RootSet: A Distributed Trust-based Knowledge Representation Framework For Collaborative Data Exchange (Demo Paper)

Chinmay Jog
International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
jog.chinmay@iiitb.org

Sweety Agrawal
International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
sweety.v.agrawal@iiitb.org

Srinath Srinivasa
International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
sri@iiitb.ac.in

ABSTRACT

In a collaborative setting, several organizations need to exchange data with each other. Trust management is a major hindrance for inter-organizational collaborations. Organizations show reluctance towards replication of data outside their own datacenters. This work demonstrates a distributed trust-based platform for data exchange, called RootSet, among several collaborating organizations. RootSet enables organizations to share data without replication. It provides *credential-based access control* to manage data access by users throughout the platform using access rules. This work showcases several features of this tool.

Keywords: Knowledge Representation, Collaborative Data Exchange, Trust Management, Distributed Knowledge Management

1. INTRODUCTION

Collaborations between organizations frequently occur in domains like healthcare, agriculture and governance to carry out variety of tasks. Most of these collaborations involve trust-based data exchange. While some collaborations may use open datasets, others may use sensitive, private information, especially in case of healthcare projects. TRUMP[1] is one such multi-organization healthcare project, which aims to build a trusted mobile platform for self-management of chronic illness in rural areas. This initiative requires to minimize the risks in sharing private information like health records[3]. In such cases, collaborators may wish to expose only specific information that is related to the collaboration. As in case of TRUMP, collaborators may also wish to restrict the access to only some other collaborators. Also, issues like legal compliance and data sensitivity may put restrictions on physical location of the data, hindering collaboration efforts. For example, government organizations are typically reluctant to replicate data outside the jurisdiction of their nation. This problem of selective data sharing without replicating data, is a very big impediment in forming many collaborations.

In this paper, we present a distributed trust-based knowledge management framework called RootSet, that enables users to share data selectively, in a collaborative setting. RootSet is based on a knowledge model called Many Worlds on a Frame(MWF)[6], which allows users to store data inside a semantic boundary called the world. Data access is controlled by defining rules on the world. In collaborations, all the collaborators can have their own MWF servers, which are interconnected to form a distributed MWF grid. Using it, the collaborators can share their data with other collaborators, without replicating it out of their data centers. Many cloud based integrity management solutions like TrustStore[8] provide a trust-layer over the existing cloud storage. These cloud storage solutions may not be able to address the restrictions based on jurisdictions. Therefore, these may still be unusable for organizations that have a binding on the physical locations of the datacenters. RootSet addresses this problem by distributing the MWF servers at the datacenters of collaborators.

The rest of this paper is organized as follows – section 2 describes the RootSet framework, followed by its features in section 3. Section 4 discusses the implementation in brief, and section 5 discusses the conclusions and future work.

[1]`http://www.trump-india-uk.org/`

The 20th International Conference on Management of Data (COMAD), 17th-19th Dec 2014 at Hyderabad, India.

2. ROOTSET

RootSet has three components – The core MWF model, the grid manager, and the access manager. MWF is the context-aware knowledge modelling framework. The grid manager is responsible for managing the MWF grid. The access manager is responsible for managing the credentials based access control. In this section, we briefly discuss about each of these components.

2.1 Many Worlds on a Frame(MWF)

Modelling context-sensitive knowledge has been a challenge for knowledge representation frameworks[6]. The MWF model addresses this problem by providing semantic boundary called *world*, that contains knowledge elements belonging to that semantic context. Each world has a type and a location. The worlds are semantically interconnected to each other using these type and location hierarchies. These semantic interconnections form the Frame in MWF. The type and the location relationships are transitive, reflexive and anti-symmetric. The type hierarchy captures the generalization-specialization semantics. Through the type hierarchy, worlds can inherit properties. The location hierarchy captures the containment semantics. Thus, through the location hierarchy, container worlds impose privileges and control access over all the contained worlds.

Roles and Associations. In MWF, worlds host data in form of roles, and associations between roles. The roles are played by role-players, which are other worlds. Associations represent the relationships between the roles. In figure 1, Person, Subject and College are worlds. Person plays a role of Student and Faculty Member in College. Student and Faculty member roles are associated with each other by the *Teaches* association. Roles and associations have attributes of their own, which form the *schema* or the structure of a world. The role players and the associations between them form the *data*.

2.2 Distributed MWF

The distributed MWF is a conglomeration of many standalone MWF servers, that are fused together to form a single frame that can span over the network. The distributed MWF is useful especially in collaborative settings, where all the collaborators can host their data on their own MWF servers. The MWF servers can then come together to form a grid, where each collaborator can decide on their own access rules to allow users access to their data.

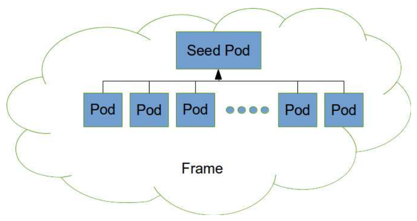


Figure 2: An MWF Grid

Pods and the MWF Grid. An MWF server is called a Pod. Each pod ships with its own database, and can connect to a grid. A grid is a set of MWF pods which are fused together in one Frame. In the grid, one pod acts as the seed pod. All other pods connect to the grid using the URI of the seed pod. This is depicted using figure 2. Other than joining the grid, no other communication needs to go through the seed pod. Therefore, the seed pod does not function as a bottleneck or a single point of failure for the grid.

Home location and Proxy of a World. In a collaboration, worlds may be extended from or placed in other worlds which are defined on other pods in the grid. In such scenarios, to avoid frequent accesses to the remotely located worlds, we may need to cache some information related to the world locally. This locally stored information represents the *proxy* of the world. It is a read-only copy of the world, in which the structure and the privilege information are cached. It does not cache any data that is stored in the world. The *home location* of the world is where the world can be edited. Any changes to the structure can only be performed at the home location. Any changes in the type or location hierarchy can also be done at the home location. While we can extend a world type from its proxy, we cannot create a new proxy for an existing proxy. A world has a unique home location, but may have any number of proxies. However, each pod may have at most one proxy of a given world.

Consistency in Distributed MWF. RootSet guarantees causal consistency on the data. Causal consistency is defined over all the world structures as well as percolation of privileges across the grid of MWF servers. By causal consistency, we mean that if an event e has occurred before an event $e\prime$ on a world w, then all the proxies of the world w will also reflect the same consequence in the same order. For example, if two users u_1 and u_2 simultaneously modify a world w, and u_1 revokes privilege of u_2 to change the world before u_2 commits the change, then only changes made by u_1 are committed. The changes made by user u_2 are not committed. All proxies of world w are notified of both the changes, and the changes reflect in the same order at all the proxies.

2.3 Credentials based access control

Traditionally, trust in terms of data exchange has been modelled on the basis of access control models. Various access control mechanisms including identity-based access[1, 5, 7], role-based access[4], credentials-based access[6] have been discussed in literature. RootSet provides an access control mechanism based on the roles played by a user in different worlds[2]. A detailed comparison between credentials based access control with other access control mechanisms is given in [2]. The credentials-based access control is based on a set of privilege rules[2]. The credentials of a user are the set of roles played by the user in different worlds. An access rule is defined using a set of credentials which when satisfied grants a privilege package. The privilege rules are of the form $PrivilegePackage_i \leftarrow r_1(t_1, l_1), r_2(t_2, l_2), \ldots, r_n(t_n, l_n)$, where r_i is a role defined in a world of type t_i and located in world l_i. A user satisfies a privilege rule when his participation set includes all the $r_i(t_i, l_i)$ tuples for the given privilege rule. On satisfying a privilege rule, a user gets a privilege package. A privilege package is a 5-bit binary string. The bits represent, in order, the five operations that a user can perform – Visibility, Privilege, Frame, Structure and Data. A visibility privilege grants read-access to a user on a world. Privilege-level privilege grants a user to add, modify or delete the privilege rules for a world. Frame level privilege grants access to modify the type and containment lineage, as well as add new worlds in the containment lineage. Structure level privilege grants access to add, modify or delete roles and associations in a world. Data level privilege grants access to add or delete role instances and association instances. Apart from these, each

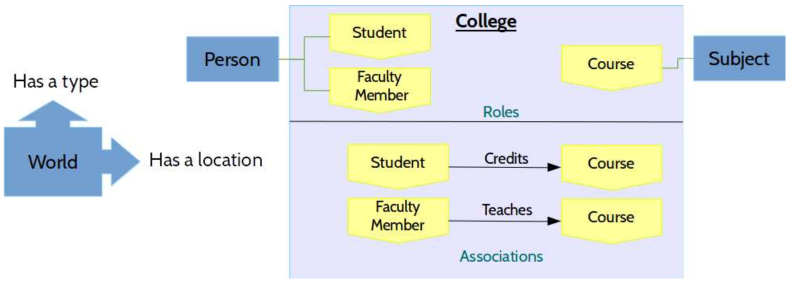
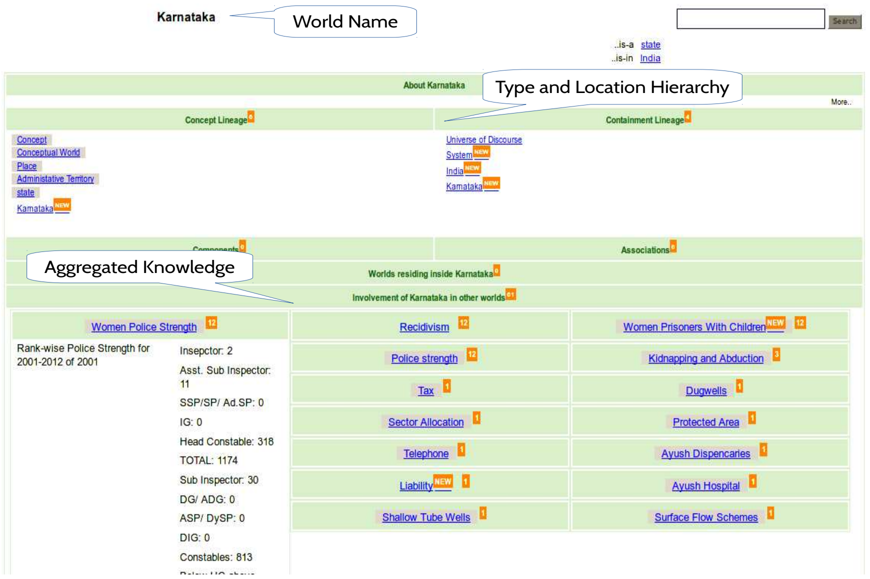


Figure 1: Many worlds on a frame



Rank-wise Police Strength for 2001-2012 of 2001

Insepctor: 2
Asst. Sub Inspector: 11
SSP/SP/ Ad.SP: 0
IG: 0
Head Constable: 318
TOTAL: 1174
Sub Inspector: 30
DG/ ADG: 0
ASP/ DySP: 0
DIG: 0
Constables: 813

Figure 3: Knowledge aggregation in Worlds

world also has a set of administrators. Administrators have the all privileges except visibility of data. A world also has a default rule. Default rule is a rule which is applied when a user does not satisfy any of the privilege rules defined for the world.

3. FEATURES

In this section, we discuss the key features of RootSet that are critical for a collaborative cross domain inter-organizational data exchange. We outline the planned and implemented features of RootSet.

Aggregation of Knowledge. A main feature of RootSet is data aggregation. If a world participates in various other worlds, it aggregates all the data, and makes it available for users at one place. This can be seen in figure 3.

Exporting worlds. RootSet provides a mechanism to export a world in XML and HTML formats. Using this feature, users can extract context specific data in an aggregated form. The export utility exports the lineage, structure and the data contained in the world.

Provenance. RootSet provides a way to trace back all the changes committed for a world, along with the privilege packages used to perform the changes. All the changes are logged in the database. Thus, a user can look back at the exact sequence of events to trace back a particular change.

Bulk load. RootSet provides a way to bulk load data and ontologies. Given mappings for each of the entity types in a structured file, it has the ability to search for worlds, create required worlds, and upload data into them. This feature helps in migrating from an existing ontology based tool onto RootSet.

4. IMPLEMENTATION

RootSet uses the Model-View-Controller architecture in its implementation. It's prototype has been built using the Ruby on Rails framework. It uses Rails 4 framework and Ruby 2.1. HTML 5 and the bootstrap Javascript framework powers its user interface. It also uses *d3.js* to display charts. It uses *sqlite3* as the database engine. It supports google authentication. It uses the Apache Solr indexing engine to locally index the worlds. The Solr server powers a full-text search capability in the implementation.

5. CONCLUSIONS AND FUTURE WORK

RootSet has been proposed to enhance multi-organizational collaborations. Some key features have been discussed. It mainly addresses the problem of sharing data in a collaborative setting. It nullifies a key challenge of giving away data outside an organization's datacenters for enabling collaborations. Implementing constraint based data-validation, exporting worlds into standard RDF representations, and a stronger risk-aware credentials based authorization are planned for future work.

6. REFERENCES

[1] S. Agarwal, B. Sprick, and S. Wortmann. Credential based access control for semantic web services. In *AAAI Spring Symposium-Semantic Web Services*, volume 1, 2004.

[2] S. Agrawal, C. Jog, and S. Srinivasa. Integrity management in a trusted utilitarian data exchange platform. In *To appear- 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2014)*. Springer, 2014.

[3] C. Burnett, P. Edwards, T. Norman, L. Chen, Y. Rahulamathavan, M. Jaffray, and E. Pignotti. Trump: A trusted mobile platform for self-management of chronic illness in rural areas. In M. Huth, N. Asokan, S. apkun, I. Flechais, and L. Coles-Kemp, editors, *Trust and Trustworthy Computing*, volume 7904 of *Lecture Notes in Computer Science*, pages 142–150. Springer Berlin Heidelberg, 2013.

[4] D. F. Ferraiolo and D. R. Kuhn. Role-based access controls. *arXiv preprint arXiv:0903.2171*, 2009.

[5] A. J. Lee. Credential-based access control. In *Encyclopedia of Cryptography and Security*, pages 271–272. Springer, 2011.

[6] S. Srinivasa, S. Agrawal, C. Jog, and J. Deshmukh. Characterizing utilitarian aggregation of open knowledge. In *Proceedings of First ACM IKDD Conference on Data Sciences, Delhi, March 2014*, pages 789–796. ACM Digital Library, 2014.

[7] S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati. Integrating trust management and access control in data-intensive web applications. *ACM Transactions on the Web (TWEB)*, 6(2):6, 2012.

[8] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic. Truststore: Making amazon s3 trustworthy with services composition. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 600–605. IEEE Computer Society, 2010.