Reputation-Based Trust Management (extended abstract)

Vitaly Shmatikov and Carolyn Talcott

Computer Science Laboratory, SRI International, Menlo Park, CA 94025 USA
{shmat,clt}@csl.sri.com

Abstract. We propose a formal model for reputation-based trust management. In contrast to conventional, credential-based trust management, in reputation-based trust management an agent's past behavior is used to decide whether to grant him access to a protected resource. We use an event semantics inspired by the actor model, and assume that each principal has only partial knowledge of the events that have occurred. Licenses are used to formalize restrictions on agents' behavior. If an agent fulfills the obligations imposed by a license and does not misuse the license by performing a forbidden action, then his reputation improves. This approach enables precise formal modeling of scenarios involving reputations, such as financial transactions based on credit histories and information sharing between untrusted agents.

1 Introduction

Reputation is a fundamental concept in many situations that involve interaction between mutually distrusting parties. Before issuing a credit card, the bank usually checks the applicant's credit history, which includes independently certified evidence that the applicant has fulfilled his prior financial obligations. On eBay, the seller's reputation, *i.e.*, the evidence that past buyers were satisfied with this seller's behavior, is considered an asset of great value. Many solutions for the "free-rider problem" [AH00] in peer-to-peer file sharing networks (the problem of users downloading a large number of files without contributing anything in return) rely on the evidence of past contributions when granting access to popular files [GJM02].

We propose a formal model that gives a precise semantics to the notion of reputation and uses it in reasoning about trust. Our approach extends an event-based semantics inspired by the actor model of distributed computation [BH77,Cli81] to incorporate *incomplete knowledge*. Resource owners are assumed to have only partial knowledge of the event history. When deciding whether whether to trust a particular agent or not, they rely on the evidence of his past behavior supplied by trusted agents. By contrast, trust in conventional trust management [BFL96] is based on access control credentials.

Inspired by license-based digital rights languages [GWW01,PW02], we use licenses to formalize both "good" and "bad" behavior. Each license restricts the behavior of the agent who accepts it by specifying the agent's obligations (what the agent *must* do) as well as the forbidden actions (what the agent *must not* do). If an agent fulfills all obligations associated with the license and avoids any of the forbidden actions, the

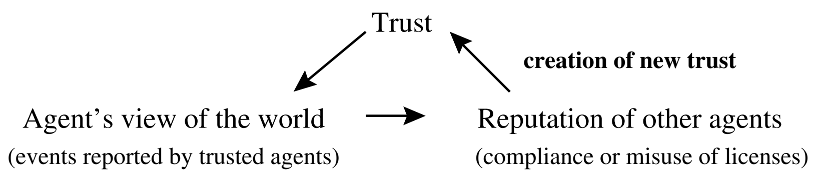

Fig. 1. Reputation-based trust management

license issuer creates a signed statement that the agent has complied with the license. Another resource owner may decide to grant a new license on the basis of this evidence, even though he has not personally observed the licensee's good behavior.

To illustrate by example, consider a consumer applying for an auto loan. The lender requests the consumer's credit history from a credit reporting bureau, and uses the information to decide whether to grant the loan and at what terms. The lender's trust in the consumer is not based on the consumer's credentials and, in contrast to conventional trust management, the credit bureau is not vouching for the consumer's creditworthiness (*i.e.*, there is no delegation of trust). The bank trusts the credit bureau to accurately report the *evidence*, *i.e.*, a summary of past events, such as the fact that the consumer had signed up for a credit card and fulfilled his obligations by making timely payments.

We provide sample rules and policies that an agent may use for building reputation-based trust. Our approach is summarized in figure 1. The proposed framework does not currently support delegation, but we expect that it will be a fairly simple extension of the basic approach. We also present two case studies, using our framework to formalize reputation-based trust in an anonymized peer-to-peer file distribution system and a multi-player game scenario, respectively. The former demonstrates how reputation is created from the evidence of license fulfillment, while the latter illustrates the use of licenses in an environment with multiple untrusted agents.

2 Model

The following list of *sorts* represents the fundamental concepts of our framework:

$\mathbf{P}$ Principals $\mathbf{E}$ Events $\mathbf{A}$ Actions $\mathbf{R}$ Resources
$\mathbf{L}$ Licenses $\mathbf{H}$ Histories $\mathbf{S}$ Statements $\mathbf{J}$ Justifications (proofs)

2.1 Principals

Principals or *agents* are individual entities (people, organizations, software agents) operating independently of each other. A principal uses his own observations and statements of those he trusts to construct a partial view of the event history and assign reputations to other principals based on the information contained in this view.

2.2 Actions and events

An *action* is an atomic interaction between two principals or a principal and a resource. For example, action use : $\mathbf{P} \times \mathbf{R} \times \mathbf{L} \to \mathbf{A}$ models a principal using a resource under a license. Action $\mathsf{says} : \mathbf{P} \times \mathbf{S}[\times \mathbf{J}] \to \mathbf{A}$ models issuing a signed statement. For each action, $\mathsf{actor} : \mathbf{A} \to \mathbf{P}$ specifies the principal who is performing the action. Function $\mathsf{observers} : \mathbf{A} \to \mathbf{2^P}$ specifies the principals who are the immediate observers of the action and thus know that the corresponding event has occurred.

An *event* is an action occurrence. Function $\mathsf{act} : \mathbf{E} \to \mathbf{A}$ maps events to associated actions. A *history* $h = \langle E_h, <_h \rangle$ is a partially ordered set of events $E_h \subset \mathbf{E}$ labelled by actions, where $<_h$ is a discrete partial order. We say that $h < h'$ if $E_h \subseteq E'_h$ and the partial order of h is the restriction of that of h'. We write h/e for the rollback of a history to the events preceding e, *i.e.*, $h/e = poset\{e' \in h \mid e' <_h e\}$.

Axioms $\mathsf{trusts}(p, p')$ model trust relationships that are used in constructing each principal's partial view of the event history. Let $\mathsf{occur}(e) : \mathbf{E} \to \mathtt{bool}$ hold *iff* e has occurred. We define a principal's *view* of a history h as the projection $\mathsf{view} : \mathbf{H} \times \mathbf{P} \to \mathbf{H}$ of h onto events that he has directly observed, or learned about from those he trusts:

$$\begin{aligned}\mathsf{view}(h,p) \stackrel{def}{=} \; & poset\{e \in h \mid p \in \mathsf{observers}(\mathsf{act}(e)) \vee \\ & (\exists p')(\mathsf{trusts}(p,p') \wedge \exists\, e' \in h \text{ s.t. } \mathsf{act}(e') = \mathsf{says}(p', \mathsf{occur}(e)))\}\end{aligned}$$

This assumes that a principal observes all signed statements issued by his friends, *e.g.*, via broadcast or a central repository of signed statements.

2.3 Resources

A *resource* is an item of value (*e.g.*, program, website, database, credential) that principals may wish to access or use. A resource has an $\mathsf{owner} : \mathbf{R} \to \mathbf{P} + *$, where $*$ is the special principal indicating that no one owns the resource, and $\mathsf{status} : \mathbf{R} \to \{\mathtt{public}, \mathtt{licensed}, \mathtt{protected}\}$. For simplicity, owner and status do not depend on history. We envision a future extension of the framework in which the owner and/or status of a resource may change over time, and owner and status have a history argument.

The owner specifies how the resource may be used by defining the predicate $\mathsf{useOk} : \mathbf{H} \times \mathbf{R} \times \mathbf{P} \times \mathbf{L} \to \mathtt{bool}$. Here $\mathsf{useOk}(h, r, p, l)$ is interpreted as "given event history h, principal p is allowed to use the resource r on the basis of license l."

If $\mathsf{status}(r) = \mathtt{public}$, then there are no restrictions on the use of the resource: $\forall p, l\ \mathsf{useOk}(h, r, p, l)$. If $\mathsf{status}(r) = \mathtt{licensed}$ and $\neg\mathsf{useOk}(h, r, p, l)$, then $\mathsf{use}(p, r, l)$ is possible, but constitutes a misuse of the license (see section 3). If $\mathsf{status}(r) = \mathtt{protected}$ and $\neg\mathsf{useOk}(h, r, p, l)$, then $\mathsf{use}(p, r, l)$ is assumed to be impossible (*e.g.*, to model cryptographic protection). There are *two* kinds of misuse in our model: an action forbidden by the license, and an action permitted by the license but forbidden by the resource itself. The latter is only meaningful if the resource is $\mathtt{licensed}$. We do not treat a failed attempt to use a protected resource without a proper license as a misuse.

3 Licenses

We use a license language inspired by [GWW01,PW02] to define the permissible behavior of principals. Licenses are very important in our framework since compliance with past licenses is the basis of a principal's reputation. Each license defines i) obligations, and ii) permitted actions. A license has one each of $\mathsf{issuer} : \mathbf{L} \to \mathbf{P}$, $\mathsf{licensee} :$

$\mathbf{L} \to \mathbf{P}$ (the principal to whom obligations and permissions apply), and $\mathsf{subject} : \mathbf{L} \to \mathbf{R}$. We do not currently model license revocation.

There are several special actions associated with licenses: $\mathsf{offer}(l)$ models offering a license, $\mathsf{accept}(p, l)$ models principal p accepting license l. Special predicate $\mathsf{associated}(e, l) : \mathbf{E} \times \mathbf{L} \to \texttt{bool} \stackrel{def}{=} \mathsf{act}(e) = \mathsf{use}(p, r, l)$ for some p and r. Licenses are assumed to have a unique id, so that each license is offered and accepted at most once in a given history.

License l is valid in a history h if it has been offered and accepted:

$$\begin{aligned}\mathsf{validLicense}(h, l) \stackrel{def}{=}\ & \exists e_1, e_2 \in h \text{ s.t. } e_1 < e_2 \wedge \\ & \mathsf{act}(e_1) = \mathsf{offer}(l) \ \wedge\ \mathsf{act}(e_2) = \mathsf{accept}(\mathsf{licensee}(l), l)\end{aligned}$$

If $h < h'$, then $\mathsf{validLicense}(h, l) \Rightarrow \mathsf{validLicense}(h', l)$. We define $\mathsf{acceptance}(h, l)$ to be the first event of the form $\mathsf{accept}(\mathsf{licensee}(l), l)$ following the offer.

Define a projection of h onto events associated with some license as

$$\mathsf{licEvents}(h, l) \stackrel{def}{=} \mathit{poset}\{e \mid e \in h \ \wedge\ \mathsf{associated}(e, l) \ \wedge\ \mathsf{validLicense}(h, l)\}$$

Permissions and obligations. Licenses permit use of resources and entail obligations such as payments or issuance of future licenses. For example, a web hosting license may state that, if the server has accepted the license and the history contains an event in which some user paid $100 to the server, the latter is obligated to issue a license permitting the user to use the service for a month.

Given a history, the set of actions permitted by a license is specified via the predicate $\mathsf{permits} : \mathbf{H} \times \mathbf{L} \times \mathbf{E} \to \texttt{bool}$, defined by the license issuer. When a license is used to access a resource, the access policy, as defined by useOk, may check permits associated with the license (as in the example of section 5), or else permits may invoke the resource's useOk to check whether a particular use is permitted or not (as in section 6).

The license issuer may impose obligations on the licensee by defining the predicate $\mathsf{violates} : \mathbf{H} \times \mathbf{L} \to \texttt{bool}$. Given history h, $\mathsf{violates}(h, l)$ *iff* h does not contain fulfillment of every obligation imposed by the license. For example, if license *l* models taking a loan, $\mathsf{violates}(h, l)$ if h contains a timestamp event corresponding to the repayment deadline, but does not contain a preceding repayment event.

We say that a principal *fulfilled* the license if, up to date, he has performed all obligations specified by the license. If a principal has fulfilled the license up to date and there are no future obligations, we say that the license is *completely fulfilled*.

$$\begin{aligned}\mathsf{fulfill}(h, l) &\stackrel{def}{=} \mathsf{validLicense}(h, l) \ \wedge\ \neg\mathsf{violates}(h, l) \\ \mathsf{cFulfill}(h, l) &\stackrel{def}{=} \forall h' \geq h\ \mathsf{fulfill}(h', l)\end{aligned}$$

Permissions associated with a license are independent of the obligations. A license circumscribes the licensee's behavior "from above" (permits restricts the set of actions he *may* do) as well from "from below" (violates specifies what he *must* do). A license may be violated passively by *not* doing something (*e.g.*, not repaying a loan), or actively misused by doing something forbidden (*e.g.*, overdrawing a credit line), or both.

$$\begin{aligned}\mathsf{misuse}(h,l,e) &\stackrel{def}{=} \neg\mathsf{permits}(h,l,e)\ \vee \\ &\quad\ \mathsf{act}(e) = \mathsf{use}(p,r,l)\ \wedge\ \neg\mathsf{useOk}(h/e,r,p,l) \text{ for some } r,p \\ \mathsf{misused}(h,l) &\stackrel{def}{=} \exists e \in h \text{ s.t. } \mathsf{misuse}(h,l,e)\end{aligned}$$

The issuer is free to define any violates and permits he wishes. Our framework *per se* does not enforce any consistency checks on these predicates, so it's up to the licensee to decide whether the restrictions encoded in the license are acceptable.

4 Reputations

Fulfillment and misuse provide us with a semantics for "good" and "bad" behavior. We interpret *reputation* of a principal as a judgment about his or her behavior made by *another* principal. Since such judgments are based on the judge's partial view of the event history, new evidence may cause the judgment to change. In particular, passive violations (*e.g.*, absence of a promised payment) may be rectified by presenting the evidence (*e.g.*, signed bank statement) that the event fulfilling the obligation has occurred.

We call evidence of good behavior *credit*, and evidence of bad behavior *demerit*. There are two forms of credit: partial (all obligations to date have been fulfilled) and full (all obligations to date have been fulfilled and there will be no future obligations), and two forms of demerit: passive (absence of evidence that an obligation has been fulfilled; can be rectified by additional evidence) and active (evidence of misuse; usually cannot be rectified unless permits is nonmonotonic).

We wish to emphasize the distinction between *events* (observed behavior) and *judgments* (a principal's evaluation of another principal's behavior). We assume that each principal has an internal logic for reasoning about good and bad behavior. Given a particular event history h and license l, let $lh = \mathsf{licEvents}(\mathsf{view}(h,p))$ be the part of h that is associated with license l and known to principal p. For example, p may use the following internal reasoning rules to derive credit and demerit judgments:

$$\frac{\mathsf{fulfill}(lh,l)}{\mathsf{partialCredit}(\mathsf{licensee}(l),l)} \qquad \frac{\mathsf{cFulfill}(lh,l)}{\mathsf{fullCredit}(\mathsf{licensee}(l),l)}$$

$$\frac{\mathsf{violates}(lh,l)}{\mathsf{passiveDemerit}(\mathsf{licensee}(l),l)} \qquad \frac{\mathsf{misused}(lh,l)}{\mathsf{activeDemerit}(\mathsf{licensee}(l),l)}$$

Once a judgment is derived, the principal may announce it by performing a says action. This models issuing a signed statement containing the judgment, which then becomes part of its subject's reputation. Other principals may use this judgment in their own internal reasoning when deciding whether to offer a license. Our framework does not restrict how issuers' policies are formulated. Below is a sample policy in which principal p offers a license to p' if the reputation of p' includes evidence of at least m fulfilled licenses and no evidence of misuse. Given history h, we overload $\mathsf{says}(p'', \ldots)$

and use it as a shorthand for $\exists e \in \mathsf{view}(h, p)$ s.t. $\mathsf{act}(e) = \mathsf{says}(p'', \ldots)$.

$$\frac{\begin{array}{l}\mathsf{trusts}(p, p_1) \;\wedge\; \ldots \;\wedge\; \mathsf{trusts}(p, p_m) \;\wedge \\ \quad \textit{(where } p_1, \ldots, p_m \textit{ are not necessarily all distinct)} \\ \mathsf{says}(p_1, \mathsf{partialCredit}(p', l_1)) \;\wedge\; \ldots \;\wedge\; \mathsf{says}(p_m, \mathsf{partialCredit}(p', l_m)) \;\wedge \\ \quad \textit{(where } l_1, \ldots, l_m \textit{ are all distinct)} \\ \forall p'' \text{ s.t. } \mathsf{trusts}(p, p'')\; \forall l, l' \text{ s.t. } l' \notin \{l_1, \ldots, l_m\} \\ \neg\mathsf{says}(p'', \mathsf{activeDemerit}(p', l)) \;\wedge\; \neg\mathsf{says}(p'', \mathsf{passiveDemerit}(p', l'))\end{array}}{\mathsf{offer}(l) \text{ where } \mathsf{issuer}(l) = p, \mathsf{licensee}(l) = p'}$$

5 Example: Peer-to-Peer File Distribution System

In this case study, we use our framework to encode a reputation policy for a peer-to-peer file distribution system, roughly similar to Gnutella or Freenet [CSWH00], implemented on top of a network of anonymizers, *e.g.*, an onion routing network [SGR97].

Consider a single file server. It consists of two resources, `ul` (upload) and `dl` (download), where `ul` and `dl` are unique names.

$$\begin{array}{lll}\mathsf{status}(\mathtt{ul}) = \mathtt{public} \;\wedge\; \mathsf{status}(\mathtt{dl}) = \mathtt{licensed} \;\wedge & & \\ \mathsf{useOk}(h, \mathtt{dl}, p, l) & \Leftrightarrow & \mathsf{validLicense}(\mathsf{view}(h, \mathsf{owner}(\mathtt{dl})), l) \;\wedge \\ & & \mathsf{issuer}(l) = \mathsf{owner}(\mathtt{dl}) \;\wedge\; \mathsf{licensee}(l) = p \;\wedge \\ & & \mathsf{permits}(\mathsf{view}(h^*, \mathsf{owner}(\mathtt{dl})), l, e^*)\end{array}$$

where e^* is a fresh event s.t. $\mathsf{act}(e) = \mathsf{use}(p, \mathtt{dl}, l)$, h^* is h extended with e^*

Informally, these axioms state that anybody can upload (since `ul` is a public resource), but downloads, modeled as uses of the `dl` resource, are only permitted with a license issued by the server's owner. Moreover, useOk checks whether, given the current event history as known to the server owner, the use is permitted by the license.

A sample license for the download resource can be defined as follows:

$$\begin{array}{l}\text{let} \quad p = \mathsf{licensee}(l) \;\wedge \\ \quad nU = \mathit{count}\{e' \in h \text{ s.t. } \mathsf{act}(e') = \mathsf{use}(p, \mathtt{ul}, l)\} \;\wedge \\ \quad nD = \mathit{count}\{e' \in h \text{ s.t. } \mathsf{act}(e') = \mathsf{use}(p, \mathtt{dl}, l)\} \text{ in} \\ \mathsf{permits}(h, l, e) \;\Leftrightarrow\; \mathsf{act}(e) = \mathsf{use}(p, \mathtt{dl}, l) \;\wedge \\ \quad (nD < nU \times 3 \quad \vee \\ \quad \exists p', l' \;\mathsf{trusts}(\mathsf{issuer}(l), p') \;\wedge\; \mathsf{says}(p', \mathsf{partialCredit}(p, l')) \;\wedge\; l' \neq l) \\ \mathsf{violates}(h, l) \;\Leftrightarrow\; nD > nU \times 2\end{array}$$

By accepting this license, the licensee promises to upload at least once for every 2 downloads. If he fails to do this, however, he is not prevented from further downloads as long as the event history contains at least 1 upload for every 3 downloads. This means that permits allows some violating actions (*i.e.*, obligations accepted by the licensee may be left unfulfilled to a limited extent). For example, if 2 uploads and 4 downloads have occurred, the 5th download will be allowed ($5 < 2 \times 3$), even though it's a violation of the obligation ($5 > 2 \times 2$), but the 6th download will not be allowed.

Also, $\mathsf{permits}$ allows unlimited downloads if the licensee has a reputation from some principal p', trusted by the server owner, in the form of credit for compliance with *another* license. The licensee can thus gain access to `dl` in two ways: by maintaining the proper ratio of uploads to downloads, or relying on the previously acquired reputation.

Credit for compliance with the license is issued according to the following rule:

$$\frac{\begin{array}{c} count\{e \in \mathsf{view}(h, \mathsf{issuer}(l)) \text{ s.t. } \mathsf{act}(e) = \mathsf{use}(\mathsf{licensee}(l), \mathtt{ul}, l)\} > 5 \wedge \\ \mathsf{fulfill}(\mathsf{view}(h, \mathsf{issuer}(l)), l) \ \wedge \ \neg \mathsf{misused}(\mathsf{view}(h, \mathsf{issuer}(l)), l) \end{array}}{\mathsf{says}(\mathsf{issuer}(l), \mathsf{partialCredit}(\mathsf{licensee}(l), l))}$$

Even though a principal with an existing reputation from a trusted source is permitted unlimited uses of the `dl` resource, he cannot increase his reputation by doing so, *i.e.*, reputation cannot be "amplified" by using it repeatedly. Only if the licensee complies with the license the "hard" way, with 1 upload for every 2 downloads and at least 5 uploads, will the license be fulfilled, and the server owner issue a new credit judgment which the licensee may then use as part of his reputation to access some other resource.

We also model the scenario in which upload and download requests arrive to the file server via a chain of anonymizers. To this end, we introduce transitive trust rules for passing reputation back down the chain. Reputation is anonymized. When a user receives credit, it will not be possible to establish *what* was uploaded, *i.e.*, evidence of license fulfillment cannot be linked to any particular use of the `ul` resource.

We assume that each anonymizer has two secret internal tables, $\mathsf{nameTranslate}$ and $\mathsf{licenseTranslate}$. The $\mathsf{nameTranslate}$ table is set up when the anonymizer chain is initialized. It contains name translations between user pseudonyms so that the user can use different names on each link of the chain, preventing an outside observer from relating requests that arrive to and leave from each anonymizer. If the user is known on the outgoing link as p, then $\mathsf{nameTranslate}[p]$ returns the user's pseudonym on the incoming link. We also assume that all anonymizers trust each other.

$$\frac{\begin{array}{l} \exists p \ \mathsf{trusts}(\mathsf{anonymizer}, p) \ \wedge \ \mathsf{says}(p, \mathsf{partialCredit}(p_1, l)) \ \wedge \\ \nexists l' \text{ s.t. } \mathsf{licenseTranslate}[l] = l' \ \wedge \ \mathsf{nameTranslate}[p_1] = p_2 \end{array}}{\begin{array}{l} \text{let } l' = \text{fresh license } id \text{ in} \\ \quad \mathsf{licenseTranslate} \leftarrow \mathsf{licenseTranslate} \cup [l \to l'] \wedge \\ \quad \mathsf{says}(\mathsf{anonymizer}, \mathsf{partialCredit}(p_2, l')) \end{array}}$$

If the anonymizer trusts some principal p, and p has issued credit to a user known as p_1 for complying with license l, the anonymizer first checks that he has not already issued credit for the same license (*i.e.*, the $\mathsf{licenseTranslate}$ table does not already contain l). He then looks up a different pseudonym p_2, under which the user is known to the preceding anonymizer in the chain, creates a fresh license id (which contains no details about the actual license for which the user received credit), and issues a credit judgment to the (pseudonymous) user for complying with the newly created license.

6 Example: Untrusted Allies

We model an online role-playing game (inspired by Clan Lord [Cla]), in which characters belong to clans that are competing in a search for valuable items. One clan can impede another by setting traps. Maps of regions that must be traversed help make the search safer and faster. A player may also be an independent agent. A clan leader wants to avoid traps and might use a strategy of trading information to discover where traps have been placed and sending scout groups to disable them. When a trap is found and successfully disabled, the scout group leader reports this to the clan leader. An independent agent wants map information to aid his own search or to trade. The agent may discover traps or learn about them by hanging out with other clans. How can the clan leader and an independent agent build trust in order to interact for mutual benefit?

We abstract access to map data to a simple use action. The clan leader issues single-use licenses for the clan's map in exchange for confirmed good information about traps. Accepting the license *obliges* the agent to provide information whether or not he accesses the map. Confirmation is in the form of the scout group leader saying that good information was received from the licensee, that is, the trap was found and disabled. The clan leader trusts the scout group leader to report receipt of good information, and for simplicity we omit consideration of bad information.

We use the following notation:

(`ClanLeader`)—the clan leader
(`IA`)—the independent agent player
(`Scout`)—the scout group leader
(`Map`)—the clan's map

We let h range over histories, and ML over single-use licenses for `IA`'s access to `Map`, and posit the following axioms:

$$\begin{array}{l}\mathsf{owner}(h, \mathtt{Map}) = \mathtt{ClanLeader} \wedge \mathsf{trusts}(\mathtt{ClanLeader}, \mathtt{Scout}) \wedge \\ \mathsf{issuer}(ML) = \mathtt{ClanLeader} \wedge \mathsf{subject}(ML) = \mathtt{Map} \wedge \mathsf{licensee}(ML) = \mathtt{IA}\end{array}$$

A license may expire after it has been accepted. We write $\mathsf{expiredLicense}(h, l)$ to represent the fact that license l has expired in history h. A single-use license expires after one use. It may also timeout. We let the action $\mathsf{expires} : \mathbf{L} \to \mathbf{A}$ be the underlying action of a license timeout event. Then we have

$$\begin{array}{l}\mathsf{expiredLicense}(h, ML) \;\Leftrightarrow\; \mathsf{validLicense}(h, ML) \wedge \\ \quad (\exists e \in h)(\mathsf{act}(e) = \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \vee \mathsf{act}(e) = \mathsf{expires}(ML))\end{array}$$

Note that if $h < h'$, then $\mathsf{expiredLicense}(h, ML) \Rightarrow \mathsf{expiredLicense}(h', ML)$.

For clan map licenses, we define useOk and $\mathsf{permits}$ as follows:

$$\begin{array}{rcl}\mathsf{useOk}(h, \mathtt{Map}, p, ML) & \Leftrightarrow & \mathsf{validLicense}(h, ML) \wedge \neg(\mathsf{expiredLicense}(h, ML)) \wedge \\ & & p = \mathsf{licensee}(ML) \wedge \\ & & (\exists e \in h)(\mathsf{act}(e) = \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML))) \\ \mathsf{permits}(h, ML, e) & \Leftrightarrow & \mathsf{act}(e) = \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \wedge \mathsf{useOk}(h/e, \mathtt{Map}, \mathtt{IA}, ML)\end{array}$$

Recall that if $\mathsf{validLicense}(h, l)$, then $\mathsf{acceptance}(h, l)$ is the (first) event in h following the offer of l with action $\mathsf{accept}(\mathsf{licensee}(l), l)$. The violates property is then defined as follows:

$$\begin{aligned}
&\mathsf{violates}(h, ML) \;\Leftrightarrow\; \mathsf{validLicense}(h, ML) \wedge \\
&\quad (\mathsf{expiredLicense}(h, ML) \;\wedge\; \neg(\exists e \in h)(\mathsf{act}(e) = \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML)))) \\
&\quad \vee \\
&\quad (\exists e_u \in h)(\mathsf{acceptance}(h, ML) < e_u \;\wedge\; \mathsf{act}(e_u) = \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \wedge \\
&\qquad\qquad \neg(\exists e \in h)(e <_h e_u \;\wedge\; \mathsf{act}(e) = \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML))))
\end{aligned}$$

The above definitions allow information to be provided before the license is accepted, or even offered. A use of ML out of scope (*i.e.*, before the acceptance event for the license) is not a violation, since it is not associated with a valid license. It is, however, a misuse.

We illustrate the definitions by presenting some simple histories and their properties. We consider several cases, depending on whether the map is licensed ($\mathsf{status}(\mathtt{Map}) = \mathtt{licensed}$) or protected ($\mathsf{status}(\mathtt{Map}) = \mathtt{protected}$). In each example we consider the history prior to and/or including expiration. We write h^X for a history extended by the $\mathsf{expires}(ML)$ event.

$$\begin{aligned}
\mathtt{h}_1 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) \,\} \\
\mathtt{h}_2 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) < \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML)) \,\} \\
\mathtt{h}_3 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) < \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML)) \\
&\qquad < \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \,\} \\
\mathtt{h}_4 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) < \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML)) \\
&\qquad < \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) < \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \,\} \\
\mathtt{h}_5 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) < \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \,\} \\
\mathtt{h}_6 &= \{\, \mathsf{offer}(ML) < \mathsf{accept}(\mathtt{IA}, ML) < \mathsf{use}(\mathtt{IA}, \mathtt{Map}, ML) \\
&\qquad < \mathsf{says}(\mathtt{Scout}, \mathsf{goodInfo}(ML)) \,\}
\end{aligned}$$

$\mathtt{h}_4$, $\mathtt{h}_5$ and $\mathtt{h}_6$ are only possible if `Map` is not `protected`. $\mathsf{useOk}(h, \mathtt{Map}, \mathtt{IA}, ML)$ is `true` only for $h = \mathtt{h}_2$ and `false` otherwise. $\mathsf{violates}(h, ML)$ is `true` for $\mathtt{h}_1^X$, $\mathtt{h}_4$, $\mathtt{h}_4^X$, $\mathtt{h}_5$, $\mathtt{h}_5^X$, $\mathtt{h}_6$, and $\mathtt{h}_6^X$, and `false` otherwise. The second use event of $\mathtt{h}_4$ and the use events of $\mathtt{h}_5$, and $\mathtt{h}_6$ are misuses according to the policy defined by useOk.

For simplicity, the above scenario focuses on a setting with one clan leader, one scout group, and one independent agent. The clan map license model can be used as a starting point for modeling how the clan leader and the independent agent might build mutual [dis]trust and use this reputation-based trust to develop simple strategies for deciding when to trade information. For example, since the clan leader trusts the scout group leader to reliably report when good/bad information has been given, such reports could be used to give credit/demerits. Furthermore, if the agent violates a license agreement, the clan leader gives the agent some demerits. Conversely, the agent gives credit to the clan leader if the map data access received as part of a license agreement is accurate, and gives demerit if the clan leader fails to hold up his end of the bargain. (Examples of trust building rules are given in section 5 for a different scenario.)

Extending the axiomatization of clan map licenses to multiple clans, scout groups, and independent agents is straightforward. Pairwise trust building rules could then be extended by rules that take into account multi-party interactions, as well as relative trustworthiness of different agents.

7 Conclusion

We have presented a formal model for reputation-based trust management that allows mutually distrusting agents to develop a basis for interaction even in the absence of a central credential authority. The model can be applied in the context of peer-to-peer applications, online games, or military simulation, among others.

We have started with a very simple model and there are several elaborations that can be considered, including: treating temporal aspects in more detail; mechanisms for allowing reputation (good or bad) to degrade over time; and refining the rules for giving credit based on the reputation of the information source.

We plan to develop a set of standard high-level policies for creating new trust judgments on the basis of reputation. Another direction of future work is to introduce economic notions such as cost-benefit ratios and their relation to reputation and trust.

Acknowledgements. The authors thank the anonymous reviewers for helpful comments and Tim McCarthy for suggesting the game application. The work was supported by ONR grant N00014-01-1-0837.

References

[AH00] E. Adar and B. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.

[BFL96] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[BH77] Henry G. Baker and Carl Hewitt. Laws for communicating parallel processes. In *IFIP Congress*, pages 987–992. IFIP, August 1977.

[Cla] Clan Lord. http://www.clanlord.com/.

[Cli81] W. D. Clinger. *Foundations of Actor Semantics*. PhD thesis, MIT, 1981. MIT Artificial Intelligence Laboratory AI-TR-633.

[CSWH00] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proc. International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 46–66. Springer-Verlag, 2000.

[GJM02] P. Golle, S. Jarecki, and I. Mironov. Cryptographic primitives enforcing communication and storage complexity. In *Proc. Financial Cryptography*, 2002.

[GWW01] C. Gunter, S. Weeks, and A. Wright. Models and languages for digital rights. In *Proc. Hawaii International Conference on Systems Sciences*, 2001.

[PW02] R. Pucella and V. Weissman. A logic for reasoning about digital rights. In *Proc. 15th IEEE Computer Security Foundations Workshop*, pages 282–294, 2002.

[SGR97] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. IEEE Symposium on Security and Privacy*, pages 44–54, 1997.