Framework for Data Tracking across Data Controllers and Processors

Zhiyuan Lai
Brown University

Yanzhi Xin
Brown University

Atlas Yu
Brown University

Abstract

Privacy protection is an increasingly challenging problem in the technology industry. While the advancement of the internet has brought more convenience and efficiency to our everyday lives, we also left unprecedented amount of personal information online, which could be collected via web tracking and used by various third-party entities for profit. However, most users of those services are not fully aware of where their personal data is sent to and there is no easy way to visualize the whole process due to the sophistication of web tracking technologies. Therefore, we would like to defend user privacy by proposing a framework to help users and data controllers to track and visualize how user data is distributed across third-party web trackers.

1 Introduction

Web tracking, a method through which web service providers and third party platforms obtains user activity data, has been a double-edged sword on the internet. On one hand, web tracking benefits online platforms by gathering user behaviors on their site, which aids targeted advertising for greater profits. For instance, Twitter uses user data to generate a list of tailored audiences, advertisers whose ads might interest that particular user, and the more accurate Twitter matches ads with users, the higher it will profit. [9] The tailored ads based business models have enabled many internet giants, such as Google and Facebook, to provide most of their services for free and made tremendous impacts on our society. On the other hand, however, the lucrative fast-expanding online ads business stimulates a sophisticated yet poorly regulated web tracking echo system, which constantly threatens our personal privacy. In 2019, Farhad Manjoo, a journalist from the New York Times, participated in a privacy research study, which monitored his digital activity online and how his data was tracked by various web trackers.[7] The result of the study was appalling. It turned out that Farhad's activity was not only monitored by the each website he visited, but a collection of third-party trackers embedded on the site to trace every single detail of his activity, including what he had looked at, how long he stayed on each section of the page, which device he used to browse each site, etc. By the end of the study, researchers recorded more than a hundred trackers that obtained Farhad's online activities in one way or the other from 47 sites he visited, not mentioning the entities behind these trackers are able to figure out Farhad's physical address, gender, age etc through statistical analysis and digital finger prints, and to associate these activities to the 10 digits identifier associated with Farhad [7] Apparently, under-regulated online tracking is sabotaging our privacy as a basic human right, causing more harm than good to internet users.

Despite the advancement in data and privacy regulations around the world, the battle against web trackers is still at an early stage. One significant challenge is tracing who have obtained user data through third-party trackers. So far, people are able to identify and block most third-party trackers on websites [3], but are not able to learn what subsequent data processors the data has been forwarded to. To address this issue, we propose a general framework based on gRPC system, which is capable of identifying all the platforms consuming data from certain web trackers.

2 Background

Third-party web trackers are scripts or objects embedded in host website pages that act directly or assist in monitoring user activities on the site. For instance, Google Analytics is a popular tracker used by many websites to help gathering user data on the site and generating insightful information for the site owners to manage their business.[6] In addition to helping site owners, many web trackers, such as Doubleclick (also owned by Google), also collect user behaviors for targeted advertisement and share the data with the other entities. [4] To prevent these ads platforms from abusing user data and to protect personal privacy, many data regulations were issued across the world, the toughest of which is the General Data Protection Regulation (GDPR).

2.1 GDPR

The General Data Protection Regulation (GDPR) is a data protection regulation of the European Union (EU). The main purpose of this regulation is to provide rights, protections and give control to individuals (otherwise referred as data subject in GDPR) over their personal data. GDPR takes effect since 25 May 2018 [2].

Since the enforcement of the GDPR, there have been three hundred and forty violations and fines issued to individuals and companies for a total of about hundred and fifty eight million euros [10]. These companies includes multinational corporation such as Google (multiple violations), British Airways and Marriott. While there are many researches and systems proposed / framework [1, 11] which aimed to help systems to have the capabilities to be compliant to the GDPR, the adoption rate is low for various reasons such as performance impact, vast changes to current systems required and many limitations. Furthermore, there is no such common framework which helps both the data controllers and the data processors to keep track of the usage of data subjects' personal data.

It is our project's aim is to explore and address this issue by developing a common framework which can be used by both the data controllers and processors to better track the usage of personal data. This framework is not only beneficial to data controllers and data processors but also to the data subjects as they will have the high level and full picture of where their personal data have been used so as to be able to better understand the usage of their personal data and invoke their rights more effectively.

2.2 gRPC

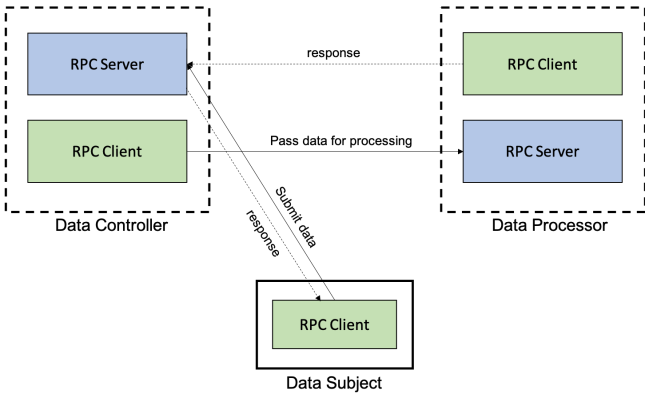

Figure 1: gRPC Diagram

For our proposed system, we use gRPC, an open source remote procedure call system that can run on any environment, as the underlying structure of the framework. Due to the unpredictable nature of operating systems types across data controllers and data processors, our design requires a universal system across all computing platforms, which makes gRPC a perfect candidate. gRPC is able to achieve universal communication among servers and clients of various OS by using Protocol Buffers (PB), an open source mechanism for serializing structured data, which is language and platform neutral. Similar to XML, Protocol Buffers are verbose, descriptive and human-readable, but, unlike XML, Protocol Buffers are smaller, faster, and more efficient than other wire-format protocols. [8] Any custom data type that needs to be serialized will be defined as a Protocol Buffer in gRPC, which gives greater flexibility for data controllers and processors to share any kind of data in any format across any platforms.

In addition, gRPC heavily promotes the use of SSL/TLS to authenticate and to encrypt all the data exchanged among clients and servers, which prevents adversaries from eavesdropping and leaking user data from outside the network.

To facilitate data tracing across platforms, we defined a list of API calls with gRPC in Table 3. We will dive into details of each API call in the next section.

3 Design

We proposed a common framework which can be used on existing systems with minimum changes. Our proposed framework in essence produces graphs with version control that tracks how data flows from the data subject to the data controller / processor. The framework tracks incoming and outgoing (sharing of) data through interception on common web services such as gRPC [5] protocols which are used by existing systems to collect and share data. This information tracker works like a proxy where traffics are forwarded to the intended receiving end. Figure 3 shows the overview of the system architecture.

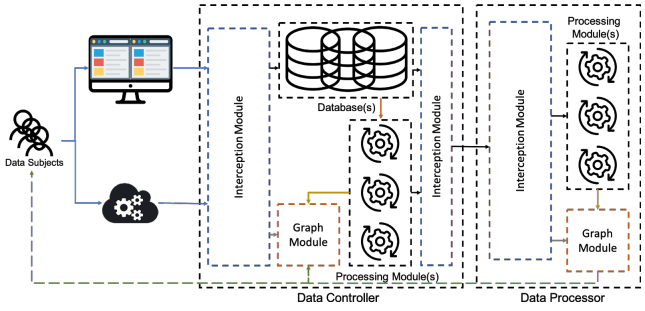

Figure 2: System Architecture

3.1 Components

There are 6 components in our proposed system and they are as follows:

1. Data Subject - Data subject in this project refers to the end users of the system. Users upload their personal information to a data controller (i.e., a server) via gRPC calls. User data is stored and processed across various entities including different data processors.
2. Data Controller - Data controller is the entity that handles upload requests from the user containing personal information and stores user data, and is the source node of the data flow graph. Data controllers may pass processed user data to other servers.
3. Data Processor - Data processor refers to every entity that stores user data passed from the data controller. It can further process user data and pass to other entities.
4. Server Interceptor - Server interceptor is a proxy for forwarding incoming requests to corresponding handler methods. Requests from users and other servers are categorized and handled by separate helper methods.
5. Processing Module - Processing module is responsible for processing user data within an entity and it imposes version control by generating new hash values for every processing / computation on user data.
6. Graph Module - Graph module within an entity is responsible for creating nodes and constructing edges of the flow graph. Graph methods are called when there relevant requests are being handled. It keeps track of the flow of user data both within a server and to other servers. It creates new nodes when new entities are introduced to the flow graph, which could be a user node, a server node, or a processing node. Internal edges between data processing nodes are created when the user's data is passed between those nodes. External edges are created when the server passes user data to other servers.

3.2 Functionalities

Creation of data node: Incoming data from for example, a website, will be intercepted by our framework. Our framework will create new nodes and incoming edges connected from the newly created node to the data controller's receiving system node in the graph with the hash value as version control label.

Creation of edges between data and system nodes: Our framework tracks any interactions / passing of the collected personal data between the data controller's systems. When data are passed / used on other systems owned by the data controller, our framework will generate an edge between the data node and the system node to represent the usage of the data by the system.

Interaction between data controllers and data processors: When (outgoing) data are passed from the data controllers to data processors for processing, our framework will create an outgoing edge from the source node to the processor node. On the data processor's end, a new node representing the data passed in will also be created. This new node (source node) will have an outgoing edge to the system node receiving this data. Any interaction between systems within the data processors will generate edges like mentioned above.

Generation of an overview graph for data subject: One of our framework's main goal is to help data subject understands the overview of what and where their data have been passed to. This was achieved by connecting all their data graph stored on the data controller to the data processor (if any). This generated graph will allow the data subject to better understand where their data have been processed so that they can better make an informed decision to invoke their rights such as the right to be forgotten or unconsent the use of their personal data of the GDPR.

3.2.1 Data Transparency

Our system achieves data transparency among servers by allowing the user to request any server in this system to gain awareness of the whereabouts of its own personal information. Within one server entity, every computation nodes that process user data and every outgoing edge will be returned to present a complete view of how this user's data is being processed by this server and where it flows outside of this entity. An example of the graph returned by a data processor is shown in Figure 5 below:

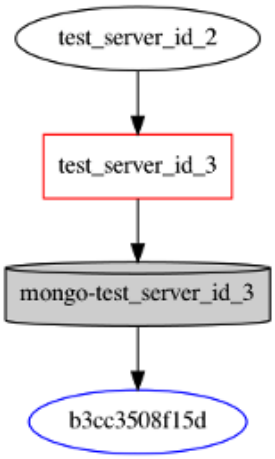

Figure 3: Visualization of the Data flow Representation on Server 3, A Data Processor

3.2.2 Flow Graph Locality

In our proposed system, user can send requests to any server in this system for the data flow graph of its own data. If the requested server is the data controller to which this user uploads its personal information to, it will return the user with a complete graph containing every entity that has this user's personal data and corresponding edges indicating how its data is being passed along among different servers. Figure 4 shows an example of a complete graph returned by the data controller to the data subject (user with an id of "test_user_id_1"):

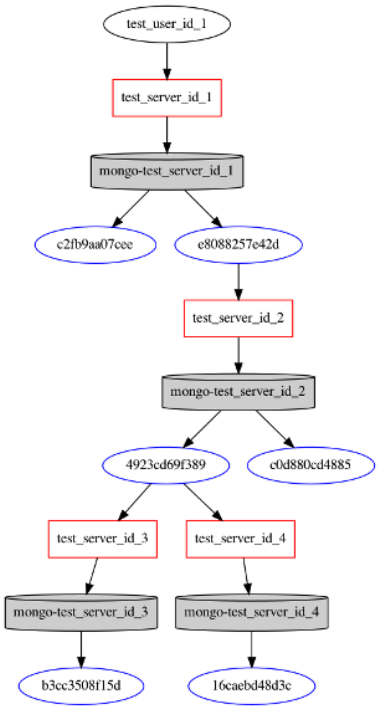

Figure 4: Visualization of the result sent back to the user

If the requested server is a data processor that gets the user data from the data controller or other data processors, it will only return a sub-graph starting from the predecessor of the processed user data to this server and all nodes / edges from there. Data processor has no previous knowledge about how user data flows.

Figure 5 shows the sub-graph starting from server 2 which is a data processor. Server 2 receives test_user_id_1's data from the data controller which is test_server_id_1 (in Figure 5). When user test_user_id_1 request data flow graph from server 2, it will only get the graph containing nodes and edges starting from server 2. The requested server will return an empty graph if it doesn't have this user's data.

Locality of flow graph protects user privacy by hiding previous paths of user data flow and only exposing the data flow starting from this server.

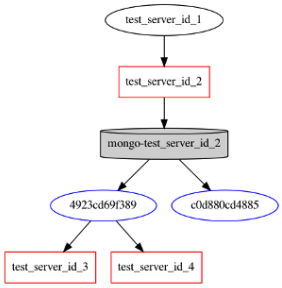

Figure 5: Visualization of the Data flow Representation on the Data Processor

3.2.3 Data Isolation

A server entity in the system can have multiple roles as it may be the data controller for one user and data processor for another user. In the case where there are more than one user in this system, a server entity can have multiple user data and process them separately with different computational nodes.

When a user requests flow graph from the server, the server only returns nodes and edges related to this user specified by the client user ID. Our system guarantees that the user only has access to its own data flow graph and has no knowledge of how other users' data is being processed and passed to other entities. Strict data isolation is imposed for every server in the system. The API for generating flow graph of the user data is only callable by the client with a specific user ID that ensures its identity.

4 Implementation

Our current project prototype is written in Golang and based on gRPC. Communication between two entities in the system is done through gRPC calls. All functionalities related to data flow construction / generation are implemented by the graph module, which is separate from the application itself.

4.1 gRPC API

Communication between user and server (data subject with data controller), server and server (data controller with data processor or data processor with data processor) are implemented with gRPC calls. Table 1 shows the gRPC calls implemented in our prototype.

4.2 Graph Internal Data Structure

Each server entity has a graph data structure that contains nodes / edges that represents user's data flow information. Table 2 shows a summary of the graph data structure used in our prototype and what it represents.

API	Description
SubmitPersonalInfo (SubmitRequest)	Invoked by user to submit personal info node
ProcessUserInfo (ProcessRequest)	Invoked by server to process user data
RequestEdges (EdgesRequest)	Invoked by server to recursively request edges
GetCombinedGraph (GetCombinedGraphRequest)	Invoked by user to get the combined graph

Table 1: gRPC Calls

Variable Name	Type	Description
id	string	server id
nodes	map[string]*Node	nodes related to a user
owners	map[string]*Node	processor of a node
predecessor	map[string]*Node	predecessor of a node
outgoingEdges	map[string][]*Node	outgoing edges from a node
successors	map[string][]*Node	successors from a node

Table 2: Graph Internal Data Structure

4.3 Graph API

API	Description
setPredecessor (userID string, predecessor *Node)	Set the predecessor of incoming user data
setOwner (userID string, controller *Node)	Set the controller ID of a user
createNodeIfNotExists (id string, label string)	Create a new node if given ID doesn't exist in the node list of this graph
passData (srcID string, dstID string)	Pass data within a server from the source node to the destination node
receiveData (srcID string)	Receives data from its predecessor
sendToOtherServer (srcID string, processorID string)	Send data to servers outside of this graph
combineAndSaveGraph (id string)	Combine and generate multiple graphs

Table 3: Graph API

Graph APIs are summarized and described at Table 3. They're used to keep track of internal and external data flow and construct the overall data flow graph.

4.4 Data Version Control

Nodes that represent user data have unique hash values used for version control within the system. Every read / write requests on a designated user data block will result in a change of hash value. When data is passed from one entity to another, this hash value is also going to change. Essentially, any operation on user data creates new data nodes with a same user ID and different hash value. User data nodes are identifiable by user ID and hash value.

5 Future Work

We have implemented 4 gRPC APIs for client / server interaction. They only provide limited functionalities for the moment and user data flow is unidirectional. For future work, we may extend gRPC calls to support more functionalities that enable bidirectional user data exchange. For example, after a user uploads its personal information to a data controller and it passes this data to a data processor, the data processor can return the processed user data with a new hash value back to the server. In addition, we would also like to use our framework on existing servers to evaluate our framework.

6 Conclusion

In conclusion, we have presented our proposed framework for data tracking across data controllers and processors. One of the advantage of adopting our framework is that it is a "plug-and-play" solution to existing applications. We designed and implemented our idea as a framework in the form of a library for users to adopt and use. It has several benefits to this design choice; 1) Existing applications only has to import our library to incorporate our framework, 2) Application developers will only add codes (by invoking our framework's APIs) and no modifications to the application's logic is required. This ensures that the application works as it should while benefiting from our framework. 3) Our framework can be easily distributed where it would be easy for data controllers and processors to adopt.

From the best of our knowledge, there is no other framework like ours that helps the data controller and data processor to keep track of data flow with the aid of graph visualization. Moreover, our framework is innovative and serves great purpose for data subjects as it allows data subjects to have an overview of where their data are actually used and went.

Task	Zhiyuan Lai	Yanzhi Xin	Atlas Yu
Project Proposal	✓	✓	✓
Presentation Slides	✓(contributed equally with Yanzhi Xin)	✓(contributed equally with Zhiyuan Lai)	-
Project Code - APIs	✓(contributed equally with Yanzhi Xin)	✓(contributed equally with Zhiyuan Lai)	-
Project Servers and Clients for POC	✓(contributed equally with Yanzhi Xin)	✓(contributed equally with Zhiyuan Lai)	-
Project MongoDB Functionalities	-	-	✓
Project Final Report	✓	✓	✓

Table 4: Summary of contributions

Acknowledgment

We would like to thank Professor Malte Schwarzkopf for his valuable guidance, insights, advice, encouragement and recommendation on this work. We would also like to thank Brown University and our classmates of this seminar class: CSCI 2390 Privacy-Concious Computer Systems for their insightful comments and feedback.

References

[1] Katriel Cohn-Gordon, Georgios Damaskinos, Divino Neto, Joshi Cordova, Benoît Reitz, Benjamin Strahs, Daniel Obenshain, Paul Pearce, and Ioannis Papagiannis. DELF: Safeguarding deletion correctness in online social networks. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, August 2020.

[2] European Commission. Reform of eu data protection rules, May 2018. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN.

[3] Tadayoshi Kohno Franziska Roesner and David Wetherall. Detecting and defending against third-party tracking on the web. *the 9th USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[4] Joanna Geary. Doubleclick (google): What is it and what does it do?, 04 2012. https://www.theguardian.com/technology/2012/apr/23/doubleclick-tracking-trackers-cookies-web-monitoring.

[5] Joshua Harry George, gRPC Specification, Anees Shaikh, and Jayant B. Kolhe. Use cases for grpc in network management. 2017.

[6] Kristi Hines. The absolute beginner's guide to google analytics, 06 2015. https://moz.com/blog/absolute-beginners-guide-to-google-analytics.

[7] Farhad Manjoo. I visited 47 sites. hundreds of trackers followed me., 08 2019. https://www.nytimes.com/interactive/2019/08/23/opinion/data-internet-privacy-tracking.html?smtyp=cur&smid=tw-nytimes.

[8] Janakiram MSV. Google's grpc: A lean and mean communication protocol for microservices, 09 2016. https://thenewstack.io/grpc-lean-mean-communication-protocol-microservices/.

[9] Miranda Wei University of Washington / University of Chicago; Madison Stamos, Sophie Veys University of Chicago; Nathan Reitinger, and Justin Goodman University of Maryland; Margot Herman University of Chicago; Dorota Filipczuk University of Southampton; Ben Weinshel University of Chicago; Michelle L. Mazurek University of Maryland; Blase Ur University of Chicago. What twitter knows: Characterizing ad targeting practices, user perceptions, and ad explanations through users' own twitter data. *Proceedings of the 29th USENIX Security Symposium.*, 08 2020.

[10] Help Net Security. 340 gdpr fines for a total of €158,135,806 issued since may 2018, July 2020. https://www.helpnetsecurity.com/2020/07/16/gdpr-fines/.

[11] Supreeth Shastri, Vinay Banakar, M. Wasserman, Arun C. S. Kumar, and Vijay Chidambaram. Understanding and benchmarking the impact of gdpr on database systems. *Proceedings of the VLDB Endowment*, 13:1064 – 1077, 2020.