Fundamenta Informaticae XX (2008) 1–25
IOS Press

Nuovo DRM Paradiso: Designing a Secure, Verified, Fair Exchange DRM Scheme

Mohammad Torabi Dashti
CWI Amsterdam
dashti@cwi.nl

Srijith Krishnan Nair
Vrije Universiteit Amsterdam
srijith@few.vu.nl

Hugo Jonker [C]
Eindhoven University of Technology
h.l.jonker@tue.nl

Abstract. We introduce Nuovo DRM, a digital rights management scheme aimed to provide formal and practical security. The scheme is based on a recent DRM scheme, which we formally specify in the μCRL process algebraic language. The original scheme stated the following security requirements: effectiveness, secrecy and resistance of content masquerading. We formalise these security requirements as well as strong fairness and formally check the original scheme against these requirements. This verification step uncovered several security weaknesses, which are addressed by Nuovo DRM. In addition to that, Nuovo DRM introduces several procedural practices to enhance the practical security of the scheme. A finite model of Nuovo DRM is subsequently model-checked and shown to satisfy its design requirements, including secrecy, fairness and resistance to content masquerading.

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 unauthorised 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.

[C] Corresponding author

This problem is currently addressed by limiting the distribution of protected contents to only the so-called *compliant* devices (e.g. iPods), that by design 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 the NPGCT scheme, where users act also as content redistributors. This potentially allows consumers to not only buy the rights to use specific 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 authorised distributor, and also a potential attacker. The main goal of NPGCT is to enable content redistribution, whilst resisting systematic content pirating. Recent sobering experience [22] has shown that DRM techniques are inherently complicated and if not enforced carefully, can infringe on customers', as well as vendors', rights. This serves 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.

1.1. Contributions

Our contribution in this paper is twofold. First, on the security side, we formally specify the NPGCT protocols and analyse them. Our analysis reveals two security flaws in the scheme, a rights-replaying flaw and a problem with fair exchange between users. We 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 present the used 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 tool set [8] to specify the protocol participants and the intruder model. The expressive power and flexibility of the μCRL language compares favourably to other specification languages. These factors enable us to keep the formalisation close to the actual implementation. Due to the complexity, the size of the scheme and the branching nature of the protocols, generating the state space is a very time-consuming process. Several approaches to handle this so-called "state space explosion" exist, such as counter-based abstractions [32] or parametrised abstraction techniques [33]. These techniques are not straightforwardly applicable to our problem however, as they focus on abstracting away state details, which in a DRM setting amounts to abstracting away rights and content – exactly the main points of interest. In order to address state space generation, we resorted to a distributed instantiation of the μCRL tool set [7] to generate and minimise 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 Dolev-Yao intruder model [14] to reflect the restricted behaviour of compliant devices in DRM systems (which are not under *full* control of the intruder, even if owned by the intruder).

1.2. Related work

Nuovo DRM introduces an optimistic fair exchange protocol. This class of protocols was introduced in [3] and since then have attracted much attention. The closest fair exchange protocol to our scheme is perhaps the probabilistic synchronous protocol [4], in that it too relies on trusted computing devices in exchange. In contrast to [4], the optimistic fair exchange protocol in Nuovo DRM is a deterministic asynchronous protocol that achieves strong (as opposed to probabilistic) fairness, but, as a drawback, it relies on impartial agents to secure unsupervised exchanges.

In this paper we do not address modelling semantics and derivations of rights associated with DRM-protected contents, which constitutes a whole separate body of research, e.g. see [34]. Instead we focus on formal analysis of transactional properties of DRM schemes. There are several works in literature on model checking (usually small instances of) optimistic fair exchange protocols, e.g. [21, 27, 35]. What makes our study unique is the size of the system that is automatically analysed as well as the capturing of some DRM-specific features of the system, like compliant devices, in the model. Constraint solving for checking fair exchange protocols proposed in [25] can detect type-flaw attacks, but is restricted to checking safety properties. Theorem-proving approaches to checking fairness of protocols [1, 6, 15] can provide a complete security proof at the cost of heavy human intervention, and thus cannot be easily integrated into the protocol design phase.

1.3. Structure of the paper

We start by explaining the notations and (cryptographic) assumptions used in the paper, in Section 2. Section 3 summarises the NPGCT scheme, which provides the basis for our refined scheme. Section 4 presents the Nuovo DRM scheme, its assumptions, its goals and security procedures. Nuovo DRM is then formally analysed in Section 5 and shown to achieve its goals. Finally, Section 6 concludes the paper with some possible future research directions.

2. Assumptions and notations

Throughout the paper, the following assumptions are used.

Trusted devices assumptions. Compliant devices are tamper-proof hardware, that though possibly operated by malicious owners, 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. A legitimate content provider, (albeit abusively) referred to as trusted third party (TTP), is assumed impartial in its behaviour and eventually available to respond to requests from compliant devices.

Cryptographic assumptions and notations. 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 $\{m\}_{sk(X)}$.

Additionally, the following notation is used:

$d1, d2$	compliant devices
P	trusted, legitimate content provider
$owner(d1)$	owner of device $d1$
$c \in Cont$	one piece of content in the set of all contents
$r \in Rgts$	one right in the finite set of all possible rights
$R_{d1}(c)$	the rights of device $d1$ for content c

It is assumed that unique descriptors (e.g. hash values) of all $c \in Cont$ are publicly known.

3. The NPGCT DRM scheme

The NPGCT scheme by Nair et al. [30] was proposed as a DRM-preserving digital content redistribution system where a consumer doubles up as a content redistributor, without this adversely affecting the protection offered by the DRM scheme. 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 content from provider P to client $d1$, the second allows $d1$ to resell contents to another client $d2$.

3.1.1. Provider-customer protocol (P2C)

The protocol is initiated by the owner of $d1$ who wishes to buy item c with rights r from provider P. From [30]:

1. $d1 \rightarrow P$: Request content
2. $d1 \leftrightarrow P$: Mutual authentication, [payment]
3. $P \rightarrow d1$: $\{c\}_K, \{K\}_{pk(d1)}, r, \sigma, \Lambda$
   σ=meta-data of c, Λ=$\{h(P, d1, c, \sigma, r)\}_{sk(P)}$

The idea of step 2 is that a multi-stage authentication protocol is inserted at this step. Furthermore, in the protocol, Λ acts as a certification that $d1$ has been granted rights r and helps in proving $d1$'s right to redistribute c to other clients. It also binds the meta-data σ to the content, which prevents masquerading attacks on c.

3.1.2. Customer-customer protocol (C2C)

This part of the protocol is initiated by the owner of $d2$ who wants to buy c with rights r' from $d1$, for which $d1$ holds certificate Λ. From [30]:

$$\begin{array}{lrl}
1. & \mathit{d2} \rightarrow \mathit{d1}: & \text{Request content}\\
2. & \mathit{d1} \leftrightarrow \mathit{d2}: & \text{Mutual authentication}\\
3. & \mathit{d1} \rightarrow \mathit{d2}: & \{c\}_{K'}, \{K'\}_{pk(\mathit{d2})}, R_{\mathit{d1}}(c), r', \sigma, \Lambda, \Lambda'\\
 & & \Lambda' = \{h(\mathit{d1}, \mathit{d2}, c, \sigma, r')\}_{sk(\mathit{d1})}\\
4. & \mathit{d2}: & \text{Verifies } \sigma, \Lambda' \text{ and } R_{\mathit{d1}}(c) \text{ using } \Lambda\\
5. & \mathit{d2} \rightarrow \mathit{d1}: & \psi, [payment]\\
 & & \psi = \{h(\mathit{d1}, P, \{c\}_{K'}, \sigma, r')\}_{sk(\mathit{d2})}
\end{array}$$

By sending ψ, $\mathit{d2}$ acknowledges[1] that it has received c with rights r' from $\mathit{d1}$, while Λ and Λ' form a certificate chain that helps to prove that $\mathit{d2}$ has been granted rights r'.

3.2. Formal analysis of NPGCT

As part of our work, we formally specified and checked the NPGCT scheme. In this section, we present the results of this analysis. The assumptions of the scheme, the security goals it was tested against, their formalisation, the protocol specification tool set 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 the attack traces found by the verification process are available online[2].

Two security flaws in the NPGCT scheme were revealed by our analysis. First, it was found that in the P2C (and similarly the C2C) protocol, a malicious customer can feed rights from a previous session to the trusted device by replaying step 3. This replaying is possible because freshness of the authentication phase is not extended to guarantee freshness of step 3 (delivery of the content-right bundle). This flaw allows $\mathit{d1}$ 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 $\mathit{d2}$ receives c in step 3, the owner of $\mathit{d2}$ can avoid paying $\mathit{d1}$ by quiting 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 potentially dishonest device owners in C2C exchanges. (Note that fairness in exchange is not a goal of NPGCT.)

4. The Nuovo DRM scheme

This section describes the proposed 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's goals

The aim of Nuovo DRM is to enable content redistribution whilst resisting systematic content pirating. Hence, Nuovo DRM provides a secure DRM scheme which encompasses content redistribution. The

[1] Term ψ was intended to be used in c2c-dispute resolution, although that notion is not further explored in [30].

[2] `http://www.cs.vu.nl/paradiso/formal.php`

security of the scheme is to address normal DRM security concerns as well as security concerns introduced by content redistribution. This is captured by the following goals: We require the Nuovo DRM scheme to achieve the following goals (which are the same as those used to analyse the NPGCT scheme in Section 3.2):

G1 (effectiveness). A protocol achieves effectiveness iff for honest participants running the protocol in a secure environment, 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 any $c \in Cont$ not intended for him. Usually, content is encrypted for intended receivers. Nuovo DRM (similar to NPGCT) limits the distribution of protected contents by encrypting them for intended compliant devices. The scheme must thus guarantee that a DRM-protected content never appears unencrypted to any known non-compliant device.

G3 (resisting content masquerading). Content masquerading occurs when content c is passed off as content c', for $c \neq c'$. Preventing this attack ensures that an intruder cannot feed c' to a device that has requested c.

G4 (strong fairness). Assume Alice owns an item m_A and Bob owns an item m_B. Informally, strong fairness states that if Alice and Bob run a protocol to exchange their items, in the end 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 DRM, 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[3], resilient communication channels (assumption A2 below) are necessary for fairness to hold [3]. For an in-depth discussion of fairness in exchange we refer the interested reader to [3].

4.2. Assumptions of Nuovo DRM

The following assumptions are made regarding the working of Nuovo DRM. Note that assumptions A1 and A2 limit the power of the intruder, as explained further in Section 5.1.4.

A1 (tamper-proof devices). 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).

A2 (resilient communication). 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 [5].

[3] Properties of systems can often 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 [2].

A3 (PKI hierarchy). There exists a hierarchy of public keys, with the public key of the root authority 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 ($d1$, $d2$ and P) implicitly refer to these authentication certificates issued by the trusted authorities.

A4 (price negotiations). 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 $d1$, the second allows $d1$ to resell content to another client $d2$. These protocols derive from the NPGCT schemes, but are updated to incorporate authentication and strong fairness.

Provider-customer protocol (P2C) The owner of $d1$ wants to buy item c with rights r from content provider P. Here $d1$ and P, but not $owner(d1)$, are assumed trusted.

$$\begin{array}{lrl}
1. & owner(d1) \rightarrow d1: & P,\ h(c),\ r \\
2. & d1 \rightarrow P: & d1,\ n_{d1} \\
3. & P \rightarrow d1: & \{n_P,\ n_{d1},\ d1\}_{sk(P)} \\
4. & d1 \rightarrow P: & \{n_{d1},\ n_P,\ h(c),\ r,\ P\}_{sk(d1)} \\
5. & P \rightarrow d1: & \{c\}_K,\ \{K\}_{pk(d1)},\ \{r,\ n_{d1}\}_{sk(P)}
\end{array}$$

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 $d1$. Following assumption A4, $owner(d1)$ and P have already reached an agreement on the price. Whether P is a legitimate provider can be checked by $d1$ and vice versa (see assumption A3). In step 2, $d1$ generates a fresh nonce n_{d1} and sends it to P, which will continue the protocol only if $d1$ is a compliant device. Message 4 completes the mutual authentication between $d1$ and P. This also constitutes a *payment order* from $d1$ 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 $d1$ receives message 5, it decrypts $\{K\}_{pk(d1)}$, extracts c and checks if it matches $h(c)$ in message 1, and n_{d1} is the same as the nonce in message 2. If these tests pass, $d1$ updates $R_{d1}(c)$ with r, e.g. r is added to $R_{d1}(c)$. Note that $R_{d1}(c)$ is not necessarily r: $d1$ could already have some rights associated with c, for instance, acquired from an earlier purchase. Since we abstract away from rights semantics (as discussed in Section 1.2), the update phase is left unspecified here.

4.3.1. Customer-customer protocol (C2C)

The owner of $d2$ wants to buy item c with rights r' from another compliant device $d1$. This protocol can be seen as a fair exchange protocol where $d1$ and $d2$ want to exchange a content-right bundle for its associated payment such that either both or neither of them receive their desired items. In deterministic protocols, however, achieving fairness has been proven to be impossible without a TTP [16]. Assuming that most participants are honest and protocols go wrong only 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 [3] and contain two sub-protocols: an optimistic sub-protocol which 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[4]. 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:

$$\begin{array}{lrl}
1. & \mathit{owner}(\mathit{d2}) \rightarrow \mathit{d2}: & \mathit{d1},\ h(c),\ r' \\
2. & \mathit{d2} \rightarrow \mathit{d1}: & \mathit{d2},\ n_{\mathit{d2}} \\
3. & \mathit{d1} \rightarrow \mathit{d2}: & \{n'_{\mathit{d1}},\ n_{\mathit{d2}},\ \mathit{d2}\}_{sk(\mathit{d1})} \\
4. & \mathit{d2} \rightarrow \mathit{d1}: & \{n_{\mathit{d2}},\ n'_{\mathit{d1}},\ h(c),\ r',\ \mathit{d1}\}_{sk(\mathit{d2})} \\
5. & \mathit{d1} \rightarrow \mathit{d2}: & \{c\}_{K'},\ \{K'\}_{pk(\mathit{d2})},\ \{r',\ n_{\mathit{d2}}\}_{sk(\mathit{d1})}
\end{array}$$

At step 5, $\mathit{d1}$ updates the right associated with c (reflecting that some part of $R_{\mathit{d1}}(c)$ has been used for reselling c) and stores the payment order signed by $\mathit{d2}$ in an atomic action. Note that the atomicity of these actions is necessary to guarantee that $\mathit{d1}$ does not store the payment order without simultaneously updating the right $R_{\mathit{d1}}(c)$.

In this protocol, a malicious $\mathit{owner}(\mathit{d1})$ can abort the protocol before sending message 5 to $\mathit{d2}$ or this message may be lost due to a hardware failure. To prevent such unfair situations for $\mathit{d2}$, we provide a recovery mechanism to obtain the lost content.

4.3.2. Recovery sub-protocol

The goal of the recovery sub-protocol is to bring compliant device $\mathit{d2}$ back to a fair state in case of a failure in delivering message 5 in the C2C protocol. Device $\mathit{d2}$ can start a recovery session (instead of receiving the content at step 5, $\mathit{d2}$ takes action $\mathit{resolves}(\mathit{d2})$) 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, $\mathit{d2}$ saves the current state and simply waits until it becomes available. Once the recovery protocol has been initiated, $\mathit{d2}$ ignores any further messages from the optimistic run of C2C. The purpose of the recovery is to ensure that $\mathit{d2}$ receives the content and rights that $\mathit{owner}(\mathit{d2})$ wanted (and ostensibly paid for).

$$\begin{array}{lrl}
5^r. & \mathit{d2}: & \mathit{resolves}(\mathit{d2}) \\
6^r. & \mathit{d2} \rightarrow P: & \mathit{d2},\ n'_{\mathit{d2}} \\
7^r. & P \rightarrow \mathit{d2}: & \{n'_P,\ n'_{\mathit{d2}},\ \mathit{d2}\}_{sk(P)} \\
8^r. & \mathit{d2} \rightarrow P: & \{n'_{\mathit{d2}},\ n'_P,\ \langle n_{\mathit{d2}},\ n'_{\mathit{d1}},\ h(c),\ r',\ \mathit{d1}\rangle,\ r'',\ P\}_{sk(\mathit{d2})} \\
9^r. & P \rightarrow \mathit{d2}: & \{c\}_{K''},\ \{K''\}_{pk(\mathit{d2})},\ \{r'',\ n'_{\mathit{d2}}\}_{SK(P)}
\end{array}$$

In this protocol $\mathit{d2}$ and P behave as if $\mathit{d2}$ is purchasing the c-r'' content-right bundle from P using the P2C protocol, except that, in message 8^r, $\mathit{d2}$ reports the failed C2C exchange it had with $\mathit{d1}$.

The way P resolves payments of failed exchanges is discussed in more detail in Section 4.4.1. Note that while payment details fall beyond the scope of our formal analysis, the recovery protocol does not.

[4] 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.

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 [3] (c.f. assumption A2).

As a final note, we emphasise 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 $d2$ 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 the following sections.

4.4. Nuovo DRM procedures

Nuovo DRM introduces several procedures to support the generic objective of Nuovo DRM to enable content redistribution whilst resisting systematic content pirating. In this section, these procedures are discussed. First, resolving of failed C2C exchanges by the provider is detailed. Next, a method is described to detect systematic content pirating. And finally, an approach to prevent interaction with compromised devices is discussed. Note that the procedures below fall beyond the scope of our formal analysis.

4.4.1. Resolving C2C disputes at the TTP

We define $\mathsf{price}\colon Rgts \to \mathbb{N}$. Given a $r \in Rgts$, $\mathsf{price}(r)$ denotes the price that has been assigned to r (see assumption A4). In the recovery protocol, the provider will agree to resolve a C2C exchange for right r'' (steps $8^r, 9^r$) iff $\mathsf{price}(r'') \geq \mathsf{price}(r')$ (from step 8^r); below we see why this condition is necessary. In line with assumption A1, we only consider compliant devices that need to resolve – the device cannot lie about $price(r')$ and $price(r'')$. In practice, resellers will usually propose prices which are lower 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 $d2$ tries to resolve an unsuccessful exchange with $d1$. As a result of the atomicity of $d1$'s actions in the optimistic sub-protocol, only the following situations are possible: Either $d1$ has updated $R_{d1}(c)$ and has the payment order of message 4 of C2C (which it is thus entitled to have), or $d1$ does not have the payment order and has not updated $R_{d1}(c)$. In the latter case, the combination of the failed optimistic protocol and its subsequent resolution simply boils down to a P2C exchange. If $d1$ owns the payment order from $d2$, two different cases are possible:

1. If $d1$ tries to encash the payment order after $d2$ has resolved, P is the one who pays the money to $d1$, as $d2$ has already paid P. Since $\mathsf{price}(r'') > \mathsf{price}(r')$, P can always pay $d1$ 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 paid (enough) for sending c to $d2$ in this particular exchange, it is indeed fair because P has already been paid by $d1$ when $d1$ bought the right to resell c.

2. If $d2$ resolves after $d1$ has encashed the payment order, P will not charge $d2$, because $d2$ has already paid the price to $d1$ and $d1$ has updated the right $R_{d1}(c)$, for which it has already (directly or indirectly) paid P.

Note that $d1$ and $d2$ cannot collude to cheat P by $d1$ offering item c to $d2$ for an extremely low price and then resolving the request to P. To make this clear, consider the following use case:

- Cost of buying song c for playing only, directly from P = \$1.00.
- Cost of buying song c for 50 resell rights from P = \$0.80 × 50 = \$40.
- Cost of buying song c for playing only, from reseller $d1$ = \$0.90.

First, this scenario describes a viable business model: $d2$ would rather buy c from $d1$ than directly from P because of the \$0.10 difference. $d1$ has incentive to act as reseller since it can make a profit of \$(0.9-0.8) × 50 if all the songs are sold. P would rather make one sale of 50 rights to $d1$ than sell to 50 $d2$s directly, to avoid all sorts of administration, processing and other per-transaction costs (it is common for services such as MasterCard or PayPal to have per-transaction charges consisting of a fixed part and a part dependent to the transaction).

If $d1$ offers c to $d2$ for \$0.01 and they resolve it to P, P would transfer the money from $d2$'s account to $d1$'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, $d1$ is the one who actually loses money. P's profit was already made when the resell rights were sold to $d1$, and $d2$ has exploited a very good offer on c. If this scenario is repeated enough, $d1$ will sell contents for \$0.01× 50 = \$0.50. At the end of the day, $d1$ paid $\$40 - \$1.00 - 0.50 = \$38.50$ more than the market price for c.

Seeing that the TTP cannot be cheated by compliant devices, even if their owners are colluding, the provider can safely be considered a TTP. The provider's interests are not harmed, and the role of TTP allows it to offer an extra service to its customers.

4.4.2. Detection of compromised devices

The security of Nuovo DRM hinges on the compliance of the certified user devices. However, it is reasonable to assume that over time, some of these devices will be compromised. In this section, we examine how to detect compromised devices. As in [30], the proposed mechanism aims at detecting powerful attackers and systematic content pirating, rather than occasionally misbehaving users. Hence, we consider a device to be compromised if it misbehaves frequently. So instead of compliance checks, the aim is to detect devices exhibiting deviant behaviour.

Nuovo DRM enables content redistribution in a controlled manner. In addition to regular attacks[5] on DRM systems, Nuovo DRM has to consider attacks on content redistribution as well. As compliant devices will not misbehave, only compromised devices can perform these attacks. A compromised device can attack content redistribution phase in two ways. First, it can overuse a right to resell content; secondly, it can try to avoid paying for content it receives (by not having sufficient funds). These are discussed in order.

Fund exchange during content redistribution is clearly a crucial point of attack. Successful attacks would undermine users' confidence in the system and the benefit to the attacker is clear: acquire more funds and spend less funds, respectively. In order to address this, we introduce the following assumptions (for compliant devices) on funds, which so far have not been considered.

[5] Content extraction attacks, related discussions and countermeasures on this are available from [30], and thus not further detailed here.

A5 (device funds). When a compliant device signs a payment order, the payment order is cash-able. This can be accomplished, for instance, by providing each compliant device with some credit, which can be spent and recharged.

A6 (traceability of funds). The banking system (responsible for encashing payment orders) cooperates with content providers[6] to catch malevolent users. Here, for the sake of simplicity, the content provider and the bank are considered as one and the same entity.

First, consider an overuse of reselling rights. To detect large scale overselling, the provider reconstructs the chain of sold rights. This is possible because of assumption A6 – to acquire payment for sold rights, devices need to contact the provider.

To this end, the provider maintains a directed weighted graph $G = (V, E)$ for each sold content-right combination, that may be resold. Each node $v \in V$ represents a device and the weighted edges of the graph ($E\colon V \times V \to \mathbb{N}$) represents right transfers between two compliant devices. For each $v \in V$, *weight difference* is the difference between outgoing weight and incoming weight. Formally:

$$\Delta(v) = \sum_{v' \in V} E(v, v') - \sum_{v' \in V} E(v', v) \tag{1}$$

Let $U \subseteq V$ be the set of nodes that have sold a content-rights bundle, but have not yet encashed the payment order. If v_c is a compromised device which engages in large scale overselling, after a reasonable amount of time, the provider will detect v_c's behaviour by noting that the weight difference of v_c plus the number of yet-to-cash rights are positive, i.e.

$$\Delta(v_c) + \sum_{u \in U} \Delta(u) > 0 \tag{2}$$

By putting time limits on encashing payment orders, a provider can control the time bound on detecting compromised devices. Such an approach requires the payment orders to be timestamped by the device issuing the order. Timestamps of compliant devices can be trusted, and the overhead to check the timestamp against the time limit is very small. Hence this solution scales well with the size of the system.

Secondly, a compromised device can refuse to pay for the content it receives. According to assumption A5, user devices are provided with (and thus aware of) credits. Therefore, the second attack could easily be detected by the banking entity (collaborating with the providers) when a device signs a payment order without having enough credit for that, as a compliant device would not cause this error.

4.4.3. Isolating compromised devices

Given that cheating – and thus compromised – devices are detected, countermeasures can be taken. Confiscation of the compromised device is of course preferred. However, in practice this will not always be possible. Instead, a Device Revocation List (DRL), containing public keys of detected compromised

[6] Though this assumption may not be universally acceptable, e.g. due to geographical diversity of content providers and banking systems used by customers, the required degree of collaboration makes the assumption practically tenable.

devices, can be used to limit interactions of compliant devices with them. To ensure correct working of device revocation, soundness of this DRL is required – no compliant device is ever listed on the DRL.

Completeness of the DRL also seems desirable: all compromised devices are listed on the DRL. However, as more and more compromised devices are detected, such a list could grow quite large over time. Given that not all devices are equally likely to interact, there is a trade-off between effectiveness and size of the DRL stored on a compliant device.

Considering these two properties immediately gives rise to two alternative ways of distributing the DRL: optimising effectiveness and optimising size, respectively. Optimising effectiveness of the DRL is done by keeping the complete DRL and updating it at all possible opportunities. In this case, each devices has a *complete copy* of the DRL. Optimising size of the DRL is done by adding only those compromised devices with which a device has had contact with earlier – only *check friends*.

Below we examine these two distribution schemes, and propose and refine variants that aim to balance these two consideration. Furthermore, estimates for the effectiveness and size of the per-device stored DRL are established.The following notation is used below:

drl The main DRL, as kept by P

drl_{d1} the DRL as kept by device $d1$

$friends_{d1}$ the list of devices with which device $d1$ has had contact. To keep the size of this list within reasonable bounds, it is reset after each contact with the provider that updates the DRL.

Updates of variables are denoted as $new\_a, new\_b, new\_c := a, b, c$, where the left-hand side denotes the variables after the update and the right-hand side expresses the values that are assigned.

complete copy: Each device keeps a copy of the entire DRL.
Update on $d1 \leftrightarrow P$: $drl_{d1} := drl$.
Update on $d1 \leftrightarrow d2$: $drl_{d1}, friends_{d1} := drl_{d1} \cup drl_{d2}, friends_{d1} \cup \{d2\}$.

friend-check: A device only lists those revoked devices, with which it has had contact.
Update on $d1 \leftrightarrow P$: $drl_{d1} := friends_{d1} \cap drl$.
Update on $d1 \leftrightarrow d2$: $friends_{d1} := friends_{d1} \cup \{d2\}$.

The next scheme strikes a balance between the two approaches above. On contacting the provider, it updates as *friend-check* (remembering the old list, however). On contact with another device, it updates as *complete copy*.

propagated list: Each device includes the DRL of all devices it has contacted.
Update on $d1 \leftrightarrow P$: $drl_{d1} := (drl_{d1} \cup friends_{d1}) \cap drl$.
Update on $d1 \leftrightarrow d2$: $drl_{d1}, friends_{d1} := drl_{d1} \cup drl_{d2}, friends_{d1} \cup \{d2\}$.

Depending on the interconnectivity of devices, the entire DRL could quickly reside on each device (equivalent to the "six degrees of separation" idea, which roughly states that everyone is at most six handshakes away from every other person on the planet). In this case, *propagated list* would be a complex version of *complete copy*. A size-limiting refinement of *propagated list* is to not simply forward all incoming DRLs, but only forward those devices on the DRL with which a device has had contact itself.

To this end, the per-device DRL is partitioned in two: self_{d1} lists those revoked devices, with which $d1$ has had contact. rest_{d1} accumulates the DRLs learned from other devices. So $\mathit{drl}_{d1} = \mathit{self}_{d1} \cup \mathit{rest}_{d1}$.

With this in mind, *propagated list* is refined as follows.

restricted propagation: each device includes the DRL of all devices it has contacted, but does not propagate this further.
Update on $d1 \leftrightarrow P$: $\mathit{self}_{d1} := \mathit{friends}_{d1} \cap \mathit{drl}$.
Update on $d1 \leftrightarrow d2$: $\mathit{rest}_{d1}, \mathit{friends}_{d1} := \mathit{rest}_{d1} \cup \mathit{self}_{d2}, \mathit{friends}_{d1} \cup \{d2\}$.

Note that, given the partitioning of the DRL into two, another approach is to validate the received DRLs before further propagation.

validated propagation: propagated lists are only included after validation by the provider.
Update on $d1 \leftrightarrow P$: $\mathit{drl}_{d1} := (\mathit{self}_{d1} \cup \mathit{rest}_{d1} \cup \mathit{friends}_{d1}) \cap \mathit{drl}$.
Update on $d1 \leftrightarrow d2$: $\mathit{rest}_{d1}, \mathit{friends}_{d1} := \mathit{rest}_{d1} \cup \mathit{self}_{d}, \mathit{friends}_{d1} \cup \{d2\}$.

Remark that this is a sanitised version of *propagated list*, as $\mathit{drl}_{d1} = \mathit{self}_{d1} \cup \mathit{rest}_{d1}$. Hence, this distribution model is not considered further.

Of the above schemes, restricted propagation seems to offer the best trade-off of list length versus usefulness of the DRL. On a further note, the impact of a corrupted DRL can be limited if each device cleans its DRL every time it contacts the provider ($\mathit{drl}_{d1} := \mathit{drl}_{d1} \cap L$). This would also allow the provider to “un-revoke” devices.

To get a feeling for the inherent effectiveness and list size of a given distribution scheme, the distribution schemes are examined in the following setting.

Use case. Given the number of devices $n = 10^7$ and the fraction of compromised devices $e = 10^{-3}$ (i.e. the number of compromised devices is in the order of 10^4). Assume that the devices interact via a network structure, each device interacting with k other devices, that no three devices interact (i.e. the number of unique neighbours of second degree is $k \cdot (k-1)$), and that secure memory can only hold a list of at most 10 id’s (due to memory limitations). Assume additionally that any compromised device is aware of the distribution of the list, and so knows which devices list it and which do not. Furthermore, we assume a compromised device can contact any other device (i.e. they do not follow the network structure, but choose with whom to interact without restrictions).

The restricted propagation scheme will only revoke compromised devices that interact with a device or with a neighbour of the device.

To calculate with how many different devices a device can have interacted, while limiting the interactions with compromised devices, consider the following.

complete copy: Each device needs to store the complete DRL, which consists of $e \cdot n = 10^4$ entries. This list cannot be stored in secure memory.

friends-check: Each device needs to store only those compromised devices it encountered, i.e. the list size is $e \cdot k$. For a maximum list size of 10, this distribution scheme can accommodate networks of up to $k = 10^4$.

propagated list: After every interaction, the DRL can only grow. Given the extremely regular, total connected network, this means that eventually, every revoked device will be listed. The time it takes for this to happen depends on the degree of separation. Within the setting sketched above, the degree of separation is approximately $\frac{\log N}{\log k-1}$.

restricted propagation: The number of entries in the per-device DRL depends on the number of interactions as follows: $e \cdot neighbours\ 1^{st} degree \cdot neighbours\ 2^{nd} degree$. For a maximum list size of 10 this means $0.001 \cdot (k^2 - k) \leq 10$, which is approximately $k^2 \leq 10^4$. Hence this suffices for interacting with up to 100 different, unique devices.

In order to measure the effectiveness of a distribution method, we consider how limiting a scheme is for a compromised device. This is equal to the number of compliant devices which lists a compromised device – i.e., the number of devices that will not interact with a compromised device. Note that due to our assumption that a compromised device (or its owner) is aware of which devices list it and may contact any other device, compromised devices may be considered to be uniformly distributed throughout the set of devices. Hence, the number of devices not interacting with a specific compromised device is equal to the average size of the per-device DRL. So given the uniformity assumption, this metric is equal to the size of the list, which was discussed above.

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]. As we base our approach on finite-state model-checking, our formal verification must have a finite number of states and thus necessarily concerns a limited instance of the system. 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 tool set (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 tool set. These steps are described in detail below.

To highlight important processing steps in the protocols, we now introduce several *abstract* actions. These are used in the formalisation process to define desired behaviours of the protocol.

$request(d1, h(c), r, P)$ Executed at step 4 in both the P2C and (with appropriate parameters) the C2C protocols by the receiving device, this action indicates the start of the exchange from the receiving device's point of view.

$paid(P, h(c), r, d1)$ Executed at step 5 of the P2C protocol by P, this action indicates reception of the payment order and sending of content to $d1$.

$update(d1, h(c), r, P)$ Executed after accepting the message of step 5 of the P2C protocol by $d1$, this action indicates the successful termination of the exchange from $d1$'s point of view.

$request(d2, h(c), r', P)$ Executed at step 8^r of the C2C recovery protocol by $d2$.

$paid(P, h(c), r', d2)$ Executed at step 9^r of the C2C recovery protocol by P.

$update(d2, h(c), r', P)$ Executed after acceptance of the message of step 9^r of the C2C recovery protocol by $d2$.

5.1. Formal specification of Nuovo DRM

The complex structure of Nuovo DRM calls for an expressive specification language. We have formalized[7] 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 labelled transition system (LTS), in which states represent process terms and edges are labelled with actions. The μCRL tool set [8, 7], together with CADP [17] which acts as its back-end, features visualisation, 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 $\mathbf{send}$ and $\mathbf{recv}$, 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 and a μCRL model of the intruder. For a complete specification of Nuovo DRM see [24]. We start with a brief introduction to μCRL, which suffices to understand the formal protocols.

5.1.1. 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 expression $p \lhd b \rhd q$, where b is a term of type $\mathbf{Bool}$ and p and q are processes, behaves like p if b is true, and like q if b is false. The predefined action δ represents a deadlock, i.e. from then on, no action can be performed. The process $\sum\limits_{d \in \Delta} P(d)$, where Δ is a (possibly infinite) data domain, behaves as $P(d_1) + P(d_2) + \cdots$.

5.1.2. The customer process

In μCRL specification 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 $d1$'s local collection of content-right bundles, n_{d1} denotes the nonce that is available to $d1$ in the current protocol round, and the function $nxt : Nonce \to Nonce$,

[7] Available from `http://www.cs.vu.nl/paradiso/formal.php`

generates a fresh random nonce, given a seed. 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 $h(c)$, the message will not be accepted.

μCRL specification 1 Customer device in the P2C protocol

$$
\begin{aligned}
\mathit{d1}(\Omega, n_{\mathit{d1}}) = \sum_{\substack{r \in Rgts \\ c \in Cont}} & \mathbf{recv}(P, h(c), r).\mathbf{send}(\mathit{d1}, n_{\mathit{d1}}). \\
& \quad \sum_{n \in Nonce} \mathbf{recv}(\{n,\ n_{\mathit{d1}},\ \mathit{d1}\}_{sk(P)}). \\
& \qquad \mathbf{send}(\{n_{\mathit{d1}},\ n,\ h(c),\ r,\ P\}_{sk(\mathit{d1})}).\mathit{request}(\mathit{d1}, h(c), r, \mathit{d1}). \\
& \qquad \sum_{K \in Key} \mathbf{recv}(\{c\}_K,\ \{K\}_{pk(\mathit{d1})},\ \{r,\ n_{\mathit{d1}}\}_{sk(P)}).\mathit{update}(\mathit{d1}, h(c), r, P). \\
& \qquad\quad \mathit{d1}(\Omega \cup \{\langle c, r\rangle\}, \mathit{nxt}(n_{\mathit{d1}}))
\end{aligned}
$$

5.1.3. Communication models

We consider two different communication models. The first is a synchronous communication model that is used to verify the effectiveness property (goal G1). No intruder is present in this model and all participants are honest. A process p can send a message m to q only if q can at the same time receive it from p. The synchronisation between these is denoted $\mathbf{com}(p, m, q)$, 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, are thus asynchronous and, moreover, a participant has no guarantees about the origins of the messages it receives.

5.1.4. Intruder model

We follow the approach of Dolev and Yao to model the intruder [14], with some deviations as 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 can remove or delay messages in favour 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 channelled through the intruder. Under the perfect cryptography assumption, this intruder has been shown to be the most powerful attacker model [11]. In our formalisation, 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. 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[8]. 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 between continuing or 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 $a + \mathit{off}$. Note that the intruder cannot violate the atomicity of actions for compliant devices. We verify the protocols in the presence of this enriched intruder model to capture possible security threats posed by these behaviours.

I2 Liveness properties of protocols cannot in general be proven in the DY model since the intruder can block all communications. To achieve fairness, essentially a liveness property (see Section 4.1), optimistic fair exchange protocols often rely on a "resilient communication channel" (RCC) assumption, see for example [26]. RCC guarantees that all transmitted messages will *eventually* reach their destination, provided a recipient for them exists [3]. The behaviour of our intruder model is limited by RCC, in that it may not indefinitely block the network. Since the intruder is a non-deterministic process in our model, in order to exclude executions that violate RCC, we impose a fairness constraint[9] on the resulting LTS. To denote communications not required by the RCC, we use the action modifier $^\diamond$ on regular communication actions (in actions $\mathbf{send}^\diamond$ and $\mathbf{com}^\diamond$). A protocol has to achieve its goals even when executions containing $\mathbf{com}^\diamond$ 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, the intruder process performs the abstract action $revealed$ when it gets access to a non-encrypted version of any DRM-protected content, to indicate violation of the secrecy requirement ($\mathbf{G2}$). This action is of course not triggered when the intruder merely renders an item using its trusted device, which is a normal behaviour in the system.

In μCRL specification 2, $Agent$ represents the set of all honest participants of the protocol and Msg represents the set of all messages. X is the intruder's knowledge set. Y contains messages buffered for delivery. The set operators $\cup$ and $\setminus$ have their usual meanings. The set $synth(X)$ represents the (infinite) set of messages that the intruder is able to synthesise from the messages in set X, e.g. by applying pairing, signing and so on. For a complete description of this model please refer to [9].

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 formulae that we have verified are in this fragment. A short account of this logic is presented below.

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

[9] 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].

μCRL specification 2 Intruder model

$$\begin{aligned}
I(X,Y) = \quad & \sum_{\substack{p\in Agent\\ m\in Msg}} \mathbf{recv}(p,m,I).I(X\cup\{m\}, Y\cup\{m\}) + \\
& \sum_{\substack{p\in Agent\\ m\in Msg}} \mathbf{send}(I,m,p).I(X, Y\setminus\{m\}) \lhd m\in Y \rhd \delta + \\
& \sum_{\substack{p\in Agent\\ m\in Msg}} \mathbf{send}^{\diamond}(I,m,p).I(X,Y) \lhd m\in \mathit{synth}(X)\setminus Y \rhd \delta + \\
& \sum_{m\in Cont} \mathit{revealed}(m).\delta \lhd m\in \mathit{synth}(X) \rhd \delta
\end{aligned}$$

$$\begin{array}{rcl}
\alpha & ::= & a\in\mathcal{A} \mid \neg\alpha \mid \alpha_1\wedge\alpha_2 \mid \alpha_1\vee\alpha_2 \\
\beta & ::= & \alpha \mid \beta_1.\beta_2 \mid \beta^* \\
\varphi & ::= & F \mid T \mid \varphi_1\vee\varphi_2 \mid \varphi_1\wedge\varphi_2 \mid \langle\beta\rangle\varphi \mid [\beta]\varphi \mid \mu X.\varphi
\end{array}$$

Table 1. (Partial) syntax of regular μ-calculus, adapted from [28]

Regular μ-calculus consists of *regular formulae* and *state formulae*. Regular formulae, describing sets of traces, are built upon *action formulae* and the standard regular expression operators. We use '.', '$\vee$', '$\neg$' and '$*$' for concatenation, choice, complement and transitive-reflexive closure, respectively, of regular formulae. The syntax of regular formulae as used in the next sections[10] is ranged over by β in Table 1. Variable a ranges over primitive actions from the set $\mathcal{A}$, such as $\mathbf{send}$ and $\mathbf{recv}$. In addition to this, the following notation is used: F denotes no action ($F = a \wedge \neg a$), T denotes any action ($T = \neg F$), and the wild-card action parameter $-$ represents any parameter of an action (e.g. $\mathbf{com}(-,-,-)$ represents any communication action).

State formulae, which express properties of states, are constructed from propositional variables, standard Boolean operators, the possibility modal operator $\langle\cdots\rangle$ (used here in the form $\langle\beta\rangle\mathsf{T}$ to express the existence of an execution of the protocol for which the regular formula β holds), the necessity modal operator $[\cdots]$ (used here in the form $[\beta]\mathsf{F}$ to express that, for all executions of the protocol, the regular formula β does not hold) and the minimal and maximal fixed point operators μ and ν. A state satisfies $\mu X.\ G$ iff it belongs to the minimal solution of the fixed point equation $X = G(X)$, G being a state formula and X a set of states. The symbols F and T are also used in state formulae. In state formulae they denote the empty set and the entire state space, respectively. The syntax of state formulae as used in the next sections is ranged over by φ in Table 1.

5.2.1. Formalisation of Nuovo's goals

This section describes the requirements stated by Nuovo DRM's goals G1-G3 in the regular μ-calculus. Below, the intent of these goals is captured formally. This serves to familiarise the reader with the core

[10] Table 1 restricts itself to the portion of the syntax used in this document. The formal syntax is fully described in [28].

formal expressions used in Section 5.3, where the goals are fully formalised. Given the complex nature of fairness (goal G4), a concise formalisation here is omitted in lieu of a full explanation in Section 5.3.

G1 (effectiveness). Effectiveness means that each purchase request is inevitably responded to, and each received item is preceded by its payment. The first requirement is encoded by stating that after a request occurs in a trace, a matching $update$ action must eventually follow. For a device $d1$ requesting content m from provider P with rights r, this is formalised as follows:

$$[\mathsf{T}^*.request(d1, m, r, P)]\ \mu X.(\langle\mathsf{T}\rangle\mathsf{T}\ \wedge\ [\neg update(d1, m, r, P)]X)$$

The second requirement is encoded by stating that no $update$ action occurs without a preceding, matching $paid$ action. For a device $d1$ updating with content m and rights r received from provider P, this is formalised as

$$[(\neg paid(P, m, r, d1))^*.update(d1, m, r, P)]\mathsf{F}$$

G2 (secrecy). G2, and the following goals, are checked in presence of an intruder. Secrecy is achieved when the intruder never acquires unprotected content. As mentioned in Section 5, the abstract action $revealed(m)$ denotes the intruder learning unprotected content m. Hence, this action should never occur. This is formalised as

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

G3 (resisting content masquerading). Content masquerading occurs when a device accepts a bundle of content and rights it did not request. Non-occurrence of this situation is formalised for device $d1$ and content m, rights r from P as

$$[(\neg request(d1, m, r, P))^*.update(d1, m, r, P)]\mathsf{F}$$

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, evaluating specific use cases in specific settings (as detailed below). Following [14], we assume perfect cryptography and do not consider attacks resulting from weaknesses of the cryptographic primitives used in protocols. Type-flaw attacks[11] are also omitted from our analysis. These can, in any case, be easily prevented [23].

Our formal analysis consists of two scenarios. A scenario explicitly describes one specific execution of the protocol, with specific actors, specific assumptions and a specific intruder model. The first scenario verifies effectiveness (G1) while using the synchronous communication model of Section 5.1, in absence of an intruder. The second scenario uses the asynchronous communication model of Section 5.1 and the

[11] 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.

modified DY intruder model of Section 5.1.4 to verify the remaining properties (G2-G4). Both scenarios operate under the assumptions of Section 4.2. Both scenarios describe a setting with two compliant devices $d1$ and $d2$, three different pieces of content and two different rights, the first right allowing to resell the second. In the second scenario, these are controlled (but not tampered) by the intruder of Section 5.1. Below, P, as before, represents the trusted content provider. The formulae 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 formalisation of these actions can be found in [24].

5.3.1. Honest scenario S_0

Scenario S_0 describes the interaction between 1 provider and 2 devices ($d1$ and $d2$). The communication network is assumed operational and no malicious agent is present. The scenario runs as follows: device $d1$ is ordered to buy an item and reselling rights from P. Then, $d1$ resells the purchased item to $d2$. As this scenario is only intended as a sanity check for the protocol, we believe correct behaviour in this scenario with two devices running multiple instances of the protocol is strong supporting evidence for that in general. For that reason, as well as the increased computational load of having more devices, we limited this scenario to only two devices. The scenario was checked using the EVALUATOR 3.0 model checker from the CADP tool set, 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:

1. Each purchase request is inevitably responded.

$$\begin{array}{ll} \forall m \in \mathit{Cont},\ r \in \mathit{Rgts}. & [\mathsf{T}^*.\mathit{request}(d1, m, r, P)]\ \mu X.(\langle\mathsf{T}\rangle\mathsf{T}\ \wedge\ [\neg \mathit{update}(d1, m, r, P)]X) \quad \wedge \\ & [\mathsf{T}^*.\mathit{request}(d2, m, r, d1)]\ \mu X.(\langle\mathsf{T}\rangle\mathsf{T}\ \wedge\ [\neg \mathit{update}(d2, m, r, d1)]X) \end{array}$$

2. Each received item is preceded by its payment.

$$\begin{array}{ll} \forall m \in \mathit{Cont},\ r \in \mathit{Rgts}. & [(\neg \mathit{paid}(P, m, r, d1))^*.\mathit{update}(d1, m, r, P)]\mathsf{F} \quad \wedge \\ & [(\neg \mathit{paid}(d1, m, r, d2))^*.\mathit{update}(d2, m, r, d1)]\mathsf{F} \end{array}$$

5.3.2. Dishonest scenario S_1

Scenario S_1 describes the interaction of an intruder I, 2 compliant devices $d1$ and $d2$, and 3 providers. The intruder controls the communication network and is the owner of devices $d1$ and $d2$. The intruder can instruct the compliant devices to purchase items and rights from the provider P, exchange items between themselves and resolve a pending transaction. Moreover, the compliant device $d1$ 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 $d1$ and $d2$. Although this is again a limited setting, the intruder capabilities are as strong as in a setting with more devices. Adding more devices would increase the number of honest users, not increase the capabilities of the intruder. This reasoning coupled with the computational power necessary to handle the generation of the state space directed us to limit the scenario thusly. The resulting model was checked using the EVALUATOR 3.0 model checker from the CADP tool set 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).

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

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

$$\begin{aligned}\forall a \in \{\mathit{d1}, \mathit{d2}\},\ m \in \mathit{Cont},\ r \in \mathit{Rgts}.\quad & [(\neg \mathit{request}(\mathit{d1}, m, r, \mathit{d2}))^*.\mathit{update}(\mathit{d1}, m, r, \mathit{d2})]\mathsf{F} \wedge \\ & [(\neg \mathit{request}(\mathit{d2}, m, r, \mathit{d1}))^*.\mathit{update}(\mathit{d2}, m, r, \mathit{d1})]\mathsf{F} \wedge \\ & [(\neg \mathit{request}(a, m, r, P))^*.\mathit{update}(a, m, r, P)]\mathsf{F}.\end{aligned}$$

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

$$\begin{aligned}\forall a \in \{\mathit{d1}, \mathit{d2}\},\ r \in \mathit{Rgts}.\quad & [\mathsf{T}^*.\mathit{update}(\mathit{d1}, m_0, r, \mathit{d2})]\mathsf{F} \wedge \\ & [\mathsf{T}^*.\mathit{update}(\mathit{d2}, m_0, r, \mathit{d1})]\mathsf{F} \wedge \\ & [\mathsf{T}^*.\mathit{update}(a, m_0, r, P)]\mathsf{F}.\end{aligned}$$

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.

$$\begin{aligned}\forall a \in \{\mathit{d1}, \mathit{d2}\},\ m \in \mathit{Cont},\ r \in \mathit{Rgts}.\quad & [(\neg \mathit{paid}(P, m, r, a))^*.\mathit{update}(a, m, r, P)]\mathsf{F} \\ & \wedge \\ & [\mathsf{T}^*.\mathit{update}(a, m, r, P).(\neg \mathit{paid}(P, m, r, a))^*. \\ & \ \ \mathit{update}(a, m, r, P)]\mathsf{F}\end{aligned}$$

Result 5. Nuovo DRM provides strong fairness in S_1 for $\mathit{d2}$, as formalised below[12]:

1. As a customer: if a compliant device pays (a provider or reseller device) for a content, it will eventually receive it[13].

   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, among other ways, by putting time limits on transactions with TTPs. Since we abstract away timing aspects here, instead, the action $\mathit{last}_{\mathit{ttp}}$ 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.

[12] Strong fairness for $\mathit{d1}$ is not guaranteed here, as it can abort the protocol prematurely. A protocol guarantees security only for the participants that follow the protocol.

[13] The fairness constraint used in the formulae corresponds to the strong notion of fairness in [19]: $\forall \theta.\ F^{\infty}\mathit{enabled}(\theta) \Rightarrow F^{\infty}\mathit{executed}(\theta)$.

$$\begin{array}{ll}
\forall m \in \mathit{Cont},\ r \in \mathit{Rgts}. & [\mathsf{T}^*.\mathit{request}(d2, m, r, P).(\neg(\mathit{update}(d2, m, r, P)))^*] \\
& \langle(\neg \mathit{com}^{\diamond}(-, -, -))^*.(\mathit{update}(d2, m, r, P))\rangle\mathsf{T} \\
& \wedge \\
\forall m \in \mathit{Cont},\ r \in \mathit{Rgts}. & [\mathsf{T}^*.\mathit{request}(d2, m, r, d1).(\neg(\mathit{resolves}(d2) \vee \mathit{update}(d2, m, r, d1)))^*] \\
& \langle(\neg \mathit{com}^{\diamond}(-, -, -))^*.(\mathit{resolves}(d2) \vee \mathit{update}(d2, m, r, d1))\rangle\mathsf{T} \\
& \wedge \\
& [(\neg \mathit{last}_{\mathit{ttp}})^*.\mathit{request}(d2, m, r, d1).(\neg \mathit{last}_{\mathit{ttp}})^*.\mathit{resolves}(d2). \\
& (\neg(\mathit{update}(d2, m, r, P) \vee \mathit{last}_{\mathit{ttp}}))^*] \\
& \langle(\neg \mathit{com}^{\diamond}(-, -, -))^*.\mathit{update}(d2, m, r, P)\rangle\mathsf{T}
\end{array}$$

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.

$$\begin{array}{ll}
\forall m \in \mathit{Cont},\ r \in \mathit{Rgts}. & [(\neg \mathit{paid}(d2, m, r, d1))^*.\mathit{update}(d1, m, r, d2)]\mathsf{F} \\
& \wedge \\
& [\mathsf{T}^*.\mathit{update}(d1, m, r, d2).(\neg \mathit{paid}(d2, m, r, d1))^*.\mathit{update}(d1, m, r, d2)]\mathsf{F}
\end{array}$$

Note that the strong fairness notion that is formalised and checked here subsumes the timeliness property of goal G4, simply because when $d2$ starts the resolve protocol, which it can autonomously do, it always recovers to a fair state without any help from $d1$.

Lemma 5.1. Nuovo DRM achieves its design goals G1-G4 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.

□

Note that Lemma 5.1 does not prove that Nuovo DRM achieves its design goals in *all* possible scenarios. It does support and provide credence for this statement.

6. Conclusions & future work

We have formally analysed the NPGCT DRM scheme and found two vulnerabilities in its protocols. The scheme was subsequently extended to address these vulnerabilities, and provide procedures for detection and revocation of compromised devices.

The extended scheme, Nuovo DRM, is inherently complicated (as many other DRM schemes are) 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 analysed and validated our design goals on a finite model of Nuovo DRM. This is of course no silver bullet: our formal verification is not complete as it abstracts away many details of the system.

To support a practical implementation of Nuovo DRM, the possibility of compromised devices has to be taken into account. To this end, procedures for both detection and revocation of compromised devices are introduced by Nuovo DRM. The distribution of revocation lists was discussed, and several alternative distribution models were compared.

As future work, we are considering several extensions to the formal analysis. For example, the accountability of the provider, which is taken as non-disputable in this study, can be verified. Additionally, possible privacy concerns of customers and the payment phase can be incorporated into the formal model.

The comparison of the various distribution models for revocation lists (especially the effectiveness of said models) is strongly influenced by the assumed uniformity of the network of connected devices. As future work, we intend to investigate the notion of effectiveness in a less uniform setting.

Acknowledgements

We are grateful to Bruno Crispo, Wan Fokkink, and Sjouke Mauw for their comments on earlier versions of this paper, to Saša Radomirović for his input on revocation lists, to Bert Lisser for his help with distributed state space generation and to the anonymous reviewers for their constructive and elaborate comments.

References

[1] Abadi, M., Blanchet, B.: Computer-Assisted Verification of a Protocol for Certified Email, *SAS '03*, 2694, 2003.

[2] Alpern, B., Schneider, F.: *Defining Liveness*, Technical Report TR 85-650, Dept. of Computer Science, Cornell University, Ithaca, NY, October 1984.

[3] Asokan, N.: *Fairness in electronic commerce*, PhD thesis, Univ. Waterloo, 1998.

[4] Avoine, G., Gärtner, F., Guerraoui, R., Vukolic, M.: Gracefully Degrading Fair Exchange with Security Modules., *EDCC '05*, 3463, Springer, 2005.

[5] Basu, A., Charron-Bost, B., Toueg, S.: Simulating Reliable Links with Unreliable Links in the Presence of Process Crashes, *ACM WDAG '96*, 1151, Springer, 1996, ISBN 3-540-61769-8.

[6] Bella, G., Paulson, L. C.: Mechanical Proofs about a Non-repudiation Protocol, *TPHOL'01*, 2152, Springer, 2001.

[7] Blom, S., Calamé, J., Lisser, B., Orzan, S., Pang, J., van de Pol, J. C., Torabi Dashti, M., Wijs, A.: Distributed Analysis with μCRL, *TACAS '07*, 4424, Springer, 2007.

[8] Blom, S., Fokkink, W., Groote, J. F., van Langevelde, I., Lisser, B., van de Pol, J. C.: μCRL: A Toolset for Analysing Algebraic Specifications, *CAV'01*, 2102, Springer, 2001, ISBN 3-540-42345-1.

[9] Cederquist, J., Torabi Dashti, M.: *An Intruder Model for Verifying Termination in Security Protocols*, Technical Report TR-CTIT-05-29, University of Twente, Enschede, The Netherlands, 2005.

[10] Cederquist, J., Torabi Dashti, M.: An Intruder Model for Verifying Liveness in Security Protocols, *FMSE '06*, ACM Press, 2006.

[11] Cervesato, I.: Data Access Specification and the Most Powerful Symbolic Attacker in MSR, *ISSS '02*, 2609, Springer, 2003.

[12] Clarke, E., Grumberg, O., Peled, D.: *Model Checking*, MIT Press, 2000.

[13] Comon, H., Shmatikov, V.: Is it possible to decide whether a cryptographic protocol is secure or not?, *J. of Telecomm. and Inform. Tech.*, **4**, 2002, 3–13.

[14] Dolev, D., Yao, A.: On the security of public key protocols, *IEEE Trans. on Information Theory*, **IT-29**(2), 1983, 198–208.

[15] Evans, N., Schneider, S.: Verifying security protocols with PVS: widening the rank function approach, *J. Logic and Algebraic Programming*, **64**(2), 2005, 253–284.

[16] Even, S., Yacobi, Y.: *Relations amoung public key signature systems*, Technical Report 175, Computer Science Department, Technicon, Haifa, Israel, 1980.

[17] Fernandez, J.-C., Garavel, H., Kerbrat, A., Mateescu, R., Mounier, L., Sighireanu, M.: CADP: A Protocol Validation and Verification Toolbox, *CAV '98*, 1102, Springer, 1996.

[18] Fischer, M., Lynch, N., Paterson, M.: Impossibility of distributed consensus with one faulty process, *J. ACM*, **32**(2), 1985, 374–382.

[19] Francez, N.: *Fairness*, Springer, 1986.

[20] Groote, J. F., Ponse, A.: The syntax and semantics of μCRL, *Algebra of Communicating Processes '94*, Workshops in Computing Series, Springer, 1995.

[21] Gürgens, S., Rudolph, C., Vogt, H.: On the Security of Fair Non-repudiation Protocols, *ISC '03*, 2851, 2003.

[22] Halderman, J., Felten, E.: Lessons from the Sony CD DRM Episode, *the 15th USENIX Security Symposium*, 2006.

[23] Heather, J., Lowe, G., Schneider, S.: How to Prevent Type Flaw Attacks on Security Protocols, *CSFW '00*, IEEE CS, 2000.

[24] Jonker, H. L., Krishnan Nair, S., Torabi Dashti, M.: *Nuovo DRM Paradiso*, Technical Report SEN-R0602, CWI, Amsterdam, The Netherlands, 2006, `http://ftp.cwi.nl/CWIreports/SEN/SEN-R0602.pdf`.

[25] Kähler, D., Küsters, R.: Constraint Solving for Contract-Signing Protocols., *CONCUR '05*, 3653, Springer, 2005.

[26] Kremer, S., Markowitch, O., Zhou, J.: An Intensive Survey of Non-repudiation protocols, *Computer Communications*, **25**(17), 2002, 1606–1621.

[27] Kremer, S., Raskin, J.: A Game-Based Verification of Non-repudiation and Fair Exchange Protocols, *CONCUR'01*, 2154, Springer, 2001.

[28] Mateescu, R., Sighireanu, M.: Efficient on-the-fly model-checking for regular alternation-free μ-calculus, *Sci. Comput. Program.*, **46**(3), 2003, 255–281, ISSN 0167-6423.

[29] Meadows, C.: Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends, *IEEE J. Selected Areas in Communication*, **21**(1), 2003, 44–54.

[30] Nair, S., Popescu, B., Gamage, C., Crispo, B., Tanenbaum, A.: Enabling DRM-preserving digital content redistribution, *7th IEEE Conf. E-Commerce Technology*, IEEE CS, 2005.

[31] Pagnia, H., Vogt, H., Gärtner, F.: Fair Exchange., *Comput. J.*, **46**(1), 2003, 55–75.

[32] Pnueli, A., Xu, J., Zuck, L. D.: Liveness with $(0, 1, \infty)$-Counter Abstraction, *CAV*, 2002.

[33] van de Pol, J. C., Valero Espada, M.: Modal Abstractions in μCRL, *AMAST*, 2004.

[34] Pucella, R., Weissman, V.: A logic for reasoning about digital rights, *CSFW '02*, IEEE CS, 2002, ISBN 0-7695-1689-0.

[35] Shmatikov, V., Mitchell, J.: Finite-state analysis of two contract signing protocols, *Theor. Comput. Sci.*, **283**(2), 2002, 419–450, ISSN 0304-3975.