Policies to Regulate Distributed Data Exchange*

Samuel R. Cauvin, Nir Oren, and Wamberto W. Vasconcelos

Dept. of Computing Science, University of Aberdeen, U.K.
{r01src15, n.oren, w.w.vasconcelos}@abdn.ac.uk

Abstract. Data sharing is becoming an integral part of many aspects of our daily lives. We propose a method for controlling access to data and knowledge through fine-grained, user-specified explicitly represented policies. We present an overview of a policy formalism and mechanisms to facilitate distributed data sharing. We provide a breakdown of how our approach defines compliance and violation, specifically providing a new outlook on violation of permissions within the context of data sharing. We also examine how our mechanisms have been adapted to support socially responsible interactions between participants, whilst still providing them with control over their own data. We also explore a series of planned experiments investigating how users understand and interact with policies in a simplified version of our formalism.

1 Introduction

Data sharing is becoming an integral part of many aspects of our daily lives. With the emergence of data-driven technologies that employ intelligent sensor devices in an environment, such as smart cities [5,45] and smart homes [18], data exchange and data sharing has to be addressed. While data sharing can provide benefits and services to users, it is important to regulate it to allow users to retain control of their data, addressing issues related to information governance. Not only is it important to give control to individual users, but to maximise the benefit to all users in data-sharing ecosystems.

Usually, data sharing is specified (and constrained) through the use of data access policies. These policies specify how data may (or may not) be accessed, changed and used. The traditional management of typical access policies tends to be centralised [1,11,14,40]. This poses a number of problems, such as information ownership and reliance on a central authority that may allow the manipulation of these policies, and which answers queries regarding current policy settings for data. A counter proposal to such a centralised form of policy management is provided in [37] which describes a distributed architecture for normative regulations.

We envisage a data-sharing economy where data can be safely exchanged between participants. In addition to data, we also consider participants sharing

* This research is partially sponsored by the EPSRC grant EP/P011829/1, funded under the UK Engineering and Physical Sciences Council Human Dimensions of Cyber Security call (2016).

data access policies amongst themselves. To achieve this, we present the following elements: an information model to support fine-grained policies, and a proposal for a distributed data-sharing infrastructure.

We present a language to specify data access policies that is based on deontic concepts such as prohibition, permission and obligation. We equipped this language with fully distributed mechanisms to support participants making decisions on how they should go about sharing data in a socially responsible manner. That is, we enable participants to anticipate consequences and minimise their negative effects where possible.

In this paper we present a language and associated mechanisms which are sufficiently expressive to capture many data exchange scenarios but that can also be presented at a higher level that we hope will aid users with less technical experience in creating and interpreting policies. This language draws on existing proposals, selecting some of their features and adding others where required, to build a minimal but sufficient feature set to address data exchange scenarios.

We aim to answer the following research questions:

Q1 What information/knowledge is needed to represent policies to regulate data sharing in a machine-processable fashion?
Q2 What mechanisms can we provide, using the information model and their representations (from Q1), to enable rational decisions about data sharing and policy-compliance?
Q3 Can our information model and their representations (from Q1) and mechanisms (from Q2) be sufficient to support data sharing in a distributed and secure fashion?

In this paper we will primarily focus on Q1, but some detail will be provided on how we are addressing Q2 and Q3.

In Section 2, we present an outline of our approach, including details of its important distributed aspects. In Section 3 we provide details of our policy language designed to regulate data exchange between peers. Section 4 introduces the mechanisms that drive our solution and how they are adapted to maximise social welfare. Section 5 discusses related work and in Section 6 we conclude with a discussion of what we have achieved and what we plan to do as future work.

2 A Data Exchange Economy

Our approach builds on work on peer-to-peer networks [4,34], in which participants (whether sensors, individuals, or companies) are peers, and where each of them is a self-interested party taking part in an economy where data is being exchanged. Peers hold a unique identifier, which is distributed by a central authority. This central authority also provides peers with neighbours to communicate with.

As we work in a fully distributed environment, each of our peers holds their own (possibly incomplete) information about other peers. Peers collect information as they interact with other peers, storing records of all interactions they

take part in. As our peers gather additional information about the peer-to-peer network (e.g., data, goals, and policies of other peers), we allow them to exchange this information, in addition to just exchanging data.

Every peer defines a set of policies that determine how they will interact with other peers. These policies can be updated as time passes to reflect changes in the peers' goals or knowledge about other peers. Our policies may express general regulatory statements such as for example, "no drug records and medical records can be obtained by the same party", or more specific, such as "I will only provide 10 records to each person". Our peers function autonomously, exchanging data with respect to their policies, and any goals (get this piece of data, send this piece of data to as many peers as possible, etc.) they have been given. Users do not influence the data exchange process directly, which is one way we ensure that all participants follow our mechanisms/transaction protocol.

Since our peers function independently of any central authority, it is important that our solution has a secure way to determine what events occurred in the past. For instance, to ensure no more than ten records of data are provided, we have to be able to verify how many records were provided in previous interactions. Without a central storage location, we turn to distributed storage. We considered current distributed ledger technologies, such as Blockchain [16,25], but concluded that these would provide too many unnecessary features. For simplicity, we created a solution where peers maintain a set of records, accessible only to themselves, of any transaction they had been a part of. These transaction records cannot be tampered with and do not need to be synchronised amongst peers. Moreover, records can only be accessed through our mechanisms.

Another challenge is how to apply and enforce penalties without having any kind of enforcing body. In our case, penalties are accrued by violating policies, and our mechanisms have built-in functionality to penalise peers. Our proposed solution establishes a barrier for entry, in which peers must pay to participate in "cycles", that is, a fixed unit of time within the network. We impose penalties as "penalty cycles" where, for the duration of that cycle, a peer's ability to participate in the network will be limited. The peer will respond to incoming messages, but will not perform any beneficial actions, such as sending data requests. This is not a penalty that can be bypassed without leaving the network, as it is built into the mechanisms through which the peer participates in the network. If a peer does leave the network, and manages to spoof their identity, they will have successfully avoided the penalty. They will, however, potentially have lost access to data they could previously access, as the policies which permitted them access will not necessarily apply to their new identity.

3 Policy Language

We have proposed an information model and associated formal syntax for a policy language designed to regulate data exchange. We will give a short overview of some of the necessary concepts for this language here, before looking more in depth at the structure of a policy.

Policies make use of predicates to describe the conditions in which they apply, and the actions which they regulate. These predicates have been designed to describe our data exchange scenarios, and provide a convenient way to capture concepts of data exchange, and support the back-end mechanisms of our framework. These predicates can be conveniently changed and adapted to fit other domains, so long as appropriate supporting mechanisms are available.

We use the following three atoms within our predicates: *identity* which refers to the identity of a specific peer (pId), or the identity of a group of peers (gId); *data* which identifies a specific type of data ($\mathbb{d}_i$), for instance, temperature records or GPS data; and *time* which uses cycles, the time taken for a peer to perform a fixed set of operations) to document the relative passing of time.

The primary interactions our peers have with each other are through transactions, that is, a record of predicates establishing an exchange between two peers regulated by policies. Peers hold a collection of predicates that represents their knowledge of the world. This collection is their knowledge base ($\widehat{\mathbb{P}}$), and is subdivided into "states" ($\mathbb{P}_i$), where each state is a collection of predicates associated with a specific time cycle i.

With these concepts in place, we can now discuss our policies in more detail. Policies are defined by peers to describe how their data may be accessed by other peers. These policies may express regulatory statements such as "no more than 10 records can be accessed by any peer", "temperature records can only be accessed by members of my family", or "GPS data may be accessed, but cannot be sent on by the recipient". A policy, π, in our formalism is a tuple of the form $\langle \mathbb{C}_A, \mathbb{C}_D, m^{src}_{tgt}\mathbb{A}, u_r, u_p \rangle$, where[24]:

- $\mathbb{C}_A$ and $\mathbb{C}_D$ refer to the non-empty set of activation and deactivation conditions, respectively. Activation conditions are the predicates (and constraints) which must hold for the policy to become "active". The policy will remain active until all of the deactivation conditions hold.
- m is the deontic modality of the policy, either P, F, or O for Permission, Prohibition, and Obligation. These are the standard deontic modalities representing, respectively, what can, must not, or must be performed.
- *src* is the chain of assignment for the policy, i.e. the identities of all those who have held and passed on the policy starting with the peer who enforces it, of the form $\{pId_1, pId_2, \ldots, pId_n\}$. This set shows not just everyone who has held this policy, but the order in which they held it (i.e., pId_1 passed it to pId_2, who passed it to pId_3).
- *tgt* is the identity of the group targeted by this policy.
- $\mathbb{A}$ is the non-empty set of actions which this policy permits, prohibits, or obliges. These actions are predicates that can, for instance, allow access to data, require a peer to adopt a policy, or prohibit a peer from sending data to anyone. $\mathbb{A}$ is of the form $\{\mathbb{a}_0, \mathbb{a}_1, \ldots, \mathbb{a}_n\}$. This set of actions is joined by implicit conjunctions, for permissions all actions must occur together, for prohibitions all actions must not occur together, and for obligations all actions must occur before a deadline. Disjunctions can be modelled by having alternative policies for each of the disjuncts.

– u_r and u_p are real numbers ($u_r, u_p \in \mathbb{R}$) representing the reward/penalty accrued by compliance or violation of this policy. We will discuss compliance/violation in the next section.

This definition of policies draws together a number of proposals. The notion of activation/deactivation conditions has been well studied [24,23]. The notion of deontic modality and an associated set of actions are a standard feature of regulatory norms [22,44]. Having specific roles targeted by policies is used in many scenarios, specifically we take this from traditional role-based access control [36]. The chain of assignment is a concept taken from blockchain, used to create a so-called "audit trail" for policies. Rewards and penalties are a game theoretic concept that we adapt to allow for utility calculations, and to provide incentives to comply with our policies. For a full version of the formalism associated with our language we refer readers to the following technical document [8].

Let us take the three example policies we gave above and represent them in our formalism. Some of the representations below have been simplified for presentation, we will note in the text below each example any simplifications which have been made.

Example 1. "No more than 10 records can be accessed by each peer"

$$\pi_1 = \begin{pmatrix} \mathit{recordsAccessed}(g_{any}, \mathbb{d}_{any}, -\infty, +\infty, n),\ n < 10, \\ \mathit{recordsAccessed}(g_{any}, \mathbb{d}_{any}, -\infty, +\infty, n),\ n \geq 10, \\ \mathsf{P}^{\{pId_1, pId_2\}}_{gId_{any}}\, \mathit{access}(\mathbb{d}_{any}, \mathit{gId}_{any}, \mathit{10}), \\ 5, 10 \end{pmatrix}$$

This policy allows any peer to access 10 records of any data. This policy is active when fewer than 10 records of any data have been accessed (for all time, between $-\infty$ and $+\infty$). This policy deactivates when 10 or more records of any data have been accessed (for all time). We have simplified this policy slightly for presentation, as the *access* action would need to refer to 10, minus the number of records accessed so far. This would be achieved in our formalism through variables and constraints.

Example 2. "Temperature records can only be accessed by members of my family"

$$\pi_2 = \begin{pmatrix} \top, \\ \bot, \\ \mathsf{P}^{\{pId_4\}}_{gId_1}\, \mathit{access}(\mathbb{d}_1, \mathit{gId}_1, \infty), \\ 2, 0 \end{pmatrix}$$

This policy allows members of a family (represented by group gId_1) to access as many temperature records (represented by data type $\mathbb{d}_1$) as they like. The policy is always active (vacuously true $\top$), and never deactivates (vacuously false $\bot$).

3.1 Policy Compliance/Violation

With a definition of a policy (and related concepts) in place, we can now discuss the notion of policy compliance and violation. That is, how do we determine

whether a peer is complying with, or violating, a given policy? In all cases, a policy must be active when a transaction takes place for a peer to be in compliance/violation with it. A policy π is active in state $\mathbb{P}_i$ if there exists a state $\mathbb{P}_j$ prior to $\mathbb{P}_i$ where the activation conditions held, and there is no state between $\mathbb{P}_j$ and $\mathbb{P}_i$ where the activation or deactivation conditions hold. If there is a more recent state where the activation conditions hold, this should be used as $\mathbb{P}_j$ instead – if we consider situations where a policy has activated, deactivated, and then activated again. We assume there exists a predicate $active(\pi, \widehat{\mathbb{P}}, i)$ which returns whether policy π is active in state $\mathbb{P}_i$ (from a sequence of states $\widehat{\mathbb{P}}$). The full specification of active() and its auxiliary functions are detailed in the following technical document [8].

We define two predicates which determine, respectively, whether a policy π was complied/violated in state $\mathbb{P}_i$ (from a sequence of states $\widehat{\mathbb{P}}$), $complied^X(\pi, \widehat{\mathbb{P}}, i)$ and $violated^X(\pi, \widehat{\mathbb{P}}, i)$, where X stands for one of the deontic modalities $\mathsf{P}, \mathsf{F}, \mathsf{O}$. We show pseudocode for permission only as we handle these differently from existing approaches as we explain next. Our definition for prohibitions are not too different to that of permission. We will provide a short informal description of obligation compliance/violation.

Compliance and violation of a permission are where our approach differs from the standard approaches in the literature [24,15,3]. The two traditional approaches are: 1) everything is prohibited unless permitted, or 2) permissions as exceptions to prohibitions. Most approaches have issues, in particular, with the notion of violating a permission. We provide a clear definition of permission violation specifically relating to our intended scenario of a data sharing environment. A permission, in our approach, is used to provide access to data by other participants. Peers will adopt policies due to their bootstrapping (by the peers designer) or as a result of interaction with other peers. In both cases, this permission becomes an obligation to provide data *when requested* that a peer is permitted to access [20,26]. So a permission is complied with when data, on request by a permitted peer, is provided by the holder of the permission. A permission is violated when the data, on request by a permitted peer, is not provided by the holder of the permission. It may appear then that a permission is really just a recast obligation, but it is closer to being a combination of the two: a permission for a peer to access data, and an obligation on the data holder to provide that data. Note that, in both scenarios, the permission is complied/violated by the *holder* of that permission. This definition of permissions is a more natural way of capturing commonly occurring phenomena of data exchange.

To more clearly illustrate this, we provide the pseudocode for $complied^{\mathsf{P}}()$ in Algorithm 1. Given the proof in Theorem 1 below, we show that we can compute $violated^{\mathsf{P}}()$ as $\neg complied^{\mathsf{P}}()$.

We establish below an important result, namely, that according to our definitions, compliance and violation of permissions are dual concepts, that is, if a permission is complied with then it cannot be violated, and vice-versa.

Theorem 1 (Permission Compliance/Violation Relation). *Given a policy π with modality P, a sequence of states $\widehat{\mathbb{P}}$, and a cycle to check compliance*

Algorithm 1 Permission Compliance

Require: A policy $\pi = \langle \mathbb{C}_A, \mathbb{C}_D, \mathsf{m}^{src}_{tgt}\mathbb{A}, u_r, u_p \rangle$, a sequence of states $\widehat{\mathbb{P}} = \langle \mathbb{P}_0, \ldots, \mathbb{P}_i, \ldots, \mathbb{P}_n \rangle$, a cycle index i
Ensure: *Complied*, a Boolean variable indicating that π was complied with

```
1: procedure COMPLIED^P()
2:     Complied ← ⊥
3:     if active(π, P̂, i) ∧ there is a request for an action a ∈ A in P_i then  ▷ Without a request, the policy is neither complied with nor violated.
4:         Complied ← ⊤
5:         for all a′ ∈ A do
6:             if a′ did not happen in state P_i then
7:                 Complied ← ⊥
8:                 break
9:             end if
10:        end for
11:    end if
12: end procedure
```

for i, compliance is equivalent to not violating. That is, $complied^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i) \equiv \neg violated^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i)$.

Proof. Our proof assumes that there has been a request for at least one of the actions in $\mathbb{A}$. ($\Rightarrow$) The actions associated with a policy π are the set of actions $\mathbb{A}$. Compliance of a permission can only occur when all actions in $\mathbb{A}$ occur in state $\mathbb{P}_i$. Violation of a permission can only occur when there exists at least one action in $\mathbb{A}$ that does not occur in state $\mathbb{P}_i$. If a permission is complied with in state $\mathbb{P}_i$ then all actions in $\mathbb{A}$ occur in that state, and there can be no action in $\mathbb{A}$ that does not occur. Therefore, if a permission is complied with in a state, it cannot possibly be violated in that state, so $complied^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i) \Rightarrow \neg violated^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i)$. ($\Leftarrow$) To prove the opposite, if a permission is not violated in state $\mathbb{P}_i$ then there no action in $\mathbb{A}$ that does not occur in that state, and all actions in $\mathbb{A}$ have occurred. Therefore, if a permission is not violated in a state, it must always be complied with in that state, so $complied^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i) \Leftarrow \neg violated^{\mathsf{P}}(\pi, \widehat{\mathbb{P}}, i)$.

As stated previously, we do not consider that our definitions of compliance/violation of prohibitions contain any details of interest. Instead, we will briefly summarise the definitions for obligations.

We establish that an obligation has been complied with by a peer *pId* in state i if, and only if, the policy π was active in that state ($active(\pi, \widehat{\mathbb{P}}, i)$), a transaction with *pId* has occurred in that state, and all of the actions ($\mathtt{a} \in \mathbb{A}$) associated with the policy have been logged as performed by *pId* in one of the states i to $i + deadline(\mathtt{a})$ (where $deadline(\mathtt{a})$ returns the number of cycles that the obliged action $\mathtt{a}$ must be completed within). Violation is similar, except it occurs when at leasts one of the actions ($\mathtt{a} \in \mathbb{A}$) associated with the policy has reached its deadline ($i + deadline(\mathtt{a})$) and does not have a corresponding entry from *pId* in any of the states i to $i + deadline(\mathtt{a})$.

4 Decision Mechanisms

Within our approach peers, once provided with policies, function autonomously. Due to this, they must be equipped with appropriate mechanisms to allow them to make decisions relating to transactions. These decisions broadly fall into two categories, depending on the peer's role in the transaction. If the peer is the one providing data (the policy holder), we call that peer the provider. If the peer is the one requesting data (from the provider), we call that peer the requestor.

4.1 Decisions by the Provider

During the course of a transaction, the provider sends a selection of their policies to the requestor. These policies are those which are relevant to the transaction in question, and represent the conditions for accessing (or not accessing) the requested data. The provider must make decisions regarding what policies to provide to the requestor during a transaction. Often this decision will be straightforward, as the provider can just provide all policies relevant to the current transaction. However, when there are conflicting policies in this set, the provider must choose which policies to send. In this case, the provider will come up with one or more conflict-free *permission* policy sets to send to the requestor. These sets can be thought of as "offers" for interactions in which the requestor can access data from the provider. Policies in these sets need not be currently active, but it must be possible for the requestor to take actions to activate them; if they are not active the requestor can suspend the transaction, complete the necessary actions, then return to complete it. For instance, a policy that can only be accessed at a certain time would be acceptable, but not one that requires the requestor to have a different identity. For each of these sets, the provider can calculate a utility value. This utility value is trivial to calculate for permissions and prohibitions, but slightly more complex for obligations. We sketch this in Definition 1, as $Profit_{Allow}(\Pi)$, which calculates the profit of a set of policies (representing an offer).

Definition 1 (Policy Profit (Provider)). *For a set of policies* $\Pi = \{\pi^1, \pi^2, \ldots, \pi^n\}$*, where* $\pi^i = \langle \mathbb{C}_A^i, \mathbb{C}_D^i, m^i {}^{src^i}_{tgt^i} \mathbb{A}^i, u_r^i, u_p^i \rangle$*, with three subsets, one for each modality,* $\Pi^{\mathsf{P}} = \{\pi^i \in \Pi | m^i = \mathsf{P}\}$*,* $\Pi^{\mathsf{F}} = \{\pi^i \in \Pi | m^i = \mathsf{F}\}$*,* $\Pi^{\mathsf{O}} = \{\pi^i \in \Pi | m^i = \mathsf{O}\}$*, we define two profit functions,* $Profit_{Allow} : \Pi \to n \in \mathbb{R}$ *and* $Profit_{Deny} : \Pi \to n \in \mathbb{R}$ *as follows:*

$$Profit_{Allow}(\Pi) = \sum_{\pi^i \in \Pi^{\mathsf{P}}} u_r^i - \sum_{\pi^j \in \Pi^{\mathsf{F}}} u_p^j + \sum_{\pi^k \in \Pi^{\mathsf{O}}} profitObl(\pi^k)$$

$$Profit_{Deny}(\Pi) = \sum_{\pi^i \in \Pi^{\mathsf{F}}} u_r^i - \sum_{\pi^j \in \Pi^{\mathsf{P}}} u_p^j$$

As noted above, the profit of obligations is less trivial to calculate, but below we outline a mechanism, *profitObl(*π*)*, which quantifies the profit of a given obligation. This calculation performs two operations, for each action associated with an obligation. First, it calculates the direct reward to the provider if the requestor completes the action. This varies depending on the type of action, for

instance, an action that obliges data to be sent to the provider has a profit equal to the value of that data (a value set by each peer for each type of data). The second operation relates to the passing of obligations. Within our solution, peers are able to pass obligations between each other. When passing on an obligation, the provider forfeits any rewards or penalties that would be accrued by compliance or violation of this obligation. To calculate the utility of passing an obligation, the provider considers the likelihood of completion using its knowledge about the state of the peer-to-peer network, and weights the reward/penalty of the obligation with this value. The probability of completion is by no means exhaustive and is at best an estimate using current (potentially inaccurate) knowledge.

The provider may also come up with a single *prohibition* policy set, that is, the set of policies that will trigger if the requestor's data request is refused (prohibitions that will be complied with, and permission that will be violated). This prohibition set is only used if it is the most profitable set, calculated by $Profit_{Deny}(\Pi)$ in Definition 1.

4.2 Decisions by the Requestor

The decisions made by the requestor are the counterpart to those made by the provider. They relate to deciding whether to accept an offer for access to data via a set of policies. When there are multiple potential policy sets sent by the provider, then the requestor also must determine which, if any, to accept. This again involves determining the utility of each of the policy sets. When the requestor has received policy sets, it does not have to worry too much about fairness to the provider, as the provider will already have eliminated any policies it deems disadvantageous (for example those that have no incentive for the provider). The requestor does however have to select the fairest policy set, for both parties, from those that were provided.

We will start by detailing how the requestor can calculate the utility of a policy set, before discussing how the fairest set can then be chosen. The utility calculation varies depending on the modality of the policy:

- The utility of a permission considers the value of data that can be accessed, and the cost of any actions required to activate the associated policy. It also considers the penalty of any policies the requestor holds to *not* access that data, and the reward of any policies the requestor holds that require access to that data.
- The utility of a prohibition considers the value of data that can no longer be accessed, *or* the cost of any actions required to *deactivate* the associated policy. It also considers the penalty of any policies the requestor holds that *requires* access to that data, and the reward of any policies the requestor holds to *not* access that data.
- The utility of an obligation considers either the cost to complete the obliged actions, or the penalty of violating the obliged actions.

The utility of a policy set is the sum of the utility of the policies within that set, weighted by the likelihood of finding another (potentially better) offer. This

weighting is determined by knowledge of other peers and what data they hold; most often this weighting will have a value close to one, having little effect, as a peer may have no knowledge of other potential data sources.

Each of the three modalities makes reference to the cost to complete an action. To discuss this calculation we must first discuss the main actions catered for in our formalism. These come in two broad forms: data related actions, and policy related actions. Data related actions involve obtaining, deleting, or providing data. Policy related actions involve the adoption or revocation of policies. Adoption of policies is how our peers are able to pass obligations and access rights between each other as currency. Considering these two types of actions give shape to how we can calculate the cost to complete a given action.

Data related actions consider the average length of a transaction (in cycles), and the value of the data (and quantity) involved. Policy related actions are more complex, as the requestor must consider currently held policies which are blocked by adopting a new policy, especially when that policy is obliged by another participant. In the case of policy revocation, they must also consider any penalties associated with not holding that policy if it has been obliged by another participant. In addition to this, all actions add the reward/penalty of the policy weighted by the probability of completing/not completing that action before its deadline. This considers knowledge of the peer-to-peer network, the average time to complete transactions, and current obligations.

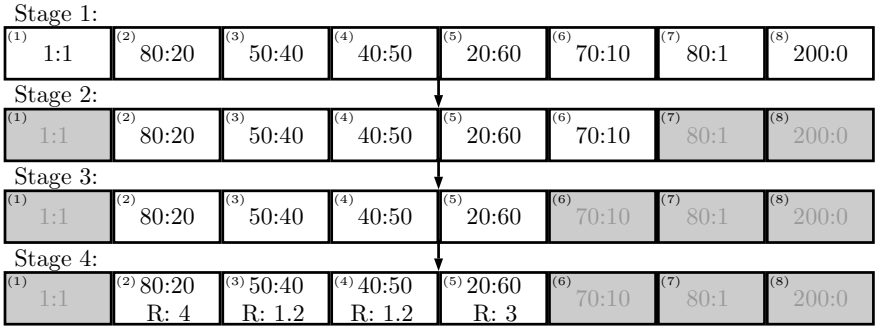

Fig. 1: An example of how the requestor chooses a policy set

As to how the requestor then chooses the fairest policy set, we use a four stage process, carried out by the requestor in the transaction. We use pareto optimality as an initial selection mechanism, and then choose the pair from those remaining with the best profit ratio. The stages are detailed below, and illustrated with an example in Figure 1:

Stage 1. Provider and requestor compute their personal utility for each policy set. Each cell in Figure 1 represents a policy set and associated utilities (Provider Utility : Requestor Utility).

Stage 2. The requestor removes those sets which have a value below the minimum profit of either party (less than 10 in this example).
Stage 3. The requestor selects the sets that have the highest utilities for both parties, i.e. the pareto frontier, the sets where neither utility could be improved without lowering the other, that is, there exists no other set with a higher utility for *both* parties.
Stage 4. The requestor then calculates the *normalised* ratio between the utilities (highest value divided by lowest value, to ensure proportional ratios). The requestor then chooses the fairest (closest to 1) set from these utilities, choosing the highest requestor utility if a tie occurs.

This operation is run by the requestor which may appear to give them an advantage, but it is performed as part of our blackbox mechanism without the influence of the participant, so it can be considered tamperproof. This operation ensures that profit is spread as evenly as possible between both parties, and prioritises fairness over total profit. We assume that for any given policy we can accurately compute the utility to ourselves.

The primary criticism of pareto optimality is that it gives no consideration to equitable distribution. This is why we only use it to identify the pareto frontier, and then go on to select the fairest of these options. Selecting the ratio that is closest to a 1:1 distribution is always the fairest, if not necessarily the best (consider two sets, 100:19 and 20:20), of the pareto frontier. While it is possible for a sub-optimal outcome to have a better ratio, it would only be removed if there is a policy set that is better for one (or both) parties without harming the other. In other words, it may filter out fairer ratios (1:2 would be subsumed by 2:5), but no party is worse off in the remaining sets.

Utility is calculated as part of a black-box mechanism, so participants cannot tamper with it. However the provider does send its utility across the network (the incentive to do this is to try and ensure they earn themselves a fair deal), so if they manage to in some way interrupt this and send false values, it does not gain them anything. Raising the values can just end up getting you a worse deal, as the requestor will end up with more to try and balance the deal. Lowering the values will reduce the chances of a policy set being picked, but in this case the provider could (and should) change their policies to have the same effect. The requestor does not send utilities and so has no chance to alter them.

5 Related Work

In our peer-to-peer system it is important for peers to have control over who, when, and how their data is shared. We achieve this through the use of policies/norms [33,35,44]. Norms are a formal representation of expected behaviours of software agents, such as prohibitions, and duties. An integral part of norms concerns deontic logic [42,22], considering permissions, prohibitions, and obligations. Norms and agents are often associated, using norms to control behaviour in societies of self-interested components [12].

Our work draws upon the concept of combining policies with data reported in [24], however our work focuses on a distributed environment and provides supporting mechanisms. The problem of unifying data and policies has, in the past, been addressed only in a centralised context [30,43]. We extend and adapt the policy language and mechanisms from our earlier work in [6] and [7].

We consider normative conflict detection [13,31] and resolution [10,38,39,29]. Conflict detection in our approach is performed when determining whether to accepted a policy related obliged action. That is, when an obligation will cause you to either adopt a new or revoke an old policy, we perform some simple conflict detection to determine how this will conflict with current goals and obligations. We do not attempt to make the policy set of each participant conflict free, only to balance the risk of potential conflict. If conflict does occur, the participant will resolve it by attempting to choose the solution with the highest utility (see Section 4).

Role Based Access Control (RBAC) shares some similarities with our work, though with a stronger focus on a controlled environment. Research has been carried out to address RBAC in a distributed environment [9,21,28], but many issues, such as a reliance on the ability to observe and control principals, have not yet been satisfactorily resolved. [9] uses user-to-user relationships to form a "path" of authorisation, but does not consider user-to-resource relationships which limits its usefulness. [21] focuses on transactions passing between two secure environments, rather than between two (potentially) insecure parties. [28] discusses automating compliance within a single secure environment, but does not discuss implementing this in a fully distributed environment.

An important concept within our solution is that of social welfare [32,27]. We use this to refer to the notion of fairness in peer transactions. A number of our mechanisms to promote social welfare are based on concepts from game theory [41]. Through our profit evaluation functions, we equate policy sets within transactions as moves in a game with associated pay-outs.

The distributed portions of our work are based on established peer-to-peer (P2P) technologies and operations [4,34]. P2P refers to networks in which "peers" communicate directly with each other, with minimal reliance on a centralised server. P2P networks can have a variety of different topologies but broadly they are either structured, where peers must organise themselves according to a set of conditions, or unstructured, where peers have a set of unrelated "neighbours" with which they communicate. Peer-to-peer simulations are based on a "network" of agents, lightweight independent processes which each act according to their own agenda, while still following a prescribed protocol [19]. Agents can be cast as a community of interconnected components, as in the Internet of Things [2,17].

6 Conclusions, Discussions, and Future Work

In this paper we proposed a solution which enables users to control how their data is used and traded within a fully distributed environment. This is achieved

through the use of fine-grained, user-specified access policies. Our solution comprises a policy formalism, associated semantics, and an outline of the supporting mechanisms. These mechanisms make allowances to maximise the utility to all parties involved in transactions, and include provisions for security without a centralised authority.

We present the most recent version of our language and mechanisms that we have been developing for some time now. Currently we are extending our mechanism (and their implementation) to accommodate recent extension to the language. We have preliminary versions of a prototype where we can simulate large-scale scenarios, however these do not reflect the latest version of our language and mechanisms This prototype will then be used to run a series of experiments to measure the performance and suitability of our language and mechanisms.

We are currently planning an evaluation to determine how participants understand and interact with policies, conducted as three experiments. Experiment 1 examines if participants understand how policy activation ($\mathbb{C}^A$) and deactivation ($\mathbb{C}^D$) conditions work by having them determine what policies are active in a specific context. Experiment 2 tests if participants can understand how a set of policies will affect the actions involved in a specific task by having them consider how a set of active policies will hinder completing a task. Experiment 3 is a variant of experiment 2 that asks participants to consider how policies affect the performance of members of a team that they are in charge of.

In terms of the mechanisms themselves, there are a number of extensions we could make to them to expand their usefulness. For instance, our framework allows for policies to dynamically change over time. At the moment these changes would have to be driven by the user. However, with all the information that our peers gather, we could enable them to make informed changes to their policy set in response to events. These policy changes could occur at any time, and could be in response to interactions, goal changes, and new knowledge about other peers and data.

Another area we could improve is to outfit our peers with more complex reasoning mechanisms to provide some limited ability to predict outcomes based on past experience. We have some provisions for this at the moment, relating to predicting potential sources for data, but we could build upon this. Our peers have the ability to request not just data, but specific pieces of knowledge about other peers in the network, so combining this with better prediction would allow our peers to form more strategic plans. Importantly, these decisions are operating on incomplete information, so our mechanisms must allow peers to take this into account.

As an extra evaluation with the extended prototype incorporating the latest version of the language, we plan to conduct a game theoretic evaluation. We have made initial explorations into standard game theoretic properties (nash equilbiria, pareto optimality, etc.), but would like to carry out a more extensive evaluation of these properties.

References

1. Artikis, A., Kamara, L., Pitt, J., Sergot, M.: A Protocol for Resource Sharing in Norm-Governed Ad Hoc Networks. In: Declarative Agent Languages and Technologies II. pp. 221–238. Springer Berlin Heidelberg (2005)
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks 54(15), 2787–2805 (2010)
3. Boella, G., van der Torre, L.: Permissions and Obligations in Hierarchical Normative Systems. In: Procs of the 9th Int'l Conference on AI and Law. pp. 109–118. ACM (2003)
4. Buford, J., Yu, H., Lua, E.K.: P2P networking and applications. Morgan Kaufmann (2009)
5. Caragliu, A., Bo, C., Nijkamp, P.: Smart Cities in Europe. Journal of Urban Technology 18(2), 65–82 (2011)
6. Cauvin, S.R., Kollingbaum, M.J., Sleeman, D., Vasconcelos, W.W.: Towards a Distributed Data-Sharing Economy. In: COIN in Agent Systems XII. pp. 3–21. Springer International Publishing (2017)
7. Cauvin, S.R., Kollingbaum, M.J., Vasconcelos, W.W.: A Peer-to-Peer Alternative to Blockchain for Managing Distributed Data Transactions. In: SmartLaw@ICAIL (2017)
8. Cauvin, S.R., Vasconcelos, W.W.: A policy formalism to facilitate distributed data exchange - technical note (2018), `https://www.dropbox.com/s/i2jbc8m1uxzprp3/policy-formalism-technical.pdf`
9. Cheng, Y., Park, J., Sandhu, R.: A user-to-user relationship-based access control model for online social networks. In: Data and applications security and privacy XXVI, pp. 8–24. Springer (2012)
10. Cholvy, L., Cuppens, F.: Solving normative conflicts by merging roles. In: Procs of the 5th Int'l Conference on A. I. and law. pp. 201–209. ACM (1995)
11. Cranefield, S.: A Rule Language for Modelling and Monitoring Social Expectations in Multi-agent Systems. In: Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems. pp. 246–258. Springer Berlin Heidelberg (2006)
12. Dignum, F.: Autonomous agents with norms. A.I. & Law 7(1), 69–79 (1999)
13. Elhag, A.A.O., Breuker, J.A.P.J., Brouwer, P.W.: On the formal analysis of normative conflicts. Information & Comms Technology Law 9(3), 207–217 (2000)
14. García-Camino, A., Rodríguez-Aguilar, J.A., Sierra, C., Vasconcelos, W.W.: A Distributed Architecture for Norm-Aware Agent Societies. In: Declarative Agent Languages and Technologies III. pp. 89–105. Springer Berlin Heidelberg (2006)
15. Governatori, G., Olivieri, F., Rotolo, A., Scannapieco, S.: Computing Strong and Weak Permissions in Defeasible Logic. Journal of Phil. Logic 42(6), 799–829 (2013)
16. Grigorik, I.: Minimum Viable Block Chain (2014), `https://www.igvita.com/2014/05/05/minimum-viable-block-chain/`
17. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems 29(7), 1645–1660 (2013)
18. Harper, R.: Inside the smart home. Springer Science & Business Media (2006)
19. Hayes, C.C.: Agents in a nutshell-a very brief introduction. IEEE Transactions on Knowledge and Data Engineering 11(1), 127–132 (1999)
20. Kanger, S.: Law and logic. Theoria 38(3), 105–132 (Dec 1972)
21. Karjoth, G., Schunter, M., Waidner, M.: Platform for enterprise privacy practices: Privacy-enabled management of customer data. pp. 69–84. Springer (2002)

22. Meyer, J.J.C., Wieringa, R.J.: Deontic Logic in Computer Science Normative System Specification. In: Int'l Workshop on Deontic Logic in Computer Science (1993)
23. Oren, N., Panagiotidi, S., Vázquez-Salceda, J., Modgil, S., Luck, M., Miles, S.: Towards a Formalisation of Electronic Contracting Environments. In: Coordination, Organizations, Institutions and Norms in Agent Systems IV. pp. 156–171 (2009)
24. Padget, J.A., Vasconcelos, W.W.: Fine-Grained Access Control via Policy-Carrying Data. ACM Trans. Internet Technol. 18(3), 31:1–31:24 (2018)
25. Postscapes: Blockchains and the Internet of Things (Mar 2016), `http://postscapes.com/blockchains-and-the-internet-of-things`
26. Pörn, I.: The Logic of Power. New York: Barnes & Noble (1970)
27. Rabin, M.: Incorporating fairness into game theory and economics. The American economic review pp. 1281–1302 (1993)
28. Sackmann, S., Kahmer, M.: ExPDT: A policy-based approach for automating compliance. Wirtschaftsinformatik 50(5), 366 (2008)
29. Santos, J.S., Zahn, J.O., Silvestre, E.A., Silva, V.T., Vasconcelos, W.W.: Detection and resolution of normative conflicts in multi-agent systems: a literature survey. Autonomous Agents and Multi-Agent Systems 31(6), 1236–1282 (Nov 2017)
30. Saroiu, S., Wolman, A., Agarwal, S.: Policy-carrying data: A privacy abstraction for attaching terms of service to mobile data. In: Procs of the 16th Int'l Workshop on Mobile Computing Systems and Applications. pp. 129–134. ACM (2015)
31. Sartor, G.: Normative conflicts in legal reasoning. AI & Law 1(2-3), 209–235 (1992)
32. Sen, A.: Collective Choice and Social Welfare: Expanded Edition. Penguin (2017)
33. Sergot, M.: A Computational Theory of Normative Positions. ACM Trans. Comput. Logic 2(4), 581–622 (Oct 2001)
34. Shen, X.S., Yu, H., Buford, J., Akon, M.: Handbook of peer-to-peer networking, vol. 34. Springer Science & Business Media (2010)
35. Shoham, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73(1), 231–252 (1995)
36. Suhendra, V.: A Survey on Access Control Deployment. In: Security Technology. pp. 11–20. Communications in Computer and Information Science, Springer (2011)
37. Vasconcelos, W.W., García-Camino, A., Gaertner, D., Rodríguez-Aguilar, J.A., Noriega, P.: Distributed norm management for multi-agent systems. Expert Systems with Applications 39(5), 5990 – 5999 (2012)
38. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Resolving conflict and inconsistency in norm-regulated virtual organizations. In: Procs of the 6th Int'l Joint Conference on Autonomous agents and multiagent systems. p. 91. ACM (2007)
39. Vasconcelos, W.W., Kollingbaum, M.J., Norman, T.J.: Normative conflict resolution in multi-agent systems. AAMAS 19(2), 124–152 (2009)
40. Viganò, F., Fornara, N., Colombetti, M.: An Event Driven Approach to Norms in Artificial Institutions. In: Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems. pp. 142–154. Springer Berlin Heidelberg (2006)
41. Von Neumann, J., Morgenstern, O.: Theory of games and economic behavior (commemorative edition). Princeton university press (2007)
42. Von Wright, G.H.: Deontic logic. Mind 60(237), 1–15 (1951)
43. Wang, X., Yong, Q., Dai, Y.h., Ren, J., Hang, Z.: Protecting Outsourced Data Privacy with Lifelong Policy Carrying. pp. 896–905 (2013)
44. von Wright, G.H.: Norm and Action: A Logical Enquiry. Routledge and Kegan Paul (1963)
45. Zheng, Y., Capra, L., Wolfson, O., Yang, H.: Urban computing: concepts, methodologies, and applications. ACM Transactions on Intelligent Systems and Technology 5(3), 38:1–38:55 (2014)