Computer-Aided Decision-Making with Trust Relations and Trust Domains (Cryptographic Applications)[⋆]

Simon Kramer[⋆⋆], Rajeev Goré, and Eiji Okamoto

[1] University of Luxembourg
simon.kramer@a3.epfl.ch
[2] Australian National University
rajeev.gore@anu.edu.au
[3] University of Tsukuba, Japan
okamoto@risk.tsukuba.ac.jp

Abstract. We propose generic declarative definitions of individual and collective trust relations between interacting agents and agent collections, and trust domains of trust-related agents in distributed systems. Our definitions yield (1) *(in)compatibility, implicational,* and *transitivity* results for trust relationships, including a *Datalog-implementability* result for their logical structure; (2) *computational complexity* results for deciding potential and actual trust relationships and membership in trust domains; (3) a positive (negative) *compositionality* result for strong (weak) trust domains; (4) a *computational design pattern* for building up strong trust domains; and (5) a negative *scalability* result for trust domains in general. We instantiate our generic trust concepts in five major cryptographic *applications* of trust, namely: Access Control, Trusted Third Parties, the Web of Trust, Public-Key Infrastructures, and Identity-Based Cryptography. We also show that *accountability induces trust.* Our defining principle for weak and strong trust (domains) is (common) belief in and (common) knowledge of agent correctness, respectively.

Keywords Access Control; accountability; applied modal logic; CADM; computational trust; cryptographic-key management; dependable distributed and multi-agent systems; PKI; TTP; Web of Trust.

1 Introduction

The subject matter of this paper is trust in dependable distributed or multi-agent systems. In this section, we introduce the motivation and goal for this matter, as well as the methodology that we employ to meet our goal.

[⋆] Cf. [40] for an early technical-report and [39] for a workshop version of this paper.

[⋆⋆] This, corresponding, author's contribution was initiated with Grant P 08742 from the Japan Society for the Promotion of Science, and completed with Grant AFR 894328 from the National Research Fund, Luxembourg, cofunded under the Marie-Curie Actions of the European Commission (FP7-COFUND), during an invited stay at the Institute of Mathematical Sciences, Chennai, India.

Table 1. Functionality guarantee of dependable distributed systems

	in spite of	**thanks to**
technology	*incorrect* design & natural forces	*correct* design
agents	*incorrect* behaviour: (un)intentional incorrect use of the technology—abuse (misuse)	*correct* behaviour: correct use of the technology

1.1 Motivation

Dependable distributed systems guarantee their functionality both in spite of and thanks to the technologies that implement these systems as well as the man or machine agents that partake in them (cf. Table 1). Examples of agents are: threads, processes, processors and their cores, real and virtual machines, system administrators and users, network nodes, etc. The functionality guarantee is conditioned on *naming* (cf. [4, Section 6.4] and [25, Section 19.1]) and *number:*

Naming on agent identity for the aspects of *anonymity* and *pseudonymity.*
Number on a minimal number of correct (or dually, a maximal number of faulty or corrupt) agents for the aspects of *fault tolerance* [49] (classical distributed computation) and *corruption tolerance* [75] (secure multiparty computation).

The notion of *agent correctness* (e.g., as induced by a *policy*) in turn depends on the system itself. Examples of correct agent behaviour are given in Table 2, in various degrees of definitional difficulty. In any case, agent correctness captures the *predictability* of each correct partaking agent that guarantees the functionality of the system to some or even all, correct or faulty, partaking agents. We opine that this predictability is ultimately one of agent behaviour rather than one of the agents' mental attitudes behind their behaviour. All that ultimately matters is the behavioural *effect* of attitudes, which themselves may have no, or no relevant effect. (A mental attitude that may indeed lead a human agent b to behave correctly—or to be trustworthy—to another human agent a is when b has the will [in case b is volitional] and capability [in case there is a need] to control b's behaviour in a's sense.) In sum, system functionality depends on agent correctness, and agents depend on each other via each other's correctness. Whence the *social* need, called *trust*, to know whether or not someone—somebody or something—behaves correctly, i.e., as expected—for some*thing* meaning [65]:

> according to design and policy, doing what is required—despite environmental disruption, human user and operator errors, and attacks by hostile parties—and not doing other things.

Note that according to game theory [9, Page 70] this need is also *rational:*

> [. . .] it isn't rational to trust people without a good reason: [. . .] trust can't be taken on trust.

Table 2. Examples of agent correctness

- algorithm, game-rule, and protocol compliance
- good book-keeping in audit and control of transactions
- fairness in the scheduling of legitimately requested services
- liveness in the sense of absence of crash, e.g., that the agent, say a, has always eventually responded to a ping from some other agent, say b, which can be expressed with a Linear-Temporal Logic [28, 27, 68, 8] formula as follows:

  $$a\ \mathsf{correct}\ b := \overline{\Box}(b\ \mathsf{pings}\ a \to \Diamond(a\ \mathsf{pongs}\ b)),$$

  where a $\mathsf{correct}$ b reads as "a is correct as far as b is concerned", and $\overline{\Box}$ is the "so-far"-modality (i.e., the modality "at all times in the past including the present") and $\Diamond$ the "eventually"-modality here
- access-control compliance [4, Chapter 4], e.g., that a has always been authorised by b and authenticated to access a given resource $r \in \mathcal{R}$, which can be expressed with an LTL formula as follows ($\mathcal{R}$ is finite):

  $$a\ \mathsf{correct}\ b := \bigwedge_{r\in\mathcal{R}} \overline{\Box} \begin{pmatrix} \begin{pmatrix} a\ \mathsf{accesses}\ r\ \land \\ \mathsf{canAuthoriseToAccess}(b,a,r) \end{pmatrix} \to \\ \begin{pmatrix} a\ \mathsf{authenticatedForAccessing}\ r \\ \land\ \mathsf{authorisedToAccess}(b,a,r) \end{pmatrix} \end{pmatrix},$$

  where abstract access could be refined w.r.t. concrete *reading* and *writing*, and authorisation w.r.t. *revocation updates*, *delegation indirections*, *security levels* [4, Chapter 8], and *security compartments* [4, Chapter 9]
- absence of cryptographic-key compromise (cf. Section 4)
- etc.

That is, we trust when we have at least belief in correctness—in which case we run the risk of mistakingly believing—or *may* trust when we have even knowledge of correctness—in which case we run no such risk. Otherwise, i.e., when we have no such belief (and absence of belief is different from disbelief), we *must* trust (we have no choice). Now at first sight, such knowledge may seem difficult to attain. Yet in our standard understanding of knowledge defined in terms of indistinguishability (an observational equivalence) of system states, an agent a simply attains knowledge about a fact ϕ in a given state s as soon as ϕ also holds in all the states that are indistinguishable from s to a (cf. Section 2). In particular, a need not—but may happen to—have control over the system.

Example 1 (Social Software). The importance of dependable distributed systems for modern society can hardly be overestimated, because *social software* [59], i.e., "the software by which society runs" (e.g., banking and commerce, health care, social networking, voting and governmental administration, etc.) runs on distributed systems. In the age of the Internet, which acts as a generator and amplifier of the virtuality of human relations, trust is crucial [16, 54, 67].

The concept of trust has at least three different aspects, namely trust *relations* and *domains,* and trust *management.* Our intuitions of them are as follows.

Trust relations An agent a trusts an agent b when a believes or even knows that b behaves correctly as far as a is concerned, independently of b's mental attitude. Hence, defining trust based on the correct behaviour of agents is more general than defining trust based on their mental attitude behind their behaviour. The reader interested in trust based on (a fixed number of kinds of) mental attitude behind agent behaviour is referred to [20], which is a substantial study of various trust relations based on belief in and knowledge of attitudes behind behaviour.

Trust domains A trust domain is a community of mutually trusting agents with the common belief in or even knowledge of the trust relationships in the community. Informally, a statement ϕ is common belief or knowledge in a community $\mathcal{C}$ when all agents in $\mathcal{C}$ believe or know that ϕ is true (call this new statement ϕ'), all believe or know that ϕ' is true (call this new statement ϕ''), all believe or know that ϕ'' is true (call this new statement ϕ'''), etc. Notice the *mutual awareness* among agents w.r.t. the commonality of their knowledge or belief, which will turn out to be computationally costly (cf. Section 3). More intuitions on common knowledge in distributed systems can be found in [30].

Trust management Trust management is (cf. [63] for a recent survey):

1. the organisation of trust relations into trust domains (compartments), i.e., the sociology;
2. the coordination of trust-building actions, i.e., the flow of trust (partial dehierarchisation and decompartmentation, e.g., by building reputation [64]).

The organisation of trust relations into trust domains requires the ability to *decide* whether or not a given relation is a trust relation, and a given domain is a trust domain. Ideally, this ability appeals to formal definitions and decidability results, in order to support the human brain with *computer-aided decision making (CADM)*, as motivated by the following example.

Example 2 (Group size and the human brain). According to [66], “150 is the cognitive limit to the number of people a human brain can maintain a coherent social relationship with.” “More generally, there are several layers of natural human group size that increase with a ratio of approximately three: 5, 15, 50, 150, 500, and 1,500”. And “[a]s group sizes grow across these boundaries, they have more externally imposed infrastructure—and more formalized security systems.”

The motivation for formal definitions now follows from the assumption that trust is a fundamental element of any coherent social relationship; and the motivation for CADM (requiring decidability results) additionally from the age-old desideratum to extend the cognitive limit of the human brain; cf. [25, Page 214]:

> Trust is the ultimate basis for all dealings that we have with other people. If you don't trust anybody with anything at all, why bother interacting with them?

It turns out that deciding trust relationships can be tractable with the aid of modern computers. However, deciding membership in trust domains, though computationally possible in theory, is computationally intractable in practice, even with the aid of clusters or grids [7], and clouds[4] of (super-)computers. What is worse: not only can we not make use of the promised power of parallel computing for deciding membership in trust domains in general, but also in particular when the candidate domains are the computing complexes themselves!

Example 3 (Cloud Computing). According to [36], in cloud computing, "users are universally required to accept the underlying premise of trust. In fact, some have conjectured that trust is the biggest concern facing cloud computing. Nowhere is the element of trust more apparent than in security, and many believe trust and security to be synonymous." Also according to [53]: "The growing importance of cloud computing makes it increasingly imperative that we grapple with the meaning of trust in the cloud and how the customer, provider, and society in general establish that trust." For more motivation, see [14, 37].

Indeed, the automatic validation of the underlying premise of cloud computing, i.e., trust, is an intractably big concern for the clouds themselves, as indicated above. However, the validation of trust can of course still be a tractably big concern for the relations between the cloud members. Anyway, what remains formally elusive is the *general and declarative meaning of trust.* As a matter of fact, the vast majority of the research literature on trust focuses on *how* to (operationally) establish and maintain trust of some, and sometimes even quantitative, form (e.g., with protocols, recommendation/reputation systems, reference monitors, trusted computing bases [29, 47], etc.), but without actually defining *what* trust of that form (declaratively) means. And the very few works that do attempt declarative—not to speak of general—definitions of trust do not provide insights in the tractability of trust domains, or applications to security such as trust in cryptographic-key management (cf. Section 6).

The bottom line is that declaratively defining the meaning of trust and obtaining estimates of the tractability of trust can be difficult. Yet formally defining trust in terms of (declarative) belief in or knowledge of behavioural correctness turns out to be natural (and generic), since humans often naturally refer to these or similar notions when informally explaining what they mean by trust. In distributed or multi-agent systems, attaining an *individual* consciousness of trust in terms of belief (weak trust) or knowledge (strong trust) from agent to agent can be computationally *tractable.* Whereas attaining a *collective* consciousness (mutual awareness) of trust in terms of *common* belief or knowledge within a greater domain (e.g., a cluster, cloud, or other collective) of agents is computationally *in*tractable. Computationally speaking, *collective trust does not scale.*

Trust domains should be family-sized, so to speak.

(Typically, the size of human families is below 15, which is still tractable.)

[4] Cloud computing is the automated outsourcing of IT infrastructure, platforms, and applications for data storage and calculation routines into evolving opaque clouds of anonymous computing units [26].

1.2 Goal

Our goal is

1. to provide generic declarative definitions of
   (a) individual and collective trust relations between interacting agents and agent collections (cf. Figure 1 and 2, respectively),
   (b) trust domains of trust-related agents (cf. Figure 3);
2. to reveal the logical structure of trust relations and domains (cf. Figure 1–3), and to demonstrate practical implementability (cf. Theorem 1);
3. to obtain computational complexity results for deciding potential and actual trust relationships and membership in trust domains (cf. Table 4);
4. to obtain compositionality and scalability results for trust domains (cf. Theorem 3 and Corollary 3, respectively);
5. to provide a computational design pattern for automatically building up bounded strong trust domains, e.g., in social networks (cf. Table 8);
6. to demonstrate the utility of our framework by showing how easy it is to instantiate it in five difficult cryptographic applications of trust, namely:
   (a) Access Control (cf. Table 2)
   (b) Trusted Third Parties (cf. Section 4.1)
   (c) the Web of Trust (cf. Section 4.2)
   (d) Public-Key Infrastructures (cf. Section 4.3)
   (e) Identity-Based Cryptography (cf. Section 4.4);
7. to reveal the relation of trust with accountability (abusefree auditability), which intuitively should induce trust (cf. Section 4.5).

Contribution To the best of our knowledge, the following items are all novel: (1) general formal definitions of collective trust relations, and thus also the (in)compatibility, implicational, and transitivity results, including a Datalog-implementability result, that we obtain for them; (2) general formal definitions of trust domains; (3) general complexity results for trust relations as well as trust domains; (4) a positive (negative) compositionality result for strong (weak) trust domains; (5) a negative scalability result for trust domains in general; (6) a computational design pattern for building up strong trust domains; and (7) a generic formalisation of trust that can be instantiated in Access Control, TTPs, the Web of Trust, PKIs, and ID-Based Cryptography and that happens to be induced by (an equally generic) formalisation of accountability. The resulting structural and (in)tractability insights are of practical importance for accountable access control and cryptographic-key management (e.g., the design of [inter]national PKIs such as those required by credit-card transactions and ePassports [44]), and could may well be important for computing clouds viewed as trust domains.

1.3 Methodology

Our methodology is to develop our formal definitions for trust relations and trust domains in a generic framework that is a semantically defined, standard modal logic of belief and knowledge (cf. Section 2). In that, we are interested in the *descriptive* (as opposed to deductive) use of an off-the-shelf, general-purpose (as opposed to special-purpose)[5] logic that is as simple as possible and as complex as necessary—both syntactically and semantically as well as computationally. Our *defining principle* for weak and strong trust is belief in and knowledge of agent correctness, respectively. We then derive our complexity results for deciding trust relationships and membership in trust domains by reduction to known results for the complexity of belief and knowledge (cf. Section 3). In spite of the substantial practical significance of our theoretical results, their derivation is quite simple (which increases their value), thanks to our modal logical framework. The difficulty was to find what we believe to be an interesting formal point of view on trust, which happens to be modal. Other points of view have, to the best of our knowledge, not resulted in even one of our contributions.

2 Formal definitions

We develop our formal definitions of trust relations and trust domains in a generic framework that is a semantically defined, standard modal logic of common belief and knowledge. The logic is *parametric* in agent correctness, to be instantiated for each considered distributed system (e.g., Table 2 and Section 4.2–4.4).

Let S designate the considered distributed system (e.g., of sub-systems).

Definition 1 (Framework). *Let*

- $\mathcal{A}$ *designate an arbitrary non-empty finite set of unique* agent names[6] *a, b, c, etc.*
- $\mathcal{C} \subseteq \mathcal{A}$ *denote (finite and not necessarily disjoint) communities of agents (referred to by their name);*
- $\mathcal{P} := \{\ a\ \mathsf{correct}\ b \mid a, b \in \mathcal{A}\ \}$ *designate our (finite) set of* atomic propositions *P for referring to agent correctness (finiteness is crucial for complexity);*
- $\mathcal{L} \ni \phi\ ::=\ P \mid \neg\phi \mid \phi \wedge \phi \mid \mathsf{CB}_{\mathcal{C}}(\phi) \mid \mathsf{CK}_{\mathcal{C}}(\phi)$ *designate our modal language of formulae ϕ, with $\mathsf{CB}_{\mathcal{C}}(\phi)$ for "it is common belief in the community $\mathcal{C}$ that ϕ", and $\mathsf{CK}_{\mathcal{C}}(\phi)$ for "it is common knowledge in the community $\mathcal{C}$ that ϕ."*

(Note that we could have indexed the relational symbol '$\mathsf{correct}$' with a parameter ranging over a finite number of trust issues *x and still have conserved all our complexity results in Section 3. Yet we have not done so for the sake of the simplicity of our exposition. The finiteness of the number of issues is crucial. Otherwise, e.g., when the issues are not mere logical constants but expressions in a formal language, not only trust domains but also trust relations become computationally intractable.) Then given the set $\mathcal{S}$ (the state space) of* system states *s induced by S (e.g., via a reachability or, in modal jargon, temporal*

[5] e.g., the famous BAN-logic, which uses but does not define trust

[6] i.e., agent names injectively map to agents (agent names are *identifiers* here)

Table 3. Satisfaction relation

$(\mathfrak{S}, \mathcal{V}), s \models P$	:iff $s \in \mathcal{V}(P)$
$(\mathfrak{S}, \mathcal{V}), s \models \neg\phi$	:iff not $(\mathfrak{S}, \mathcal{V}), s \models \phi$
$(\mathfrak{S}, \mathcal{V}), s \models \phi \wedge \phi'$	:iff $(\mathfrak{S}, \mathcal{V}), s \models \phi$ and $(\mathfrak{S}, \mathcal{V}), s \models \phi'$
$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CB}_{\mathcal{C}}(\phi)$	:iff for all $s' \in \mathcal{S}$, if $s \ \mathrm{D}_{\mathcal{C}}^{+} \ s'$ then $(\mathfrak{S}, \mathcal{V}), s' \models \phi$
$(\mathfrak{S}, \mathcal{V}), s \models \mathsf{CK}_{\mathcal{C}}(\phi)$	:iff for all $s' \in \mathcal{S}$, if $s \ \mathrm{E}_{\mathcal{C}}^{*} \ s'$ then $(\mathfrak{S}, \mathcal{V}), s' \models \phi$

accessibility relation)[7]*, we define the* satisfaction relation $\models$ *of our framework in Table 3. There,*

- *":iff" abbreviates "by definition, if and only if";*
- $(\mathfrak{S}, \mathcal{V})$ *designates a (modal)* model *of our framework;*
- $\mathfrak{S} := (\mathcal{S}, \{\mathrm{D}_a\}_{a \in \mathcal{A}}, \{\mathrm{E}_a\}_{a \in \mathcal{A}})$ *designates the (modal)* frame *with appropriate (for the system S)*
  - *serial*[8]*, transitive, and Euclidean*[9] *relations* $\mathrm{D}_a \subseteq \mathcal{S} \times \mathcal{S}$ *of* doxastic accessibility *(used for defining* belief*),*
  - *equivalence relations* $\mathrm{E}_a \subseteq \mathcal{S} \times \mathcal{S}$ *of* epistemic accessibility *(e.g., state* indistinguishability*, used for defining* knowledge*),*

  such that $\mathrm{D}_a \subseteq \mathrm{E}_a$ *for any* $a \in \mathcal{A}$*;*
- $\mathcal{V} : \mathcal{P} \to 2^{\mathcal{S}}$ *designates the* valuation function *(returning for every* $P \in \mathcal{P}$ *the set of states where P is true) to be defined according to the appropriate notion of agent correctness for the system S (e.g., see Section 4);*
- $\mathrm{D}_{\mathcal{C}}^{+}$ *designates the transitive closure of* $\bigcup_{a \in \mathcal{C}} \mathrm{D}_a$*;*
- $\mathrm{E}_{\mathcal{C}}^{*}$ *designates the reflexive transitive closure of* $\bigcup_{a \in \mathcal{C}} \mathrm{E}_a$*.*

Note that defining (common) belief and knowledge abstractly with a serial, transitive, and Euclidean relation, and an equivalence relation, respectively, has emerged as a common practice that gives greater generality over more concrete approaches [52, Section 7.1]: the concrete definitions of the accessibility relations can be freely determined for a given distributed system provided they comply with the prescribed, characteristic properties. Typically, these definitions involve the projection of global states onto agents' local views [24]. For example, let $a \in \mathcal{A}$, and let π_a designate such a projection (function) for a. Then, epistemic accessibility in the sense of state indistinguishability can be defined such that for all $s, s' \in \mathcal{S}$,

$$s \ \mathrm{E}_a \ s' \text{ :iff } \pi_a(s) = \pi_a(s'),$$

which guarantees that E_a is an equivalence relation. Specific applications, might require additional constraints. For example, we might want to stipulate the a

[7] For example, suppose that there is a set $\mathcal{S}_i$ of initial states for every system S, T designates the system's reachability or, synonymously, temporal accessibility relation, and T^* designates the reflexive transitive closure of T. Then, $\mathcal{S}$ is induced by S in the sense that $\mathcal{S} := \{ \ s \mid \text{there is } s_i \in \mathcal{S}_i \text{ such that } s_i \ \mathrm{T}^* \ s \ \}$.

[8] for all $s \in \mathcal{S}$, there is $s' \in \mathcal{S}$ s.t. $s \ \mathrm{D}_a \ s'$

[9] for all $s, s', s'' \in \mathcal{S}$, if $s \ \mathrm{D}_a \ s'$ and $s \ \mathrm{D}_a \ s''$ then $s' \ \mathrm{D}_a \ s''$

posteriori constraint that

$$\text{if } s \ \mathrm{E}_a \ s' \text{ then for all } b \in \mathcal{A},\ (\mathfrak{S}, \mathcal{V}), s \models b \ \mathsf{correct}\ a \text{ iff } (\mathfrak{S}, \mathcal{V}), s' \models b \ \mathsf{correct}\ a.$$

Doxastic accessibility $\mathrm{D}_a \subseteq \mathrm{E}_a$ can be defined from E_a by weakening the reflexivity of E_a to seriality as appropriate for the considered application.

Further note the following macro-definitions: $\phi \vee \phi' := \neg(\neg\phi \wedge \neg\phi')$, $\top := a \ \mathsf{correct}\ a \vee \neg(a \ \mathsf{correct}\ a)$, $\bot := \neg\top$, $\phi \rightarrow \phi' := \neg\phi \vee \phi'$, $\phi \leftrightarrow \phi' := (\phi \rightarrow \phi') \wedge (\phi' \rightarrow \phi)$, $\mathsf{B}_a(\phi) := \mathsf{CB}_{\{a\}}(\phi)$ (for "a believes that ϕ"), $\mathsf{K}_a(\phi) := \mathsf{CK}_{\{a\}}(\phi)$ (for "a knows that ϕ"), $\mathsf{CD}_\mathcal{C}(\phi) := \mathsf{CK}_\mathcal{C}(\phi) \vee \mathsf{CK}_\mathcal{C}(\neg\phi)$ (for "$\mathcal{C}$ can epistemically decide ϕ"), $\mathsf{D}_a(\phi) := \mathsf{CD}_{\{a\}}(\phi)$, $\mathsf{CN}_\mathcal{C}(\phi) := \neg\mathsf{CB}_\mathcal{C}(\phi) \wedge \neg\mathsf{CB}_\mathcal{C}(\neg\phi)$ (for "$\mathcal{C}$ is doxastically neutral with respect to ϕ"), and $\mathsf{N}_a(\phi) := \mathsf{CN}_{\{a\}}(\phi)$.

Likewise, but more interestingly, we now obtain our generic declarative definitions of trust as mere macro-definitions, i.e., as simple syntactic constructions of semantically defined building blocks, as depicted by Figure 1–3 (conjunction over $\mathcal{C} = \emptyset$ being $\top$). The common organisational principle in these figures is the classic Square of Oppositions [73]. (The principle is applicable, because knowledge and belief modalities are universal quantifiers in disguise, cf. Table 3.) Recall that neither pairs of contraries nor pairs of subcontraries are necessarily contradictory. However when indeed contradictory, we have marked them explicitly as such together with the name of the justifying logical law in parenthesis (**D** or **T**, recalled in the farther Fact 1 and 2, respectively). Arrows signify the above macro-defined material conditional, and are either justified with a logical law and/or the inclusion of doxastic in epistemic accessibility, or not at all when the material conditional holds by propositional logic in a single step. Source states (i.e., states with only outgoing arrows) are distinguished by means of grey-shading, and sink states (i.e., states with only incoming arrows) are so by means of double boxing. Observe that the anchor of all the definitions in

1. these three successive figures is our, like the NSA's [4], synonymic definition of *trustworthiness as correctness* for individual agents in Figure 1;
2. Figure 2 is the conjunctive lifting of correctness for individual agents from Figure 1 to correctness for agent collections in Figure 2;
3. Figure 3 is the definition of a trust domain simply as the reflexive instance of the corresponding collective trust relation in Figure 2.

Notice that the absence of the **D**-law for common belief within agent collections strictly greater than the singleton set (cf. Fact 1), corresponds to the absence of **D**-arrows in Figure 2 for arbitrary finite collections, and leads the trust and the distrust state to become sink states. This absence is consecutively inherited by Figure 3. We recall that the further scientific exploration of distrust and mistrust was strongly encouraged by [50]. The reader is invited not to confuse *dis*trust (using *verb-phrase* negation) with *absence of* trust, (using *sentence* negation).

Remark 1 (Trust & risk [57]). In the case of [dis]trust, which is belief-based, we run the risk of a wrong (i.e., false positive [negative]) apprehension of agent correctness. In the case of strong (dis)trust, we do not run this risk, because strong (dis)trust is knowledge-based, which intuitively corresponds to a right apprehension. So belief-based trust is lack of trust, which "[. . .] is simply a risk, and that can sometimes be handled by standard risk-management techniques such as insurance" [25, Section 13.2.1]. Here we would want to insure beliefs in [in]correctness. As usual, risk quantifies as the cost of an event (here the cost of mistakingly believing in agent correctness) times the probability of the occurrence of the event.

Theorem 1 (Datalog implementability). *The logical structure of the trust relations and trust domains revealed by Figure 1–3 is implementable axiomatically as a (terminating) Datalog program [55].*

Proof. An adequate (sound and complete) and consistent, though non-minimal (and thus more efficient!) axiomatisation is obtained by simply transcribing each

– arrow as a Datalog rule, e.g., the arrow

  $$a \ \mathsf{doestrust}\ b \rightarrow b \ \mathsf{trustworthy}\ a$$

  is transcribed as the Datalog rule

  ```
  trustworthy(a,b) :- doestrust(a,b).
  ```

– (undirected) line labelled with "contradictions" or "contradictory" between a given pair of vertices as a Datalog rule with as body the logical conjunction of the two vertices and as head `contradiction`; for example, the line between a doestrust b and a doesdistrust b is transcribed as the Datalog rule

  ```
  contradiction :- doestrust(a,b),doesdistrust(a,b).
  ```

– undirected line labelled with "subcontraries" or "subalterns" between a given pair of vertices as a Datalog rule with as body the logical conjunction of the two vertices and as head `subcontrary` and `subaltern`, respectively; for example, the line between a doesnotdistrust b and a doesnottrust b is transcribed as the Datalog rule

  ```
  subcontrary :- doesnotdistrust(a,b),doesnottrust(a,b).
  ```

The axiomatisation is consistent thanks to the definitional grounding of each trust predicate in doxastic or epistemic logic. However if one wishes, one can also simply forget this grounding, and use the predicates abstractly, i.e., without their modal meaning but with their practical implementation as Datalog programs.

Application-specific axiomatisations (relational trust theories) can be obtained by extending each Datalog program with a finite set (a relational data base) of additional axioms (ground facts) stipulating which specific relations and domains are supposed to hold between and collect up which concrete agents, respectively. Answers to queries as to which relations and domains are implied by each specific data base can then be efficiently computed by the Datalog engine [18]. In the sequel of this paper, we shall focus on the positive notions of (weak) trusts and (strong) doestrust rather than on the other, negative notions, for the ethical reason that the desiderata for well-intended engineering are the positive notions.

Fig. 1. Related squares of *individual* trust-relation oppositions

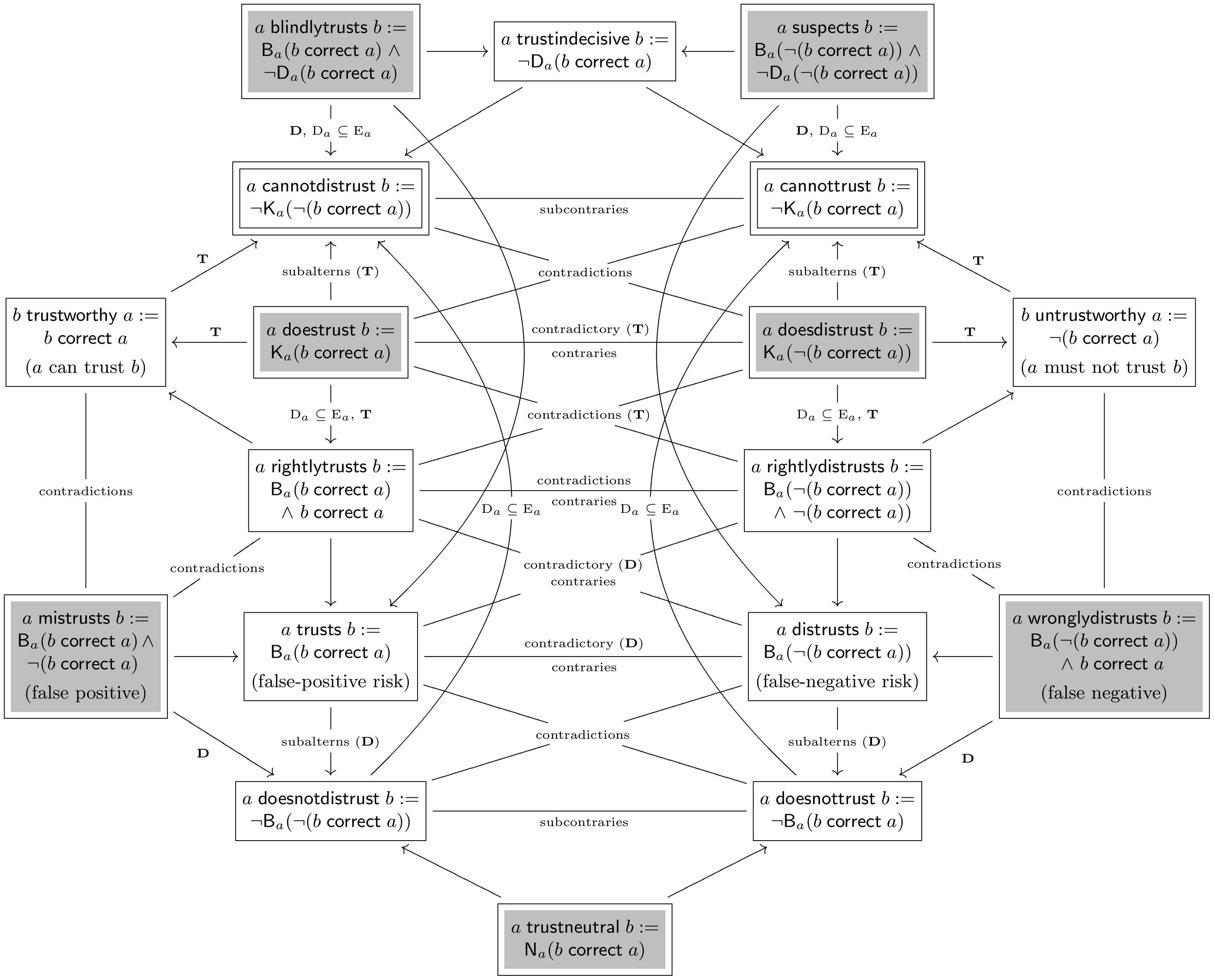
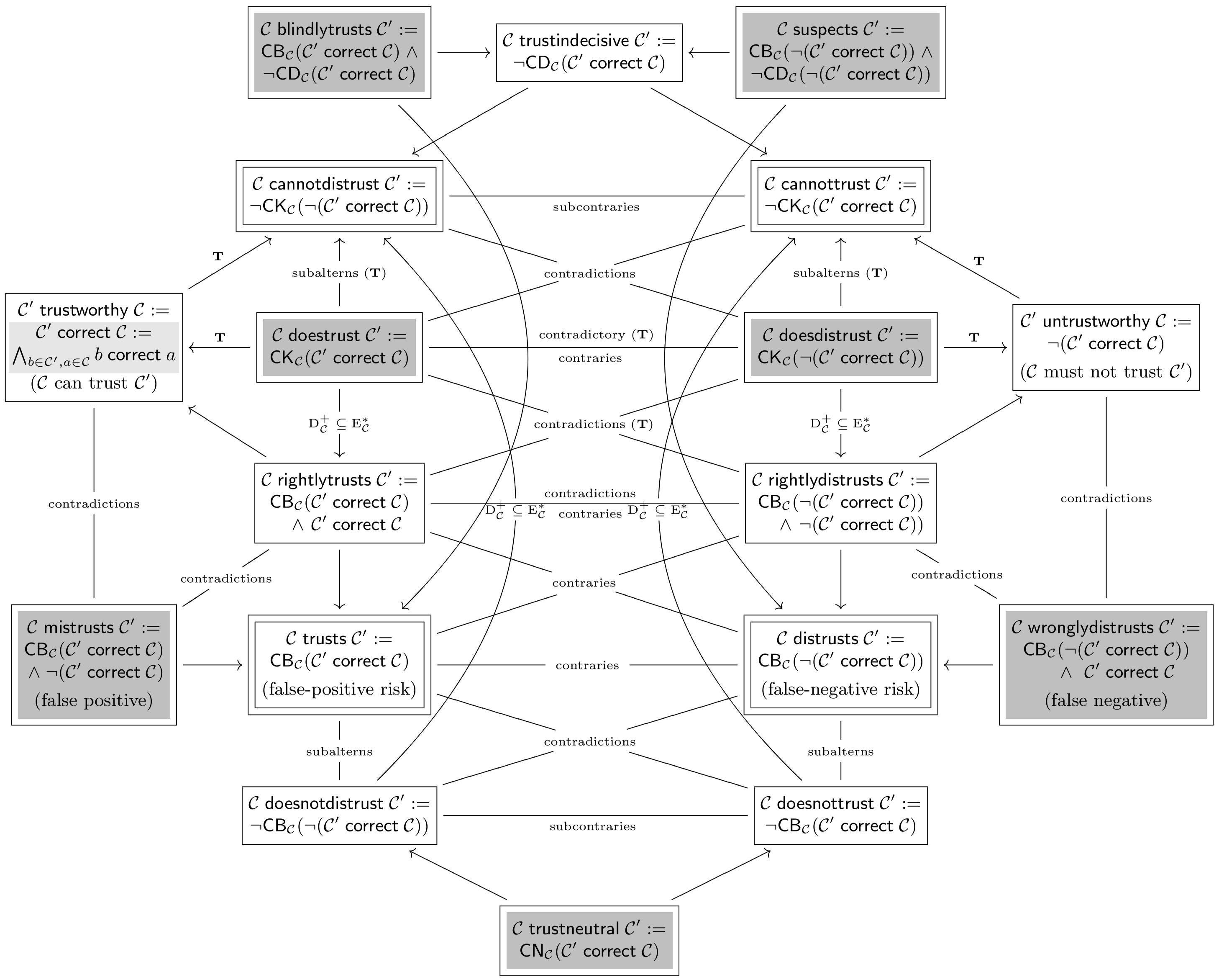
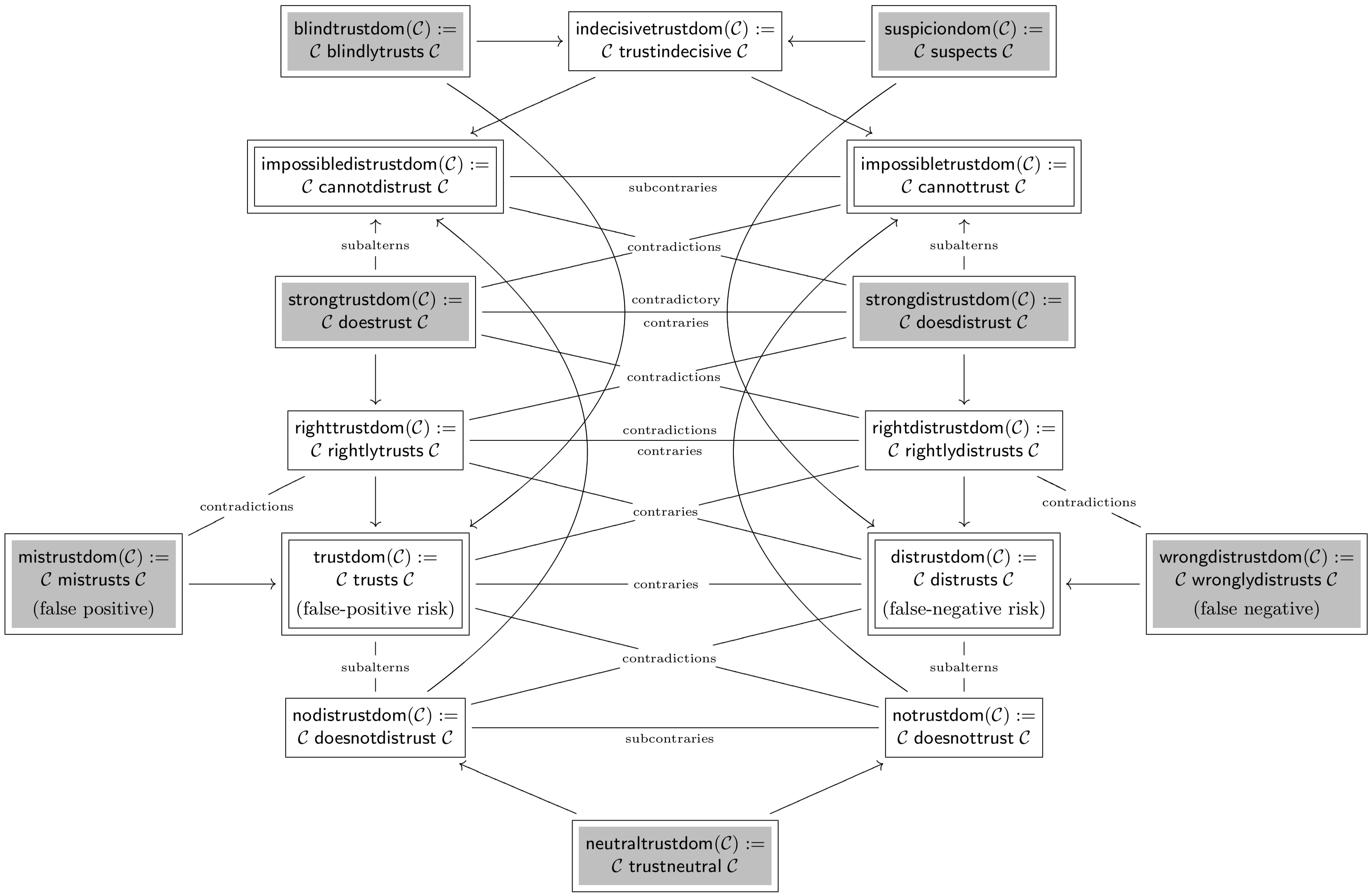

a blindlytrusts b := $\mathsf{B}_a(b \text{ correct } a) \wedge \neg\mathsf{D}_a(b \text{ correct } a)$
a trustindecisive b := $\neg\mathsf{D}_a(b \text{ correct } a)$
a suspects b := $\mathsf{B}_a(\neg(b \text{ correct } a)) \wedge \neg\mathsf{D}_a(\neg(b \text{ correct } a))$
$\mathbf{D}$, $\mathrm{D}_a \subseteq \mathrm{E}_a$
$\mathbf{D}$, $\mathrm{D}_a \subseteq \mathrm{E}_a$
a cannotdistrust b := $\neg\mathsf{K}_a(\neg(b \text{ correct } a))$
subcontraries
a cannottrust b := $\neg\mathsf{K}_a(b \text{ correct } a)$
$\mathbf{T}$
subalterns ($\mathbf{T}$)
contradictions
subalterns ($\mathbf{T}$)
$\mathbf{T}$
b trustworthy a := b correct a (a can trust b)
$\mathbf{T}$
a doestrust b := $\mathsf{K}_a(b \text{ correct } a)$
contradictory ($\mathbf{T}$)
contraries
a doesdistrust b := $\mathsf{K}_a(\neg(b \text{ correct } a))$
$\mathbf{T}$
b untrustworthy a := $\neg(b \text{ correct } a)$ (a must not trust b)
$\mathrm{D}_a \subseteq \mathrm{E}_a$, $\mathbf{T}$
contradictions ($\mathbf{T}$)
$\mathrm{D}_a \subseteq \mathrm{E}_a$, $\mathbf{T}$
contradictions
a rightlytrusts b := $\mathsf{B}_a(b \text{ correct } a) \wedge b \text{ correct } a$
contradictions
contraries
$\mathrm{D}_a \subseteq \mathrm{E}_a$
$\mathrm{D}_a \subseteq \mathrm{E}_a$
a rightlydistrusts b := $\mathsf{B}_a(\neg(b \text{ correct } a)) \wedge \neg(b \text{ correct } a))$
contradictions
contradictions
contradictory ($\mathbf{D}$)
contraries
contradictions
a mistrusts b := $\mathsf{B}_a(b \text{ correct } a) \wedge \neg(b \text{ correct } a)$ (false positive)
a trusts b := $\mathsf{B}_a(b \text{ correct } a)$ (false-positive risk)
contradictory ($\mathbf{D}$)
contraries
a distrusts b := $\mathsf{B}_a(\neg(b \text{ correct } a))$ (false-negative risk)
a wronglydistrusts b := $\mathsf{B}_a(\neg(b \text{ correct } a)) \wedge b \text{ correct } a$ (false negative)
$\mathbf{D}$
subalterns ($\mathbf{D}$)
contradictions
subalterns ($\mathbf{D}$)
$\mathbf{D}$
a doesnotdistrust b := $\neg\mathsf{B}_a(\neg(b \text{ correct } a))$
subcontraries
a doesnottrust b := $\neg\mathsf{B}_a(b \text{ correct } a)$
a trustneutral b := $\mathsf{N}_a(b \text{ correct } a)$

Fig. 2. Related squares of *collective* trust-relation oppositions



$\mathcal{C}$ blindlytrusts $\mathcal{C}'$:= CB$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$) ∧ ¬CD$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)
$\mathcal{C}$ trustindecisive $\mathcal{C}'$:= ¬CD$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)
$\mathcal{C}$ suspects $\mathcal{C}'$:= CB$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$)) ∧ ¬CD$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$))
$\mathcal{C}$ cannotdistrust $\mathcal{C}'$:= ¬CK$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$))
subcontraries
$\mathcal{C}$ cannottrust $\mathcal{C}'$:= ¬CK$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)
T
contradictions
subalterns (T)
subalterns (T)
T
$\mathcal{C}'$ trustworthy $\mathcal{C}$:= $\mathcal{C}'$ correct $\mathcal{C}$:= $\bigwedge_{b\in\mathcal{C}',a\in\mathcal{C}}$ b correct a ($\mathcal{C}$ can trust $\mathcal{C}'$)
T
$\mathcal{C}$ doestrust $\mathcal{C}'$:= CK$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)
contradictory (T)
contraries
$\mathcal{C}$ doesdistrust $\mathcal{C}'$:= CK$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$))
T
$\mathcal{C}'$ untrustworthy $\mathcal{C}$:= ¬($\mathcal{C}'$ correct $\mathcal{C}$) ($\mathcal{C}$ must not trust $\mathcal{C}'$)
$\mathrm{D}^+_\mathcal{C} \subseteq \mathrm{E}^*_\mathcal{C}$
contradictions (T)
$\mathrm{D}^+_\mathcal{C} \subseteq \mathrm{E}^*_\mathcal{C}$
$\mathcal{C}$ rightlytrusts $\mathcal{C}'$:= CB$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$) ∧ $\mathcal{C}'$ correct $\mathcal{C}$
contradictions
$\mathrm{D}^+_\mathcal{C} \subseteq \mathrm{E}^*_\mathcal{C}$
contraries
$\mathrm{D}^+_\mathcal{C} \subseteq \mathrm{E}^*_\mathcal{C}$
$\mathcal{C}$ rightlydistrusts $\mathcal{C}'$:= CB$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$)) ∧ ¬($\mathcal{C}'$ correct $\mathcal{C}$))
contradictions
contradictions
contradictions
contraries
contradictions
$\mathcal{C}$ mistrusts $\mathcal{C}'$:= CB$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$) ∧ ¬($\mathcal{C}'$ correct $\mathcal{C}$) (false positive)
$\mathcal{C}$ trusts $\mathcal{C}'$:= CB$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$) (false-positive risk)
contraries
$\mathcal{C}$ distrusts $\mathcal{C}'$:= CB$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$)) (false-negative risk)
$\mathcal{C}$ wronglydistrusts $\mathcal{C}'$:= CB$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$)) ∧ $\mathcal{C}'$ correct $\mathcal{C}$ (false negative)
subalterns
contradictions
subalterns
$\mathcal{C}$ doesnotdistrust $\mathcal{C}'$:= ¬CB$_\mathcal{C}$(¬($\mathcal{C}'$ correct $\mathcal{C}$))
subcontraries
$\mathcal{C}$ doesnottrust $\mathcal{C}'$:= ¬CB$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)
$\mathcal{C}$ trustneutral $\mathcal{C}'$:= CN$_\mathcal{C}$($\mathcal{C}'$ correct $\mathcal{C}$)

Fig. 3. Related squares of *trust-domain* oppositions


blindtrustdom($\mathcal{C}$) := $\mathcal{C}$ blindlytrusts $\mathcal{C}$
indecisivetrustdom($\mathcal{C}$) := $\mathcal{C}$ trustindecisive $\mathcal{C}$
suspiciondom($\mathcal{C}$) := $\mathcal{C}$ suspects $\mathcal{C}$
impossibledistrustdom($\mathcal{C}$) := $\mathcal{C}$ cannotdistrust $\mathcal{C}$
subcontraries
impossibletrustdom($\mathcal{C}$) := $\mathcal{C}$ cannottrust $\mathcal{C}$
subalterns
contradictions
subalterns
strongtrustdom($\mathcal{C}$) := $\mathcal{C}$ doestrust $\mathcal{C}$
contradictory
contraries
strongdistrustdom($\mathcal{C}$) := $\mathcal{C}$ doesdistrust $\mathcal{C}$
contradictions
righttrustdom($\mathcal{C}$) := $\mathcal{C}$ rightlytrusts $\mathcal{C}$
contradictions
contraries
rightdistrustdom($\mathcal{C}$) := $\mathcal{C}$ rightlydistrusts $\mathcal{C}$
contradictions
contraries
contradictions
mistrustdom($\mathcal{C}$) := $\mathcal{C}$ mistrusts $\mathcal{C}$ (false positive)
trustdom($\mathcal{C}$) := $\mathcal{C}$ trusts $\mathcal{C}$ (false-positive risk)
contraries
distrustdom($\mathcal{C}$) := $\mathcal{C}$ distrusts $\mathcal{C}$ (false-negative risk)
wrongdistrustdom($\mathcal{C}$) := $\mathcal{C}$ wronglydistrusts $\mathcal{C}$ (false negative)
subalterns
contradictions
subalterns
nodistrustdom($\mathcal{C}$) := $\mathcal{C}$ doesnotdistrust $\mathcal{C}$
subcontraries
notrustdom($\mathcal{C}$) := $\mathcal{C}$ doesnottrust $\mathcal{C}$
neutraltrustdom($\mathcal{C}$) := $\mathcal{C}$ trustneutral $\mathcal{C}$

Remark 2 (Trust graphs). Weak and strong trust domains can be represented as *complete* sub-graphs (which when maximal are *cliques*) of graphs of trust and strong-trust relationships—bearing common belief and knowledge, respectively. The trust and strong-trust graph of a pointed modal model $(\mathfrak{S}, \mathcal{V}), s$ is

$$\langle \mathcal{A}, \{\ (a, b) \in \mathcal{A} \times \mathcal{A} \mid (\mathfrak{S}, \mathcal{V}), s \models a \ \mathsf{relationship}\ b\ \} \rangle,$$

where $\mathsf{relationship} \in \{\mathsf{trusts}, \mathsf{doestrust}\}$. It is conceivable to take trust graphs as accessibility relations of a modal language for speaking *about* trust relations. Indeed, the majority of approaches to trust actually adopts trust relations as *given*—as trust graphs. Whereas we here actually *define* and thus explicate trust relations and domains in terms of belief in and knowledge of agent correctness.

We could also define *weak-strong* and *strong-weak* trust domains, i.e., as the common knowledge of weak and the common belief in strong trust relations, respectively. The difference between weak and strong trust is induced by the difference between belief and knowledge, respectively: weak trust possibly is wrong (i.e., mistaken belief), whereas strong trust necessarily is right (i.e., truthful belief). Social network systems furnish evidence for the adequacy of defining trust domains in terms of common knowledge or at least belief. As a matter of fact, the enumeration of "friends" on a member page in such systems constitutes a public announcement to the readers of that page who are logged in, who see all logged-in readers, etc. (cf. also [69]). And it is common knowledge in the community of (dynamic) epistemic logicians that public announcements of (verifiable) elementary facts induce common knowledge within the addressed public (cf. [71] for a public announcement). So, suppose that you are a member of Facebook, and $\mathcal{C}$ designates the set consisting of you and those of your "friends" that are enumerated on your member page. Further, fix the current moment in time, and call it s. (We may talk about time here; see Footnote 7.) Then the formula $\bigwedge_{a,b \in \mathcal{C}} a\ \mathsf{doestrust}\ b \leftrightarrow \mathsf{strongtrustdom}(\mathcal{C})$ is true at s (with no outermost $\mathsf{CK}_{\mathcal{C}}$ operator on the left since the aforementioned public announcement implies it). The formula is a (bi-)conditional because we have not fixed the notion of agent correctness for Facebook (*they* should). Note that the trust relations between you and your "friends" are symmetric, because each "friend" had to give their consent for having the privilege of being enumerated as such on your member page in Facebook. In general, each agent's notion of correctness for other agents should be publicly announced and thus become common knowledge, in order for everybody to be able to choose whether or not to comply with these notions.

Definition 2 (Truth & Validity [10]). *The formula ϕ is* true *(or* satisfied*) in the model $(\mathfrak{S}, \mathcal{V})$ at the state $s \in \mathcal{S}$:iff $(\mathfrak{S}, \mathcal{V}), s \models \phi$. The formula ϕ is* satisfiable *in the model $(\mathfrak{S}, \mathcal{V})$:iff there is $s \in \mathcal{S}$ such that $(\mathfrak{S}, \mathcal{V}), s \models \phi$. The formula ϕ is* globally true *(or* globally satisfied*) in the model $(\mathfrak{S}, \mathcal{V})$, written $(\mathfrak{S}, \mathcal{V}) \models \phi$, :iff for all $s \in \mathcal{S}$, $(\mathfrak{S}, \mathcal{V}), s \models \phi$. The formula ϕ is* satisfiable *:iff there is a model $(\mathfrak{S}, \mathcal{V})$ and a state $s \in \mathcal{S}$ such that $(\mathfrak{S}, \mathcal{V}), s \models \phi$. The formula ϕ is* valid*, written $\models \phi$, :iff for all models $(\mathfrak{S}, \mathcal{V})$, $(\mathfrak{S}, \mathcal{V}) \models \phi$.*

Fact 1 (Common belief) *Being defined in terms of a serial, transitive, and Euclidean relation,* $\mathsf{CB}_{\mathcal{C}}$ *is K45 and* $\mathsf{CB}_{\{a\}}$ *KD45 for any* $a \in \mathcal{C} \subseteq \mathcal{A}$*, i.e.:*

K: $\models \mathsf{CB}_{\mathcal{C}}(\phi \to \phi') \to (\mathsf{CB}_{\mathcal{C}}(\phi) \to \mathsf{CB}_{\mathcal{C}}(\phi'))$ *(Kripke's law)*
D: $\models \mathsf{CB}_{\{a\}}(\phi) \to \neg\mathsf{CB}_{\{a\}}(\neg\phi)$ *(consistency of beliefs, seriality)*
4: $\models \mathsf{CB}_{\mathcal{C}}(\phi) \to \mathsf{CB}_{\mathcal{C}}(\mathsf{CB}_{\mathcal{C}}(\phi))$ *(positive introspection, transitivity)*
5: $\models \neg\mathsf{CB}_{\mathcal{C}}(\phi) \to \mathsf{CB}_{\mathcal{C}}(\neg\mathsf{CB}_{\mathcal{C}}(\phi))$ *(negative introspection, Euclideanness)*
N: *if* $\models \phi$ *then* $\models \mathsf{CB}_{\mathcal{C}}(\phi)$ *(necessitation).*

Further, let $\mathsf{EB}_{\mathcal{C}}(\phi) := \bigwedge_{a \in \mathcal{C}} \mathsf{B}_a(\phi)$ *("everybody in* $\mathcal{C}$ *believes that* ϕ*"). Then:*

- $\models \mathsf{CB}_{\mathcal{C}}(\phi) \to \mathsf{EB}_{\mathcal{C}}(\phi)$
- $\models \mathsf{CB}_{\mathcal{C}}(\phi) \to \mathsf{EB}_{\mathcal{C}}(\mathsf{CB}_{\mathcal{C}}(\phi))$
- $\models \mathsf{CB}_{\mathcal{C}}(\phi \to \mathsf{EB}_{\mathcal{C}}(\phi)) \to (\mathsf{EB}_{\mathcal{C}}(\phi) \to \mathsf{CB}_{\mathcal{C}}(\phi))$.

For details see [52, Section 7.1].

The difference between belief and knowledge is that belief possibly is wrong (cf. the **D** law), whereas knowledge necessarily is right (cf. the following **T** law).

Fact 2 (Common knowledge) *Being defined in terms of an equivalence relation,* $\mathsf{CK}_{\mathcal{C}}$ *is S5 for any* $\mathcal{C} \subseteq \mathcal{A}$*, i.e.:*

K: $\models \mathsf{CK}_{\mathcal{C}}(\phi \to \phi') \to (\mathsf{CK}_{\mathcal{C}}(\phi) \to \mathsf{CK}_{\mathcal{C}}(\phi'))$ *(Kripke's law)*
T: $\models \mathsf{CK}_{\mathcal{C}}(\phi) \to \phi$ *(truth law, reflexivity)*
4: $\models \mathsf{CK}_{\mathcal{C}}(\phi) \to \mathsf{CK}_{\mathcal{C}}(\mathsf{CK}_{\mathcal{C}}(\phi))$ *(positive introspection)*
5: $\models \neg\mathsf{CK}_{\mathcal{C}}(\phi) \to \mathsf{CK}_{\mathcal{C}}(\neg\mathsf{CK}_{\mathcal{C}}(\phi))$ *(negative introspection)*
N: *if* $\models \phi$ *then* $\models \mathsf{CK}_{\mathcal{C}}(\phi)$ *(necessitation).*

Further, let $\mathsf{EK}_{\mathcal{C}}(\phi) := \bigwedge_{a \in \mathcal{C}} \mathsf{K}_a(\phi)$ *("everybody in* $\mathcal{C}$ *knows that* ϕ*"). Then:*

- $\models \mathsf{CK}_{\mathcal{C}}(\phi) \to \mathsf{EK}_{\mathcal{C}}(\mathsf{CK}_{\mathcal{C}}(\phi))$
- $\models \mathsf{CK}_{\mathcal{C}}(\phi \to \mathsf{EK}_{\mathcal{C}}(\phi)) \to (\phi \to \mathsf{CK}_{\mathcal{C}}(\phi))$.

For details see [52, Section 7.1].

Note that depending on the properties of the employed communication lines, common knowledge may have to be pre-established, i.e., off those lines [30].

Fact 3 (Knowledge implies belief) *For all* $\mathcal{C} \subseteq \mathcal{A}$, $\models \mathsf{CK}_{\mathcal{C}}(\phi) \to \mathsf{CB}_{\mathcal{C}}(\phi)$*. In particular when* $\mathcal{C} = \{a\}$, $\models \mathsf{K}_a(\phi) \to \mathsf{B}_a(\phi)$*.*

Proof. By the fact that for all $a \in \mathcal{A}$, $\mathrm{D}_a \subseteq \mathrm{E}_a$ (cf. Definition 1).

Trust relations and trust domains can be related as follows.

Fact 4 (Trust relations & domains) *In trust domains, trust relations are* universal *(i.e., correspond to the Cartesian product on those domains). That is, for all* $a, b \in \mathcal{C}$, $\models \mathsf{trustdom}(\mathcal{C}) \to a \mathrel{\mathsf{trusts}} b$ *and* $\models \mathsf{strongtrustdom}(\mathcal{C}) \to a \mathrel{\mathsf{doestrust}} b$*.*

Proof. By inspection of definitions.

Hence in trust domains, trust relations are equivalence relations. (The universal relation contains all other relations.)

Corollary 1. *In trust domains, trust relations are $(a, b, c \in \mathcal{C} \subseteq \mathcal{A})$:*

- reflexive, *i.e.,* $\models \mathsf{trustdom}(\mathcal{C}) \rightarrow a \mathrel{\mathsf{trusts}} a$ *and* $\models \mathsf{strongtrustdom}(\mathcal{C}) \rightarrow a \mathrel{\mathsf{doestrust}} a$
- symmetric, *i.e.,* $\models \mathsf{trustdom}(\mathcal{C}) \rightarrow (a \mathrel{\mathsf{trusts}} b \rightarrow b \mathrel{\mathsf{trusts}} a)$ *and* $\models \mathsf{strongtrustdom}(\mathcal{C}) \rightarrow (a \mathrel{\mathsf{doestrust}} b \rightarrow b \mathrel{\mathsf{doestrust}} a)$
- transitive, *i.e.,* $\models \mathsf{trustdom}(\mathcal{C}) \rightarrow ((a \mathrel{\mathsf{trusts}} b \land b \mathrel{\mathsf{trusts}} c) \rightarrow a \mathrel{\mathsf{trusts}} c)$ *and* $\models \mathsf{strongtrustdom}(\mathcal{C}) \rightarrow ((a \mathrel{\mathsf{doestrust}} b \land b \mathrel{\mathsf{doestrust}} c) \rightarrow a \mathrel{\mathsf{doestrust}} c)$*).*

A more interesting condition for the transitivity of trust relations than their universality is the existence of an agent (say b) acting as a *trust reference* for the trustworthiness of another agent (say c) to a third agent (say a).

Lemma 1 (Trust reference). *For all* $a, b, c \in \mathcal{A}$:

1. $\models \underbrace{\mathsf{B}_a(c \mathrel{\mathsf{correct}} b \rightarrow c \mathrel{\mathsf{correct}} a)}_{\textit{belief in agent-correctness inclusion}} \rightarrow (\mathsf{B}_a(b \mathrel{\mathsf{doestrust}} c) \rightarrow a \mathrel{\mathsf{trusts}} c)$
2. $\models \underbrace{\mathsf{K}_a(c \mathrel{\mathsf{correct}} b \rightarrow c \mathrel{\mathsf{correct}} a)}_{\textit{knowledge of agent-correctness inclusion}} \rightarrow (\mathsf{K}_a(b \mathrel{\mathsf{doestrust}} c) \rightarrow a \mathrel{\mathsf{doestrust}} c)$*.*

Proof. By $\mathbf{T}(\mathsf{K}_b)$ and $\mathbf{K}(\mathsf{B}_a)$, and $\mathbf{T}(\mathsf{K}_b)$ and $\mathbf{K}(\mathsf{K}_a)$, respectively.

That is, b acts as a reference of c's trustworthiness to a when a believes or knows that (1) b's notion of correctness about c is included a's, and (2) b may trust c.

Remark 3. Observe that the replacement of $b \mathrel{\mathsf{doestrust}} c$ by the weaker $b \mathrel{\mathsf{trusts}} c$ in Lemma 1 does not yield a validity, due to the absence of the $\mathbf{T}$-law for B_b.

The existence of a trust reference is a sufficient condition for the *transitivity* of trust relationships—the links in *trust chains* or *paths* in trust graphs. See [47, Section 32.2.2] for an example of a chain of trust in Trusted Computing.

Proposition 1 (Trust transitivity).

1. *For all* $a, b, c \in \mathcal{A}$,

$$\models \left(\begin{array}{l} (\underbrace{\mathsf{B}_a(c \mathrel{\mathsf{correct}} b \rightarrow c \mathrel{\mathsf{correct}} a)}_{\textit{belief in agent-correctness inclusion}} \land \mathsf{B}_a(b \mathrel{\mathsf{doestrust}} c)) \\ \lor \left(\begin{array}{l} \underbrace{\mathsf{B}_a((c \mathrel{\mathsf{correct}} b \land b \mathrel{\mathsf{correct}} a) \rightarrow c \mathrel{\mathsf{correct}} a)}_{\textit{belief in agent-correctness transitivity}} \\ \land \underbrace{(\mathsf{B}_b(c \mathrel{\mathsf{correct}} b) \rightarrow \mathsf{B}_a(c \mathrel{\mathsf{correct}} b))}_{\textit{inclusion of agent-correctness belief}} \end{array} \right) \end{array} \right) \rightarrow \underbrace{((a \mathrel{\mathsf{trusts}} b \land b \mathrel{\mathsf{trusts}} c) \rightarrow a \mathrel{\mathsf{trusts}} c)}_{\textit{trust transitivity}}.$$

2. For all $a, b, c \in \mathcal{A}$,

$$\models \left(\begin{array}{l} \underbrace{(\mathsf{K}_a(c\ \mathsf{correct}\ b \to c\ \mathsf{correct}\ a) \wedge \mathsf{K}_a(b\ \mathsf{doestrust}\ c))}_{\textit{knowledge of agent-correctness inclusion}} \\ \vee \left(\begin{array}{l} \underbrace{((c\ \mathsf{correct}\ b \wedge b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a)}_{\textit{agent-correctness transitivity}} \\ \wedge\ \underbrace{(c\ \mathsf{correct}\ a \to \mathsf{K}_a(c\ \mathsf{correct}\ a))}_{\textit{agent-correctness knowledge}} \end{array} \right) \\ \vee \left(\begin{array}{l} \underbrace{\mathsf{K}_a((c\ \mathsf{correct}\ b \wedge b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a)}_{\textit{knowledge of agent-correctness transitivity}} \\ \wedge\ \underbrace{(\mathsf{K}_b(c\ \mathsf{correct}\ b) \to \mathsf{K}_a(c\ \mathsf{correct}\ b))}_{\textit{inclusion of agent-correctness knowledge}} \end{array} \right) \end{array} \right) \to$$

$$\underbrace{((a\ \mathsf{doestrust}\ b \wedge b\ \mathsf{doestrust}\ c) \to a\ \mathsf{doestrust}\ c)}_{\textit{strong-trust transitivity}}.$$

Proof. The sufficiency for trust transitivity of the condition $\mathsf{B}_a(c\ \mathsf{correct}\ b \to c\ \mathsf{correct}\ a) \wedge \mathsf{B}_a(b\ \mathsf{doestrust}\ c)$ and $\mathsf{K}_a(c\ \mathsf{correct}\ b \to c\ \mathsf{correct}\ a) \wedge \mathsf{K}_a(b\ \mathsf{doestrust}\ c)$ is a trivial consequence of Lemma 1; the sufficiency of the condition $\mathsf{B}_a((c\ \mathsf{correct}\ b \wedge b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a) \wedge (\mathsf{B}_b(c\ \mathsf{correct}\ b) \to \mathsf{B}_a(c\ \mathsf{correct}\ b))$ and $\mathsf{K}_a((c\ \mathsf{correct}\ b \wedge b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a) \wedge (\mathsf{K}_b(c\ \mathsf{correct}\ b) \to \mathsf{K}_a(c\ \mathsf{correct}\ b))$ follows from $\mathbf{K}(\mathsf{B}_a)$ and $\mathbf{K}(\mathsf{K}_a)$, respectively; and the sufficiency of the condition $((c\ \mathsf{correct}\ b \wedge b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a) \wedge (c\ \mathsf{correct}\ a \to \mathsf{K}_a(c\ \mathsf{correct}\ a))$ follows from $\mathbf{T}(\mathsf{K}_a)$ and $\mathbf{T}(\mathsf{K}_b)$. (Recall that there is no $\mathbf{T}$-law for belief.)

Note that trust relationships are *not* in general transitive. As a simple counterexample consider that if a trusts b and b trusts c but b's notion of correctness is not included in a's then a need not trust c just because b trusts c.

Theorem 2 (Transitivity correspondence). *For all $a, b, c \in \mathcal{A}$,*

$$\models \underbrace{\begin{pmatrix} (c\ \mathsf{trustworthy}\ b \to b\ \mathsf{doestrust}\ c)\ \wedge \\ (b\ \mathsf{trustworthy}\ a \to a\ \mathsf{doestrust}\ b)\ \wedge \\ (c\ \mathsf{trustworthy}\ a \to a\ \mathsf{doestrust}\ c) \end{pmatrix}}_{\substack{\textit{trustworthiness transparency} \\ \textit{(perfect knowledge of agent correctness)}}} \to$$

$$\left(\begin{array}{c} \underbrace{((c\ \mathsf{trustworthy}\ b \wedge b\ \mathsf{trustworthy}\ a) \to c\ \mathsf{trustworthy}\ a)}_{\textit{trustworthiness transitivity}}\ \leftrightarrow \\ \underbrace{((a\ \mathsf{doestrust}\ b \wedge b\ \mathsf{doestrust}\ c) \to a\ \mathsf{doestrust}\ c)}_{\textit{strong-trust transitivity}} \end{array} \right).$$

Proof. By propositional logic jointly from the simply-to-prove fact that

$$\models \begin{pmatrix} (c\ \mathsf{correct}\ b \to \mathsf{K}_b(c\ \mathsf{correct}\ b)) \land \\ (b\ \mathsf{correct}\ a \to \mathsf{K}_a(b\ \mathsf{correct}\ a)) \end{pmatrix} \to \begin{pmatrix} ((a\ \mathsf{doestrust}\ b \land b\ \mathsf{doestrust}\ c) \to a\ \mathsf{doestrust}\ c) \\ \to\ ((c\ \mathsf{correct}\ b \land b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a) \end{pmatrix}$$

and the already-proved fact that $\models (c\ \mathsf{correct}\ a \to \mathsf{K}_a(c\ \mathsf{correct}\ a)) \to ((c\ \mathsf{correct}\ b \land b\ \mathsf{correct}\ a) \to c\ \mathsf{correct}\ a) \to ((a\ \mathsf{doestrust}\ b \land b\ \mathsf{doestrust}\ c) \to a\ \mathsf{doestrust}\ c))$ (cf. Proposition 1.2).

Let us now turn to trust domains.

Proposition 2 (Trust domains).

0. Emptiness: $\models \mathsf{trustdom}(\emptyset)$ *and* $\models \mathsf{strongtrustdom}(\emptyset)$

1. Decomposition:
$\models \mathsf{trustdom}(\mathcal{C} \cup \mathcal{C}') \to (\mathsf{trustdom}(\mathcal{C}) \land \mathsf{trustdom}(\mathcal{C}'))$
$\models \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}') \to (\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}'))$

2. Intersection:
$\models (\mathsf{trustdom}(\mathcal{C}) \land \mathsf{trustdom}(\mathcal{C}')) \to \mathsf{trustdom}(\mathcal{C} \cap \mathcal{C}')$
$\models (\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \to \mathsf{strongtrustdom}(\mathcal{C} \cap \mathcal{C}')$

3. Antitonicity:
if $\mathcal{C} \subseteq \mathcal{C}'$ *then* $\models \mathsf{trustdom}(\mathcal{C}') \to \mathsf{trustdom}(\mathcal{C})$ *and* $\models \mathsf{strongtrustdom}(\mathcal{C}') \to \mathsf{strongtrustdom}(\mathcal{C})$

4. Mutuality:
$\models \mathsf{trustdom}(\mathcal{C} \cup \mathcal{C}') \to (\mathcal{C}\ \mathsf{trusts}\ \mathcal{C}' \land \mathcal{C}'\ \mathsf{trusts}\ \mathcal{C})$
$\models \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}') \to (\mathcal{C}\ \mathsf{doestrust}\ \mathcal{C}' \land \mathcal{C}'\ \mathsf{doestrust}\ \mathcal{C})$

Proof. Straightforward from definitions.

The following theorem provides an insight into trust domains.

Theorem 3 (Strong trust domains). *Merging two strong trust domains is* conditionally compositional *in the sense that if it is common knowledge in their union that they include each other's notions of agent correctness then a necessary and sufficient condition for their merger is that it be common knowledge in their union that each one is a strong trust domain. Formally, for all* $\mathcal{C}, \mathcal{C}' \subseteq \mathcal{A}$,

$$\underbrace{\models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix} \displaystyle\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'} (b\ \mathsf{trustworthy}\ a \to b\ \mathsf{trustworthy}\ c) \land \\ \displaystyle\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}} (b\ \mathsf{trustworthy}\ a \to b\ \mathsf{trustworthy}\ c) \end{pmatrix}}_{\textit{common knowledge of trustworthiness compatibility}} \to$$

$$\underbrace{(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \leftrightarrow \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}'))}_{\textit{trust-domain compositionality}}.$$

Proof. See Appendix A. Lemma 1.2 is crucial for the composability part, which is expressed by the formula $\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \to \mathsf{strongtrustdom}(\mathcal{C}\cup\mathcal{C}')$.

Note that there is no analogous result for weak trust domains (cf. Remark 3). That is, merging *weak* trust domains is *non*-compositional in the sense that the result of merging two weak trust domains is not necessarily again a weak trust domain — not even when there is common knowledge (instead of only common belief) in the union of both domains that each domain is a weak trust domain. Nevertheless, we can salvage the following corollary result.

Corollary 2 (Weak and strong trust domains).

$$\models \underbrace{\mathsf{CB}_{\mathcal{C}\cup\mathcal{C}'}\left(\begin{array}{l}\displaystyle\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{trustworthy}\ a \to b\ \mathsf{trustworthy}\ c)\ \land\\ \displaystyle\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{trustworthy}\ a \to b\ \mathsf{trustworthy}\ c)\end{array}\right)}_{\textit{common belief in trustworthiness compatibility}} \to$$
$$\underbrace{(\mathsf{CB}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \to \mathsf{trustdom}(\mathcal{C}\cup\mathcal{C}'))}_{\textit{strong- into weak-trust-domain composability}}.$$

Proof. Similarly to the proof in Table 10 of Appendix A, by invoking Lemma 1.1 in lieu of Lemma 1.2.

Theorem 3 yields a simple, though computationally intrinsically costly *design pattern* for recursively building up strong trust domains (cf. Section 5). Again due to the absence of an analogous theorem for weak trust domains, there is no analogous design pattern either. Hence, there really is a strong practical interest in strong trust domains. As a matter of fact, this practical interest is even stronger because checking membership in strong trust domains will turn out to be computationally no more complex (up to a constant) than checking membership in weak trust domains (cf. Section 3).

We continue to define *potential* trust between two agents a, called *potential truster*, and b, called *potential trustee*, and within communities $\mathcal{C}$. The idea is to define *potentiality* as satis*fiability.*

Definition 3 (Potential trust).

- *There is a potential weak (strong) trust relationship between a and b in the system S :iff a trusts b (a $\mathsf{doestrust}$ b) is satisfiable in the model $(\mathfrak{S}, \mathcal{V})$ induced by S.*
- *The community $\mathcal{C}$ is a potential weak (strong) trust domain in the system S :iff $\mathsf{trustdom}(\mathcal{C})$ ($\mathsf{strongtrustdom}(\mathcal{C})$) is satisfiable in the model $(\mathfrak{S}, \mathcal{V})$ induced by S.*

Similarly, we define *actual* trust between two agents a, called *truster*, and b, called *trustee*, and within communities $\mathcal{C}$. The idea is to define (two degrees of) *actuality* as (two degrees of) satis*faction.*

Definition 4 (Actual trust).

- *There is a weak (strong) trust relationship between a and b in the model $(\mathfrak{S}, \mathcal{V})$ at the state $s \in \mathcal{S}$:iff a trusts b (a $\mathsf{doestrust}$ b) is satisfied in $(\mathfrak{S}, \mathcal{V})$ at s.*
- *There is a weak (strong) trust relationship between a and b in the model $(\mathfrak{S}, \mathcal{V})$:iff a trusts b (a $\mathsf{doestrust}$ b) is globally satisfied in $(\mathfrak{S}, \mathcal{V})$.*
- *The community $\mathcal{C}$ is a weak (strong) trust domain in the model $(\mathfrak{S}, \mathcal{V})$ at the state $s \in \mathcal{S}$:iff $\mathsf{trustdom}(\mathcal{C})$ ($\mathsf{strongtrustdom}(\mathcal{C})$) is satisfied in $(\mathfrak{S}, \mathcal{V})$ at s.*
- *The community $\mathcal{C}$ is a weak (strong) trust domain in the model $(\mathfrak{S}, \mathcal{V})$:iff $\mathsf{trustdom}(\mathcal{C})$ ($\mathsf{strongtrustdom}(\mathcal{C})$) is globally satisfied in $(\mathfrak{S}, \mathcal{V})$.*

Since satisfaction implies satisfiability, but not vice versa, actual trust implies potential trust, but not vice versa. For example, if two agents do not even know each other then they can not be in an actual trust relationship. However, they may be in a potential trust relationship: maybe in another system state, their trust potential can become actualised. On the other hand, in a given system two agents may well know each other but not be in a potential trust relationship: the system design might be such that trust between them is impossible. In any case, our complexity results in the next section open the possibility of automated discovery (detection) of potential trustees (actual mistrustees), by exploiting our previous results about trust transitivity and trust references in this section.

3 Complexity results

We obtain our complexity results for deciding trust relationships and membership in trust domains by reduction to known results for the complexity of common belief and knowledge (cf. [32] and [31]). As usual for logics, the valuation function (here $\mathcal{V}$) acts as an oracle, which is assumed to decide in a single step whether or not an atomic proposition is true at the current state in the model induced by the considered distributed system S. However, deciding agent correctness is non-trivial and depends on S. For example, in our applications (cf. Section 4), deciding agent correctness can be at least polynomial in the size of the current state, which depending on the system modelling, may contain the history of system events. Another example is the case study in [42], where deterministically deciding agent correctness is quadratic in the size of the current state (containing the history of system events). That is, *state size is system-specific.* For example, when states contain the history of system events, state size can be defined as history length. Hence, we may have to account for the complexity of deciding agent correctness in the complexity of deciding trust relationships and membership in trust domains.

So, suppose that the truth of each atomic proposition can be deterministically decided in a polynomial number $f_S(|s|)$ of steps in the size $|s|$ of the state s of the model $(\mathfrak{S}, \mathcal{V})$ induced by S. The ideal case is when an agent produces its own, formal proof of correctness whenever requested to do so. (Proof-checking is tractable, whereas proof-finding is intractable.) We recall that since potential

Table 4. Computational time complexities

		Trust relations		**Trust domains**	
	degree	*weak* a trusts b	*strong* a doestrust b	*weak* trustdom$(\mathcal{C})$	*strong* strongtrustdom$(\mathcal{C})$
actual	local satisfaction in models $(\mathfrak{S}, \mathcal{V})$ and states s	$\mathcal{O}(f_S(\lvert s \rvert))$		$\mathcal{O}(f_S(\lvert s \rvert) \cdot 2^{\lvert \mathcal{C} \rvert})$	
	global satisfaction in models $(\mathfrak{S}, \mathcal{V})$				
potential	satisfiability in models $(\mathfrak{S}, \mathcal{V})$				

Recall that there is a universal quantification over states s in the definition of global satisfaction, and an existential quantification over states s in the definitions of satisfiability in models.

trust is defined in terms of satisfiability, and actual trust in terms of satisfaction, and since decidability of satisfiability implies decidability of satisfaction, decidability of potential trust implies decidability of actual trust, and complexities of potential trust are upper bounds for complexities of actual trust. Furthermore, satisfiability and validity are inter-solvable (ϕ is valid iff $\neg\phi$ is not satisfiable), and satisfiability complexities yield satisfaction (model-checking) complexities.

Theorem 4. *The computational time complexities of deterministically deciding trust relations is $\mathcal{O}(f_S(|s|))$ for potential and actual, weak and strong trust (cf. Table 4).*

Proof. Notice that our definitions of trust relations refer to a finite number (i.e., $|\mathcal{P}|$) of atomic propositions P, and that each definition uses exactly one atomic proposition (e.g., b correct a) and exactly one modal operator (e.g., B_a or K_a). Now according to [32], the complexity of the satisfiability of formulae $\mathsf{B}_a(\phi)$ and $\mathsf{K}_a(\phi)$ in a language with a finite number of atomic propositions and a bounded nesting depth of modal operators B_a and K_a is in (oracle) linear time in the length (here constantly 1) of the formula. Hence, the complexity of the satisfiability of formulae expressing weak and strong trust relations is even in (oracle) constant time, and thus $\mathcal{O}(f_S(|s|))$ without oracle. Yet $\mathcal{O}(f_S(|s|))$ is an absolute lower bound and thus the complexity of *all* trust relationships.

We can learn at least two lessons from these results. The first lesson is that we do have to account for the complexity of deciding agent correctness in the complexity of deciding trust relationships. The second lesson is that, surprisingly, deciding agent correctness is, up to a constant, equionerous to deciding potential and actual as well as weak and strong trust relationships.

Theorem 5. *The computational time complexities of deterministically deciding trust domains is $\mathcal{O}(f_S(|s|) \cdot 2^{|\mathcal{C}|})$ for potential and actual, weak and strong trust (cf. Table 4).*

Proof. According to [31], the complexity of the satisfiability of formulae $\mathsf{CB}_{\mathcal{C}}(\phi)$ and $\mathsf{CK}_{\mathcal{C}}(\phi)$ is in (oracle) deterministic single exponential time in the length of the sub-formula ϕ. The intuition is that formulae $\mathsf{CB}_{\mathcal{C}}(\phi)$ and $\mathsf{CK}_{\mathcal{C}}(\phi)$ correspond to formulae of infinitely deeply nested operators B_a and K_a with $a \in \mathcal{C}$, respectively, and that in that case, a finite number of atomic propositions does not help. In our case, the length of the conjunctive sub-formula in trustdom$(\mathcal{C})$ and strongtrustdom$(\mathcal{C})$ is polynomial in the size $|\mathcal{C}|$ of the community $\mathcal{C}$. Further, the complexity of each conjunct is $\mathcal{O}(f_S(|s|))$, which we assumed to be polynomial. Yet a single exponential of a polynomial cost is still "only" a single exponential cost. Hence, the complexity of the satisfiability of formulae $\mathsf{trustdom}(\mathcal{C})$ and $\mathsf{strongtrustdom}(\mathcal{C})$ is deterministic single exponential time in the size of $\mathcal{C}$ times $f_S(|s|)$. That is, the complexity of membership in potential weak and strong trust domains is $\mathcal{O}(f_S(|s|) \cdot 2^{|\mathcal{C}|})$. Finally according to [31], the operators $\mathsf{CB}_\mathcal{C}$ and $\mathsf{CK}_\mathcal{C}$ force satisfying models of a size that is exponential in $|\mathcal{C}|$. Hence, $\mathcal{O}(f_S(|s|) \cdot 2^{|\mathcal{C}|})$ is the complexity of *all*, potential or actual, membership problems.

Corollary 3. *Computationally, composition of trust domains does not scale.*

Proof. By Theorem 3 and 5.

4 Application to cryptographic-key management

We instantiate our generic trust concepts in five major cryptographic applications of trust, namely: Access Control (cf. Table 2), Trusted Third Parties (TTPs), the Web of Trust, Public-Key Infrastructures (PKIs), and ID-Based Cryptography. For the latter three, we will have to define the valuation function $\mathcal{V}$ on the atomic propositions a $\mathsf{correct}$ b about agent correctness (cf. Definition 1). (We will also have to refine the definition of trust domains for the latter two.) That is, ***each notion of agent correctness is***[10] ***specific to each system rather than general to all systems.*** (Thus, ***trust is system-specific to some extent.***) However, we can define agent correctness *generically* for the Web of Trust and PKIs, with the aid of the following, common auxiliary logic, called *AuxLog*. The logic is a modal fixpoint logic [13] operating on points that are agents $a \in \mathcal{A}$ rather than states $s \in \mathcal{S}$. (The definition of agent correctness requires fixing the current state and varying the agents.) AuxLog is *parametric* in a binary relation $R \subseteq \mathcal{A} \times \mathcal{A}$ to be fixed separately for the Web of Trust and PKIs, but with the commonality of depending on a fixed state s ($R \in \{\mathrm{DTI}_s, \mathrm{CERT}_s\}$). In other words, we define:

1. trust relations from belief in or even knowledge of agent correctness;
2. agent correctness in the Web of Trust and PKIs with the aid of AuxLog as

> the existence of reliable designated-trusted-introducer and certification relationships, respectively, whose reliability is grounded in the actual secrecy of the related agents' private keys (the *root* or *anchor of trust*).

That is, we define trust relations (and domains) from the (common) knowledge of the existence of certain other, application-specific reliable relationships.

Definition 5 (Auxiliary Logic). *Let $\mathcal{X}$ designate a countable set of propositional variables C, and let*

$$\mathcal{L}' \ni \alpha ::= \mathsf{OK} \mid \mathsf{k}_b \mid C \mid \neg\alpha \mid \alpha \wedge \alpha \mid \overline{\Box}\alpha \mid \boldsymbol{\nu} C(\alpha)$$

[10] like policies, which induce notions of agent correctness, as mentioned in Section 1.1

designate the language $\mathcal{L}'$ of AuxLog, where $b \in \mathcal{A}$ and all free occurrences of C in α of $\boldsymbol{\nu} C(\alpha)$ are assumed to occur within an even number of occurrences of $\neg$ to guarantee the existence of (greatest) fixpoints (expressed by $\boldsymbol{\nu} C(\alpha)$) [13]. Then, given a relation $R \subseteq \mathcal{A} \times \mathcal{A}$, decidable in deterministic constant time, but structurally arbitrary, and an auxiliary interpretation $\llbracket \cdot \rrbracket : \mathcal{X} \cup \{\mathsf{OK}, \mathsf{k}_b\} \to 2^{\mathcal{A}}$ partially pre-defined as[11]

$$\begin{aligned} \llbracket \mathsf{OK} \rrbracket &:= \{\ a \in \mathcal{A} \mid \textit{at most } a \textit{ can access } a\textit{'s private key}\ \} \\ \llbracket \mathsf{k}_b \rrbracket &:= \{\ a \in \mathcal{A} \mid \textit{at least } a\textit{'s address is known to } b\ \}, \end{aligned}$$

the interpretation $\| \cdot \|_{\llbracket \cdot \rrbracket} : \mathcal{L}' \to 2^{\mathcal{A}}$ of AuxLog-propositions is as follows:

$$\begin{aligned} \|\mathsf{OK}\|_{\llbracket \cdot \rrbracket} &:= \llbracket \mathsf{OK} \rrbracket \\ \|\mathsf{k}_b\|_{\llbracket \cdot \rrbracket} &:= \llbracket \mathsf{k}_b \rrbracket \\ \|C\|_{\llbracket \cdot \rrbracket} &:= \llbracket C \rrbracket \\ \|\neg\alpha\|_{\llbracket \cdot \rrbracket} &:= \mathcal{A} \setminus \|\alpha\|_{\llbracket \cdot \rrbracket} \\ \|\alpha \wedge \alpha'\|_{\llbracket \cdot \rrbracket} &:= \|\alpha\|_{\llbracket \cdot \rrbracket} \cap \|\alpha'\|_{\llbracket \cdot \rrbracket} \\ \|\overline{\Box}\alpha\|_{\llbracket \cdot \rrbracket} &:= \{\ a \in \mathcal{A} \mid \textit{for all } b \in \mathcal{A}, \textit{ if } b\ R\ a \textit{ then } b \in \|\alpha\|_{\llbracket \cdot \rrbracket}\ \} \\ \|\boldsymbol{\nu} C(\alpha)\|_{\llbracket \cdot \rrbracket} &:= \bigcup \{\ A \subseteq \mathcal{A} \mid A \subseteq \|\alpha\|_{\llbracket \cdot \rrbracket_{[C \mapsto A]}}\ \}, \end{aligned}$$

where $\llbracket \cdot \rrbracket_{[C \mapsto A]}$ maps C to A and otherwise agrees with $\llbracket \cdot \rrbracket$.

Further, $\alpha \vee \alpha' := \neg(\neg\alpha \wedge \neg\alpha')$, $\top := \alpha \vee \neg\alpha$, $\bot := \neg\top$, $\alpha \to \alpha' := \neg\alpha \vee \alpha'$, $\alpha \leftrightarrow \alpha' := (\alpha \to \alpha') \wedge (\alpha' \to \alpha)$, $\overline{\Diamond}\alpha := \neg\overline{\Box}(\neg\alpha)$, and, notably, $\boldsymbol{\mu} C(\alpha(C)) := \neg\boldsymbol{\nu} C(\neg\alpha(\neg C))$.

Finally, for all $a \in \mathcal{A}$ and $\alpha \in \mathcal{L}'$,

$$\langle(\mathcal{A}, R), \llbracket \cdot \rrbracket\rangle, a \vDash \alpha \quad \textit{:iff} \quad a \in \|\alpha\|_{\llbracket \cdot \rrbracket},$$

and for all $\alpha \in \mathcal{L}'$,

$$\vDash \alpha \quad \textit{:iff} \quad \textit{for all } \llbracket \cdot \rrbracket \textit{ and } a \in \mathcal{A},\ \langle(\mathcal{A}, R), \llbracket \cdot \rrbracket\rangle, a \vDash \alpha.$$

The reader is invited not to confuse the auxiliary satisfaction relation $\vDash$ of AuxLog with $\models$, the main one from Definition 1. Further note that AuxLog is a member of the family of μ-calculi over the modal system $\mathbf{K}$, which is characterised by the laws of propositional logic and the modal laws $\vDash \overline{\Box}(\alpha \to \alpha') \to (\overline{\Box}\alpha \to \overline{\Box}\alpha')$ and "if $\vDash \alpha$ then $\vDash \overline{\Box}\alpha$". The reason is that, as mentioned,

[11] The phrasing of OK can be made formal provided that a notion of data space $\mathcal{D}$ (including keys) and data derivation for agents is fixed, e.g., à la Dolev-Yao [22]. Then data derivation can be formalised as a relation $\vdash \subseteq 2^{\mathcal{D}} \times \mathcal{D}$ (the first formalisation was in terms of closure operators [60]), and the phrase "at most a can access a's private key" as "for all $b \in \mathcal{A}$, if k is the private key of a and $\mathrm{m}_b(s) \vdash k$ then $b = a$", where $\mathrm{m}_b(s)$ returns the set of data that b generated or received as such in s. Note however that depending on the structure of $\mathcal{D}$, the computability of $\vdash$ may range from polynomial time to undecidability [70].

Table 5. (Trustworthy) Trusted Third Parties

$\mathsf{wTTP}(c,a,b) := \mathsf{CB}_{\{a,b,c\}}(a\ \mathsf{trustindecisive}\ c \land b\ \mathsf{trustindecisive}\ c)$	c is a weak TTP of a and b
$\mathsf{sTTP}(c,a,b) := \mathsf{CK}_{\{a,b,c\}}(a\ \mathsf{trustindecisive}\ c \land b\ \mathsf{trustindecisive}\ c)$	c is a strong TTP of a and b
$\mathsf{wtTTP}(c,a,b) := \mathsf{CB}_{\{a,b,c\}}(\mathsf{trustdom}(\{c,a\}) \land \mathsf{trustdom}(\{c,b\}))$	c is a weakly trustworthy TTP of a and b
$\mathsf{stTTP}(c,a,b) := \mathsf{CK}_{\{a,b,c\}}(\mathsf{strongtrustdom}(\{c,a\}) \land \mathsf{strongtrustdom}(\{c,b\}))$	c is a strongly trustworthy TTP of a and b

$R \subseteq \mathcal{A} \times \mathcal{A}$ is structurally arbitrary. Hence, no more structural properties than those of the modal system **K**, i.e., none, can generally be assumed to hold for $\overline{\Box}$. As a corollary, the model-checking problem, i.e., "Given $a \in \mathcal{A}$ and $\alpha \in \mathcal{L}'$, is it the case that $\langle(\mathcal{A}, R), \llbracket\cdot\rrbracket\rangle, a \vDash \alpha$?" is decidable in deterministic polynomial time in the size of α. See [13] for details.

4.1 Trusted Third Parties

The concept of a Trusted Third Party (TTP) is a folklore concept for much of information security, e.g., in PKIs as registration (e.g., the town hall or the local post office, the HR department of a company), certification (e.g., a company, the IT department of a company), and key-escrow authorities [21], for many protocols for authentication and key establishment [12], as well as for secure multiparty computation [75]. Recall that "a secure multiparty computation for function f can be viewed as an implementation of a trusted third party T, which, upon receipt of the input values $x_1, \ldots, x_n$ from parties $P_1, \ldots, P_n$, respectively, produces the output value $y = f(x_1, \ldots, x_n)$. Party T is trusted for (i) providing the correct value for y and (ii) [n]ot revealing any further information to parties $P_1, \ldots, P_n$" [72]. In our terminology of agent correctness, the conjunction of Condition (i) and (ii) informally stipulates what it means for T to be correct. Notice that the above definition of secure multiparty computation merely defines the *object* of trust (i.e., agent correctness), but not trust itself (which, in our definition, is belief *in* or even knowledge *of* agent correctness). To the best of our knowledge, the concept of a TTP has never been formally defined, i.e., mathematically analysed into its conceptual constituents. Here, we are able to define the TTP-concept in terms of our trust concepts (cf. Table 5), and instantiate TTPs in the Web of Trust (cf. Section 4.2) and Public-Key Infrastructures (PKIs) (cf. Section 4.3). More precisely, we define the concepts of weak and strong as well as weakly and strongly *trustworthy* TTPs. Trustworthy TTPs are TTPs that may or even must *deserve* the trust of their trusters—and vice versa. Note that the trustworthiness of TTPs (e.g., the certification authorities in a PKI) is absolutely crucial, without which whole security architectures (e.g., an international PKI for ePassports [44]) can break down. Observe that thanks to Theorem 3, the two sides, e.g., $\{c,a\}$ and $\{c,b\}$, in a strongly (but not in a weakly) trustworthy TTP constitute a (strong) trust domain as a whole, i.e., as $\{c,a\} \cup \{c,b\}$ on the sufficient condition $\mathsf{CK}_{\{c,a\}\cup\{c,b\}}(\bigwedge_{x,y\in\{c,a\},z\in\{c,b\}}(y\ \mathsf{correct}\ x \to y\ \mathsf{correct}\ z) \land \bigwedge_{x,y\in\{c,b\},z\in\{c,a\}}(y\ \mathsf{correct}\ x \to y\ \mathsf{correct}\ z))$.

4.2 The Web of Trust

In the (decentralised) Web of Trust, as defined by Philip Zimmermann (cf. [78] and [21]) for PGP in 1992, any agent can independently establish its own domain of trusted correspondents by publicly designating (e.g., on their homepage) so-called *trusted introducers*, who by this very act become commonly known as such. In PGP, the designation of a trusted introducer is implemented as the (publicly exportable) signing of the designated trusted introducer's public key with the designator's private key. Additional assurance can be provided by [5]. The role of an agent a's trusted introducer b is to act as a *guarantor* for the trustworthiness of a, and by that, to catalyse the building up of trust relationships between a and those agents c who are only potential (not yet actual) trustees of a but who are (already) actual trustees of b. Notice the importance of distinguishing between potential and actual trust (cf. Definition 3 and 4). Thus, the more guarantors (actual trustees) an agent (as an actual truster) has, the more potential trustees the agent (as a potential truster) has. In the Web of Trust, agents are (socially speaking) trustworthy, or (technically speaking) ***correct if and only if all their designated trusted introducers are, and at most they (the correct agents) can access their (own) private key.*** (Agents with untrustworthy introducers or a corrupt private key are untrustworthy.) Notice the possible *mutuality* in this social notion of agent correctness.

We model the designated-trusted-introducer relationships between agents in system states $s \in \mathcal{S}$ with a family of relations (a kind of data base) $\mathrm{DTI}_s \subseteq \mathcal{A} \times \mathcal{A}$ such that

$$b \;\mathrm{DTI}_s\; a \text{ :iff } b \text{ is a designated trusted introducer of } a \text{ in } s.$$

The valuation function $\mathcal{V}$ on the propositions $a \;\mathsf{correct}\; b$ can then be formally defined with the aid of AuxLog as follows:

$$\boxed{\mathcal{V}(a \;\mathsf{correct}\; b) := \{\ s \mid \langle(\mathcal{A}, \mathrm{DTI}_s), \emptyset\rangle, a \vDash \boldsymbol{\nu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C))\ \},}$$

where $\emptyset$ designates the empty auxiliary interpretation (C is bound!). The greatest-fixpoint assertion $\langle(\mathcal{A}, \mathrm{DTI}_s), \emptyset\rangle, a \vDash \boldsymbol{\nu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C))$ says that a is in the greatest fixpoint of the interpretation of *the property C such that:*

> if a satisfies C (i.e., a is in the interpretation of C) then a satisfies $\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C)$, which in turn says that if a is known to b then at most a can access a's private key and for all $b \in \mathcal{A}$, if b is a designated trusted introducer of a in the state s then b satisfies C.

Observe that all (=1) free occurrences of C in $\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C)$ of $\boldsymbol{\nu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C))$ occur within an even (=0) number of occurrences of $\neg$. Hence, our definition is formally well-defined. Further observe that the possible mutuality in our notion of agent correctness corresponds to the co-inductiveness of the greatest fixpoint, which allows direct (*self-designation*) and indirect (*mutual designation*) loops in the designated-trusted-introducer relationships. The interpretation of the corresponding (inductive) least-fixpoint formula would wrongly not allow such loops. Of course, the language of AuxLog allows for other, more complex definitions of agent correctness, e.g., ones disallowing self-designation,[12] and/or ones with a more complex notion of being OK. Our present definition is just an inceptive example proposal. The *co*-inductive definition has the following iterative paraphrase from *above* (iterated *de*construction).

> Everybody is correct (the Web of Trust is born in the plenum, so to say);
> except for the following agents (*ex*clude those which are clearly not OK):
> 0. agents with a corrupt private key (Type 0 agents);
> 1. agents with a designated trusted introducer of Type 0 (Type 1 agents);
> 2. agents with a designated trusted introducer of Type 1 (Type 2 agents);
> 3. etc.

Clearly, weak or strong trust relations in the Web of Trust must be universal within an agent's domain of correspondents in the sense of Fact 4: designated trusted introducers are trusted; and they would not act as such, if they did not trust their designator and their designator's other designated trusted introducers, etc. Hence, our trust relations and trust domains from Figure 1–3 as well as our weakly and strongly trustworthy TTPs from Table 5 are fit for the Web of Trust without further adaptation.

4.3 Public-Key Infrastructures

In Public-Key Infrastructures (PKIs), centralised *certificate authorities* (CAs) act as guarantors for the trustworthiness of the public key of their clients by issuing certificates that bind the public key of each client (the legitimate key owner) to the client's (unique) name (cf. [25, Chapters 18–20], [23], [46], [21], and [1–3]). (The legitimacy of a key ownership is warranted by the *registration authority* with which the key owner authenticated in person, i.e., validated the correspondence of her name to her person.) According to [25, Page 284]:

> Key management is the most difficult problem in cryptography, and a PKI system is one of the best tools that we have to solve it with. But everything depends on the security of the PKI, and therefore on the trustworthiness of the CA.

In PKIs, agents are (socially speaking) trustworthy, or (technically speaking) ***correct if and only if all their certified agents are, and at most they (the correct agents) can access their (own) private key.*** (Agents who certify incorrect agents or agents with a corrupt private key are incorrect.) Notice the absence of mutuality in this notion of agent correctness; it is intrinsically

[12] Yet whether or not you declare yourself as trustworthy may well be legally critical (cf. for example such hand-written self-declarations in certain immigration procedures).
Anyway, the disallowance of self-designation could be implemented by introducing a binding operator $\downarrow$ à la hybrid logic [6] into the language of AuxLog, such that $\langle(\mathcal{A}, R), [\![\cdot]\!]\rangle, a \vDash \downarrow C(\alpha)$:iff $\langle(\mathcal{A}, R), [\![\cdot]\!]_{[C\mapsto\{a\}]}\rangle, a \vDash \alpha$, and stipulating that $\mathcal{V}(a \;\mathsf{correct}\; b) := \{\ s \mid \langle(\mathcal{A}, \mathrm{DTI}_s), \emptyset\rangle, a \vDash \boldsymbol{\mu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \neg{\downarrow} C'(\overline{\Diamond} C') \wedge \overline{\Box} C))\ \}$.

unilateral. However, the notion of PKI-trust to be built from this notion of agent correctness will be again bilateral (i.e., mutual, and thus symmetric): the certifying correct agent trusts the certified correct agent, and vice versa.

We model the relationships from certifying agents to certified agents (which may themselves be certifying agents to agents certified by them, etc.) in system states $s \in \mathcal{S}$ with a family of relations (again a kind of data base) $\mathrm{CRT}_s \subseteq \mathcal{A} \times \mathcal{A}$ such that

$$b \; \mathrm{CRT}_s \; a \text{ :iff } b \text{ is certified by } a \text{ in } s,$$

where "b is certified by a in s" means "a has issued a valid certificate for b in s", i.e., a certificate that is non-revoked and non-suspended in s and signed by a with the private key of a. The valuation function $\mathcal{V}$ on the propositions $a \; \mathsf{correct} \; b$ can then be formally defined with the aid of AuxLog as follows:

$$\boxed{\mathcal{V}(a \; \mathsf{correct} \; b) := \{ \; s \mid \langle (\mathcal{A}, \mathrm{CRT}_s), \emptyset \rangle, a \vDash \boldsymbol{\mu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C)) \; \}.}$$

The least-fixpoint assertion $\langle (\mathcal{A}, \mathrm{CRT}_s), \emptyset \rangle, a \vDash \boldsymbol{\mu} C(\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C))$ says that a is in the least fixpoint of the interpretation of *the property C such that:*

> if a satisfies $\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C)$ (i.e., a is in the interpretation of $\mathsf{k}_b \rightarrow (\mathsf{OK} \wedge \overline{\Box} C)$)—which in turn says that if a is known to b then at most a can access a's private key and for all $b \in \mathcal{A}$, if b is certified by a in the state s then b satisfies C—then a satisfies C.

Observe that certification is unilateral (i.e., non-mutual, and thus not symmetric) in the sense that certification relationships must not be directly (*self-certification*) nor indirectly (*mutual certification*) looping, which forces a least-fixpoint formulation. The PKI-approach to trustworthiness is thus diametrically opposed to the approach of the Web of Trust. This opposition is reflected first, in the least/greatest fixpoint "duality" of the two paradigms; and second, in the fact that PKIs are based on (ultimately national) *authority* (hierarchical CAs), whereas the Web of Trust is based on (borderless) *peership* (peer-guarantors). (At the international level, it makes sense to link two national [root] CAs via the *bona officia* of a trustworthy trusted third party [a national *bridge CA*], or to organise a group of national CAs as a Web of Trust [*cross* or *mesh certification*] such as the PKIs required by ePassports [44].) Yet again of course, as with the Web of Trust, the language of AuxLog allows for other, more complex definitions of agent correctness, e.g., ones with self-certification for the root-CA[13] (the *trust anchor*), and/or ones with a more complex notion of being OK (e.g., including the possibility of *key escrow*). Again, our present definition is just an inceptive example proposal. The *in*ductive definition has the following iterative paraphrase from *below* (iterated *con*struction).

[13] Self-certification for the root-CA could be implemented by introducing an atomic proposition root true at and only at the root-CA agent, and stipulating that $\mathcal{V}(a \; \mathsf{correct} \; a) := \{ \; s \mid \langle (\mathcal{A}, \mathrm{CRT}_s), \emptyset \rangle, a \vDash (\mathsf{root} \rightarrow \overline{\Diamond} \mathsf{root}) \wedge \boldsymbol{\mu} C(\mathsf{OK} \wedge \overline{\Box} C) \; \}$.

Table 6. Public-Key Infrastructure trust domains

$\mathsf{wPKI}(\mathcal{C}) := \mathsf{CB}_{\mathcal{C}}(\bigwedge_{a,\, b\, \in\, \mathcal{C} \text{ and } (a \leq b \text{ or } b \leq a)} a \mathrel{\mathsf{trusts}} b)$	$\mathcal{C}$ is a weak PKI trust domain
$\mathsf{sPKI}(\mathcal{C}) := \mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,\, b\, \in\, \mathcal{C} \text{ and } (a \leq b \text{ or } b \leq a)} a \mathrel{\mathsf{doestrust}} b)$	$\mathcal{C}$ is a strong PKI trust domain.

> Nobody is correct (PKIs are born *ex nihilo*, so to say); except for the following agents (*in*clude those which are clearly OK): agents without a corrupt private key (Type 0 agents), whose certified agents are also of Type 0 (Type 1 agents), whose certified agents are again also of Type 0 (Type 2 agents), etc. (In other words, being of Type 0 is an invariant in the transitive closure of certification relationships.)

Notice the structural difference between this paraphrase for agent correctness in PKIs and the previous one for agent correctness in the Web of Trust. The "duality" is not pure.

As suggested, CAs are commonly organised in a hierarchy, which induces *structured trust domains* in the form of finite trees. Recall that a finite tree is a partially-ordered set (here say $\langle \mathcal{C}, \leq \rangle$) with a bottom (top) element such that for each element in the set, the down-set (up-set) of the element is a finite chain [19]. In PKI trust domains, trust relations are symmetric (up- and downwards the tree branches) and transitive (along the tree branches) but not universal (after all, a tree is a tree and not felt fabric), and the root CA corresponds to the tree root, the intermediate CA's correspond to the intermediate tree nodes, and the clients to the tree leafs. Hence we can fit our weak and strong trust domains to PKI trust domains $\langle \mathcal{C}, \leq \rangle$ by simply stipulating that the conjunction in the respective definition respect the finite-tree structure $\leq$ of $\mathcal{C}$, and reflect the symmetry and transitivity of the trust relations. And that is all: see Table 6. We can now instantiate our weakly and strongly trustworthy TTPs from Table 5 for PKIs as

$$\begin{aligned} \mathsf{wtTTP}_{\mathsf{PKI}}(c,a,b) &:= \mathsf{CB}_{\{a,b,c\}}(\mathsf{wPKI}(\{c,a\}) \land \mathsf{wPKI}(\{c,b\})) \\ \mathsf{stTTP}_{\mathsf{PKI}}(c,a,b) &:= \mathsf{CK}_{\{a,b,c\}}(\mathsf{sPKI}(\{c,a\}) \land \mathsf{sPKI}(\{c,b\})), \end{aligned}$$

respectively, with $\leq\, := \{(c,a),(c,b)\}$ as (tree) domain structure. Fits for trust domains with other structures (e.g., buses, chains, rings, stars, etc.) can be made by similarly simple stipulations (e.g., for computing clouds, which have not a fixed but a dynamic, i.e., an evolving structure).

In sum, a weak or strong PKI trust domain $\mathcal{C}$ is built from a *certification hierarchy* $\leq$ of certifying (CAs) and certifiable agents $a \in \mathcal{C}$ such that for all states $s \in \mathcal{S}$ there is a (possibly empty) *certification record* $\{\ b \in \mathcal{C} \mid b \mathrel{\mathrm{CRT}_s} a\ \}$. Thereby, the certification hierarchy acts as a constraining skeleton (there is no such skeleton in the Web of Trust; it is free) for the potential and the actual trust relationships in $\mathcal{C}$, and, by that, the respective memberships in the trust domain $\mathcal{C}$ itself. And the certification records act as evidential support for the actuality of the trust relationships and the memberships in the trust domain.

Table 7. Generic definition of accountability

$$\begin{array}{|r@{\;}l|}\hline \mathsf{Accountability} &:= \mathsf{Abusefreeness} \land \mathsf{Auditability}\\ \mathsf{Abusefreeness} &:= \forall a \forall b \Box (a\ \mathsf{correct}\ b \rightarrow \exists/\forall c \mathsf{P}_{(a,c)}(a\ \mathsf{correct}\ b))\\ \mathsf{Auditability} &:= \forall a \forall b \Box (\neg(a\ \mathsf{correct}\ b) \rightarrow \exists/\forall c \Diamond \Box \mathsf{P}_{(b,c)}(\neg(a\ \mathsf{correct}\ b)))\\ \hline\end{array}$$

4.4 Identity-Based Cryptography

Identity-Based Cryptography is a variation of Public-Key Cryptography in which the intending sender of a message derives the (public) encryption key from the public identity (e.g., a telephone number, an email address, etc., or a combination thereof) of the intended recipient [35]. In our setting, we abstractly model an agent a's public identity with the symbol 'a'. For the sake of the security of ID-based encryption, an ID-based private key must not be derivable from its corresponding public counterpart without an additional trap-door information. This trap-door information is owned by a central CA (cCA $\in \mathcal{A}$), which therefore can derive the private keys of all its certified agents! (Thus ID-based domains have a star structure: $\leq$ is an n-ary tree of depth 1 with as root cCA and as leaves the n agents certified by cCA.) Hence for ID-Based Cryptography, the definition of an agent being OK (cf. Section 4) must be weakened, e.g.,

$$[\![\mathsf{OK}]\!] := \{\ a \in \mathcal{A} \mid \text{at most } a \text{ and cCA can access } a\text{'s private key}\ \}.$$

Note that as a consequence, the notion of trust (and thus the value of the trustworthiness of cCA) is weakened! Of course, a restrengthening is possible, e.g., by stipulating that cCA has not *used* the private keys of its certified agents (except possibly for *key escrow*). In sum, the flexibility of ID-Based Cryptography for its users (the certified agents) is paid with a devaluation of the trustworthiness of its provider (cCA).

4.5 Accountable access control and cryptographic-key management

In [42], a generic definition of *accountability* is given, which can be instantiated in one fell swoop with our present notions of agent correctness for access control from Section 1.1 and cryptographic-key management from Sections 4.2–4.4 as displayed in Table 7. There, $\Box$ means "henceforth" and $\Diamond$ "eventually" in the sense of standard temporal logic, and $\mathsf{P}_{(a,b)}$ "a can prove to b that" in the sense of a standard (though even interactive) S4 notion of Gödel-style provability [38]. For accountability, abusefreeness means that correct agents can defend themselves against false accusations of incorrectness, and auditability that incorrect agents can eventually be found out. The connection to trust is that if b knows a's proof of a $\mathsf{correct}$ b then b knows that a is correct as far as b is concerned, and so abusefreeness induces strong trust in the sense of Figure 1 (cf. [42] for details).

5 Trust building

Building trust in the sense of building *from absence of trust* (as opposed to *re*building from *dis*trust, cf. 2nd paragraph after Definition 1) is possible if and only if there is at least *potential* trust in the sense of Definition 3. That is, given $a, b \in \mathcal{A}$ and $\mathcal{C} \subseteq \mathcal{A}$, the formulae a trusts b or a $\mathsf{doestrust}$ b, and $\mathsf{trustdom}(\mathcal{C})$ or $\mathsf{strongtrustdom}(\mathcal{C})$ must at least be *satisfiable in the model induced by the considered system*, in order for a weak or strong trust relationship from a to b to possibly exist, and in order for $\mathcal{C}$ to possibly be a weak or strong trust domain, respectively. Yet thanks to the computability of our notions of trust, a computer can aid us in our decision of whether or not building trust from a current absence of trust in a given system, and between a given pair of agents, or within a given (small) group of agents is actually possible.

When it is indeed possible to actualise a certain potential trust, the next question is how to actually *build up* the trust. Consider that a potential trustee (say a) has at least two non-mutally-exclusive possibilities of earning the trust of a potential truster (say b):

1. by *behaving correctly* according to the notion of agent correctness a $\mathsf{correct}$ b in the system where the computer has found the considered kind of trust to be actually possible. (Behaving correctly can include doing nothing, when the considered notion of agent correctness does not exclude such inactivity.)
2. by *producing evidence* (e.g., a recommendation) for *or proof* (e.g., a signed log file: [17], [42]) of behavioural correctness, whenever requested to do so by the potential truster (here in the role of an auditor).

The potential truster may then decide to trust the potential trustee based on the observation of the potential trustee's behaviour, and/or based on the knowledge of certain data, i.e., evidence or even proof produced by the potential trustee. Depending on the value of the information obtained, a relationship of weak or strong trust is established from the potential truster to the potential trustee, who have now become *de facto* an actual truster and trustee, respectively. This process of building up oriented trust relations can be extended to building up symmetric trust relationships, and trust domains of agents in such relationships. As a matter of fact, Theorem 3 yields a *design pattern*, called RD (cf. Table 8), for building up strong trust domains from a possible absence of trust in a given system via recursive descent. RD relies on the communication abstraction of *public announcement* (cf. 3rd paragraph after Definition 1), which we use as a black-box plug-in. We recall that the effect of a public announcement of a fact is to induce the common knowledge of that fact with the addressed public by minimally changing the epistemic state of each member (cf. [71] for details). Depending on the communication context, the mechanism ranges from trivial (e.g., in a public assembly) to difficult (e.g., with communication asynchrony), possibly requiring implementation as a full-fledged communication protocol. So, although 'announcement' may suggest triviality rather than difficulty, public announcements may well be non-trivial to implement in a given context, which is

Table 8. The design pattern RD

1. **Input:** a pointed model $(\mathfrak{S},\mathcal{V}),s$ and $\mathcal{C}_1,\mathcal{C}_2 \subseteq \mathcal{A}$ such that

$$(\mathfrak{S},\mathcal{V}),s \models \mathsf{CK}_{\mathcal{C}_1\cup\mathcal{C}_2}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}_1,c\in\mathcal{C}_2}(b\ \mathsf{correct}\ a \to b\ \mathsf{correct}\ c)\ \wedge\\ \bigwedge_{a,b\in\mathcal{C}_2,c\in\mathcal{C}_1}(b\ \mathsf{correct}\ a \to b\ \mathsf{correct}\ c)\end{pmatrix};$$

2. **Divide:** for $i \in \{1,2\}$ do {
   when $(\mathfrak{S},\mathcal{V}),s \models \neg\mathsf{strongtrustdom}(\mathcal{C}_i)$:
   (a) divide $\mathcal{C}_i$ freely into $\mathcal{C}_{i.1}$ and $\mathcal{C}_{i.2}$;
   (b) $s := \mathrm{RD}((\mathfrak{S},\mathcal{V}),s,\mathcal{C}_{i.1},\mathcal{C}_{i.2})$; };
3. **Conquer:** announce to the community $\mathcal{C}_1 \cup \mathcal{C}_2$ that $\mathsf{strongtrustdom}(\mathcal{C}_1) \wedge \mathsf{strongtrustdom}(\mathcal{C}_2)$ is true (choose appropriate communication channels and an appropriate protocol), which takes the system from s to some $s' \in \mathcal{S}$ such that s' is reachable (cf. Footnote 7) from s and

$$(\mathfrak{S},\mathcal{V}),s' \models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \wedge \mathsf{strongtrustdom}(\mathcal{C}'));$$

4. **Output:** $s' \in \mathcal{S}$;
5. **Effect:** $(\mathfrak{S},\mathcal{V}),s' \models \mathsf{strongtrustdom}(\mathcal{C}_1 \cup \mathcal{C}_2)$.

why we call RD just a *design pattern* rather than a full-fledged algorithm: Building up trust domains from possible absence of trust generally is computationally costly, in particular when made by means of the recursive-descent design pattern RD.

Theorem 6. *The complexity of building up trust domains is exponential in the number of potential members.*

Proof. By the fact that membership in trust domains has to be checked for each potential new member anew, which is exponential in the size of the trust domain being checked in a given model and at a given state (cf. Table 4).

In contrast, it is common knowledge (among humans) that for *destroying* actual trust relationships, and thus also trust domains, a single (side) step is sufficient — metaphorically speaking. And *re*building trust from *dis*trust is difficult. It might require *forgiving* in the sense of forgetting past violations of agent correctness, which would actually reduce rebuilding trust from distrust to building trust from absence of trust in a blank memory—after wiping the bad memories so to say.

6 Related work

There is a huge literature on notions of trust that are not formal in the sense of formal languages and semantics, and also on trust management, which however is not the subject matter of this paper.

We are aware of the following formal work related to ours.

6.1 Trust relations

As explained in the introduction, [20] presents various kinds of trust relations based on belief in and knowledge of mental attitudes behind agent behaviour, which is less general than belief in or knowledge of agent behaviour *simpliciter*.

In [77], trust relationships are defined as four-tuples of a set R of trusters, a set E of trustees, a set C of conditions, and a set P of properties (the actions or attributes of the trustees). Thereby, conditions and properties are fully abstract, i.e., without pre-determined form. According to the authors, "Trust relationship T means that under the condition set C, truster set R trust that trustee set E have the properties in set P.", where the meaning of "trust that" is formally primitive and thus left to interpretation. Given that trust can possibly be wrong, a plausible interpretation of "trust that" could be "believe that" (rather than "know that"). The authors then define operations on trust relationships in terms of set-theoretic operations on the projections of their tuples. We attempt to relate the authors' notion of trust relationship to our notion of weak trust relation (based on belief rather than knowledge) by coercing their definition of T in the following macro-definition of our logic, assuming their P is included in our $\mathcal{P}$:

$$T(C,R,E,P) := (\bigwedge_{c\in C} c) \rightarrow \bigwedge_{r\in R} \mathsf{B}_r(\bigwedge_{e\in E}\bigwedge_{p\in P} p(e)).$$

If the authors agree with this coercion, we have the following definability result:

$$\models T(\{\top\},\{a\},\{b\},\{\cdot\ \mathsf{correct}\ a\}) \leftrightarrow a\ \mathsf{trusts}\ b,$$

where $\cdot\ \mathsf{correct}\ a$ is an attribute in the authors' sense.

In [74], "trust is a state at which the host believes, expects, or accepts that the effects from the cleint are the positive", although belief is not formal in the sense of modal logic. The authors' idea is that "the host's trust on a client is obtained based on the trust evaluation of the client by the host. When the host trusts a client, the host will believe, expect or accept that the client will do no harm to the host in the given context". Hence, this notion of trust is agent-centric in the sense of being defined in terms of *specific,* (local) effects at an agent's location. This is a less general notion of trust than ours, which is generic in the sense of being defined in terms of correct agent behaviour within *arbitrary* systems. Also, we recall again that agent correctness is a standard primitive in the distributed-systems community [49].

In [15], a domain-theoretic model of trust relations between agents is presented. In that model, a given directed trust relation from a truster to a trustee is abstracted as a value reflecting the truster's degree of trust in the trustee. Thus, [15]'s notion of trust is, as opposed to ours, quantitative, but, as opposed to ours, lacks a behavioural (e.g., in terms of agent correctness) and doxastic/epistemic explication (in terms of belief/knowledge). The purpose of the model is the definition of a unique global trust state arising from the trust relations between the agents via trust policies. Complexities for computing portions of that global state are given in [43].

In [51] and [48], axiomatic frameworks for the special purpose of modelling PKIs with the relevant trust relationships are presented. However, the authors do not attempt to actually explicate what trust is. In fact, their respective framework is restricted to the mere *declaration* of the involved trust relationships. The frameworks also ignore that PKIs involve in addition to the trust relations themselves the belief in or even knowledge of these relations.

After the appearance of our work in [40] and [39], appeared [56], in which the author proposes an epistemic semantics for what we would call individual trust relations. The author's interpretation of trust is: "If agents know that a target is trustworthy under an interpretation, that agent trusts the target." In comparison, being trustworthy means being correct (behaving correctly) in our interpretation (cf. Figure 1). The author then also applies his trust definitions to PKIs and the Web of Trust.

6.2 Trust domains

To the best of our knowledge, the only formal piece of work on trust domains is [76], based on description logic. However, the authors' definition remains a conceptual *modelling,* and moreover one that is limited to PKIs.

6.3 Trust in statements

Five formal notions of trust loosely related to ours are the following. In [33], so-called *trust in belief* and *trust in performance* are formalised in the situation calculus, such that "axioms and theorems can be used to infer whether a thing can be believed". Whereas we define (weak) trust *in agents*, and moreover *in terms of belief* taken off-the-shelf as a standard primitive. In [34], the ideas of trust in belief and trust in performance are taken up again similarly. In [62], trust in a distributed system is captured as the trusted statements about the system, which are added as axioms to the axiomatics of a standard logic of belief. Then in [45], trust in the judgement about the truth of a statement about a distributed system is captured by means of a modal operator $T_{ij}(\phi)$ with the intended meaning "that agent i trusts the judgement of j on the truth of ϕ." Finally, the authors of [61] view "trust as second-order property qualifying first-order relations." That is, "trust is not a relation occurring among the agents of a system. Rather it is a way in which such relations may occur." Their view leads the authors to conceive the following sophisticated definition of trust:

> Assume a set of first-order relations functional to the achievement of a goal. Assume that one such relation holds between two agents, such that one of them (the trustor) has to achieve the given goal while the other (the trustee) is able to perform some tasks in order to achieve that goal. If the trustor chooses to achieve his goal through the task performed by the trustee, and if the trustor considers the trustee a trustworthy agent, then the relation has the property of being advantageous for the trustor. Such a property is a second-order property called trust that affects the first-order relations occurring between agents.

While trust may well have second-order character, we beg to disagree with the authors that trust should not be a relation occurring among agents of a system. In our, and most other authors' views, trust is *also* a relation between agents.

7 Conclusion

Assessment We have delivered simple smooth definitions and revealed the logical structure of weak and strong trust relations and trust domains, and potential and actual trust relationships and membership in trust domains, as well as practical complexity, compositionality, scalability, and transitivity results—and all that in a single standard minimal framework. Moreover, we have provided a design pattern for building up strong trust domains. All our definitions have the advantage of being both declarative *and* computational, as well as parametric in the notion of agent correctness. Thanks to being declarative, our definitions are independent of the manifold manifestations of trust establishment (e.g., via recommendations and/or reputation). They are meaningful in any concrete distributed system with a notion of agent correctness and state space. We recall that agent correctness is a primitive in the distributed-systems community [49], and that state space is forced in a world of digital computers. A surprising insight gained from our computational analysis of trust is that given weak trust, strong trust is for free (up to a constant) from the point of view of complexity theory. A further insight gained from our analysis is that trust can be related to accountability so that accountability induces trust. Finally, we have shown that our trust domains are fit as such for TTPs and the Web of Trust, and that with a minor modification in the form of a constraint, they can be made to fit PKIs, ID-Based Cryptography, and others.

Future work We could unify our notions of weak and strong trust relation (domain) in a notion of *graded trust relation* (*domain*) defined in terms of graded (common) belief instead of plain (common) belief and plain (common) knowledge, respectively [41]. Informally, knowledge is belief with 100% certitude. And with the additional introduction of *temporal* modalities, it becomes possible to study the evolution of the quality and quantity of trust in a given distributed system, by observing the evolution of the grade of each trust relation and trust domain in the system. This could be especially interesting for cryptographic protocols, whose function is, according to [25, Section 13.4, Page 217],

> [...] to minimize the amount of trust required. This is important enough to repeat. The function of cryptographic protocols is to minimize the amount of trust required.

Finally, we could build actual trust-management systems for trust relations and trust domains in our present sense of building trust *from absence of trust* and in a future sense of *re*building trust from *dis*trust.

Acknowledgements The first author thanks R. Ramanujam and S.P. Suresh, and S. Arun-Kumar and S. Prasad, as well as their respective students for delightful discussions on agent correctness and trust.

References

1. Asia PKI consortium. `http://www.apkic.org/`.
2. EuroPKI. `http://www.europki.org/`.
3. Federal PKI. `http://csrc.nist.gov/groups/ST/crypto_apps_infra/pki/index.html`.
4. R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* Wiley, second edition, 2008.
5. R. Anderson, B. Crispo, J.-H. Lee, C. Manifavas, V. Matyas, and F. Petitcolas. *The Global Internet Trust Register.* The MIT Press, 1999.
6. C. Areces and B. ten Cate. *Handbook of Modal Logic*, chapter Hybrid Logics. Volume 3 of Blackburn et al. [11], 2007.
7. A. Arenas, M. Wilson, and B. Matthews. On trust management in grids. In *Proceedings of the ACM Conference on Autonomic Computing and Communication Systems*, 2007.
8. M. Bauland, M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer. The tractability of model checking for LTL: The good, the bad, and the ugly fragments. *ACM Transactions on Computational Logic*, 12(2), 2011.
9. K. Binmore. *Game Theory: A Very Short Introduction.* Oxford University Press, 2007.
10. P. Blackburn and J. van Benthem. *Handbook of Modal Logic*, chapter Modal Logic: A Semantic Perspective. Volume 3 of Blackburn et al. [11], 2007.
11. P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning.* Elsevier, 2007.
12. C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment.* Springer, 2003.
13. J. Bradfield and C. Stirling. *Handbook of Modal Logic*, chapter Modal Mu-Calculi. Volume 3 of Blackburn et al. [11], 2007.
14. C. Cachin, I. Keidar, and A. Shraer. Trusting the cloud. Technical Column of ACM SIGACT News, June 2009.
15. M. Carbone, M. Nielsen, and V. Sassone. A formal model for trust in dynamic networks. In *Proceedings of the IEEE Conference on Software Engineering and Formal Methods*, 2003.
16. V.G. Cerf. Trust and the Internet. *IEEE Internet Computing*, 14(5), 2010.
17. A. Chuvakin. *Beautiful Security: Leading Security Experts Explain How They Think*, chapter Beautiful Log Handling. In Oram and Viega [58], 2009.
18. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3), 2001.
19. B.A. Davey and H.A. Priestley. *Introduction to Lattices and Order.* Cambridge University Press, 1990 (2002).
20. R. Demolombe. Reasoning about trust: A formal logical framework. volume 2995 of *LNCS*. Springer, 2004.
21. Y.G. Desmedt, editor. *Secure Public Key Infrastructure.* Springer, 2012. To appear.
22. D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12), 1983.
23. C. Ellison and B. Schneier. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal*, 16(1), 2000.
24. R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

25. N. Ferguson, B. Schneier, and T. Kohno. *Cryptography Engineering: Design Principles and Practical Applications.* Wiley, 2010.
26. European Research Consortium for Informatics and Mathematics. Cloud computing: Platforms, software and applications. *ERCIM News*, 83, 2010.
27. D.M. Gabbay, M. Finger, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 2 of *Oxford Logic Guides.* Oxford University Press, 2000.
28. D.M. Gabbay, I. Hodkinson, and M. Reynolds. *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1 of *Oxford Logic Guides.* Oxford University Press, 1994.
29. E. Gallery and C.J. Mitchell. Trusted computing: Security and applications. *Cryptologia*, 33(3), 2009.
30. J.Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *Journal of the ACM*, 37(3), 1990.
31. J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3), 1992.
32. J.Y. Halpern and Y. Moses. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2), 1995.
33. J. Huang and M.S. Fox. An ontology of trust — formal semantics and transitivity. In *Proceedings of the ACM Conference on Electronic Commerce*, 2006.
34. J. Huang and D. Nicol. A calculus of trust and its application to PKI and identity management. In *Proceedings of the ACM Symposium on Identity and Trust on the Internet*, 2009.
35. M. Joye and G. Neven. *Identity-Based Cryptography.* IOS Press, 2009.
36. L.M. Kaufman. Data security in the world of cloud computing. *IEEE Security & Privacy*, 7(4), 2009.
37. K.M. Khan and Q. Malluhi. Establishing trust in cloud computing. *IEEE IT Professional*, 12(5), 2010.
38. S. Kramer. A logic of interactive proofs (formal theory of knowledge transfer). Technical Report 1201.3667, arXiv, 2012. `http://arxiv.org/abs/1201.3667`.
39. S. Kramer, R. Goré, and E. Okamoto. Formal definitions and complexity results for trust relations and trust domains. In *Proceedings of the ESSLLI-affiliated Workshop on Logics in Security*, 2010.
40. S. Kramer, R. Goré, and E. Okamoto. Formal definitions and complexity results for trust relations and trust domains fit for TTPs, the Web of Trust, PKIs, and ID-Based Cryptography. Logic Column of ACM SIGACT News, March 2010.
41. S. Kramer, C. Palamidessi, R. Segala, A. Turrini, and C. Braun. A quantitative doxastic logic for probabilistic processes and applications to information-hiding. *Journal of Applied Non-Classical Logic*, 19(4), 2009.
42. S. Kramer and A. Rybalchenko. A multi-modal framework for achieving accountability in multi-agent systems. In *Proceedings of the ESSLLI-affiliated Workshop on Logics in Security*, 2010.
43. K. Krukowa and A. Twigg. The complexity of fixed point models of trust in distributed networks. *Theoretical Computer Science*, 389(3), 2007.
44. D. Lekkas and D. Gritzalis. e-Passports as a means towards a globally interoperable public key infrastructure. *Journal of Computer Security*, 18(3), 2010.
45. C.-J. Liau. Belief, information acquisition, and trust in multi-agent systems — a modal logic formulation. *Artificial Intelligence*, 149(1), 2003.
46. A. Lioy, M. Marian, N. Moltchanova, and M. Pala. PKI past, present, and future. *Journal of Information Security*, 5(1), 2006.

47. A. Lioy and G. Ramunno. *Handbook of Information and Communication Security*, chapter Trusted Computing. Springer, 2010.
48. Ch. Liu, M. Ozols, and T. Cant. An axiomatic basis for reasoning about trust in PKIs. volume 2119 of *LNCS*. Springer, 2001.
49. N.A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.
50. S. Marsh and M.R. Dibben. Trust, untrust, distrust and mistrust — an exploration of the dark(er) side. volume 3477/2005 of *LNCS*. Springer, 2005.
51. U. Maurer. Modelling a public-key infrastructure. volume 1146 of *LNCS*. Springer, 1996.
52. J.-J. Meyer and F. Veltnam. *Handbook of Modal Logic*, chapter Intelligent Agents and Common Sense Reasoning. Volume 3 of Blackburn et al. [11], 2007.
53. B. Michael. In clouds shall we trust? *IEEE Security & Privacy*, 7(5), 2009.
54. K.W. Miller, J. Voas, and P. Laplante. In trust we trust. *IEEE Computer*, 43(10), 2010.
55. J. Minker. Logic and databases: A 20 year retrospective. volume 1154/1996 of *LNCS*. Springer, 1996.
56. T. Muller. Semantics of trust. volume 6561 of *LNCS*. Springer, 2011.
57. P.J. Nickel and K. Vaesen. *Handbook of Risk Theory: Epistemology, Decision Theory, Ethics, and Social Implications of Risk*, chapter Risk and Trust. Springer, 2012.
58. A. Oram and J. Viega, editors. *Beautiful Security: Leading Security Experts Explain How They Think*. O'Reilly, 2009.
59. R. Parikh. Social software. *Synthese*, 132, 2002.
60. L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1), 1998.
61. G. Primiero and M. Taddeo. A modal type theory for formalizing trusted communications. *Journal of Applied Logic*, 10(1), 2012.
62. P.V. Rangan. An axiomatic basis of trust in distributed systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1988.
63. S. Ruohomaa and L. Kutvonen. Trust management survey. volume 3477 of *LNCS*. Springer, 2005.
64. A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2), 2007.
65. F.B. Schneider, editor. *Trust in Cyberspace*. The National Academic Press, 1998.
66. B. Schneier. Security, group size, and the human brain. *IEEE Security & Privacy*, 7(4), 2009.
67. B. Schneier. *Liars and Outliers: Enabling the Trust That Society Needs to Thrive*. Wiley, 2012.
68. Ph. Schnoebelen. The complexity of temporal logic model checking. In *Proceedings of Advances in Modal Logic*, volume 4, 2003.
69. J. Seligman, F. Liu, and P. Girard. Logic in the community. volume 6521 of *LNAI*. Springer, 2011.
70. A. Tiu, R. Goré, and J. Dawson. A proof theoretic analysis of intruder theories. *Logical Methods in Computer Science*, 6(3), 2010.
71. H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2007.
72. H.C.A. van Tilborg, editor. *Encyclopedia of Cryptography and Security*, pages 398–400. Springer, 2005.
73. J. Woleński. Applications of squares of oppositions and their generalizations in philosophical analysis. *Logica Universalis*, 2(1), 2007.

74. D. Xiu and Z. Liu. A formal definition for trust in distributed systems. volume 3650 of *LNCS*. Springer, 2005.
75. A. Yao. Protocols for secure computations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1982.
76. H. Yu, C. Jin, and H. Che. A description logic for PKI trust domain modeling. In *Proceedings of the IEEE Conference on Information Technology and Applications*, 2005.
77. W. Zhao, V. Varadharajan, and G. Bryan. Analysis and modelling of trust in distributed information systems. volume 3803 of *LNCS*. Springer, 2005.
78. Ph. Zimmermann and J. Callas. *Beautiful Security: Leading Security Experts Explain How They Think*, chapter The Evolution of PGP's Web of Trust. In Oram and Viega [58], 2009.

A A formal proof

In order to simplify our presentation of the proof of Theorem 3, we recall the following standard definition.

Definition 6 (Semantic consequence). *The formula* $\phi' \in \mathcal{L}$ *is* a semantic consequence of $\phi \in \mathcal{L}$*, written* $\phi \Rightarrow \phi'$*, :iff for all models* $(\mathfrak{S}, \mathcal{V})$ *and states* $s \in \mathcal{S}$*, if* $(\mathfrak{S}, \mathcal{V}), s \models \phi$ *then* $(\mathfrak{S}, \mathcal{V}), s \models \phi'$*.*

Fact 5 $\models \phi \rightarrow \phi'$ *if and only if* $\phi \Rightarrow \phi'$

Proof. By expansion of definitions.

The proof of Theorem 3 is now as follows:

1. $\mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}') \Rightarrow \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}'))$ cf. Table 9

2. $\models \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}') \rightarrow \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}'))$ 1, Fact 5

3. $$\begin{pmatrix} \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'} \begin{pmatrix} \bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \land \\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \end{pmatrix} \land \\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \end{pmatrix} \Rightarrow \quad \text{cf. Table 10}$$ $\mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}')$

4. $$\models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'} \begin{pmatrix} \bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \land \\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \end{pmatrix} \rightarrow$$ $(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \rightarrow \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}'))$ 3, Fact 5

5. $$\models \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'} \begin{pmatrix} \bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \land \\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a \rightarrow b\ \mathsf{correct}\ c) \end{pmatrix} \rightarrow$$ $(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \land \mathsf{strongtrustdom}(\mathcal{C}')) \leftrightarrow \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}'))$ 2, 4.

Table 9. Decomposability of strong trust domains

$$\begin{array}{r}
\mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}') \Leftrightarrow \\
\text{(by the definition of } \mathsf{strongtrustdom}(\mathcal{C} \cup \mathcal{C}')\text{)} \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C} \cup \mathcal{C}'} a \ \mathsf{doestrust} \ b) \Rightarrow \\
\text{(by the definition of } \bigwedge_{a,b \in \mathcal{C} \cup \mathcal{C}'}\text{)} \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C}} a \ \mathsf{doestrust} \ b \wedge \bigwedge_{a,b \in \mathcal{C}'} a \ \mathsf{doestrust} \ b) \Rightarrow \\
\text{(by } \mathbf{4}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'})\text{)} \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C}} a \ \mathsf{doestrust} \ b \wedge \bigwedge_{a,b \in \mathcal{C}'} a \ \mathsf{doestrust} \ b)) \Rightarrow \\
\text{(by the distributivity of } \mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'} \text{ over } \wedge, \mathbf{N}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}), \text{ and } \mathbf{K}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}) \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C}} a \ \mathsf{doestrust} \ b) \wedge \mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C}'} a \ \mathsf{doestrust} \ b)) \Rightarrow \\
\text{(by } \models \mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\phi) \rightarrow \mathsf{CK}_{\mathcal{C}}(\phi), \mathbf{N}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}), \text{ and } \mathbf{K}(\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}) \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,b \in \mathcal{C}} a \ \mathsf{doestrust} \ b) \wedge \mathsf{CK}_{\mathcal{C}'}(\bigwedge_{a,b \in \mathcal{C}'} a \ \mathsf{doestrust} \ b)) \Leftrightarrow \\
\text{(by the definition of } \mathsf{strongtrustdom}(\mathcal{C}) \text{ and } \mathsf{strongtrustdom}(\mathcal{C}')\text{)} \\
\mathsf{CK}_{\mathcal{C} \cup \mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C}) \wedge \mathsf{strongtrustdom}(\mathcal{C}'))
\end{array}$$

Table 10. Conditional composability of strong trust domains

$$\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\wedge\\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\end{pmatrix}\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{strongtrustdom}(\mathcal{C})\wedge\mathsf{strongtrustdom}(\mathcal{C}'))\Leftrightarrow$$

(by the definition of $\mathsf{strongtrustdom}(\mathcal{C})$ and $\mathsf{strongtrustdom}(\mathcal{C}')$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\wedge\\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\end{pmatrix}\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\mathsf{CK}_{\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b))\end{pmatrix}\Leftrightarrow$$

(by the distributivity of $\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}$ over $\wedge$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b))\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b))\end{pmatrix}\Rightarrow$$

(by $\mathbf{T}(\mathsf{CK}_{\mathcal{C}})$, $\mathbf{T}(\mathsf{CK}_{\mathcal{C}'})$, $\mathbf{N}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$, and $\mathbf{K}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Leftrightarrow$$

(by the idempotency of $\wedge$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\end{pmatrix}\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c))\wedge\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\end{pmatrix}\Leftrightarrow$$

(by the distributivity of $\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}$ over $\wedge$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b\end{pmatrix}\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b\end{pmatrix}\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Rightarrow$$

(by $\mathbf{4}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Rightarrow$$

(by $\models\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\phi)\rightarrow\mathsf{CK}_{\mathcal{C}'}(\phi)$, $\models\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\phi)\rightarrow\mathsf{CK}_{\mathcal{C}}(\phi)$, $\mathbf{N}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$, and $\mathbf{K}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{CK}_{\mathcal{C}}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Rightarrow$$

(by $\models\mathsf{CK}_{\mathcal{C}'}(\phi)\rightarrow\mathsf{EK}_{\mathcal{C}'}(\phi)$, $\models\mathsf{CK}_{\mathcal{C}}(\phi)\rightarrow\mathsf{EK}_{\mathcal{C}}(\phi)$, $\mathbf{N}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$, and $\mathbf{K}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{EK}_{\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\mathsf{EK}_{\mathcal{C}}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Leftrightarrow$$

(by the definition of $\mathsf{EK}_{\mathcal{C}'}$ and $\mathsf{EK}_{\mathcal{C}}$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{c\in\mathcal{C}'}\mathsf{K}_c\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{c\in\mathcal{C}}\mathsf{K}_c\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b\end{pmatrix})\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Leftrightarrow$$

(by the distributivity of K_c over $\wedge$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}\mathsf{K}_c(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C},c\in\mathcal{C}'}\mathsf{K}_c(a\ \mathsf{doestrust}\ b)\end{pmatrix}\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}\begin{pmatrix}\bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}\mathsf{K}_c(b\ \mathsf{correct}\ a\rightarrow b\ \mathsf{correct}\ c)\\ \wedge\ \bigwedge_{a,b\in\mathcal{C}',c\in\mathcal{C}}\mathsf{K}_c(a\ \mathsf{doestrust}\ b)\end{pmatrix}\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Rightarrow$$

(by Lemma 1, $\mathbf{N}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'})$, and $\mathbf{K}(\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}$)

$$\begin{pmatrix}\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{b\in\mathcal{C},c\in\mathcal{C}'}(c\ \mathsf{doestrust}\ b))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}}a\ \mathsf{doestrust}\ b)\wedge\\ \mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{b\in\mathcal{C}',c\in\mathcal{C}}(c\ \mathsf{doestrust}\ b))\wedge\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}'}a\ \mathsf{doestrust}\ b)\end{pmatrix}\Leftrightarrow$$

(by the distributivity of $\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}$ over $\wedge$)

$$\mathsf{CK}_{\mathcal{C}\cup\mathcal{C}'}(\bigwedge_{a,b\in\mathcal{C}\cup\mathcal{C}'}a\ \mathsf{doestrust}\ b)\Leftrightarrow$$

(by the definition of $\mathsf{strongtrustdom}(\mathcal{C}\cup\mathcal{C}')$)

$$\mathsf{strongtrustdom}(\mathcal{C}\cup\mathcal{C}')$$