A Semantic Framework for Secure and Efficient Contact Tracing of Infectious Diseases

Payton Schubel*, Zhiyuan Chen†, Adina Crainiceanu‡, Karuna P Joshi† and Don Needham‡
* Duke University Durham, NC, Email: {payton.schubel}@duke.edu
† University of Maryland Baltimore County, Baltimore, MD, Email: {zhchen, karuna.joshi}@umbc.edu
‡United States Naval Academy, Annapolis, MD, Email: {adina, needham}@usna.edu

2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) | 978-1-6654-0126-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/BIBM52615.2021.9669663

***Abstract*—Contact tracing is the process of identifying people who came into contact with an infected person ("case") and collecting information about these contacts. Contact tracing is an essential part of public health infrastructure and slows down the spread of infectious diseases. Existing contact tracing methods are extremely time and labor intensive due to their reliance on manually interviewing cases, contacts, and locations visited by cases. Additionally, complex privacy regulations mean that contact tracers must be extensively trained to avoid improper data sharing. App-based contact tracing, a proposed solution to these problems, has not been widely adopted by the general public due to privacy and security concerns. We develop a secure, semantically rich framework for automating the contact tracing process. This framework includes a novel, flexible ontology for contact tracing and is based on a semi-federated data-as-a-service architecture that automates contact tracing operations. Our framework supports security and privacy through situation-aware access control, where distributed query rewriting and semantic reasoning are used to automatically add situation based constraints to protect data. In this paper, we present our framework along with the validation of our system via common use cases extracted from CDC guidelines on COVID-19 contact tracing.**

***Index Terms*—Ontology, Contact tracing, Access control, Automation, Semantic web**

I. Introduction

Contact tracing, which identifies and notifies individuals who had close contact with a confirmed or probable case of an infectious disease, must be improved to effectively combat large scale pandemics like COVID-19. Current contact tracing process requires interviewing cases, contacts, and the places cases visited to determine who was exposed to an infectious disease. The procedure is time consuming and labor intensive and becomes less effective during the peaks of epidemics [1]. Current efforts to improve manual contact tracing frequently focus on better recruitment and training, not automation [2].

Apps proposed to combat contact tracing's inefficiency during the COVID-19 pandemic generally have low use when voluntarily adopted, since users are concerned with the apps' ability to maintain their privacy and security [3].

We have developed a novel contact tracing system to address these challenges. Our system builds on existing contact tracing methods where public health officials investigate cases and inform contacts. We contribute the following:

1) We built a novel and flexible ontology for contact tracing of infectious diseases. To the best of our knowledge, this is the first ontology designed for practical contact tracing of infectious respiratory diseases.
2) Our framework is based on a semi-federated Data-as-a-Service architecture which allows many contact tracing operations to be automated using SPARQL queries over data stored at multiple member locations.
3) We incorporate query rewriting to support security and privacy by situation-aware access control.
4) We validate our solution using use cases extracted from CDC guidelines [4].
5) We implement our solution using Apache Jena Fuseki.

Section II presents the architecture of our framework. Section III describes the ontology. Section IV describes automating some contact tracing operations. Section V explains how we enforce security and privacy using situation-aware access control. Section VI describes validating our solution. Section VII concludes the paper and discusses future work.

II. System Architecture

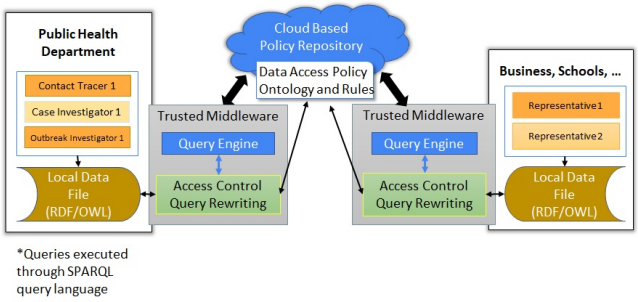

Fig. 1. An overview of the proposed system architecture, modified from [5]

We assume that all entities store data for the contact tracing process in an RDF format that is consistent with our contact tracing ontology. Our architecture is similar to our prior work [5], illustrated in Figure 1. The main difference is that our system is semi-federated, meaning most communications go through the organization in charge of contact tracing.

Organizations can implement personalized access control rules that apply when querying their specific data in addition to proposed universal access control rules. Rules are stored in a cloud-based repository for easy access. A trusted middleware contains the query engine and the query rewriting module proposed in our original work to implement the rules [5].

We use the open source Apache Jena Fuseki code to implement this architecture. Each member in the system has a Jena/Fuseki server which provides a SPARQL 1.1 service endpoint.

III. Ontology

There have been several ontologies implemented to classify infectious diseases or collect patient data [6]–[8]. However, none of these ontologies have the depth, breadth, or flexibility required to map detailed exposure scenarios as indicated by CDC guidelines [4].

We used the ontology development tool Protégé and the W3C Web Ontology Language (OWL) to design a practical contact tracing ontology. This ontology prioritizes flexibility to try to model all possible contact tracing scenarios. Additionally, we wanted to collect all information relevant to contact tracers. The ontology class structure developed from these principles is presented in Figure 2.

When selecting major classes, we considered what information contact tracers needed to know as per CDC guidelines [4]. This resulted in creating *Personal_Information* and its subclasses. Information is classified as *open*, *closed*, or *protected* based on how carefully that information is protected by rules and privacy regulations.

We considered the practical information users need to trace contacts. For example, public health officials need contact information in order to inform individuals of their disease status. Also, public health officials need certain information about those places e.g. their address, purpose, or layout in order to investigate disease exposure there. Classes were created to account for these needs.

We also considered how to link information in order to automatically determine disease exposure. To be exposed to a disease, one would have to be at the same location as an infected case at the same time. `Visits` were used to link people to places they went at certain times so that the system could contact trace. `Regular_Visits` are recurring visits that allow users to contact trace without reentering each recurrence into the system.

Finally, we considered how to make the ontology flexible. We present an approach similar to the PLATYS ontology, which classifies places using a secondary type class (e.g. PlaceType) to determine what kind of place is described by the instance [9]. Adding a new type of place (e.g. movie theater) simply requires a new instance of PlaceType as (e.g. 'movie theater') and associating a Place instance with this new PlaceType. This structure was used to classify places, organizations, and data in our ontology.

The complete ontology, including major relationships and data type properties, can be found in [10].

IV. Automation

A. Background

We list several potential applications of our ontology below. We implemented and tested the first application, and our system provides support to execute others as well.

- Accessing place or organization data directly
- Assigning people, visits, or places to users based on characteristics such as role, location, or caseload
- Identifying disease contacts based on visits/locations
- Purging data from the system after a set period of time

B. Example Use Case

We consider Alice, a student with confirmed COVID-19. We wish to request data from Alice's school to determine contacts for COVID-19.

We assume that data relevant for contact tracing is stored on two servers. The public health server stores information collected from Alice directly. The school's server stores classes, attendance, and rosters. A query sent to the school needs to return personal and contact information students who attend class with Alice, as well as attendance records for her class.

Suppose Alice is identified as `Person1a` on the public health server. The complete query is provided in [10], with segments provided below. First, Alice's name and contact information is pulled from the public health server in order to identify her.

```
1  #Collect information to identify Alice
2  path:Person1a path:hasFirstName ?fname;
3                path:hasLastName  ?lname.
4  ?contact rdfs:subPropertyOf* path:
       hasContactInformation.
5  path:Person1a ?contact ?contactInfo.
```

Since each server maintains its own database, the same entity may have different identifiers on different servers. The query needs to link these entities. After determining Alice's school and entering the school's server, Alice's name and contact information links her to the school's server, as shown below. The variable `?PUI` refers to Alice in the school's server, and it has the same name and contact information as `Person1a`. Other entities can be linked in a similar way. Then other individuals exposed to Alice (`?Person`) are determined from the visits (classes) Alice attended.

```
1  SERVICE <http://192.168.100.5:3030/ds>{
2  #Link Alice in school data based on fname,
       lname, contactInfo
3  ?PUI path:hasFirstName ?fname;
4       path:hasLastName ?lname;
5  ?contact ?contactInfo.
6  #Find people at Visits with Alice
7  ?Visit2 path:visitedBy ?PUI, ?Person;
```

Once those exposed to Alice are identified, their personal information can be determined. This query is presented prior to query-rewriting, where situation-specific privacy and security conditions are added to the query.

V. Situation-Aware Access Control

Situation-aware access control allows access to certain data based on dynamic situations, such as whether a contact tracer is assigned to a specific case.

Each member of the federated system can implement rules for accessing their specific data, in addition to universal rules applied generally. A non-exhaustive list of example rules are as follows:

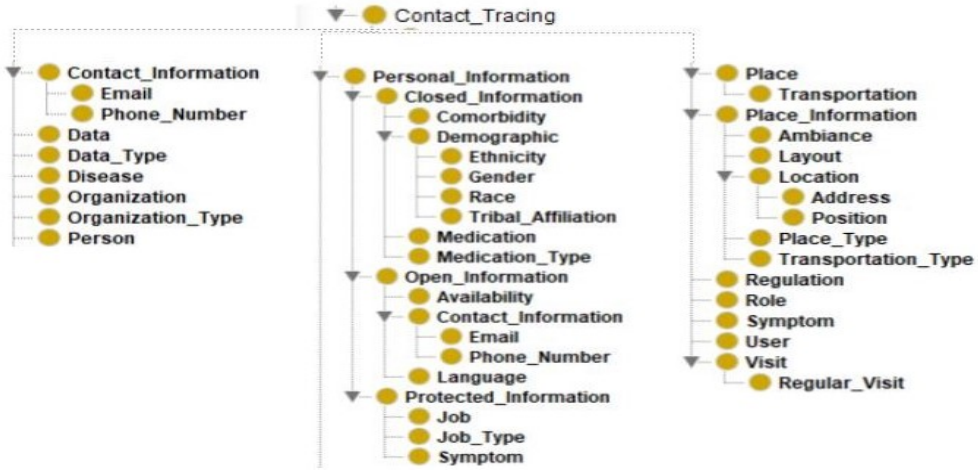

Fig. 2. The complete class structure of the contact tracing ontology

- Users only can access a Person's closed or protected personal information if a Person is a case for some disease.
- Users can access information related to an entity only if they are assigned to that entity.
- Information regulated by certain privacy regulations like HIPAA can only be accessed with explicit permission.

Once access control rules are defined, they can be enforced by the query rewriting algorithm that extends the method in [5].

Our original query-rewriting solution does not consider expanded queries with keywords such as `UNION`, `FILTER`, or `OPTIONAL` [5]. We can extend our previous solution by recognizing that queries with the `UNION` keyword can be rewritten as two nearly identical queries, with the query rewriting algorithm applied to each rewritten query. Additionally, `FILTER`s will not return more results than an unfiltered query, so restriction can ignore filters. Finally, access control rules for data queried using `OPTIONAL` can be added within the `OPTIONAL` block.

We again consider Alice, a student and confirmed case of COVID-19. This is the same example used in section IV-B. Consider UserC who wants to query the school's information. We consider the following:

- Since Alice's school is regulated by FERPA, Users need explicit permission to access the school's data.
- Only Users with a role of Contact Tracer or Outbreak Investigator can query data from the school.
- Contact Tracers accessing data in the system must be assigned to the case Alice or to the Visit itself.
- Outbreak Investigators accessing data in the system must be assigned to an entity containing the Visit's location (e.g. assigned to a district which contains the school which contains the classroom) or to the Visit itself.

The rewritten query, which has the same content as the query provided in section IV-B while implementing access control rules, is provided in [10]. Sections of the query illustrating the access control rules are provided below. First, UserC confirmed to be assigned to Alice (`Person1a`) or the location being investigated (`?Place` or `?entity`). Then UserC's ID is retrieved in order to link UserC in the school's server.

```
1   {path:UserC path:hasRole path:
        Contact_Tracer
2   {path:UserC path:assigned path:Person1a.
3   path:Person1a path:isCaseFor path:Covid
        -19.}
4   UNION{path:UserC path:assigned ?Visit.}}
5   UNION{path:UserC path:hasRole path:
        Outbreak_Investigator.
6   {path:UserC path:assigned ?Visit.}
7   UNION{{path:UserC path:assigned ?Place.}
8   UNION{path:UserC path:assigned ?entity.}}}
9   #Use UserID to check access at the school
10  path:UserC path:userID ?ID.
```

After confirming their assignment, UserC's ID is sent to the school to confirm that they have permission to access information about the class (`?Visit2`). `?User` is UserC, since they have the same ID.

```
1   SERVICE <http://192.168.100.5:3030/ds>{
2    ?User path:userID ?ID.
3    ?User path:canAccess ?Visit2
```

Once access has been confirmed, the query can return the information about contacts for COVID-19 in Alice's class.

VI. Validation

Setup: We simulated two servers using Apache Jena Fuseki on two virtual machines: a public health server to store contact tracing data, and a contact tracing partner (e.g: a school, an employer, a movie theater etc). Each VM has 2 GB RAM and 32 GB hard disks. Both VMs were hosted by a machine with 16GB RAM and an Intel i7-8665U 1.90GHz processor. We used the Micro Owl Reasoner to leverage the ontology's logic.

We validated our solution by applying it to use cases reflecting common situations contact tracers encounter [4]. The situations considered are:

Use Case	No. of triples on Partner's server	Result Size	No. of Rules Used
School	1158	4	3
Household	Not using partner's server	6	3
Work	1072	5	3
Social Event	1086	28	3
Venue Exposure	1086	3	3

TABLE I
DATA SET CHARACTERISTICS, RESULT SIZES, AND NUMBER OF RULES USED IN EACH USE CASE.

1) School: This is the use case described in section IV-B to find contacts at school.
2) Household: Finding contacts that share a household with the case.
3) Work: Finding contacts exposed at work.
4) Social Event: Finding contacts exposed at a social event (e.g. a movie).
5) Venue Exposure: A venue querying whether it or any of its locations were exposed to a disease.

These queries were implemented by considering the fictional case Alice, who exposed contacts at school, work, home, and the movies. All situations were successfully implemented with our ontology. All queries and data sets used to validate these use cases can be found in our repository [10].

The public health server contained 1152 triples. Table I lists the number of triples on each partner's server, the result size, and number of access control rules used for each use case.

Since our algorithm has not yet been modified to account for `UNION`, `FILTER` and `OPTIONAL`, the situation-aware access control rules for these use cases were added manually [5].

Results: Figure 3 presents the average execution time of each query over 10 trials with the highest and lowest execution times dropped. These trials were conducted with access control constraints.

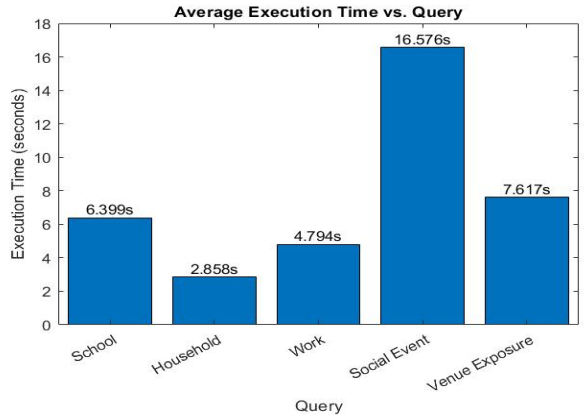

Fig. 3. The average execution time of each query over 10 trials with the highest and lowest execution times dropped.

While Social Event query has a slightly higher execution time, it is still under 20 seconds on average. The execution times of these use cases are reasonable given the modest hardware used in the experiments and demonstrate the feasibility of implementing our contact tracing system. However, it is difficult to tell how the efficiency of our system compares to other contact tracing systems without a larger data set. Greater efficiency can be achieved by building additional indexes or optimizing the queries.

VII. Conclusion and Future Work

Existing contact tracing systems are either inefficient or lack public trust due to security concerns. In this paper we propose an approach to increase the efficiency of the contact tracing process while protecting privacy. Our framework supports both situation-aware access control via query rewriting and automation of some contact tracing tasks in a semi-federated data-as-a-service system.

In our future work, we plan to collect use cases from contact tracing domain experts to ensure our ontology can handle those situations. Additionally, we plan to implement extensions to the query rewriting algorithm proposed in our previous work to handle more complex queries [5] and run larger scale experiments.

Acknowledgment

This research is partially supported by the NSF REU Site grant CNS-2050999 for Smart Computing and Communications as well as Office of Naval Research grant N00014-18-1-2452. The authors thank project director Dr. Nirmalya Roy and UMBC's Mobile, Pervasive and Sensor Computing Lab for hosting this research.

References

[1] E. Clark, E.Y. Chiao, and S. Amirian, "Why contact tracing efforts have failed to curb coronavirus disease 2019 (COVID-19) transmission in much of the United States," in *Clinical Infectious Diseases,* vol. 72 no. 9 pp. e415-e419, May 2021. Volume 72, Issue 9, 1 May 2021.

[2] P. Koetter *et al.*, "Implementation and process of a COVID-19 contact tracing initiative: leveraging health professional students to extend the workforce during a pandemic," in *American Journal of Infection Control,* vol. 48 no. 12 pp. 1451-1456, Dec 2020.

[3] F. Legendre, M. Humbert, A. Mermoud, and V. Lenders, "Contact tracing: an overview of technologies and cyber risks," 2020. [Online]. Available: https://arxiv.org/abs/2007.02806

[4] Centers for Disease Control and Prevention. "Contact Tracing," June 2021. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/php/contact-tracing/index.html

[5] S. Oni, Z. Chen, A. Crainiceanu, K. P. Joshi and D. Needham, "A Framework for Situation-Aware Access Control in Federated Data-as-a-Service Systems Based on Query Rewriting," in *IEEE International Conference on Services Computing*, 2020, pp. 1-11.

[6] D, McGagh, H. Liyanage, S. de Lusignan and J. Williams, "COVID-19 surveillance ontology." Jan 2021, accessed June 21, 2021. [Online]. Available: https://bioportal.bioontology.org/ontologies/COVID19

[7] D. Biswanath and M. DeBellis, "An ontology for collection and analysis of COVID-19 data." Oct 2020, accessed June 21, 2021. [Online]. Available: https://bioportal.bioontology.org/ontologies/CODO

[8] L. Bonino, "WHO COVID-19 rapid version CRF semantic data model." June 2020, accessed June 21, 2021. [Online]. Available: https://bioportal.bioontology.org/ontologies/COVIDCRFRAPID

[9] L. Zavala *et al.*, "Platys: From Position to Place- Oriented Mobile Computing," in *AI Magazine*, vol. 36 no. 2. pp. 50-62. 2015.

[10] P. Schubel, "Contact-tracing," August 2021. [Online]. Available: https://github.com/pns13-umbc/contact-tracing