

The Reputation Lag Attack

Sean Sirur¹ and Tim Muller²

¹ University of Oxford, UK
`sean.sirur@stx.ox.ac.uk`

² University of Nottingham, UK
`tim.muller@nottingham.ac.uk`

Abstract. Reputation systems and distributed networks are increasingly common. Examples are electronic marketplaces, IoT and ad-hoc networks. The propagation of information through such networks may suffer delays due to, e.g., network connectivity, slow reporting and rating-update delays. It is known that these delays enable an attack called the reputation lag attack. There is evidence of impact of reputation lag attacks on existing trust system proposals. There has not been in-depth formal analysis of the reputation lag attack. Here, we present a formal model capturing the core properties of the attack: firstly, the reputation of an actor failing to reflect their behaviour due to lag and, secondly, a malicious actor exploiting this for their personal gain. This model is then used to prove three key properties of the system and the attacker: if there is no decay of reputation, then the worst-case attacker behaviour is to cooperate initially, then wait, then behave badly; increasing communication between users was found to always be of benefit to the users; performing a specified number of negative interactions given any instance of the system is an NP-hard problem.

Keywords: Reputation Lag · Reputation · Trust System · Attack · Malicious Peer

1 Introduction

Ratings can be found as a basis for trust in various networks including e-commerce and social media. Typically, actors will rate their interactions with one another. These individual ratings are propagated through the system, and considered by others when judging who is trustworthy. Timely and effective propagation is necessary for actors to accurately judge each other. Non-ideal networks introduce lag due to network connectivity, people providing ratings late, or other reasons. An attacker can exploit this lag by engaging actors who, due to lag, have not received news of the attacker's prior negative behaviour and still consider them trustworthy. Broadly, we define a reputation lag attack as any instance where an attacker exploits a lag in the propagation of their negative reputation to allow them to perform negative actions they otherwise couldn't have.

No substantial research or well-reported instances of the reputation lag attack exist (not much work has followed up [7], which introduced the notion). We do not know the scale, prevalence and effect on vulnerable networks remains unknown. Nonetheless, existing research [8] shows that the attack is viable on proposed trust systems. Attacks on trust systems often combine different types, and, e.g., fake ratings, Sybil accounts or camouflaging tactics are more obvious, so combined attacks may have been classified as these.

There is general theoretical model of reputation lag attacks. A formal model provides insight into the attacks, even without data. This paper takes a first step towards defining a general formal model of reputation lag attacks. The model successfully captures the core mechanism of the reputation lag attack: some user(s) must trust an attacker who they would not have trusted had no lag been present in the system. Three primary insights were gained from the model. Firstly, if users judge all actions equally regardless of when they occurred, there exists an ordering to the attacker’s actions which is always superior to any other ordering: the attacker first behaves positively; waits for that reputation to spread through the system; and then attempts to behave negatively as much as possible before being rejected by the users. This drastically reduces the search space of possible optimal sequences of actions for the attacker. Secondly, increasing the rate of communication between users relative to the attacker is always detrimental to the attacker in the average case. Finally, how to successfully performing a specified number of negative actions for a given instance of the system is an NP-hard decidability problem for the attacker.

2 Related Work

Distributed systems can use ratings or recommendations between actors as a basis for trust, where an actor’s reputation is defined through these ratings [18]. In such systems, reputation is imperative to actors’ decision-making processes, for example in marketplace environments [5,17]. The delay present in the propagation of these ratings was first identified as a vulnerability by Kerr and Cohen [7] as the “reputation lag attack”. The vulnerability is not present in previous surveys on reputation systems [4,12]. Hoffman, Zage and Nita-Rotaru [4] is an example of how the attack often went unrecognised. The authors decompose reputation systems into their constituent parts and discuss the vulnerabilities present in each. They are prudent in making rating propagation explicit within the *dissemination* stage. However, no notion of lag is considered here so the authors miss a likely environment for exploitation (focusing primarily on transmission integrity).

Even once discovered, the reputation lag attack remained largely unnoticed by the trust community, appearing in some subsequent surveys (e.g. [6,14]) but not others (e.g. [9]). The first analysis is performed by Kerr and Cohen [8], when investigating the success of dishonest sellers against various proposed trust systems in a simulated marketplace. They conclude that the reputation lag attack, though somewhat successful, was largely less so than other attacks, acquiring

less profit and beating fewer trust systems. This finding comes with two major caveats, however: Firstly, the authors’ intuitive but informal definition of reputation lag attacks assumes the attacker must at some point behave honestly. We find that no such restriction is necessary when defining the attack. The second caveat is the implementation of the lag. Every sale suffers from a constant lag before the buyer learns whether they have been cheated. There is no lag in the propagation of this information, however. This makes it difficult to separate how reputation lag effects buyers’ decision given that every sale is subject to the same reputation lag effect. An implication of the above two caveats is that according to their analysis the “re-entry attack” was more successful. We argue that, due to limitations in analysis, the attack was functionally identical to the “reputation lag attack”, except the attacker never needed to behave honestly (with even the author’s noting this). An issue is that, beyond the initial intuition, it is not always clear what a reputation lag attack entails. We feel the issues faced in existing research motivate an abstract formal model to avoid conflating the idiosyncrasies of an attack’s implementation with its analysis.

The reputation lag attack is not restricted to traditional reputation systems with many distributed networks being vulnerable to it. Commonly, strong security guarantees exist through the use of trusted authorities or shared secrets. In some networks, however, it is necessary to establish trust between nodes on a more ad-hoc basis [16]. Any delay in the communication of trust establishing information (and perhaps other “hopping” protocols) would be vulnerable to reputation lag attacks. For example, while research on reputation lag attacks in these contexts are not widespread, many instances of such networks encounter malicious peers and it is possible that the mechanisms against these attackers (e.g. distributed warning systems) are vulnerable to reputation lag attacks. Examples include peer-to-peer networks used for file-sharing (Gnutella, BitTorrent) [10,19]; ad-hoc networks (mesh networks, vehicle-to-vehicle communication) [2,13]; hardware networks (BGP/routing, IoT) [11]; and overlay networks (Tor, I2P) [1].

3 Preliminaries

In this section, some mathematical tools are defined for use later in the paper.

Sequences of events are an important notion through the paper as they are used to describe the sequential behaviour in time of the model presented herein. First, we define sequences recursively:

Definition 1 (Sequence). *A sequence $\sigma \in \Sigma$ over an alphabet \mathcal{C} is recursively defined:*

$$\sigma := \begin{cases} \emptyset \\ \sigma :: c \in \mathcal{C}. \end{cases}$$

We may write $\sigma :: \sigma'$ as a shorthand, where $\sigma :: \emptyset = \sigma$ and $\sigma :: (\sigma' :: c) = (\sigma :: \sigma') :: c$. It is useful to reason about the length of a sequence, $|\bar{\sigma}|$, by letting $|\emptyset| = 0$ and $|\bar{\sigma} :: c| = |\bar{\sigma}| + 1$. This provides a mechanism for both differentiating the number

of elements in sequences and assigning positions to elements of the sequence. To refer to elements or subsequences of a sequence by their position, we introduce indexing as follows:

$$\sigma_t = \begin{cases} \sigma'_t & t < |\sigma| \wedge \sigma = \sigma' :: c \\ c & t = |\sigma| \wedge \sigma = \sigma' :: c \end{cases} \quad (1)$$

$$\sigma_{x \sim y} = \begin{cases} \sigma_{x \sim y-1} :: \sigma_y & x < y \\ \sigma_x & x = y \end{cases} \quad (2)$$

It is useful to discuss the number of occurrences of a particular subset of elements in a sequence as well as the order in which that particular subset occurs irrespective of the other elements e.g. when analysing the behaviour of that particular subset alone. This is done by extracting the subsequence of a particular element from within a sequence. The function $\sqsubset: 2^{\mathcal{C}} \times \Sigma \rightarrow \Sigma$ returns the subsequence of $\bar{\sigma}$ consisting only of the members $c \in C$ of the set of elements $C \subseteq \mathcal{C}$:

$$C \sqsubset \sigma = \begin{cases} (C \sqsubset \sigma') :: c & \text{if } \sigma = \sigma' :: c \text{ where } c \in C \\ (C \sqsubset \sigma') & \text{if } \sigma = \sigma' :: c' \text{ where } c' \notin C \\ \emptyset & \text{if } \sigma = \emptyset \end{cases} \quad (3)$$

It is useful to be able to compare two sequences, where one sequence is essentially the same as another sequence, except it has certain additional actions sprinkled in. Intuitively $\sigma \prec_C \sigma'$ means that we can transform σ' into σ , by removing certain elements $c \in C$ from σ' .

$$\sigma_1 \prec_C \sigma_2 = \begin{cases} \sigma_1 = \emptyset & \text{if } \sigma_2 = \emptyset \\ \sigma'_1 \prec_C \sigma'_2 & \text{if } \sigma_2 = \sigma'_2 :: c' \text{ and } \sigma_1 = \sigma'_1 \\ \sigma_1 \prec_C \sigma'_2 & \text{if } \sigma_2 = \sigma'_2 :: c \text{ for } c \in C \end{cases} \quad (4)$$

Probability theory plays a significant part in the paper as the system is defined on continuous-time stochastic processes. If X is a (continuous) random variable, then $X(\omega)$ represents the outcome of X , $p(X = x)$ represents the probability density at x , $P(X < x)$ represents the probability that the outcome of X is below x ; so $P(X < x) = \int_{-\infty}^x p(X = x) dx$.

The relevant stochastic processes can be modelled using continuous-time Markov chains [15]. Intuitively, a CTMC is a series of random variables indexed with a time t , representing the state at time t . More recent states are not influenced by older states, as the process must be memoryless [15].

Definition 2. A continuous-time Markov chain [15] is a continuous series of random variables $(S)_t$ for $t \in \mathbb{R}$, such that for $x > y > z$, $P(S_x = s | S_y = t) = P(S_x = s | S_y = t, S_z = u)$.

4 Model

Our aim is to model the reputation lag attack. Honest users may communicate information to each other, but when and how often depends on external factors,

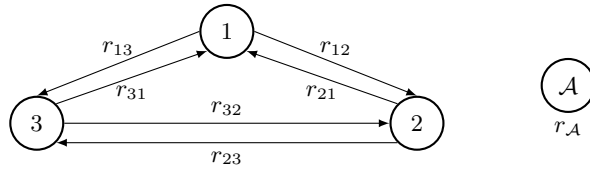


Fig. 1. A graph of users $\mathcal{U} = \{1, 2, 3\}$ and the independent attacker \mathcal{A} with rate $r_{\mathcal{A}}$.

such as internet connectivity, configuration settings or preference. We assume that honest users do not communicate strategically, and thus model them as stochastic processes. The attacker behaves strategically and tries to act in a way to maximise how often he can cheat others, relative to cooperating with others. However, the attacker is still bound to physical limitations, and cannot act at infinite speeds. The first step is to construct a model that defines how often certain users tend to communicate, as well as how often the attacker may be able to act. We refer to this model as the *abstract model*. The *concrete model* (Section 4.3) is an instantiation of the abstract model, and tells us the exact communication between users.

4.1 Abstract Model

The abstract model does not tell us what is being communicated, or what actions have occurred. In order to be able to reason about the concrete communications and actions (and thus the attacker’s strategy), we need to instantiate the attacker’s actions appropriately. The concrete model is defined in Section 4.3 to facilitate this. The final step will be to define and reason about the behaviour of the attacker.

The abstract model defines when two users communicate but not what they communicate. Attacker behaviour is not explicit in the abstract model, only when the attacker has an opportunity to act. We introduce the notion of abstract traces, which specify how and when users and the attacker communicate, but not what they communicate. Users may communicate at different rates. The attacker’s independent communication rate describes the rate at which they receive the opportunity to act.

Definition 3. An abstract system $\bar{\psi} \in \bar{\Psi}$ consists of a tuple $\bar{\psi} = (\mathcal{U}, R, r_{\mathcal{A}}, \bar{M})$: a set of users $\mathcal{U} = \{i \in \{1, \dots, n\} \mid n \in \mathbb{N}_{>0}\}$; an $n \times n$ matrix R describing the communication rates between users, with $r_{ij} \in \mathbb{R}_{\geq 0}$ and $r_{ii} = 0$; and the rate $r_{\mathcal{A}} \in \mathbb{R}_{\geq 0}$ with which the attacker acts.

As a shorthand, we write $\mathbf{r} = r_{\mathcal{A}} + \sum_{i,j \in \mathcal{U}} r_{ij}$ and $r_S = \bigcup_{i,j \in \mathcal{U}} r_{ij} \cup r_{\mathcal{A}}$.

An abstract trace is a sequence of abstract interactions between users and the attacker. It describes in what order interactions occurred for some particular instance of the stochastic system described in subsection 5. It is comprised of either the empty trace; an interaction between two users; an abstract attacker

action; or the concatenation of two other traces.. The trace semantics takes the form of sets of messages assigned to users, representing which messages those users have received:

Definition 4 (Abstract Trace). *An abstract trace $\bar{\sigma} \in \bar{\Sigma}$ is a sequence over the alphabet $\bar{\mathcal{C}} = \{c_{ij} \mid i, j \in \mathcal{U}\} \cup \{c_A\}$.*

As a shorthand, we may write r_c to mean r_{ij} or r_A if $c = c_{ij}$ or $c = c_A$, respectively.

The abstract model defines a stochastic system (or *probabilistic run*) describing who interacts when. In this system, the actors communicate at intervals. The time between each communication is independent of the time between the preceding communications. Formally, every action in the abstract alphabet can be modelled as a series of random variables representing the time between occurrences of that action.

Definition 5. *A probabilistic run of the abstract system $\bar{\psi}$ consists of collection of series of random variables satisfying the Markov property. For each $c \in \bar{\mathcal{C}}$, the probability density functions of the corresponding series ($m \geq 0$) of random variables are:*

$$p(\tau_m^c = t) = r_c e^{-r_c t}$$

We let λ_k^c be a random variable representing the time in which the k^{th} c -action occurred: $\lambda_k^c(\omega) = \sum_{0 \leq i \leq k} \tau_i^c(\omega)$.

The probabilistic run can be viewed as a distribution over possible traces. In particular, we can say that the probabilistic run defines a (continuous-time) Markov chain, where the state consists of the current trace. First, we define the

Definition 6. *The abstract system execution is a continuous series of random variables $(S)_t$ for $t \geq 0$, such that $S_0(\omega) = \emptyset$, and for every t , there exists $t' < t$ such that either $S_t(\omega) = S_{t'}(\omega)$ or $S_t(\omega) = S_{t'}(\omega) :: c$. The latter case occurs if and only if $t' \leq \lambda^{c'} k(\omega) \leq t \implies c' = c$.*

The random variable S_{10} would give you the distribution of all abstract traces of the abstract system running for 10 time units. The intuition is that the state only changes at times where the probabilistic run determines an action occurs. The definition implicitly assumes that no two actions happen at exactly the same time (and the probability of this occurring is indeed 0).

The abstract system execution is a continuous-time Markov chain:

Proposition 1. *The abstract system execution satisfies, for $x > y > z$, that $P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y) = P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y, S_z = \bar{\sigma}_z)$*

Proof. The definition of an abstract system execution trivially implies that $\bar{\sigma}_x = \bar{\sigma}_y :: \bar{\sigma}'$, for some $\bar{\sigma}'$. If $\bar{\sigma}_x = \bar{\sigma}_y$, then $P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y) = P(\forall_{c,m} \tau_c^m \notin [y, x]) = P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y, S_z = \bar{\sigma}_z)$. If $\bar{\sigma}_x = \bar{\sigma}_y :: c$, then $P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y) = P(\forall_{c' \neq c, m} \tau_{c'}^m \notin [y, x] \wedge \exists_m^1 \tau_c^m \in [y, x]) = P(S_x = \bar{\sigma}_x | S_y = \bar{\sigma}_y, S_z = \bar{\sigma}_z)$. If $\bar{\sigma}'$ isn't \emptyset or in $\bar{\mathcal{C}}$, then we can take $x > y' > y$ and recursively apply the argument.

In the abstract trace, the probability distribution for the “next” action remains the same. Specifically, the probability is proportional to the rate.

Proposition 2. *For any times t_1, t_2 , the probability $p(S_{t_2} = \bar{\sigma} :: c | S_{t_1} = \bar{\sigma}) \propto r_c$.*

Proof. Another way to state the theorem is that $\frac{p(S_{t_2} = \bar{\sigma} :: c_1 | S_{t_1} = \bar{\sigma})}{p(S_{t_2} = \bar{\sigma} :: c_2 | S_{t_1} = \bar{\sigma})} = \frac{r_{c_1}}{r_{c_2}}$. Assume w.l.o.g. that $\bar{\sigma}$ contains k_1 instances of c_1 actions, and k_2 of c_2 . Due to Proposition 1, we can assume w.l.o.g. that $\lambda_{k_1}^{c_1} = \lambda_{k_2}^{c_2}$. The ratio to prove is then equal to $p(\tau_{c_1}^{k_1+1} < \tau_{c_2}^{k_2+1}) : p(\tau_{c_1}^{k_1+1} > \tau_{c_2}^{k_2+1})$, or $\frac{r_{c_1}}{r_{c_1} + r_{c_2}} : \frac{r_{c_2}}{r_{c_1} + r_{c_2}}$, proving the proposition.

4.2 Reputation

Before defining the concrete model, it is important that we capture the notion of reputation. We do this through the judgement function δ . We define a judgement function as any function that defines a metric over ratings (messages) that establishes the reputation of the attacker. Users will not accept interactions with a disreputable attacker. Despite this generalised definition of δ , there are some properties which we consider key to the definition of a rational judgement function:

1. Only information known by a user can be made when judging an incoming interaction on behalf of that user.
2. Positive actions must be rewarded and negative actions punished.
3. The judgement function must accept interactions from an attacker with no known prior behaviour to ensure they can enter the system.

There are many additional properties a judgement function could satisfy. Furthermore, it would be simple to extend the model to allow different users to utilise different judgement functions thus representing the various tolerances different users may have to the attacker’s behaviour. However, the basic δ defined for all users in this paper takes only the number of \top and \perp interactions known to user i as arguments. This means, for instance, it is independent of the order in which messages were received and the time at which interactions occurred.

Definition 7. *The function $\delta : \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{R}$ is an arbitrary function with the following properties:*

$$\delta(m', n) > \delta(m, n) \text{ when } m' > m \tag{5}$$

$$\delta(m, n') < \delta(m, n) \text{ when } n' > n \tag{6}$$

$$\delta(0, 0) \geq 0 \tag{7}$$

4.3 The Concrete Model

In the concrete model, the attacker instantiates their abstract actions with concrete actions consisting of an action applied to a user. Actions can have a positive impact or a negative impact, increasing or decreasing the actor’s reputation

respectively. Positive actions are denoted \top and negative ones \perp . When the attacker interacts with a user through one of the above actions, a message, held by that user, is generated. This message contains information regarding what action the attacker performed. It is then propagated through the graph of users, who then use this information to judge the trustworthiness of the attacker. This captures the notion of reputation. A user receiving messages from different users reporting e.g. a positive action, must be able to distinguish which particular action is being reported. We assume users can distinguish different actions and model this using a unique index for each action.

Definition 8. *The concrete system $\psi \in \Psi$ is composed of a tuple $\psi = (\mathcal{U}, R, r_A, M, \Theta, \delta, \mathcal{A}, \Gamma,)$: set of users $\mathcal{U} = \{1, \dots, n\}$; an $n \times n$ matrix R describing the communication rates between users, with $r_{ij} \in \mathbb{R}_{\geq 0}$ and $r_{ii} = 0$; the rate $r_A \in \mathbb{R}_{\geq 0}$ with which the attacker acts; a set of concrete messages $M = \{(\theta, i, x) \mid \theta \in \Theta, i \in \mathcal{U}, x \in \mathbb{N}\}$; two possible results $\Theta = \{\top, \perp\}$; a judgement function δ ; an attacker function \mathcal{A} ; and an instantiation function Γ .*

As a shorthand, we write $\mathbf{r} = r_A + \sum_{i,j \in \mathcal{U}} r_{ij}$ and $r_S = \bigcup_{i,j \in \mathcal{U}} r_{ij} \cup r_A$.

Every abstract trace $\bar{\sigma}$ has a set of corresponding concrete traces. A concrete trace σ is an abstract trace $\bar{\sigma}$ that has had every abstract attacker action $c_{\mathcal{A}}$ substituted with a concrete attacker action. Concrete actions consist of interacting positively or negatively with a user i (c_i^{\top} or c_i^{\perp}) or skipping a turn (c^-).

Definition 9 (Concrete Trace). *An concrete trace σ is a sequence over the alphabet $\mathcal{C} = \{c_{ij} \mid i, j \in \mathcal{U}\} \cup \{c_i^{\theta} \mid \theta \in \Theta, i \in \mathcal{U}\} \cup \{c^-\}$.*

The family of functions defining the set messages known by users in the concrete system is defined:

Definition 10. *The function $\mu_{i>0}^{O \subseteq \Theta} : \Sigma \rightarrow 2^M$ returns the set of messages held by user i concerning actions of a type in T given that the trace σ occurred:*

$$\mu_i^O(\sigma) = \begin{cases} \mu_i^O(\sigma') \cup \mu_j^O(\sigma') & \sigma = \sigma' :: c_{ji} \\ (\theta, i, |\sigma|) \cup \mu_i^O(\sigma') & (\sigma = \sigma' :: c_i^{\theta}) \wedge (\theta \in O) \\ \mu_i^O(\sigma') & \sigma = \sigma' :: c, \text{ for other } c \\ \emptyset & \sigma = \emptyset \end{cases} \quad (8)$$

We introduce the shorthand:

$$\mu^O(\sigma) = \bigcup_{i=1}^{\mathcal{U}} \mu_i^O(\sigma) \quad (9)$$

$$\delta_i(\sigma) = \delta(|\mu_i^{\top}(\sigma)|, |\mu_i^{\perp}(\sigma)|) \quad (10)$$

$$\delta(\sigma) = \delta(|\mu^{\top}(\sigma)|, |\mu^{\perp}(\sigma)|) \quad (11)$$

5 Reputation Lag attack

Above we have defined the environment in which the reputation lag attack can occur. Now, we define the attacker model and the attack itself.

5.1 The Attacker Model

In this model, the attacker \mathcal{A} is captured as a function which outputs a trace of concrete attacker actions which, when substituted into an abstract trace, instantiates a set of abstract attacker action. The attacker function can instantiate each abstract action with one of three actions c_i^\top , c_i^\perp or c^- . The attacker aims to maximise their profit. Informally, profit is simply defined as any function monotonically increases with the attacker’s positive interactions and decreases with their negative interactions i.e. the number of negative interactions the attacker has succeeded in committing given the number of positive interactions they have invested into the system. As such, an “optimal attacker” is defined as an attacker which, for all abstract traces $\bar{\sigma}$, can commit the maximum number of negative interactions when restricted to a particular number of positive interactions and/or given an abstract trace of finite length i.e. maximise their profit given finite resources and/or finite time.

Definition 11. *The attacker’s profit function $\varrho : \Sigma \rightarrow \mathbb{R}$ is subject to the following constraints:*

$$\varrho(\sigma) < \varrho(\sigma') \text{ when } \mu^\top(\sigma) > \mu^\top(\sigma') \text{ if } \mu^\perp(\sigma) = \mu^\perp(\sigma') \quad (12)$$

$$\varrho(\sigma) > \varrho(\sigma') \text{ when } \mu^\perp(\sigma) > \mu^\perp(\sigma') \text{ if } \mu^\top(\sigma) = \mu^\top(\sigma') \quad (13)$$

In this model, different types of attacker can be delineated by how much information they have of the system when making decisions i.e. what subset of the system is considered by the attacker $\mathcal{A}(s \subseteq \psi)$. For example, it is important to consider how much of the abstract trace $\bar{\sigma}$ and attacker is aware of when making decisions. Note, for the purposes of a security analysis, assuming an apparently overestimated attacker is useful for testing the constraints of the system and providing strong guarantees of the system’s resilience against attack. We will consider this when choosing \mathcal{A} :

1. *Attacker Model 1* in which the attacker is omniscient to the past and future i.e. the attacker can view the full abstract trace $\bar{\sigma}$ at will. This attacker’s power is somewhat unrealistic as it grants the attacker the ability to see the future when making decisions but, as stated, for the purposes of a security analysis this is not unreasonable. For instance, an eavesdropping attacker that monitors a system long enough to notice a pattern in the system behaviour could be captured somewhat realistically by this model. Thus, this is the model considered in this paper.
2. *Attacker Model 2* in which the attacker is an eavesdropper to the entire system but is only aware of the past when making decisions i.e. for each abstract attacker action $\bar{\sigma}_x = c_{\mathcal{A}}$, the attacker can view $\bar{\sigma}_{1 \sim x-1}$.
3. *Attacker Model 3* in which the attacker is blinded to every interaction in $\bar{\sigma}$ which is not an attacker action i.e. the attacker only sees $c_{\mathcal{A}} \sqsubset \bar{\sigma}$.

Similarly, the attacker’s knowledge of other aspects of the system such as the user rates R or the judgement function δ can also be allowed or restricted to

different extents. However, given the fact that the attacker is solely concerned with instantiating an abstract trace such that their profit is maximised, for the most powerful attacker it is sufficient to only consider the abstract trace $\bar{\sigma}$ and the and judgement function δ .

Using this information the attacker generates a trace of concrete attacker actions $\dot{\sigma}$ such that $|\dot{\sigma}| = |c_{\mathcal{A}} \sqsubset \bar{\sigma}|$. This attack trace is then substituted into the full abstract trace to make a concrete trace σ :

Definition 12 (Attack Trace). *An attack trace $\dot{\sigma} \in \dot{\Sigma}$ is a sequence over the alphabet $\mathcal{C} = \{c_i^\theta \mid \theta \in \Theta, i \in \mathcal{U}\} \cup \{c^\ominus\}$.*

Definition 13. *Attack traces are constructed by the attacker function $\mathcal{A} : \bar{\Sigma} \times (\mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{R}) \rightarrow \dot{\Sigma}$ defined thus:*

$$\mathcal{A}(\bar{\sigma}, \delta) = \begin{cases} \emptyset \\ \dot{\sigma} :: c_i^\theta & \text{where } (i > 0) \wedge (\theta \in \Theta) \\ \dot{\sigma} :: c^\ominus \end{cases} \quad (14)$$

We give no explicit definition of the function itself and instead explore its properties in section 6. The attack trace $\dot{\sigma}$ is then substituted into the abstract trace $\bar{\sigma}$ to construct the concrete trace σ . This is performed by an instantiation function. To model the fact that users will not accept interactions with a disreputable attack, the instantiation function will substitute any rejected attacker actions with a c^\ominus action. User i judges the attacker at each attacker action $c_i^{\theta \in \Theta}$ via δ with only the information known to them:

Definition 14. *The instantiation function $\Gamma : \bar{\Sigma} \times \dot{\Sigma} \rightarrow \Sigma$ is defined:*

$$\Gamma(\bar{\sigma}, \dot{\sigma}, \delta) = \begin{cases} \emptyset & \text{if } \bar{\sigma} = \dot{\sigma} = \emptyset \\ c_{ij} :: \Gamma(\bar{\sigma}', \dot{\sigma}, \delta) & \text{if } \bar{\sigma} = c_{ij} :: \bar{\sigma}' \\ c :: \Gamma(\bar{\sigma}', \dot{\sigma}', \delta) & \text{if } (\bar{\sigma} = c_{\mathcal{A}} :: \bar{\sigma}') \wedge (\dot{\sigma} = c :: \dot{\sigma}') \wedge (\delta_i(\sigma') \geq 0) \\ c^\ominus :: \Gamma(\bar{\sigma}', \dot{\sigma}', \delta) & \text{if } (\bar{\sigma} = c_{\mathcal{A}} :: \bar{\sigma}') \wedge (\dot{\sigma} = c :: \dot{\sigma}') \wedge (\delta_i(\sigma') < 0) \end{cases} \quad (15)$$

If none of the attacker's actions are denied by a user, we deem that attack trace *complete* for $\bar{\sigma}$. Otherwise, we deem it *incomplete* for $\bar{\sigma}$.

5.2 The Reputation Lag Attack

Here the reputation lag itself is defined in terms of the model presented thus far. Informally, a reputation lag attack occurs when the attacker is allowed to perform a (presumably malicious) interaction with a user who would have rejected the interaction had they had perfect information of the attacker's prior actions. By construction, any example of imperfect user knowledge within this model stems

directly from a failure of the system to propagate the messages in a timely manner. While this definition is very high-level, it successfully captures every instance which could be considered a reputation lag attack.

If an attacker interaction with user i is accepted by that user using only the information (messages) known to them but is rejected when using all the information present in the system, then a reputation lag attack has occurred.

First, we define an omniscient instantiation function in which users judge the attacker with all the information available in the system:

Definition 15. *The omniscient instantiation function $\Gamma^* : \bar{\Sigma} \times \dot{\Sigma} \rightarrow \Sigma$ is defined:*

$$\Gamma^*(\bar{\sigma}, \dot{\sigma}, \delta) = \begin{cases} \emptyset & \text{if } \bar{\sigma} = \dot{\sigma} = \emptyset \\ c_{ij} :: \Gamma^*(\bar{\sigma}', \dot{\sigma}, \delta) & \text{if } \bar{\sigma} = c_{ij} :: \bar{\sigma}' \\ c :: \Gamma^*(\bar{\sigma}', \dot{\sigma}', \delta) & \text{if } (\bar{\sigma} = c_{\mathcal{A}} :: \bar{\sigma}') \wedge (\dot{\sigma} = c :: \dot{\sigma}') \wedge (\delta(\sigma') \geq 0) \\ c^- :: \Gamma^*(\bar{\sigma}', \dot{\sigma}', \delta) & \text{if } (\bar{\sigma} = c_{\mathcal{A}} :: \bar{\sigma}') \wedge (\dot{\sigma} = c :: \dot{\sigma}') \wedge (\delta(\sigma') < 0) \end{cases} \quad (16)$$

Here we define the reputation lag attack indicator. If the attacker has an increased profit when instantiated normally compared to when instantiated by the omniscient function (i.e. if the attacker has successfully exploited the lag for their own gain), a reputation lag attack has occurred.

Definition 16. *The reputation lag attack indicator is defined:*

$$\text{RLA}(\bar{\sigma}, \dot{\sigma}, \delta, \varrho) = \begin{cases} 1 & \varrho(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta)) > \varrho(\Gamma^*(\bar{\sigma}, \dot{\sigma}, \delta)) \\ 0 & \varrho(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta)) = \varrho(\Gamma^*(\bar{\sigma}, \dot{\sigma}, \delta)) \end{cases} \quad (17)$$

6 Results

The primary motivation for the above formalism is to provide insight into the reputation lag attack. Here we elicit the three following key properties of the attack: the definition of δ defined herein is shown to be vulnerable to a superior ordering of attacker actions; increasing the rate of user communication is shown to never be detrimental to the users and could be detrimental to the attacker in the average case; the decidability problem of whether the attacker can perform a specified number of negative actions is shown to be have an NP-hard computational complexity.

6.1 Attack Ordering

The order in which the attacker executes their actions has an impact on their success. In a structured attack trace, the attacker goes through three phases: a \top phase, a \sqcup phase and a \perp phase. In a given phase, the attacker only executes actions of that type. We show that structured attack traces, under the particular

δ defined in the above model, are always superior to unstructured strategies. Intuitively, this results from the fact that positive actions occurring earlier in the trace gives them more chance to be propagated whilst negative occurring later in the trace have less time to be propagated.

Definition 17. Define the reflexive partial order \prec on attacker actions, s.t. $c_i^\top \prec c^- \prec c_j^\perp$, for all i, j . We define the partial order \prec as $\dot{\sigma} :: c :: c' :: \dot{\sigma}' \prec \dot{\sigma} :: c' :: c :: \dot{\sigma}'$ iff $c \prec c'$.

Proposition 3. For every $\dot{\sigma}$, there is a minimal element $\dot{\sigma}' \prec \dot{\sigma}$, and this minimal element has the property that it is a structured attack trace.

Proof. If $\dot{\sigma}'$ is structured, then there is no $\dot{\sigma}' = \dot{\sigma}_1 :: c :: c' :: \dot{\sigma}_2$ where $c' \prec c$, so $\dot{\sigma}'$ is a minimal element. Vice versa, any minimal element may not be of the shape $\dot{\sigma}' = \dot{\sigma}_1 :: c :: c' :: \dot{\sigma}_2$ either, so it must be structured.

Theorem 1. For all abstract traces $\bar{\sigma}$; attack traces $\dot{\sigma}$ and $\dot{\sigma}' \prec \dot{\sigma}$ (where $\dot{\sigma}$ is complete for $\bar{\sigma}$); users i ; and locations $x \leq |\bar{\sigma}|$: $\delta_i(\Gamma(\bar{\sigma}, \dot{\sigma}', \delta)) \geq \delta_i(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta))$

Proof. Consider two adjacent attacker actions $\dot{\sigma}_a$ and $\dot{\sigma}_{a+1}$. We denote their corresponding positions in the abstract (or concrete) traces as $\bar{\sigma}_y$ and $\bar{\sigma}_z$ respectively. Two cases follow from this: $\dot{\sigma}_a \prec \dot{\sigma}_{a+1}$ and $\dot{\sigma}_{a+1} \prec \dot{\sigma}_a$.

Case 1 ($\dot{\sigma}_a \prec \dot{\sigma}_{a+1}$): As \prec is reflexive, we may construct $\dot{\sigma}' = \dot{\sigma}$. In this, it follows trivially that $\delta_i(\Gamma(\bar{\sigma}, \dot{\sigma}', \delta)) \geq \delta_i(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta))$

Case 2 ($\dot{\sigma}_{a+1} \prec \dot{\sigma}_a$): We construct $\dot{\sigma}'$ by swapping the two elements in question: $\dot{\sigma}' = \dot{\sigma}_{1 \sim a-1} :: \dot{\sigma}_{a+1} :: \dot{\sigma}_a :: \dot{\sigma}_{a+2 \sim |\dot{\sigma}|}$. This implies that $\sigma_{1 \sim x-1} = \sigma'_{1 \sim x-1}$ and $\sigma_{y+1 \sim |\sigma|} = \sigma'_{y+1 \sim |\sigma'|}$. We consider the case where $\dot{\sigma}_{a+1} = \dot{\sigma}'_a = c_i^\top$. We notice two things: firstly, user i is still aware of the c_i^\top action at the time of σ'_z and so has lost no information in comparison to σ .

Secondly, we notice that the earlier introduction of c_i^\top creates the opportunity for it to be propagated between σ'_y and σ'_z and thus possibly even further. Essentially, more users $u \neq i$ may be aware of the c_i^\top in σ' than in σ from point y onward: for all users u and locations $x \geq y$, $\mu_u^\top(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta)_{1 \sim x}) \subseteq \mu_u^\top(\Gamma(\bar{\sigma}, \dot{\sigma}', \delta)_{1 \sim x}) \implies |\mu_u^\top(\Gamma(\bar{\sigma}, \dot{\sigma}', \delta)_{1 \sim x})| \geq |\mu_u^\top(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta)_{1 \sim x})|$.

A symmetrical argument in the case that $\dot{\sigma}_a = \dot{\sigma}'_{a+1} = c_i^\perp$ leads us to conclude that the each user is aware of less or equal negative actions after the swap: for all users u and locations $x \geq |\sigma|$, $|\mu_u^\perp(\Gamma(\bar{\sigma}, \dot{\sigma}', \delta)_{1 \sim x})| \leq |\mu_u^\perp(\Gamma(\bar{\sigma}, \dot{\sigma}, \delta)_{1 \sim x})|$. We also see from both arguments that the case in which one of the swapped attacker actions are c^- , that particular action has no effect on the knowledge of the users i.e. the inequality holds trivially. By the definition of \prec , it is not possible for two actions to be swapped where both actions are of the same type (e.g. where both are positive).

The increased awareness of positive interactions and the decreased awareness of negative ones coupled with the monotonicity of δ implies that, for all

users i and locations $x \leq |\sigma|$, $(|\mu_u^\top(\Gamma(\bar{\sigma}, \sigma', \delta)_{1 \sim x})| \geq |\mu_u^\top(\Gamma(\bar{\sigma}, \sigma, \delta)_{1 \sim x})|) \wedge (|\mu_u^\top(\Gamma(\bar{\sigma}, \sigma', \delta)_{1 \sim x})| \leq |\mu_u^\top(\Gamma(\bar{\sigma}, \sigma, \delta)_{1 \sim x})|)$
 $\implies \delta_i(\Gamma(\bar{\sigma}, \sigma', \delta)_{1 \sim x}) \geq \delta_i(\Gamma(\bar{\sigma}, \sigma, \delta)_{1 \sim x})$.

By transitivity of \prec , the proof for pairs holds for all traces. Proposition 3 shows that the optimal ordering is structured. \square

Thus, for all complete attack traces, structured traces are superior. We restrict our theorem to complete attack traces as incomplete traces contain counterexamples and any optimal attack trace will be complete. Structuring may not affect profit but it reduces the search space of attack traces as the optimal strategy must be structured. However, structuring is dependent on the judgement function. A judgement function in which reputation decays with time would be sensitive to abrupt changes in behaviour such as in a structured attack trace, thus making time-dependent judgement a simple but effective mitigation.

6.2 Effect of Communication Rates on the Attacker Strategy

Considering an optimal (and thus structured) attack trace σ , we show that increasing the communication rates is detrimental to the attacker. We make the assumption that we are in a state where the knowledge of the deals has spread to all users, and the attacker is in the cheat stage of his structured attack. The reason for this assumption, is that the question of how to most effectively cheat many users, is the core question behind analysis reputation lag attacks.

We formulate our theorem as follows:

Theorem 2. *Let $\bar{\psi}, \bar{\psi}'$ be a pair of abstract systems differing only in their respective matrices R, R' , such that $R_{ij} < R'_{ij}$. Let $S_e = S'_e$ represent the initial development of the system (including deals and skips). At any time $t > 0$, for all attacker traces σ consisting only of cheats, $p(\text{RLA}(S'_t, \sigma, \delta, \rho)) \leq p(\text{RLA}(S_t, \sigma, \delta, \rho))$.*

Proof. First we prove that if $\sigma \prec_{\{c_{ij}\}} \sigma'$, then σ allows at least as many reputation lag attacks as σ' . If $|\sigma_1| = k$, $\sigma = \sigma_1 :: \sigma_2$ and $\sigma' = \sigma_1 :: c_{ij} :: \sigma_2$, then for every h , either $\mu_h(\sigma') = \mu_h(\sigma)$ or $\mu_h(\sigma') = \mu_h(\sigma) \cup \mu_j(\sigma_1)$. The latter means that the messages j told i after σ_1 have reached h , the former means that they have not. In either case $\mu_h(\sigma') \subseteq \mu_h(\sigma)$. For any two traces with $\sigma \prec_{\{c_{ij}\}} \sigma'$ we can iteratively apply this argument for the additional messages.

Second, partition the set of all concrete traces Σ into Σ_0 and Σ_1 , such that if $\sigma \prec_{\{c_{ij}\}} \sigma'$, then either $\sigma, \sigma' \in \Sigma_0$, $\sigma, \sigma' \in \Sigma_1$ or $\sigma \in \Sigma_0, \sigma' \in \Sigma_1$. Since $R_{ij} < R'_{ij}$, it follows that $\sum_{\sigma \in \Sigma_0} S_t(\sigma) \geq \sum_{\sigma \in \Sigma_0} S'_t(\sigma)$ and $\sum_{\sigma \in \Sigma_1} S_t(\sigma) \leq \sum_{\sigma \in \Sigma_1} S'_t(\sigma)$. In particular, we can select the partition Σ_0 to be those traces up to time t' where the attacker can perform all his actions, and Σ_1 those traces where he cannot. S'_t will assign more probability to the latter than S_t . \square

6.3 NP-Hardness

While strategies may exist to improve the attacker's profit in different scenarios, it is important to consider the feasibility of the optimal attacker. A polynomial

optimal strategy for a given judgement function would be the ideal goal for any attacker. However, we show below that, for any judgement function, the computation complexity of constructing a strategy which can perform a specified number of negative interactions is NP-hard.

Theorem 3. *For any judgement function δ , the decidability problem of whether it is possible to perform m cheats without performing deals is NP hard.*

Proof. We provide a reduction to the 3-SAT problem, which is NP-complete [3]. In this proof outline, we provide the reduction itself, but omit the full proof that it is indeed a reduction. The full proof is a tedious exercise in bookkeeping, whereas the reduction provides insight in why it is NP-hard.

Let $X = x_1 \dots x_k$ be the set of variables in the 3-sat problem. Let ℓ_{ij} be the j^{th} literal in the i^{th} clause. Assume there are n clauses.

For each variable take a user for its positive and negative atom, take a user for each literal in the formula, and we add an additional pair of users for every variable: $\{u_x, u_{\neg x} | x_i \in X\} \cup \{v_{ij} | \ell_{ij}\} \cup \{v_{x_i}, v_{\neg x_i} | x_i \in X\}$. The last pairs of users form pseudo clauses – they represent the clause $x \vee \neg x$. There are a total of $n+k$ (pseudo) clauses. The set $U^{\geq h} = \{v_{ij} | i \geq h, \ell_{ij}\} \cup \{v_{x_i}, v_{\neg x_i} | i + n \geq h, x_i \in X\}$. As a short-hand, we say a set of users performs a kill-communication, if they communicate their messages to all other users ($O(n^2)$).

Consider an abstract trace $\bar{\sigma}$ with the following shape:

- The trace starts with k attacker actions.
- Then, for $h \in [1, \dots, k+n]$ do:
 - Users $U^{\geq h}$ perform a kill-communication.
 - The users u_x, u_y, u_z , corresponding to the inverse of literals $\ell_{h1}, \ell_{h2}, \ell_{h3}$ send a single communication to the respective literals.
 - The attacker gets 1 action.
 - All users u_x and $u_{\neg x}$ communicate to $\ell_{h1}, \ell_{h2}, \ell_{h3}$.
- After performing these $k+n$ steps, the attacker gets k more actions.

The decidability question for k variables and n clauses is whether the attacker can perform $n+3k$ actions. The size of the trace $\bar{\sigma}$ is $O(n^3)$, which is polynomial. \square

7 Conclusion

The formalism captured the two core properties of the attack: firstly, users inaccurately judging the reputation of the attacker due to incomplete knowledge caused by reputation lag and, secondly, a malicious actor exploiting this for their personal gain. The primary aim of the paper was to gain insight regarding the attacker’s strategies to help with both mitigation and detection of the attack. There were three key outcomes: Theorem 1 shows how to re-order the attacker’s actions, to increase the power of the attack: the attacker behaves positively; waits for this reputation to spread to as many users as possible; then begins

behaving as negatively as possible before the users reject them. The fact that our judgement function, which models reputation, does not have a decay factor is a crucial ingredient for this theorem. Trust/reputation with decay factors may be somewhat more resistant against these attacks. Theorem 2 showed that increasing user communication rates cannot be detrimental to users and may be detrimental to the attacker. Intuitively, this follows directly from the definition of the reputation lag attack which relies on poor user communication. However, as also evidenced by Theorem 3, the issue of determining the effectiveness of attacks is difficult, so it is important to have a proof of our intuitions. Finally, Theorem 3 showed that performing a specified number of negative interactions given any instance of the system $\bar{\sigma}$ is an NP-hard problem, implying the optimal attacker is computationally unfeasible.

Our definitions were chosen to be sufficiently abstract to cover a variety of systems, and is readily extendable to more than just particular reputation systems. The class of systems we consider are those systems where users communicate certain information about how other users have acted in the past, where good behaviour offsets bad behaviour, and too much bad behaviour leads to refusing to interact. We argue that many systems that consist of a distributed set of entities communicating knowledge about one another could be defined through the presented model with some modification.

Here we discuss further study. There is much to learn about the system and its effect on the attacker e.g. the effects of different judgement functions or how profit functions affect them. Investigating the combination of the reputation lag attack with other attacks may be of use for learning in which environments the attack is likely to be. Identifying real-world examples of the attack would also offer insight into the effectiveness of the model. Further investigation into the attacker's strategy is a vital next step as understanding identifying likely attack patterns is imperative not only to mitigating but also to detecting reputation lag attacks in the wild.

This paper provided a formal model of the reputation lag attack. The formalism captured the core properties of reputation lag attacks. The formalism allowed us to prove three interesting properties of the reputation lag attack: deals before cheats, communication benefits users but not the attacker, and finding optimal attacks is NP-hard. However, the analysis presented here is still in early stages. We hope to apply some of our techniques in practice, as well as continue to strengthen the formalism. The attacker's strategies and their relationship with the system as a whole warrants further study.

References

1. Egger et al.: Practical attacks against the i2p network pp. 432–451 (2013). https://doi.org/10.1007/978-3-642-41284-4_22
2. Fogue et al.: Securing warning message dissemination in vanets using cooperative neighbor position verification. *IEEE Transactions on Vehicular Technology* **64**(6), 2538–2550 (2015). <https://doi.org/10.1109/TVT.2014.2344633>

