On Robustness of Trust Systems

Tim Muller, Yang Liu, Sjouke Mauw, Jie Zhang

On Robustness of Trust Systems

Tim Muller[1], Yang Liu[1], Sjouke Mauw[2], and Jie Zhang[1]

[1] Nanyang Technological University
{tmuller,yangliu,zhangj}@ntu.edu.sg
[2] University of Luxembourg
sjouke.mauw@uni.lu

Abstract. Trust systems assist in dealing with users who may betray one another. Cunning users (attackers) may attempt to hide the fact that they betray others, deceiving the system. Trust systems that are difficult to deceive are considered more robust. To formally reason about robustness, we formally model the abilities of an attacker. We prove that the attacker model is maximal, i.e. 1) the attacker can perform any feasible attack and 2) if a single attacker cannot perform an attack, then a group of attackers cannot perform that attack. Therefore, we can formulate robustness analogous to security.

1 Introduction

Robustness refers to the ability of a trust system to function properly under all circumstances. Users may purposely perform actions to attempt to prevent the trust system from functioning properly.

For example, a famous food critic travels around the country visiting restaurants. The critic tastes the food and enjoys the service. A restaurant with good food and good service gets positive reviews. Potential customers read the reviews, knowing that the food critic has a keen eye and are eager to try the restaurants he recommends. Some restaurants recognise the famous food critic, and go above and beyond to provide the critic better food and service than usual. The positive impression sketched by the critic in his review does not translate well to the regular customer, who gets substandard food and service. The restaurant got an unfair advantage over its neighbour, who provides equal quality food and service to all customers. The restaurant exploited the mechanism of the procedure which provides restaurant reviews, and the system malfunctioned.

In trust systems, the intrinsic interactions have the property that one party can betray another party. Such behaviour is unfair and dishonest on the interpersonal level. However, on the level of the trust system, this is expected behaviour. The trust system can deal with users betraying other users; if trusted users would never betray, then trust systems can trivially trust anyone. However, as in the aforementioned example regarding the food critic, there are behaviours that are more than merely unfair and dishonest on the interpersonal level, rather, they deceive the entire system. We study such deceptions of the system, and refer to them as *attacks* - examples are listed in Section 2.1.

Trust systems that are less vulnerable to attacks are deemed more *robust*. A general and formal definition of robustness of trust systems helps in detecting

and fixing vulnerabilities, and possibly to verify exploit-freeness. In this paper, we precisely establish the notion of robustness of trust systems, and its aspects.

Robustness of trust systems is related to robustness of (other) software systems, and related to security of software systems. Robustness of a software system typically refers to the capability of the system to deal with or recover from unexpected input. Given two similar algorithms performing a division, the algorithm that checks for and deals with divide-by-zero issues is more robust than the algorithm which does not. Robustness correlates, in software systems, with the number of different input that leads to faulty states.

Security of a software system typically refers to the impossibility of reaching a particular class of faulty states, regardless of input. Secrecy of a message, as a classical example, holds when there are no actions that the attacker can perform that lead to a state where the attacker knows the message. Security is the absence of input that leads to a faulty state.

The notion of robustness of trust systems, as many currently hold (e.g. [24,13]), is somewhere in between these two notions. Lack of robustness in software systems is bad, primarily because a legitimate user may experience problems if he accidentally inputs the wrong data. Lack of security is bad, primarily because an attacker may seek to input the wrong data. Robust trust systems seek to prevent the latter, as we do not want attackers to exploit the workings of the trust system. However, in security, legitimate users are assumed to always input data that does not lead to faulty states (they are assumed to follow protocol). In robust trust systems, we cannot make such assumptions about legitimate users' input - any user could fail at any time. In other words, in robust trust systems we must assume that there are attackers that purposely enter bad input (like security), but we may not assume that non-attackers will not enter bad input (like standard robustness). Our proposal is based on this key notion; the notion that an attackers' devious choice of input should be no more likely to lead to a faulty state than a normal users' randomly selected (and potentially bad) input.

In computer security - be it symbolic security [3] or provable security [15] - the notions surrounding the attacker model and security properties are well established. Deviations from the default attacker model are subject to extra scrutiny. We can learn from computer security that we should not formulate an ad-hoc notion of robustness. Our main contribution is exactly that - an attacker model and notions of robustness properties that are independent of the trust system at hand. Moreover, we learn from computer security that the strategy of the attackers in the attacker model should not be limited by our own ideas regarding attacks on a system. We prove that our attacker model indeed captures all possible strategies of an attacker. Furthermore, again akin to security, we prove that it suffices to verify a robustness property under one attacker, and that robustness against multiple attackers follows automatically.

In Section 2, we introduce the notions and formalisms that we use throughout the paper. Notably, we define types of users and accounts, their states, actions and behaviours and how they synthesise abstract trust systems. In Section 2.1, we list attacks on trust systems found in the literature to establish archetypical

attacks that we want to capture with our notions. In Section 3, we define the possible actions of an attacker, called malicious behaviour, in the attacker model. We show that the attacker model matches our intuition, and that it has maximal strength. In Section 4, we define notions related to robustness of trust systems. We define robustness properties, analogous to security properties. We show that if a robustness property holds for one attacker, it holds for several attackers.

2 Formalisation of Trust Systems

Trust systems revolve around interactions with an asymmetric power balance, and trust is employed to deal with this imbalance in the form of *trust opinions* ($\mathcal{O}$ is the set of trust opinions). Trust opinions are the building blocks of trust systems, and indicate the likelihood that the target allows the interactions to *succeed* or *fail*. A *subject* constructs a trust opinion about a *target*, before allowing the target to control the interaction. The trust opinion determines whether the subject *accepts* an interaction with a target, possibly together with the *payoff of success* and *payoff of failure* (α and β are the sets of payoff of success and failure, respectively). Before a subject can accept an interaction, the target needs to *offer* the interaction. The offer (implicitly or explicitly) sets the pay-off of success and of failure.

Since we study the robustness, we want to reason about behaviour that the designer did not anticipate. We refer to behaviour anticipated by the designer as *ideal behaviour*. Formal *correctness* is a property of a trust system which holds when the trust system provides mathematically correct probabilities in trust opinions, provided all users are ideal. The Beta model [18,11] is an example of a system where formal correctness holds. The Beta model produces mathematically correct results in an ideal trust system [20]. Robustness, as defined more precisely later, is roughly the ability of the system to deal with behaviour that is not ideal. Robustness is sometimes viewed as an extension of correctness, but we argue that it should be seen as a trade-off against correctness, in Section 4.

As motivated in Section 3, even the non-ideal behaviour that concerns robustness is, in some way, restricted. We refer to these non-ideal behaviours as *malicious behaviour*, and a user with malicious behaviour as an *attacker*. There are goals (e.g. obtain a large profit) that attackers should not be more likely to achieve than ideal users. Since, if the malicious users are more likely to achieve these goals, then they have an unfair advantage. We refer to the negation of these goals as *robustness properties* - after their similarity to security properties. A malicious behaviour that breaks a robustness property is an *attack*.

Before we can reason formally about such behaviours, we need to formalise a system in which these behaviours are expressed. We adopt a semantic view of the trust system, philosophically related to transition system spaces [1]. A trust system space (Definition 1) encompasses all trust systems that arise when actual users are instantiated in the trust system space. Figure 1 contains an example of a part of a trust system space.

A trust system space consists of a set of users, a set of actions, a set of states and a set of transitions. There are conditions on each of these sets (e.g. an

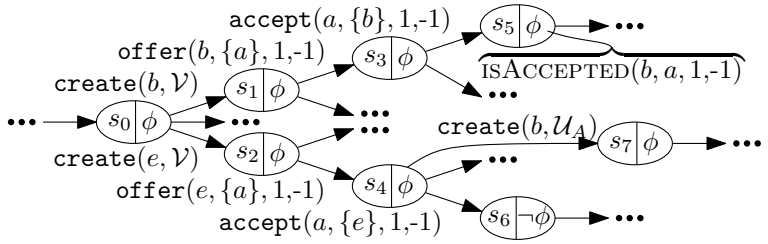

Fig. 1. Fraction of an example of a trust system space.

account must be created before it can make an offer), and we call such sets *valid* when the conditions are satisfied. First we identify which sets of users, actions, states and transitions are valid, then we define trust system spaces.

Every trust system has a set of users, which in turn control accounts of given types. We say that a set of users $\mathcal{U}$ is *valid* when there is a set of accounts $\mathcal{V}$, such that every account belongs to exactly one user. Formally, there is a function $\text{OWNER} : \mathcal{V} \to \mathcal{U}$, if $\text{OWNER}(a) = u$ then u *owns* a. The set of subjects and the set of targets are subsets of the set of accounts, $\mathcal{V}_S \subseteq \mathcal{V}$ and $\mathcal{V}_T \subseteq \mathcal{V}$, respectively.

Every trust system space has a collection of actions, and every step of the system is an action, as depicted in Figure 1. Each action has an *executor* and *listeners* - denoted by its first and second parameter, respectively. The owner of the executor is the *originator* and the owners of the listeners are the *recipients*. We identify a collection of parameterised actions that are sufficiently ubiquitous in trust systems that we elevate them to a special status:

- $\texttt{create} : \mathcal{V} \times \mathcal{P}(\mathcal{V})$; $\texttt{create}(a, B)$ denotes creation of account a, with accounts in B being notified.
- $\texttt{offer} : \mathcal{V}_T \times \mathcal{P}(\mathcal{V}_S) \times \alpha \times \beta$; $\texttt{offer}(a, B, x, y)$ denotes that target a makes an offer to subjects in B, with payoff of success x and payoff of failure y.
- $\texttt{accept} : \mathcal{V}_S \times \{\{b\} | b \in \mathcal{V}_T\} \times \alpha \times \beta$; $\texttt{accept}(a, B, x, y)$ denotes that subject a accepts the offer made by b (with $B = \{b\}$) with outcome (x, y).
- $\texttt{succeed} : \mathcal{V}_T \times \{\{b\} | b \in \mathcal{V}_S\} \times \alpha \times \beta$ and $\texttt{fail} : \mathcal{V}_T \times \{\{b\} | b \in \mathcal{V}_S\} \times \alpha \times \beta$; $\texttt{succeed}(a, B, x, y)$ and $\texttt{fail}(a, B, x, y)$ denote that target a succeeds or fails the offer accepted by b (with $B = \{b\}$) with outcome x and y, respectively.
- $\texttt{recommend} : \mathcal{V}_S \times \mathcal{P}(\mathcal{V}_S) \times \mathcal{V}_T \times \mathcal{O}$; $\texttt{recommend}(a, B, c, t)$ denotes that subject a claims to B that his trust opinion about c equals t.

The set of actions $\mathcal{A}$ is valid, if it contains at least these six groups of actions.

A trust system space has a set of states $\mathcal{S}$, which determines which actions are possible and which predicates hold, as depicted in Figure 1. For a state space $\mathcal{S}$ of a trust system space to be *valid*, there must exist a projection function π, which projects the system state onto the state of a single user. In other words, $\pi : \mathcal{U} \times \mathcal{S} \to \mathcal{S}_U$, where $\mathcal{S}_U$ is the state space of a single user. We shorthand $\pi(u, s)$ to $\pi_u(s)$. If $\pi_u(s) = \pi_u(t)$, then we say that s and t are *indistinguishable* to user u. We identify three state predicates $\text{EXISTS}_s(a)$, $\text{isOffered}_s(a, b, x, y)$ and $\text{isAccepted}_s(a, b, x, y)$, which are supposed to hold in the state s when account a has been created, target b offers (x, y) to a, and a accepts the offer (x, y) from b, respectively. In Figure 1, $\text{isAccepted}_{s_5}(b, a, 1, -1)$ holds, for example.

The set of transitions $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is the core of the definition of a trust system space. Transitions are labelled edges that represent steps from one state to another. We identify two types of requirements on *valid* transitions:

First, there are requirements regarding proper functioning of the trust systems. In a trust system space, (L1) only the originator and the recipients of an action are aware of it, (L2) the result of performing an action in a given state is deterministic, (L3) if two states are indistinguishable to a user, then the effects of an action are identical to that user, (L4) if two states are indistinguishable to a user, that user can perform the same actions (L5) accounts that are created remain existent (to avoid impersonation after deletion) and (L6) offers and acceptances by given users remain unchanged when these users are unaware that an action happened.

L1. For all transitions $(s, \mathtt{a}, t) \in \mathcal{T}$, for all $u \in \mathcal{U}$ that are neither the originator nor a recipient of action $\mathtt{a}$, $\pi_u(s) = \pi_u(t)$.
L2. For all transitions $(s, \mathtt{a}, t) \in \mathcal{T}$ and $(s, \mathtt{a}, t') \in \mathcal{T}$, $t = t'$.
L3. For all transitions $(s, \mathtt{a}, t) \in \mathcal{T}$ and $(s', \mathtt{a}, t') \in \mathcal{T}$, if $\pi_u(s) = \pi_u(s')$ then $\pi_u(t) = \pi_u(t')$.
L4. For all transitions $(s, \mathtt{a}, t) \in \mathcal{T}$, where $u \in \mathcal{U}$ is the originator of $\mathtt{a}$, for all states $s' \in \mathcal{S}$ where $\pi_u(s) = \pi_u(s')$, there is a state $t' \in \mathcal{S}$, such that $(s', \mathtt{a}, t')$.
L5. For all transitions $(s, \mathtt{a}, t)$, if $\textsc{exists}_s(b)$ then $\textsc{exists}_t(b)$.
L6. For all transitions $(s, \mathtt{a}, t)$, if $b \in \mathcal{V}$ and $c \in \mathcal{V}$ are neither executors nor listeners, then $\textsc{isOffered}_s(b, c, x, y)$ iff $\textsc{isOffered}_t(b, c, x, y)$, and $\textsc{isAccepted}_s(b, c, x, y)$ iff $\textsc{isAccepted}_t(b, c, x, y)$, for all $x \in \alpha$, $y \in \beta$.

Second, there are requirements regarding the actions that users can perform. It is (R1) always possible to create new subject or target accounts, (R2) always possible for targets to make any offer, (R3) always possible for subjects to accept existing offers, (R4) always possible for targets to succeed or fail accepted offers and (R5) always possible for subjects to make any recommendations.

R1. For some $\mathcal{C} \subseteq \mathcal{P}(\mathcal{V})$, $\mathcal{C} \neq \emptyset$, for every user $u \in \mathcal{U}$, state $s \in \mathcal{S}$ and set of accounts $C \in \mathcal{C}$, for some $a \in \mathcal{V}_S$ (with $\neg\textsc{exists}_s(a)$ and $\textsc{owner}(a) = u$), $b \in \mathcal{V}_T$ (with $\neg\textsc{exists}_s(b)$ and $\textsc{owner}(b) = u$) and $t, t' \in \mathcal{S}$, we have $(s, \mathtt{create}(a, C), t) \in \mathcal{T}$ and $(s, \mathtt{create}(b, C), t') \in \mathcal{T}$.
R2. For every state $s \in \mathcal{S}$, target $a \in \mathcal{V}_T$ with $\textsc{exists}_s(a)$, set of accounts $C \in \mathcal{P}(\mathcal{V})$, payoff of success $x \in \alpha$ and payoff of failure $y \in \beta$, for some states $t \in \mathcal{S}$, we have $(s, \mathtt{offer}(a, C, x, y), t) \in \mathcal{T}$.
R3. For every state $s \in \mathcal{S}$, subject $a \in \mathcal{V}_S$, target $b \in \mathcal{V}_T$, payoff of success $x \in \alpha$ and payoff of failure $y \in \beta$ with $\textsc{isOffered}_s(a, b, x, y)$, for some $t \in \mathcal{S}$, we have $(s, \mathtt{accept}(a, \{b\}, x, y), t) \in \mathcal{T}$.
R4. For every state $s \in \mathcal{S}$, target $a \in \mathcal{V}_T$, subject $b \in \mathcal{V}_S$, payoff of success $x \in \alpha$ and payoff of failure $y \in \beta$, with $\textsc{isAccepted}_s(a, b, x, y)$, for some $t \in \mathcal{S}$ (or $t' \in \mathcal{S}$), we have $(s, \mathtt{succeed}(a, \{b\}, x, y), t) \in \mathcal{T}$ (or $(s, \mathtt{fail}(a, \{b\}, x, y), t') \in \mathcal{T}$).
R5. For every state $s \in \mathcal{S}$, subject $a \in \mathcal{V}_S$ with $\textsc{exists}_s(a)$, target $c \in \mathcal{V}_T$ with $\textsc{exists}_s(c)$, trust opinion $o \in \mathcal{O}$ for some $B \in \mathcal{P}(\mathcal{V})$ and $t \in \mathcal{S}$, we have $(s, \mathtt{recommend}(a, B, c, o), t) \in \mathcal{T}$.

Remark 1. We introduce one additional requirement for purely technical reasons, namely that the trust system space is finitely branching; that there are finitely many outgoing transitions in every state. See the technical report [19].

A trust system space is like a chess rule book; a chess rule book defines the users (white and black), the actions ("move pawn C4-C5", etc.), the states (placement of pieces on the board) and the transitions ("move pawn C4-C5" is only possible if it is white turn, if there is a pawn on C4, if C5 is free, and if the resulting state does not put white's king in check).

Definition 1 (Trust System Space). *A trust system space is a 4-tuple* $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{T})$*, where* $\mathcal{U}$*,* $\mathcal{S}$*,* $\mathcal{A}$ *and* $\mathcal{T}$ *are valid sets of users, states, actions and transitions.*

A trust system space is not an actual trust system - similar to how a chess rule book is not a game of chess, To obtain a trust system, users need to be instantiated with a strategy - similar to how a game of chess needs two players with a strategy. We first introduce the notion of a strategy, then we define how a strategy can be applied to a trust system space to obtain an actual trust system.

The users' strategies determine the relative probability of their available actions, as well as the expected time they spend in a certain state. The expected time is also known as a rate (in rated transition systems [14] and continuous-time Markov chains [22]). Rates are an effective way to model the behaviour of independent entities without a global scheduler. Intuitively, each user can increase the probability of performing an action, (only) by increasing the rate of that action.

We define strategies and behaviour based on rates as follows:

Definition 2 (Strategy and Behaviour). *A* strategy *is a function* $f \in \mathcal{F} = \mathcal{S}_U \to (\mathcal{A} \nrightarrow \mathbb{R}_{\geq 0})$*, which assigns a rate to every action available in a state to the user. A* combined strategy *is a function* $\gamma \in \Gamma = \mathcal{U} \to \mathcal{F}$*.*
A behaviour *is a distribution over strategies. A discrete behaviour has probability mass function* $B \in \mathcal{B} = \mathcal{F} \to [0, 1]$*. If* B *is a discrete behaviour, then its* support*,* $\mathrm{supp}(B)$*, is the set of strategies where* $B(f) > 0$*. A* combined behaviour *is a distribution over combined strategies. A discrete combined behaviour has probability mass function* $\theta \in \Theta = \Gamma \to [0, 1]$*.*

In chess, a strategy is, e.g., Kasparov's strategy, and combined strategy is Kasparov versus Fischer[3]. A behaviour is, e.g., "some grandmaster's strategy", and a combined behaviour is "some grandmaster versus an unknown player".

The strategy provides rates, rather than probabilities. Given a combined strategy, we can *normalise* the rates to probabilities by dividing the rate of a transition from state s by the sum of the rates of the other transitions s. The normalisation of γ is $\overline{\gamma}(u, s, a) = \frac{\gamma(u)(\pi_u(s))(a)}{\sum_{u' \in \mathcal{U} \wedge a' \in \mathcal{A}} \gamma(u')(\pi_{u'}(s))(a')}$, where $\gamma(u)(s)(a)$ is taken as 0 when undefined. An *assignment of behaviour* is a shorthand way of defining a combined behaviour. The assignment of behaviour $\overline{\theta} : \mathcal{U} \to \mathcal{B}$, is shorthand for θ, with $\theta(\gamma) = \prod_{u \in \mathcal{U}} \overline{\theta}(u)(\gamma(u))$.

[3] Kasparov and Fischer are famous chess players.

The result of applying behaviour to a trust system space is a rated transition system (with an initial state), as defined in [14]:

Definition 3 (Rated Trust System). *A* rated trust system *is a rated transition system $(S, A, \mathbf{s}_0, W)$, where S is a set of states, A is a set of actions, $\mathbf{s}_0$ an initial state and $W : S \times A \times S \to \mathbb{R}_{\geq 0}$ is a rate function.*
Let $M = (\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{T})$ be a trust system space, $\overline{\theta}$ be an assignment of behaviour and $s_0 \in \mathcal{S}$ a state. Then $[\![M, \overline{\theta}, s_0]\!]$ is a rated transition system $(S, A, \mathbf{s}_0, W)$ where: 1) $S \subseteq \mathcal{S} \times \Theta$, 2) $A = \mathcal{A}$, 3) $\mathbf{s}_0 = (s_0, \theta)$, and 4) when $(s, \mathtt{a}, t) \notin \mathcal{T}$, W satisfies $W((s, f), \mathtt{a}, (t, g)) = 0$ for all $f, g : \Theta$, and when $(s, \mathtt{a}, t) \in \mathcal{T}$ where u originates $\mathtt{a}$, W satisfies $W((s, f), \mathtt{a}, (t, g)) = E_{\mathtt{a}}(f)$ for $f, g : \Theta$, where 4a) the expected rate $E_{\mathtt{a}}(f) = \sum_{\gamma \in \mathrm{supp}(f)} f(\gamma) \cdot \gamma(u)(\pi_u(s))(\mathtt{a})$ and 4b) $g(\gamma) = \frac{\overline{\gamma}(u,s,\mathtt{a}) \cdot f(\gamma)}{\sum_{\delta \in \Gamma} \overline{\delta}(u,s,\mathtt{a}) \cdot f(\delta)}$.

The rated trust system is the natural result of applying combined behaviour to a trust system space in an initial state. The state of a rated trust system is determined by both the state in the trust system space and the combined behaviour. The rated transitions can be interpreted as a labelling on the trust system space, where transitions that do not occur in the trust system receive rate 0, and rates of other transitions are determined by the combined behaviour in a straightforward manner, via (4a). Note that after a transition, a combined strategy that assigns low normalised probability to that transition is less likely to be the actual combined strategy, via Bayes' theorem (4b). This is similar to concluding that an opponent that sacrifices his queen without apparent benefit, is unlikely to have Kasparov's strategy.

In this section, we have defined trust system spaces, which define the possible input and the relation between input and output. We define strategies (and distributions thereof) to model how users choose the input they provide. We further define rated trust systems, which model a running trust system with users with strategies.

2.1 Known Attacks

We are setting out to provide a general, formal definition of robustness. In order to ensure the applicability and relevance of such a definition, we must keep real attacks in mind. We identify the following attacks:

- The on/off attack, in which the attacker builds his trust value, then fails in one or more interactions and depletes his trust value, and slowly rebuild his trust value with time on his side [23]. This attack is particularly powerful on systems where subjects forget behaviour of targets over time.
- The value imbalance attack, in which an attacker builds his trust value in low stake interactions, and depletes his trust value in high stake interactions [13,10]. This works on systems where the stakes of interactions vary. On some game-theoretical systems, this is expected behaviour and not an attack.

- The reputation lag attack, in which an attacker builds his trust value, then fails interactions in quick succession, before his lowered trust value propagates through the system [13,10].
- The discrimination attack, which is essentially our food critic example from the introduction [10]. It is also known as the conflicting behaviour attack [23].
- The re-entry attack, which is akin to the on/off attack, except depleted accounts are replaced by newly created accounts [13,10].
- The distraction attack, in which the attacker creates many superfluous offers, requests or recommendations in order to prevent its victim to perform relevant actions. It is a special case of the denial-of-service attack in [8].
- The proliferation attack, in which the attacker creates many target accounts and thus represents a large portion of all target accounts, meaning that he may receive a large portion of all interactions too.
- The composite trust attack, which asserts that there are composite interactions [20] where multiple targets are involved. In some cases, the action of the attacker does not influence the outcome of the interaction. The attacker to abuses the action to manipulate his trust value.
- The unfair ratings attack, in which the attacker manipulates the reputation of some targets by providing unfair ratings [10,8,23]. This attack has many subdivisions, some of which are studied in great detail. [25,9].
- The shilling attack, in which the attacker matches the profiles of (groups of) users, to ensure his unfair ratings carry more weight [17]. This attack works for recommender systems.
- The Sybil attack is an extremely powerful attack, in which the attacker creates multiple accounts to perform combinations of the aforementioned attacks [6].

3 Malicious Behaviour

As mentioned in the introduction, our notion of robustness hinges on notions of ideal behaviour and malicious behaviour. Each user, ideal or malicious, performs certain actions with a certain probability, at certain times, depending on its state. A function that assigns probability to actions given a state is called a strategy (see Definition 2). Which strategies are ideal depends on the trust system at hand. Typically, trust systems make assumptions about the behaviours of users, denoted in the form of an ideal behaviour. At a formal level, we simply assert that we are given an ideal behaviour I together with the trust system space M.

Ideally, a trust system can compute the probabilities of future actions of ideal users. That is, some trust systems (such as [11,18]) provide formally correct answers, provided that users adhere to the assumptions of the system. We refer to the ability of a trust system to provide formally correct answers for ideal users as *correctness*. However, we are interested in users who do not exhibit ideal behaviour. Robustness is the inability of non-ideal behaviour to achieve something that the ideal users cannot. It is clear that these non-ideal behaviours have limitations. As in computer security, we need to construct a model of which non-ideal behaviours an attacker can exhibit. In formal computer security, the

default attacker model is called the Dolev-Yao model [5]. The Dolev-Yao model defines what strategies an attacker (in the security domain) may perform - and by elimination, which he may not perform. We define a default attacker - inspired by the Dolev-Yao model - in the trust domain, by providing the set of strategies that the attacker may perform.

The Dolev-Yao attacker is maximally powerful in that he can accomplish anything that a group of attackers can accomplish. This is in fact a key property of the Dolev-Yao attacker, as we do not need to model groups of attackers communicating and coordinating. Modelling one attacker is, provably, enough. Our attacker exhibits the same property, as proven in Theorem 2.

First, we define the attacker model informally, but precisely. An *attacker* is a user with a malicious behaviour. At any time in a *malicious behaviour*:

C1. The attacker has a complete understanding of the system. The attacker (only) has access to all private information of his accounts. The attacker can reason with this information.
C2. The attacker can create accounts.
C3. The attacker can offer any interactions with any of his target accounts.
C4. The attacker can decide to succeed or fail at any interaction with any of his target accounts.
C5. The attacker can make arbitrary recommendations with any of his subject accounts.
C6. The attacker can perform any auxiliary actions (including accepting offers) for subject or targets, with his subject or target accounts, respectively.

The first ability, C1, ensures that our attacker model captures the attacker that uses the available information to optimize his decisions. Effectively, C1 disallows security through obscurity - a well-known anti-pattern in computer security. The other abilities, C2-C5 match at least some of the actions in the attacks from Section 2.1. Account creation, C2, is required for Sybil attacks. Offering interactions, C3, is required for proliferation attacks. The ability to succeed or fail at will,f C4, is required for on/off attacks. The ability to make arbitrary recommendations, C5, is required for unfair ratings attacks. Hence, we see that any reasonable attacker model capturing the attacks from Section 2.1 has at least these capabilities. Finally, the capability to perform any of the auxiliary actions, C6, is included mostly for completeness' sake. It is clear that a real attacker would abuse auxiliary actions, if this would help him achieve his goal. Hence our model should include this capability.

Before introducing the attacker, we need to formalise what it means for a user (i.e. the attacker) to have volition. A user with volition can pick its own strategy in the trust system. We model this by letting a volitional trust system be a set of rated trust systems, each generated by a malicious strategy. A choice of strategy equates to a choice of a rated trust system from a volitional trust system.

Definition 4 (Volitional Trust System). *Let M be a trust system space, I be an ideal behaviour, s_0 be the initial state, and e be a user. A* volitional trust system *is a set of rated trust systems denoted $\Upsilon^{e \triangleleft F}_{(M,I,s_0)}$, for some $F \subseteq \mathcal{F}$. Let $[\![M, \theta', s_0]\!] \in \Upsilon^{e \triangleleft F}_{(M,I,s_0)}$ iff $\overline{\theta'}(u \neq e) = I$ and $\overline{\theta'}(e)(f \in F) = 1$.*

The volitional trust system $\Upsilon^{e \triangleleft F}_{(M,I,s_0)}$ only contains rated trust systems based on the trust system space M and initial state s_0, where all users except e have behaviour I. The only difference between the elements in $\Upsilon^{e \triangleleft F}_{(M,I,s_0)}$ is the strategy of e, which can be any strategy in F.

We define the maximal attacker model as follows:

Definition 5 (Maximal Attacker Model). *The* maximal attacker model *is* $\Upsilon^{e \triangleleft \mathcal{F}}_{(M,I,s_0)}$*, for trust system space* M*, ideal behaviour* I*, initial state* s_0 *and attacker* e*.*

We may refer to a volitional trust system with an attacker as a *subverted trust system*. If an attacker behaviour f can be imagined within the restraints of our action alphabet $\mathcal{A}$ and the attacker's state space $\mathcal{S}_U$, then there is an subverted trust system $\upsilon \in \Upsilon^{e \triangleleft \mathcal{F}}_{(M,I,s_0)}$ where the attacker uses strategy f.

We define the intuitive attacker model as follows:

Definition 6 (Intuitive Attacker Model). *The* intuitive attacker model *is a volitional trust system* $\Upsilon^{e \triangleleft X}_{(M,I,s_0)}$*, where: First, for all* $(S, A, \mathbf{s}_0, W) \in \Upsilon^{e \triangleleft X}_{(M,I,s_0)}$*,* $\mathbf{s}, \mathbf{s}' \in S$ *and* $\mathtt{a} \in A$*, if* $\pi_u(\mathbf{s}) = \pi_u(\mathbf{s}')$ *then* $W(\mathbf{s}, \mathtt{a}, \mathbf{t}) = W(\mathbf{s}', \mathtt{a}, \mathbf{t}')$ *- for those* $\mathbf{t}, \mathbf{t}' \in S$ *with* $W(\mathbf{s}, \mathtt{a}, \mathbf{t}) \neq 0$ *and* $W(\mathbf{s}', \mathtt{a}, \mathbf{t}') \neq 0$*. Second, for all collections of transitions* $(s_0, \mathtt{a}_0, t_0), \ldots, (s_n, \mathtt{a}_n, t_n) \in \mathcal{T}$ *where* e *is the originator of all* $\mathtt{a}_i$ *and there is no pair of transitions* $(s_i, \mathtt{a}_i, t_i), (s_j, \mathtt{a}_j, t_j)$ *with both* $\pi_e(s_i) = \pi_e(s_j)$ *and* $\mathtt{a}_i = \mathtt{a}_j$*, there is a volitional trust system* $(S, A, \mathbf{s}_0, W) \in \Upsilon^{e \triangleleft X}_{(M,I,s_0)}$ *such that every* $W((s_i, f_i), \mathtt{a}_i, (t_i, g_i))$ *is equal to any predetermined value* r_i*.*

The first rule limits the attacker's behaviour in indistinguishable states, i.e., his private information, according to C1. The second rule captures C2-C6, as the set of strategies contains any combination of rates for all actions (including create - C2, offer- C3, succeed and fail - C4, recommend - C5 - and others - C6) that respect rule C1. Since the rate of each action can have an arbitrarily large value, the attacker can perform the action with arbitrary probability smaller than one.

Remark 2. The rate is the inverse of time, in an exponential distribution (see, e. g. [2]) - this forms the theoretical basis of rated transitions systems [14] and continuous-time Markov chains [22]. If we accept that the attacker acts according to an exponential distribution, then any positive time corresponds to a rate (as the inverse of time). In this case our notion that the attacker can perform any of his actions (C2-C6) with any probability is trivially satisfied.

Arguably we may reject the notion that the attacker acts according to an exponential distribution. If an attack exists for this attacker, but not for an attacker that acts according to an exponential distribution, then the attack is purely based on exact timing (but not on expected timing). However, we should compare the attacker with an ideal user that also does not act according to an exponential distribution. Thus, the attack cannot purely be based on exact timing. If we, nevertheless, reject the notion that the attacker acts according to an exponential distribution, we must generalise the notion of subverted trust models to hybrid automata [7] or probabilistic timed automata [16] - both automata with both time and probability.

Our intuitive notion of an attacker (Definition 6) corresponds with the attacker that is strongest by definition (Definition 5):

Theorem 1. *The maximal and intuitive attacker models are equal.*

Proof. See [19].

Malicious behaviour can only be performed by attackers and, by definition, not by ideal users. However, not all malicious behaviour is an attack. Consider a user that only creates an additional account, but never uses that account to make offers, interactions and recommendations, nor uses the private information of that account. Such a user is an attacker on a system where additional account creation is not ideal behaviour, but the behaviour does not accomplish anything, and thus is not an attack. In Section 4, we define additional notions to define attacks and robustness. For now, it suffices to realise that C1-C6 (Definition 5/6) do not define attacks, but rather the toolset of an attacker.

In computer security, security is relative to so-called security properties. A typical example of a security property is secrecy of a certain message m. The system is secure, when there is no reachable state in which the security property is violated. In the next section, we define robustness in a similar way, differing from security only where necessary.

4 Robustness

We have motivated why we need a formal generic definition of robustness of trust systems. So far, we have introduced the formal machinery and the attacker model, similar to (symbolic) formal security [3] - a methodology that has proven itself in practice. Our definition of robustness properties are also similar to formal computer security. However, there is an alternative way to reason about robustness in a formal and general way, e.g. in [12]. We refer to the alternative approach as the game-theoretical approach, because it gives attackers a utility function and it assumes rationality.

The game-theoretical approach is elegant and powerful, however, our approach has two advantages over the game-theoretical approach: First, we do not have a utility function, but robustness properties. That means that, e.g., distorting trust opinions is bad in itself, rather than because the utility function increases, due to an increased probability that the user interacts with the attacker, due to the distorted trust opinion. In the game-theoretical model, therefore, the notion that distorting trust opinions is an attack, relies on assumptions about the system and the users, whereas intuitively, distorting trust opinions is an attack regardless of the attacker's gains. Second, the game-theoretical notion of robustness is an extension of the notion of correctness - game-theoretic robustness fails trivially in incorrect systems. There are three drawbacks of having robustness as an extension of correctness, rather than a trade-off: Firstly, in many existing systems, correctness cannot be proven, hence robustness cannot be compared. Secondly, a trust system may have goals other than correctness and robustness, thus having users make suboptimal choices by design, and trivially having superior strategies for an attacker. Thirdly, viewing robustness and

correctness as a trade-off more naturally represents design decisions in creating trust systems. For example, not incorporating recommendations in trust opinions makes the system robust against unfair rating attacks at the expense of correctness of trust opinions [21].

A robustness property is a predicate that holds in a collection of system states. A robustness property is a predicate, for which it is undesirable that attackers are more likely to satisfy it than ideal users; e.g., "gain 1000$ in failed interactions". Typically, the probability of breaking the robustness property is non-zero, however, the probability of an ideal user breaking such a property is also non-zero. A trust system is robust when an attacker is no more likely to break the property than an ideal user. The rationale is that the designer modelled the system with ideal users in mind, hence whatever probability the ideal user has to break a property, that probability is acceptable.

In formal computer security, there are tools (e.g. ProVerif [4]) that can determine whether a security property holds, given a specification of the protocol. The algorithms used by the tools are of intractable complexity, but solve the problem at hand sufficiently often to be of practical value. Such tools would be a valuable asset in determining the robustness of a system. The first step towards automated verification, is a standardised formalisation of the problem. We shall use the notion of volitional trust systems and the notion of robustness properties to define robustness. This is fully analogous to how, in formal computer security, the notions of protocols and security properties define security.

We define the notion of a robustness property in a trust system space:

Definition 7 (Robustness Property). *A* robustness property ϕ *is a state predicate over the trust system space. If* ϕ *holds in a state* s *in a trust system space, then* ϕ *also holds in any state* (s, g) *in a rated trust system.*

Observe that the assignment of probabilities has no impact on which properties hold after a given sequence of actions.

We define the notion of probability of reaching ϕ:

Definition 8 (Probability of Reaching ϕ). *Given a rated trust system* $(S, A, \mathbf{s}_0, W)$*, the* probability of reaching ϕ *is recursively defined as* $p_\phi(\mathbf{s}_0)$*, where for* $\mathbf{s} \in S$*:* $p_\phi(\mathbf{s}) = 1$ *if* $\phi(\mathbf{s})$*, and* $p_\phi(\mathbf{s}) = \frac{\sum_{\mathbf{t} \in S, \mathbf{a} \in A} W(\mathbf{s},\mathbf{a},\mathbf{t}) \cdot p_\phi(\mathbf{t})}{\sum_{\mathbf{t} \in S, \mathbf{a} \in A} W(\mathbf{s},\mathbf{a},\mathbf{t})}$ *if* $\neg\phi(\mathbf{s})$.

The equation defining the probability of reaching ϕ does not necessarily terminate. Nevertheless, the value of $p_\phi(s_0)$ is well-defined[4], even if computation is infeasible. For predicates ϕ that only hold in a finite number of states, $p_\phi(s_0)$ can always be computed.

Now, we are interested in two volitional trust systems in particular, one where the user with volition is ideal, and one where the user with volition is malicious. In Section 3, we have defined what volitional trust systems result from malicious behaviours - equivalently in Definitions 5 and 6. A volitional trust system on based on a volitional ideal user is defined as:

[4] Assuming absence of cycles in the trust system space, which follows from perfect recall.

Definition 9 (Ideal Trust System). *An ideal trust system based on trust system space M, ideal behaviour I, initial state s_0 and user e is the volitional trust system $\Upsilon^{e \lhd \mathrm{supp}(I)}_{(M,I,s_0)}$.*

Based on these two types of volitional trust systems, we can verify whether a robustness property holds in a trust system:

Definition 10 (Robustness of ϕ). *In a trust system space M with ideal behaviour I and initial state s_0,* robustness of ϕ *holds when the maximal probability of reaching $\neg\phi$ in a subverted trust system $\upsilon \in \Upsilon^{e \lhd \mathcal{F}}_{(M,I,s_0)}$ is no greater than the maximal probability of reaching $\neg\phi$ in an ideal trust system $\upsilon' \in \Upsilon^{e \lhd \mathrm{supp}(I)}_{(M,I,s_0)}$.*

Observe that robustness of ϕ only holds regardless of trust system space (M) and initial state (s_0) when the support of the ideal behaviour is equal to the set of all strategies. In other words, robustness of ϕ trivially holds, when all possible strategies are ideal ($\mathrm{supp}(I) = \mathcal{F}$). However, correctness is more difficult to achieve for larger sets of ideal strategies. Thus, as remarked before, robustness and correctness are a trade-off.

The Dolev-Yao attacker, in security, is sufficiently powerful, that any attack that can be performed by a group of attackers, can be performed by a single attacker. Our attacker has the same property, albeit under the assumption that users do not discriminate between users a priori. (Non-discrimination implies that substituting a user for another user with identical behaviour does not essentially change anything.)

Theorem 2. *For any robustness property ϕ and trust system space, that do not discriminate users, the probability of reaching ϕ under two cooperating attackers is equal to the probability of reaching ϕ under one attacker.*

Proof. See [19].

There are two ways to interpret the implications of Theorem 2. The first is the straightforward interpretation, that our attacker model is sufficiently strong to capture attacks with multiple attackers. It is obvious that this result follows from the capability of creating accounts arbitrarily. The alternative interpretation is relevant when the capability to create accounts at liberty is rejected. Our result shows that if an attack exists for colluding attackers, this attack exists for our maximal attacker. Thus, if robustness of ϕ holds in a system where accounts can be created freely, then ϕ holds in an otherwise identical system where several attackers, each with a single account, collude.

Example 1 (Verifying Robustness). In order to verify robustness of a system, we need to specify a trust system space, specify ideal behaviour and specify robustness properties. In practice, these may come in another specification language than assumed in this paper, in which case translation is necessary. We let the trust system space contain the fraction depicted in Figure 1. After applying the ideal strategies of the owners of a, b and e, we obtain a rated transition system. The rated transition system can be represented, simply by labelling the edges in

the graph with rates. The property ϕ in Figure 1 corresponds to "the attacker's (OWNER(e)'s) offer is not accepted before the ideal target's (OWNER(b)'s)". Both the subverted trust system and the ideal trust system are sets of rated trust systems, thus collections of different labellings of edges. For every labelling, we can compute the probability that we end up in a state where $\neg\phi$ holds. Now, we can compute the maximal probability that we end up in a state where $\neg\phi$ holds in a set of labellings. We can compare the maximal probability in the subverted trust system with the maximal probability in the ideal trust system. If they are equal, robustness holds, if they differ, robustness does not hold.

The robustness property is a qualitative property, not a quantitative property. There is a straightforward way to introduce a quantitative aspect to robustness properties:

Definition 11 (Quantitative Robustness Properties). *In a trust system space M with ideal behaviour I and initial state s_0, the* amount of robustness *of ϕ is defined as the difference between the maximal probability of reaching ϕ in a subverted trust system $\upsilon \in \Upsilon^{e \lhd \mathcal{F}}_{(M,I,s_0)}$, and the maximal probability of reaching ϕ in an ideal trust system $\upsilon' \in \Upsilon^{e \lhd \mathrm{supp}(I)}_{(M,I,s_0)}$.*

The advantage of the quantitative robustness properties, is that it allows reasoning about robustness of systems where qualitative robustness does not hold. Our quantitative robustness property is more useful than a quantification based on the number/set of strategies that can break the security property. Even if there is only one attack available for the attacker, the attacker can select this strategy. Thus if that strategy exceeds the maximal ideal strategy by 0.5, then the attacker has an unfair advantage of 0.5. When there are multiple attacks available, the attacker can only select one strategy - presumably the most effective one.

5 Conclusion

We have introduced formal machinery that allows us to express the notion of trust systems semantically, in the form of trust system spaces. We argue that robustness refers to the distance between a system operating under the designers' assumptions and a real system. The designers' assumptions come in the form of ideal behaviour. When applying (ideal) behaviour to users, non-determinism is replaced by probability, transforming trust system spaces into rated trust systems. The attacker is a special user, whose behaviour is not ideal, but malicious.

Not all non-ideal behaviour can be performed by an attacker. Hence, we provide a model of malicious behaviour, in the form of two equivalent attacker models. Both models define a toolset of strategies of the attacker, in the form of a volitional trust system. One attacker model is based on an intuitive understanding of what an attacker should be able to do. The other attacker model contains, by definition, all strategies that can be performed within a trust system space. They mutually support each other's validity via their equivalence. The definition of the attacker model is one of the main contributions of the paper.

Behaviour being malicious is not sufficient for it to be an attack. An attack is a malicious behaviour that breaks a property with a probability exceeding that

of an ideal behaviour. A robustness property is a state predicate that attackers want to break. We introduce probabilistic notions of reachability of a property. We define robustness with respect to a certain robustness property based on the probabilistic reachability of the negation of the state predicate. The definition of robustness with respect to a robustness property (Definition 10) is another of the main contributions of the paper. We further extend the robustness property to a quantified variant, that allows comparison between two systems that both fail to uphold a certain robustness property.

We prove that a multitude of attackers is no more powerful than a single attacker (Theorem 2). This notion is crucial to our initial choice to model all users except the attacker as ideal users, which is, therefore, validated by Theorem 2. The choice to restrict ourselves to one attacker severely simplifies the analysis of robustness properties - both for manual analysis and for possible future automated verification tools.

We identify four different, albeit intertwined, directions of future work. First, to analyse the robustness of real trust systems - to link theory to practice, e.g. in cloud computing, e-commerce or vehicular networks. Second, to research theoretical implications of our approach, e.g. complexity, expressivity, extensions or simplifications. Third, to implement our ideas to allow automated verification (based on tools as PRISM[5] or PAT[6]). Fourth, to find (or at least characterise) trust systems that satisfy given robustness properties.

Acknowledgements. This work is supported by "Formal Verification on Cloud" project under Grant No: M4081155.020. This work is partially supported by the A*Star SERC grant (1224104047) awarded to Dr. Jie Zhang.

References

1. Jos C.M. Baeten, Twan Basten, and Michel A. Reniers. *Process algebra: equational theories of communicating processes*, volume 50. Cambridge university press, 2010.
2. Patrick Billingsley. *Probability and measure*. Wiley, 3 edition, 1995.
3. Bruno Blanchet. Security protocol verification: Symbolic and computational models. In *Proceedings of the First international conference on Principles of Security and Trust*, pages 3–29. Springer, 2012.
4. Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. In *Logic in Computer Science*, pages 331–340. IEEE, 2005.
5. Danny Dolev and Andrew C Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.
6. John R Douceur. The sybil attack. In *Peer-to-peer Systems*, pages 251–260. Springer, 2002.
7. Thomas A Henzinger. *The theory of hybrid automata*. Springer, 2000.
8. Kevin Hoffman, David Zage, and Cristina Nita-Rotaru. A survey of attack and defense techniques for reputation systems. *ACM Comput. Surv.*, 42(1):1–31, 2009.
9. Siwei Jiang, Jie Zhang, and Yew-Soon Ong. An evolutionary model for constructing robust trust networks. In *Autonomous Agents and Multi-Agent Systems*, pages 813–820. IFAAMAS, 2013.

[5] http://www.prismmodelchecker.org [6] http://www.patroot.com

10. Audun Jøsang and Jennifer Golbeck. Challenges for robust trust and reputation systems. In *Security and Trust Management, Saint Malo, France*, 2009.
11. Audun Jøsang and Roslan Ismail. The beta reputation system. In *Bled Electronic Commerce Conference*, pages 41–55, 2002.
12. Reid Kerr and Robin Cohen. Towards provably secure trust and reputation systems in e-marketplaces. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 172. ACM, 2007.
13. Reid Kerr and Robin Cohen. Smart cheaters do prosper: defeating trust and reputation systems. In *Autonomous Agents and Multiagent Systems*, volume 2, pages 993–1000. International Foundation for Autonomous Agents and Multiagent Systems, 2009.
14. Bartek Klin and Vladimiro Sassone. Structural operational semantics for stochastic process calculi. In *Foundations of Software Science and Computational Structures*, pages 428–442. Springer, 2008.
15. Neal Koblitz and Alfred Menezes. Another look at "provable security". Cryptology ePrint Archive, Report 2004/152, 2004.
16. M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. In Y. Lakhnech and S. Yovine, editors, *Proc. Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant Systems (FORMATS/FTRTFT'04)*, volume 3253 of *LNCS*, pages 293–308. Springer, 2004.
17. Shyong K Lam and John Riedl. Shilling recommender systems for fun and profit. In *International Conference on World Wide Web*, pages 393–402. ACM, 2004.
18. Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *System Sciences, 2002. HICSS.*, pages 2431–2439. IEEE, 2002.
19. Tim Muller, Yang Liu, Sjouke Mauw, and Jie Zhang. On robustness of trust systems. Technical report, Nanyang Technological University. `http://pat.sce.ntu.edu.sg/tim/papers/robustnesstechreport.pdf`, 2014.
20. Tim Muller and Patrick Schweitzer. A formal derivation of composite trust. In *Foundations and Practice of Security*, volume 7743, pages 132–148. 2013.
21. Tim Muller and Patrick Schweitzer. On beta models with trust chains. In *Trust Management VII*, volume 401, pages 49–65. 2013.
22. William J Stewart. *Introduction to the numerical solution of Markov chains*, volume 41. Princeton University Press Princeton, 1994.
23. Yan Lindsay Sun, Zhu Han, Wei Yu, and KJ Ray Liu. Attacks on trust evaluation in distributed networks. In *Information Sciences and Systems*, pages 1461–1466. IEEE, 2006.
24. Jie Zhang and Robin Cohen. Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach. *Electronic Commerce Research and Applications*, 7(3):330–340, 2008.
25. Lizi Zhang, Siwei Jiang, Jie Zhang, and Wee Keong Ng. Robustness of trust models and combinations for handling unfair ratings. In *Trust Management VI*, pages 36–51. Springer, 2012.