scratch memory and non-volatile storage. These requirements are typically met by current technologies (e.g. iPods). A legitimate content provider, (abusively) referred to as trusted third party (TTP), is assumed impartial in its behavior and eventually available to respond to requests from compliant devices.

Cryptographic assumptions In our analysis the cryptographic operations are assumed to be ideal à la Dolev-Yao [14]: we assume access to a secure one-way collision-resistant hash function h; therefore $h(x)$ uniquely describes x. A message m encrypted with symmetric key K is denoted $\{m\}_K$, from which m can only be extracted using K. Notations $pk(X)$ and $sk(X)$ denote the public and private keys of entity X, respectively. In asymmetric encryption we have $\{\{m\}_{sk(X)}\}_{pk(X)} = \{\{m\}_{pk(X)}\}_{sk(X)} = m$. Encrypting with $sk(X)$ denotes signing and, for convenience, we let m be retrievable from $\{m\}_{sk(X)}$.

Notations C and D denote compliant customer devices, respectively owned by $owner(C)$ and $owner(D)$. P denotes a trusted legitimate content provider. A DRM-protected content is denoted by M. The finite set of all protected contents is denoted *Cont*. It is assumed that unique descriptors (e.g. hash values) of all $M \in Cont$ are publicly known. The (finite) set of all possible rights in the protocols is denoted *Rgts*. The term $R_X(M)$ represents the rights of device X for content M.

3 The NPGCT DRM scheme

The NPGCT scheme was proposed as a DRM-preserving digital content redistribution system where a consumer doubles up as a content reseller. In this section we briefly describe the NPGCT scheme and then present the results of its formal analysis. For a detailed specification of NPGCT see [30].

3.1 NPGCT protocols

The scheme consists of two main protocols: the first distributes contents from provider P to client C, the second allows C to resell contents to another client D.

Provider-customer protocol (P2C) The protocol is initiated by the owner of C who wants to buy item M with rights R from provider P. From [30]:

1. $C \rightarrow P$: Request content
2. $C \leftrightarrow P$: Mutual authentication, [payment]
3. $P \rightarrow C$: $\{M\}_K, \{K\}_{pk(C)}, R, \sigma, \Lambda$
   σ=meta-data of M, Λ=$\{h(P, C, M, \sigma, R)\}_{sk(P)}$

Here Λ acts as a certification that C has been granted rights R and helps in proving C's right to redistribute M to other clients. It also binds the meta-data σ to the content, which prevents masquerading attacks on M.

Customer-customer protocol (C2C) This part of the protocol is initiated by the owner of D who wants to buy M with rights R' from C. From [30]:

1. $D \rightarrow C$: Request content
2. $C \leftrightarrow D$: Mutual authentication
3. $C \rightarrow D$: $\{M\}_{K'}, \{K'\}_{pk(D)}, R_C(M), R', \sigma, \Lambda, \Lambda'$
   $\Lambda' = \{h(C, D, M, \sigma, R')\}_{sk(C)}$
4. D : Verifies σ, Λ' and $R_C(M)$ using Λ
5. $D \rightarrow C$: $\psi, [payment]$
   $\psi = \{h(C, P, \{M\}_{K'}, \sigma, R')\}_{sk(D)}$

By ψ, D acknowledges that it has received M with rights R', while Λ and Λ' form a chain that helps to prove that D has been granted rights R'.

3.2 Formal analysis of NPGCT

We have formally specified and model checked the NPGCT scheme. In this section, due to space constrains, we only present the results of this analysis. The assumptions and security goals of the scheme, their formalization, the protocol specification toolset and the model checking technology used here are similar to those used for Nuovo DRM, which are discussed in the following sections. Details of this analysis along with found attack traces are available online [1].

Two security flaws in the NPGCT scheme were revealed in our analysis. First, in the P2C (and similarly the C2C) protocol, a malicious customer could feed rights from a previous session to the trusted device, because the authentication phase is not extended to guarantee freshness of the content-right bundle that is subsequently delivered. This flaw allows C to accumulate rights without paying P for it. As a remedy, fresh nonces from the authentication phase can be used in Λ to ensure the freshness of the whole exchange, c.f. Section 4.

Second, in the C2C protocol, payment is not bound to the request/receive messages exchanged between two customers. Thus, once D receives M in step 3, the owner of D can avoid paying C by aborting the protocol. Since this exchange is unsupervised, the owners of compliant devices are forced to trust each other to complete transactions. While it is reasonable to extend such trust to a legitimate content provider, it should not be assumed for device owners in C2C exchanges.

4 The Nuovo DRM scheme

This section describes an extension to the NPGCT, dubbed Nuovo DRM, which in particular addresses the security concerns identified in Section 3.2. Here we confine to informal descriptions; a formal specification is discussed in Section 5.

4.1 Nuovo DRM's goals

We require the Nuovo DRM scheme to achieve the following goals (the same goals as those used to analyze the NPGCT scheme in Section 3.2):
G1. Effectiveness A protocol achieves effectiveness iff when honest participants run the protocol, it terminates successfully, i.e. a desired content-right bundle is exchanged for the corresponding payment order. Effectiveness is a sanity check for the functionality of the protocol and is therefore checked in a reliable communication system with no attacker.
G2. Secrecy Secrecy states that no outsider may learn "secret" items, which are usually encrypted for intended receivers. Nuovo DRM (similar to NPGCT) limits the distribution of protected contents by encrypting them for intended compliant devices. This scheme must thus guarantee that a DRM-protected content never appears in plain to any non-compliant device.
G3. Resisting content masquerading Content masquerading occurs when content M is passed off as content M', for $M \neq M'$. Preventing this attack ensures that an intruder cannot feed M' to a device that has requested M.
G4. Strong fairness Assume Alice owns m_A and Bob owns m_B. Informally, strong fairness states that if Alice and Bob run a protocol to exchange their items, finally either both or neither of them receive the other party's item [31]. Strong fairness usually requires the contents exchanged in the system to be *strongly generatable*: in Nuovo, a content provider can provide the exact missing content if the exchange goes amiss. Strong fairness also guarantees *timeliness*, which informally states that, in a finite amount of time, honest protocol participants can safely terminate their role in the protocol with no help from malicious parties. As this is a liveness property[1], resilient communication channels (assumption A2 below) are necessary for fairness to hold [4]. For an in-depth discussion of fairness in exchange we refer the interested reader to [4].

4.2 Nuovo DRM's assumptions

Nuovo DRM is based on the following assumptions:
A1. Consumer compliant devices are assumed tamper-proof. Owners of compliant devices are however untrusted. They may collude to subvert the protocol. They can, in particular, arbitrarily switch off their own devices ("crash failure model" in distributed computing terminology).

[1] Properties of systems can be divided into two classes: *safety* properties, stating unwanted situations do not happen, and *liveness* properties, stipulating desired events eventually happen. For a formal definition of these property classes see [3].

A2. We assume an asynchronous resilient communication model with no global clock, i.e. the communication media deliver each transmitted message intact in a finite but unknown amount of time. Resilience is necessary when aiming for fairness [18], and is realizable under certain reasonable assumptions [6].
A3. There exists a hierarchy of public keys, with the public key of the root embedded in each compliant device and available to content providers. Using such an infrastructure, a device can prove its identity or verify other devices' identities without having to contact the root. Participant identities (C, D and P) implicitly refer to these authentication certificates issued by the root.
A4. Protocol participants negotiate the price of content in advance. In Nuovo DRM, the price of the content being traded is bundled with the requested rights.

4.3 Nuovo DRM protocols

As in NPGCT, our scheme consists of two main protocols: the first distributes content from provider P to client C, the second allows C to resell content to another client D.
Provider-customer protocol (P2C) The owner of C wants to buy item M with rights R from content provider P. Here C and P, but not $owner(C)$, are assumed trusted.

1. $owner(C) \rightarrow C : P,\ h(M),\ R$
2. $C \rightarrow P : C,\ n_C$
3. $P \rightarrow C : \{n_P,\ n_C,\ C\}_{sk(P)}$
4. $C \rightarrow P : \{n_C,\ n_P,\ h(M),\ R,\ P\}_{sk(C)}$
5. $P \rightarrow C : \{M\}_K,\ \{K\}_{pk(C)},\ \{R,\ n_C\}_{sk(P)}$

In the first step, the hash of the desired content, retrieved from a trusted public directory, with a right and the identity of a legitimate provider are fed to the compliant device C. Following assumption A4, $owner(C)$ and P have already reached an agreement on the price. Whether P is a legitimate provider can be checked by C and vice versa (see assumption A3). In step 2, C generates a fresh nonce n_C and sends it to P, which will continue the protocol only if C is a compliant device. Message 4 completes the mutual authentication between C and P. This also constitutes a *payment order* from C to P. After receiving this message, P checks if R is the same as previously agreed upon (assumption A4) and only if so, stores the payment order (for future/immediate encashing) and performs step 5 after generating a random fresh key K. When C receives message 5, it decrypts $\{K\}_{pk(C)}$, extracts M and checks if it matches $h(M)$ in message 1, and n_C is the same as the nonce in message 2. If these tests pass, C updates $R_C(M)$ with R, e.g. R is added to $R_C(M)$. Note that $R_C(M)$ is not necessarily R: C could already have some rights associated with M, for instance, acquired from an earlier purchase. Since we abstract away from rights semantics (see our related work), the update phase is left unspecified here.

We now define a set of *abstract* actions to highlight important steps of the protocol. These are used in the formalization process to define desired behaviors of the protocol. For the P2C protocol, C performs the abstract action

request$(C, h(M), R, P)$ at step 4, indicating the start of the exchange from C's point of view. At step 5, P performs *issue*$(P, h(M), R, C)$, denoting the receipt of the payment order and sending the content to C. Finally C performs *update*$(C, h(M), R, P)$ upon accepting message 5, denoting the successful termination of the exchange from C's point of view. These abstract actions are further discussed in Section 5.

Customer-customer protocol (C2C) The owner of D wants to buy item M with rights R' from another compliant device C. This protocol can be seen as a fair exchange protocol where C and D want to exchange a content-right bundle for its associated payment so that either both or none of them receive their desired items. In deterministic protocols, however, achieving fairness is proved to be impossible without a TTP [16]. Assuming that most participants are honest and protocols go wrong infrequently, it is reasonable to use protocols which require TTP's intervention only when a conflict has to be resolved. These are usually called *optimistic* fair exchange protocols [4] and contain two sub-protocols: an optimistic sub-protocol is executed between untrusted devices, and if a participant cannot finish this protocol run, it will initiate a recovery sub-protocol with a designated TTP.[2] Our C2C protocol is an optimistic fair exchange protocol which uses the content provider P as the TTP. The optimistic exchange sub-protocol is as follows:

1. $owner(D) \rightarrow D : C,\ h(M),\ R'$
2. $D \rightarrow C : D,\ n_D$
3. $C \rightarrow D : \{n'_C,\ n_D,\ D\}_{sk(C)}$
4. $D \rightarrow C : \{n_D,\ n'_C,\ h(M),\ R',\ C\}_{sk(D)}$
5. $C \rightarrow D : \{M\}_{K'},\ \{K'\}_{pk(D)},\ \{R',\ n_D\}_{sk(C)}$

This protocol is similar to the P2C protocol and only the abstract actions are described here: at step 4, D takes the action *request*$(D, h(M), R', C)$ when sending out the message which represents its payment. At step 5, C performs *issue*$(C, h(M), R', D)$ and in the same atomic action updates the right associated with M (reflecting that some part of $R_C(M)$ has been used for reselling M) and stores the payment order signed by D. Note that the atomicity of these actions is necessary to guarantee that C does not store the payment order without simultaneously updating the right $R_C(M)$. Upon accepting message 5, D performs *update*$(D, h(M), R', C)$.

In this protocol, a malicious *owner*(C) can abort before sending message 5 to D or this message can get lost due to a hardware failure. To prevent such unfair situations for D, we provide a recovery mechanism to obtain the lost content.

Recovery sub-protocol The goal is to bring the compliant device D back to a fair state in case of a failure in delivering message 5 in the C2C protocol.

[2] Fair exchange is attained by ensuring either successful termination (recovery) or failure (abortion) for both parties. In Nuovo DRM, if neither party terminates successfully, nothing is exchanged and failure is already attained. Hence, no particular "abort" protocol is necessary.

D can start a recovery session with the content provider P at any time after sending message 4 in the C2C protocol. If a connection with the provider is not available, D saves the current state and simply waits till it becomes available. Once the recovery protocol has been initiated, D ignores messages from the optimistic run of C2C. The purpose of the recovery is to ensure that D receives the content and rights that $owner(D)$ wanted (and ostensibly paid for).

$$
\begin{array}{ll}
5^r. & D : resolves(D) \\
6^r. & D \rightarrow P : D,\ n'_D \\
7^r. & P \rightarrow D : \{n'_P,\ n'_D,\ D\}_{sk(P)} \\
8^r. & D \rightarrow P : \{n'_D,\ n'_P,\ \langle n_D,\ n'_C,\ h(M),\ R',\ C\rangle,\ R'',\ P\}_{sk(D)} \\
9^r. & P \rightarrow D : \{M\}_{K''},\ \{K''\}_{pk(D)},\ \{R'',\ n'_D\}_{SK(P)}
\end{array}
$$

In this protocol D and P behave as if D is purchasing the M-R'' content-right bundle from P using the P2C protocol, except that, in message 8^r, D reports the failed C2C exchange it had with C. The following abstract actions are performed here: $request(D, h(M), R', P)$ is performed by D at step 8^r. At step 9^r, P performs $issue(P, h(M), R', D)$ and upon accepting message 9^r, D performs $update(D, h(M), R', P)$. The way P resolves (payments of) failed exchanges deserves detailed explanation. This however falls beyond the scope of our formal analysis; see Appendix A for a detailed discussion.

One can argue that the recovery sub-protocol may also fail due to lossy communication channels. As a way to mitigate this, persistent communication channels for content providers can be built, e.g., using an FTP server as an intermediary. The provider would upload the content, and the device would download it from the server. In order to guarantee fairness, such resilient communication channels are generally unavoidable [4] (c.f. assumption A2).

As a final note, we emphasize that only tamper-proof compliant devices are considered here (assumption A1). These protocols can be trivially attacked if the devices are tampered with (e.g. a corrupted D would be able to initiate a recovery protocol even after a successful exchange). Methods for revoking circumvented devices and resisting systematic content pirating are described in [24, 30].

5 Formal analysis

In this section we describe the steps followed to formally verify that Nuovo DRM achieves its design goals. Our approach is based on finite-state model checking [12], which (usually) requires negligible human intervention and, moreover, produces concrete counterexamples, i.e. attack traces, if the design fails to satisfy a desired property. It can therefore be effectively integrated into the design phase. However, a complete security proof of the system cannot, in general, be established by model checking. For an overview on formal methods for verifying security protocols see [29]. Our formal verification can be seen as a sequence of steps: first, we specify the protocol and the intruder model in the μCRL process algebraic language and generate the corresponding model using the μCRL toolset (version 2.17.12). Second, we state the desired properties in the regular

(alternation-free) μ-calculus, and, finally, check the protocol model with regard to the properties in the CADP toolset. Below, these steps are described in detail.

5.1 Formal specification of Nuovo DRM

The complex structure of Nuovo DRM calls for an expressive specification language. We have formalized the Nuovo DRM scheme in μCRL, a language for specifying and verifying distributed systems and protocols in an algebraic style [20]. A μCRL specification describes a labeled transition system (LTS), in which states represent process terms and edges are labeled with actions. The μCRL toolset [9, 8], together with CADP [17] which acts as its back-end, features visualization, simulation, symbolic reduction, (distributed) state space generation and reduction, model checking and theorem proving capabilities.

We model a security protocol as an asynchronous composition of a finite number of non-deterministic named processes. These processes model roles of honest participants in the protocol. Processes communicate by sending and receiving messages. A message is a pair $m = (q, c)$, where q is the identity of the intended receiver process (so that the network can route the message to its destination) and c is the content of the message. To send or receive a message m, a participant p performs the actions $\mathbf{send}(p, m)$ or $\mathbf{recv}(p, m)$, respectively. Apart from **send** and **recv**, all other actions of processes are assumed internal, i.e. not communicating with other participants. These are symbolic actions that typically denote security claims of protocol participants (e.g. *update* in Section 4.3). Here, we only present a μCRL specification of the honest customer role in the P2C protocol. For a complete specification of Nuovo DRM see [24]. We start with a very short introduction to μCRL.

The μCRL specification language In a μCRL specification, processes are represented by process terms, which describe the order in which the actions may happen in a process. A process term consists of action names and recursion variables combined by process algebraic operators. The operators '$\cdot$' and '$+$' are used for the sequential and alternative composition ("choice") of processes, respectively. The process $\sum_{d \in \Delta} P(d)$, where Δ is a (infinite) data domain, behaves as $P(d_1) + P(d_2) + \cdots$.

The customer process In μCRL spec 1 we specify the customer's compliant device role in the P2C protocol of the Nuovo DRM scheme. In this specification, *Nonce* and *Key* represent the finite set of nonces and keys available in the protocol, respectively. The set Ω is C's local collection of content-right bundles, n_C denotes the nonce that is available to C in the current protocol round, and the function $nxt : Nonce \rightarrow Nonce$, given a seed, generates a fresh random nonce. To simplify the presentation we remove the identities of senders and intended receivers from messages. Note that any discrepancy in the received content is automatically detected in this code: in the last message, if the first part does not agree with the initial $h(M)$, the message will not be accepted.

Communication models We consider two different communication models. The first is a synchronous communication model that is used to verify the

μCRL spec 1 Customer device in the P2C protocol

$$\mathrm{C}(\Omega, n_C) = \sum_{\substack{R \in Rgts \\ M \in Cont}} \mathbf{recv}(P, h(M), R).\mathbf{send}(C, n_C).$$
$$\sum_{n \in Nonce} \mathbf{recv}(\{n,\ n_C,\ C\}_{sk(P)}).$$
$$\mathbf{send}(\{n_C,\ n,\ h(M),\ R,\ P\}_{sk(C)}).request(C, h(M), R, C).$$
$$\sum_{K \in Key} \mathbf{recv}(\{M\}_K,\ \{K\}_{pk(C)},\ \{R,\ n_C\}_{sk(P)}).update(C, h(M), R, P).$$
$$\mathrm{C}(\Omega \cup \{\langle M, R\rangle\}, nxt(n_C))$$

effectiveness property (goal G1). In this model there is no intruder and all participants are honest. A process p can send a message m to q only if q at the same time can receive it from p. The synchronization between these is denoted **com**, which formalizes the "$p \to q : m$" notation of Sections 3 and 4. In order to verify the properties G2–G4, an asynchronous communication model is used where the intruder has complete control over the communication media. When a process p sends a message m with the intention that it should be received by q, it is in fact the intruder that receives it, and it is only from the intruder that q may receive m. The communications between participants of a protocol, via the intruder, is thus asynchronous and, moreover, a participant has no guarantees about the origins of the messages it receives.

Intruder model We follow Dolev and Yao's approach to model the intruder [14], with some deviations that are described below. The Dolev-Yao (DY) intruder has complete control over the network: it intercepts and remembers all transmitted messages, it can encrypt, decrypt and sign messages if it knows the corresponding keys, it can compose and send new messages using its knowledge and it can remove or delay messages in favor of others being communicated. As it has complete control over communication media, we assume it plays the role of the communication media. All messages are thus channeled through the intruder. Under the perfect cryptography assumption, this intruder has been shown to be the most powerful attacker model [11]. In our formalization, this intruder is a non-deterministic process that exhausts all possible sequences of actions, resulting in an LTS which can subsequently be formally checked. Note that the intruder is not necessarily an outside party: it may be a legitimate, though malicious, player in the protocol.

The intruder model used here is different from the DY intruder in two main aspects (for a formal specification of our intruder model see [24]). These differences stem from the characteristics of the DRM scheme and its requirements:

I1. Trusted devices, that play a crucial role in these protocols, significantly limit the power of the intruder[3]. However, the intruder has the ability to deliberately turn off its (otherwise trusted) devices. This has been reflected in our model by allowing the devices controlled by the intruder to non-deterministically choose

[3] In our formalization we ignore the possibility of tampering trusted devices. Countermeasures for such cases are discussed in [24, 30].

between continuing and quitting the protocol at each step, except when performing atomic actions. Therefore, in the model, all non-atomic actions a of the devices operated by the intruder are rewritten with $a + \mathit{off}$. Thus, the intruder cannot turn compliant devices off while these devices are performing an atomic action. We verify the protocols in the presence of this enriched intruder model to capture possible security threats posed by these behaviors.

I2. Liveness properties of protocols can in general not be proved in the DY model, since the intruder can block all communications. To achieve fairness, which inherently comprises a liveness property (see Section 4.1), optimistic fair exchange protocols often rely on a "resilient communication channels" (RCC) assumption, e.g. see [26]. RCC guarantee that all transmitted messages will *eventually* reach their destination, provided a recipient for them exists [4]. The behavior of our intruder model is limited by RCC, i.e. it cannot indefinitely block the network.[4] Since the intruder is a non-deterministic process in our model, to exclude executions that violate RCC, we impose a fairness constraint [5] on the resulting LTS. Besides, the action $\mathbf{com}^{\dagger}$, used in Section 5.3, represents communications not required by RCC. A protocol has to achieve its goals even when executions containing $\mathbf{com}^{\dagger}$ actions are avoided. A formal treatment of these issues is beyond the scope of this paper and can be found in [10].

As a minor deviation from DY, to indicate violation of the secrecy requirement, the intruder process performs the abstract action *revealed* when it gets access to a non-encrypted version of any DRM-protected content. This action is of course not triggered when the intruder merely renders an item using its trusted device, which is a normal behavior in the system.

5.2 Regular μ-calculus

The design goals of Nuovo DRM (G1-G4) are encoded in the regular μ-calculus [28]. This logic covers the Nuovo DRM's design goals in its entirety, both safety and liveness, and naturally incorporates data parameters that are exchanged in the protocols. The alternation-free fragment of the regular μ-calculus can be efficiently model checked [28], and all the formulas that we have verified are in this fragment. Below, a short account of this logic is presented.

Regular μ-calculus consists of *regular formulas* and *state formulas*. Regular formulas, describing sets of traces, are built upon *action formulas* and the standard regular expression operators. We use '.', '$\vee$', '$\neg$' and '*' for concatenation, choice, complement and transitive-reflexive closure, respectively, of regular formulas. State formulas, expressing properties of states, are built upon propositional variables, standard boolean operators, the possibility modal operator $\langle \cdots \rangle$ (used in the form $\langle R \rangle \mathsf{T}$ to express the existence of an execution of the protocol

[4] For instance, a wireless channel provides RCC for mobile devices, assuming that jamming can only be locally sustained.

[5] Two different notions of fairness are used in this paper: fairness in exchange (see G4) and fairness constraint of an LTS, which informally states that each process of the system has to be given a fair chance to execute [12].

for which the regular formula R holds), the necessity modal operator $[\cdots]$ (used in the form $[R]\mathsf{F}$ to express that, for all executions of the protocol, the regular formula R does not hold) and the minimal and maximal fixed point operators μ and ν. A state satisfies $\mu X.\ F$ iff it belongs to the minimal solution of the fixed point equation $X = F(X)$, F being a state formula and X a set of states. The symbols F and T are used in both action formulas and state formulas. In action formulas they represent *no action* and *any action* and in state formulas they denote the empty set and the entire state space, respectively. The wild-card action parameter '$-$' represents any parameter of an action.

5.3 Analysis results

In this section we describe the results obtained from the formal analysis of the Nuovo DRM scheme. Our analysis has the following properties: the intruder is allowed to have access to unbounded resources of data (like fresh nonces), should it need them to exploit the protocol. We consider only a finite number of concurrent sessions of the protocol, i.e. each participant is provided a finite number of fresh nonces to start new exchange sessions. Although this does not, in general, constitute a proof of security for a protocol, in many practical situations it suffices. As security of cryptographic protocols is not decidable (e.g. see [13]), a trade-off has to be made between completeness of the proofs and their automation. Our analysis method is fully automatic. Following [14], we assume perfect cryptography and do not consider attacks resulting from weaknesses of the cryptographic apparatus used in protocols. Type-flaw attacks[6] are also omitted from our analysis. These can, in any case, be easily prevented [23].

Our formal analysis consists of two scenarios. The first verifies effectiveness (G1) while using the synchronous communication model of Section 5.1. The second scenario uses the asynchronous communication model of Section 5.1 to verify the remaining properties (G2-G4). Both scenarios consist of two compliant devices C and D that are controlled (but not tampered) by the intruder of Section 5.1. Below, P, as always, represents the trusted content provider. The formulas in the following results use abstract actions to improve the readability of the proved theorems. These actions are explained in Sections 4.3 and 5.1. A complete formalization of these actions can be found in [24].

Honest scenario S_0: the communication network is assumed operational and no malicious agent is present. C is ordered to buy an item from P. Then, C resells the purchased item to D. This scenario was checked using the Evaluator 3.0 model checker from the cadp toolset, confirming that it is deadlock-free, and effective as specified below.

Result 1 *Nuovo DRM is effective for scenario* S_0*, meaning that it satisfies the following properties:*

[6] A type-flaw attack happens when a field in a message that was originally intended to have one type is interpreted as having another type.

1. *Each purchase request is inevitably responded.*

$$\forall m \in Cont,\ r \in Rgts.\ [\mathsf{T}^*.request(C,m,r,P)]\ \mu X.(\langle\mathsf{T}\rangle\mathsf{T}\ \wedge\ [\neg update(C,m,r,P)]X) \\ \wedge \\ [\mathsf{T}^*.request(D,m,r,C)]\ \mu X.(\langle\mathsf{T}\rangle\mathsf{T}\ \wedge\ [\neg update(D,m,r,C)]X)$$

2. *Each received item is preceded by its payment.*

$$\forall m \in Cont,\ r \in Rgts. \\ [(\neg issue(P,m,r,C))^*.update(C,m,r,P)]\mathsf{F}\ \wedge\ [(\neg issue(C,m,r,D))^*.update(D,m,r,C)]\mathsf{F}$$

Dishonest scenario S_1: the intruder controls the communication network and is the owner of the compliant devices C and D. The intruder can instruct the compliant devices to purchase items from the provider P, exchange items between themselves and resolve a pending transaction. Moreover, the compliant device C can non-deterministically choose between following or aborting the protocol at each step, which models the ability of the intruder to turn the device off (see I1 in Section 5.1). We model three concurrent runs of the content provider P, and three sequential runs of each of C and D. The resulting model was checked using the EVALUATOR 3.0 model checker from the CADP toolset and the following results were proven.

Result 2 *Nuovo DRM provides secrecy in scenario S_1, i.e. no protected content is revealed to the intruder (see Section 5.1).*

$$\forall m : Cont.\ [\mathsf{T}^*.revealed(m)]\mathsf{F}$$

Result 3 *Nuovo DRM resists content masquerading attacks in S_1, ensuring that a compliant device only receives the content which it has requested.*

$$\forall a \in \{C,D\},\ m \in Cont,\ r \in Rgts.\ [(\neg request(C,m,r,D))^*.update(C,m,r,D)]\mathsf{F} \wedge \\ [(\neg request(D,m,r,C))^*.update(D,m,r,C)]\mathsf{F} \wedge \\ [(\neg request(a,m,r,P))^*.update(a,m,r,P)]\mathsf{F}.$$

Besides, the intruder cannot feed the self-fabricated content m_0 to compliant devices:

$$\forall a \in \{C,D\},\ r \in Rgts.\ [\mathsf{T}^*.update(C,m_0,r,D)]\mathsf{F} \wedge \\ [\mathsf{T}^*.update(D,m_0,r,C)]\mathsf{F} \wedge \\ [\mathsf{T}^*.update(a,m_0,r,P)]\mathsf{F}.$$

Result 4 *Nuovo DRM provides strong fairness in S_1 for P, i.e. no compliant device receives a protected content, unless the corresponding payment has been made to P.*

$$\forall a \in \{C,D\},\ m \in Cont,\ r \in Rgts.\ [(\neg issue(P,m,r,a))^*.update(a,m,r,P)]\mathsf{F} \\ \wedge \\ [\mathsf{T}^*.update(a,m,r,P).(\neg issue(P,m,r,a))^*. \\ update(a,m,r,P)]\mathsf{F}$$

Result 5 *Nuovo DRM provides strong fairness in S_1 for D, as formalized below*[7]*:*

[7] Strong fairness for C is not guaranteed here, as it can quit the protocol prematurely. A protocol guarantees security only for the participants that follow the protocol.

1. *As a customer: if a compliant device pays (a provider or reseller device) for a content, it will eventually receive it.* [8]
   Note that there are only finitely many TTPs available in the model, so the intruder, in principle, can keep all of them busy, preventing other participants from resolving their pending transactions. This corresponds to a denial of service attack in practice, which can be mitigated by putting time limits on transactions with TTPs. As we abstract away from timing aspects here, instead, the action $last_{ttp}$ is used to indicate that all TTPs in the model are exhausted by the intruder. In other words, as long as this action has not occurred yet, there is still at least one TTP available to resort to.

$$\begin{array}{ll}
\forall m \in Cont,\ r \in Rgts. & [\mathsf{T}^*.request(D,m,r,P).(\neg(update(D,m,r,P)))^*] \\
& \langle(\neg com^\dagger(-,-,-))^*.(update(D,m,r,P))\rangle\mathsf{T} \\
& \wedge \\
\forall m \in Cont,\ r \in Rgts. & [\mathsf{T}^*.request(D,m,r,C).(\neg(resolves(D) \vee update(D,m,r,C)))^*] \\
& \langle(\neg com^\dagger(-,-,-))^*.(resolves(D) \vee update(D,m,r,C))\rangle\mathsf{T} \\
& \wedge \\
& [(\neg last_{ttp})^*.request(D,m,r,C).(\neg last_{ttp})^*.resolves(D). \\
& (\neg(update(D,m,r,P) \vee last_{ttp}))^*] \\
& \langle(\neg com^\dagger(-,-,-))^*.update(D,m,r,P)\rangle\mathsf{T}
\end{array}$$

2. *As a reseller: no compliant device receives a content from a reseller device, unless the corresponding payment has already been made to the reseller.*

$$\begin{array}{ll}
\forall m \in Cont,\ r \in Rgts. & [(\neg issue(D,m,r,C))^*.update(C,m,r,D)]\mathsf{F} \\
& \wedge \\
& [\mathsf{T}^*.update(C,m,r,D).(\neg issue(D,m,r,C))^*.update(C,m,r,D)]\mathsf{F}
\end{array}$$

Note that the strong fairness notion that is formalized and checked here subsumes the timeliness property of goal G4, simply because when D starts the resolve protocol, which it can autonomously do, it always recovers to a fair state without any help from C.

Theorem 1. *Nuovo DRM achieves its design goals in scenarios S_0 and S_1.*

Proof. G1 is achieved based on result 1. Result 2 implies G2. Result 3 guarantees achieving G3. Results 4 and 5 guarantee G4.

6 Conclusions

We have formally analyzed the NPGCT DRM scheme and found two vulnerabilities in its protocols. The scheme is subsequently extended to address these vulnerabilities. The extended scheme, namely Nuovo DRM, as many other DRM systems, is inherently complicated and, thus, error prone. This calls for expressive and powerful formal verification tools to provide a certain degree of confidence in the security and fairness of the system. We have analyzed and validated our design goals on a finite model of Nuovo DRM. There is of course no silver bullet:

[8] The fairness constraint used in the formulas corresponds to the strong notion of fairness in [19]: $\forall\theta.\ F^\infty\, enabled(\theta) \Rightarrow F^\infty\, executed(\theta)$.

our formal verification is not complete as it abstracts away many details of the system. For instance, as future work, we are considering analyzing the accountability of the provider, which is taken as non-disputable in this study, addressing possible anonymity concerns of customers and incorporating the payment phase into the formal model. We are currently working on a practical implementation of Nuovo DRM using existing technologies, see [1].

Acknowledgments We are grateful to Bruno Crispo and Wan Fokkink, for their comments on earlier versions of this paper, and to Bert Lisser for his help with distributed state space generation.

References

1. DRM paradiso. `http://www.few.vu.nl/~srijith/paradiso/` and `formal.php` therein.
2. M. Abadi and B. Blanchet. Computer-assisted verification of a protocol for certified email. In *SAS '03*, volume 2694 of *LNCS*, pages 316–335, 2003.
3. B. Alpern and F. Schneider. Defining liveness. Technical Report TR 85-650, Dept. of Computer Science, Cornell University, Ithaca, NY, October 1984.
4. N. Asokan. *Fairness in electronic commerce*. PhD thesis, Univ. Waterloo, 1998.
5. G. Avoine, F. Gärtner, R. Guerraoui, and M. Vukolic. Gracefully degrading fair exchange with security modules. In *EDCC '05*, volume 3463 of *LNCS*, pages 55–71. Springer, 2005.
6. A. Basu, B. Charron-Bost, and S. Toueg. Simulating reliable links with unreliable links in the presence of process crashes. In *ACM WDAG '96*, volume 1151 of *LNCS*, pages 105–122. Springer, 1996.
7. G. Bella and L. C. Paulson. Mechanical proofs about a non-repudiation protocol. In *TPHOL'01*, volume 2152 of *LNCS*, pages 91–104. Springer, 2001.
8. S. Blom, J. Calame, B. Lisser, S. Orzan, J. Pang, J. van de Pol, M. Torabi Dashti, and A. Wijs. Distributed analysis with μCRL. In *TACAS '07*, 2007. to appear.
9. S. Blom, W. Fokkink, J. F. Groote, I. van Langevelde, B. Lisser, and J. van de Pol. μCRL: A toolset for analysing algebraic specifications. In *CAV'01*, volume 2102 of *LNCS*, pages 250–254. Springer, 2001.
10. J. Cederquist and M. Torabi Dashti. An intruder model for verifying liveness in security protocols. In *FMSE '06*, pages 23 – 32. ACM Press, 2006.
11. I. Cervesato. Data access specification and the most powerful symbolic attacker in MSR. In *ISSS '02*, volume 2609 of *LNCS*, pages 384–416. Springer, 2003.
12. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2000.
13. H. Comon and V. Shmatikov. Is it possible to decide whether a cryptographic protocol is secure or not? *J. of Telecomm. and Inform. Tech.*, 4:3–13, 2002.
14. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Trans. on Information Theory*, IT-29(2):198–208, 1983.
15. N. Evans and S. Schneider. Verifying security protocols with PVS: widening the rank function approach. *J. Logic and Algebraic Programming*, 64(2):253–284, 2005.
16. S. Even and Y. Yacobi. Relations amoung public key signature systems. Technical Report 175, Computer Science Department, Technicon, Haifa, Israel, 1980.
17. J.-C. Fernandez, H. Garavel, A. Kerbrat, R. Mateescu, L. Mounier, and M. Sighireanu. CADP: A protocol validation and verification toolbox. In *CAV '98*, volume 1102 of *LNCS*, pages 437–440. Springer, 1996.

18. M. Fischer, N. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
19. N. Francez. *Fairness*. Springer, 1986.
20. J. F. Groote and A. Ponse. The syntax and semantics of μCRL. In *Algebra of Communicating Processes '94*, Workshops in Computing Series, pages 26–62. Springer, 1995.
21. S. Gürgens, C. Rudolph, and H. Vogt. On the security of fair non-repudiation protocols. In *ISC '03*, volume 2851 of *LNCS*, pages 193–207, 2003.
22. J. Halderman and E. Felten. Lessons from the Sony CD DRM episode. In *the 15th USENIX Security Symposium*, pages 77–92, 2006.
23. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *CSFW '00*, pages 255 – 268. IEEE CS, 2000.
24. H. Jonker, S. Krishnan Nair, and M. Torabi Dashti. Nuovo DRM paradiso. Technical Report SEN-R0602, CWI, Amsterdam, The Netherlands, 2006. `ftp.cwi.nl/CWIreports/SEN/SEN-R0602.pdf`.
25. D. Kähler and R. Küsters. Constraint solving for contract-signing protocols. In *CONCUR '05*, volume 3653 of *LNCS*, pages 233–247. Springer, 2005.
26. S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. *Computer Communications*, 25(17):1606–1621, 2002.
27. S. Kremer and J. Raskin. A game-based verification of non-repudiation and fair exchange protocols. In *CONCUR'01*, volume 2154 of *LNCS*, pages 551–565. Springer, 2001.
28. R. Mateescu and M. Sighireanu. Efficient on-the-fly model-checking for regular alternation-free μ-calculus. *Sci. Comput. Program.*, 46(3):255–281, 2003.
29. C. Meadows. Formal methods for cryptographic protocol analysis: Emerging issues and trends. *IEEE J. Selected Areas in Communication*, 21(1):44–54, 2003.
30. S. Nair, B. Popescu, C. Gamage, B. Crispo, and A. Tanenbaum. Enabling DRM-preserving digital content redistribution. In *7th IEEE Conf. E-Commerce Technology*, pages 151–158. IEEE CS, 2005.
31. H. Pagnia, H. Vogt, and F. Gärtner. Fair exchange. *Comput. J.*, 46(1):55–75, 2003.
32. R. Pucella and V. Weissman. A logic for reasoning about digital rights. In *CSFW '02*, pages 282 – 294. IEEE CS, 2002.
33. V. Shmatikov and J. Mitchell. Finite-state analysis of two contract signing protocols. *Theor. Comput. Sci.*, 283(2):419–450, 2002.

A Resolving C2C disputes at the TTP

Here we describe how the TTP handles payment orders in resolving C2C disputes. This topic is however outside our current formalization of Nuovo DRM.

We define $\mathsf{price} : \mathit{Rgts} \to \mathbb{N}$. Given $R \in \mathit{Rgts}$, $\mathsf{price}(R)$ denotes the price that has been assigned to R (see assumption A4). In the resolve sub-protocol, P agrees with R'' iff $\mathsf{price}(R'') \geq \mathsf{price}(R')$ (c.f. Section 4.3); below we see why this check is necessary. In practice, resellers usually propose prices which are less than the main vendor's price for buying that single item, hence automatically satisfying this requirement.

We require P to maintain a persistent log of the resolved disputes. Assume that D tries to resolve an unsuccessful exchange with C. As a result of the atomicity of C's actions in the optimistic sub-protocol, only the following situations

are possible: Either C has updated $R_C(M)$ and has the payment order of message 4 of C2C (which it is thus entitled to have), or C does not have the payment order and has not updated $R_C(M)$. In the latter case, the combination of the failed optimistic protocol and its subsequent resolve simply boils down to a P2C exchange. If C owns the payment order from D, two different cases are possible:

1. If C tries to encash the payment order after D has resolved, P is the one who pays the money to C, as D has already paid P. Since $\mathsf{price}(R'') > \mathsf{price}(R')$, P can always pay C its share of the transaction. Therefore, the actual payment to P in this particular exchange sums up to $\mathsf{price}(R'') - \mathsf{price}(R')$. Note that although P is not finally (enough) paid for sending M to D in this particular exchange, it is indeed fair because P has already been paid by C when C received the right to resell M.
2. If D resolves after C has encashed the payment order, P will not charge D, because D has already paid the price and C has updated the right $R_C(M)$, for which it has already (directly or indirectly) paid P.

Note that C and D cannot collude to cheat P by C offering item M to D for an extremely low price and then resolving the request to P. To make this clear, consider the following model:

- Cost of buying song M for playing only, directly from P = \$1.00.
- Cost of buying song M for 50 resell rights from P = \$0.80 $\times$ 50 = \$40.
- Cost of buying song M for playing only, from reseller C = \$0.90.

First, this scenario describes a viable business model: D would rather buy M from C than directly from P because of the \$0.10 difference. C has incentive to act as reseller since it can make a profit of \$(0.9-0.8) $\times$ 50 if all the songs are sold. P would rather make one sale of 50 rights to C than sell to 50 Ds directly, to avoid all sorts of administration, processing and other per-transaction costs.

If C offers M to D for \$0.01 and they resolve it to P, P would transfer the money from D's account to C's without being paid in this exchange (P would accept resolving such unnecessary disputes to assure its customers that in case of real problems, they can resort to P). However, C is the one who actually loses money, because the profit of P has been granted when the resell rights were sold to C, and D has exploited a very good offer on M. If this scenario is repeated enough, C will sell contents for \$0.01$\times$ 50=\$0.5. At the end of the day, C can only render M, a music for which it has actually paid \$40-1.0-0.5=\$38.5 more than what was actually needed.