Project Report: Consumer Cloud Storage Providers as Solid Pods

Lauren McKeon, *Brown University* Oren Kohavi, *Brown University*

Abstract

In this paper, we introduce Google Drive Pod (GDP) a proof-of-concept application designed to enforce decentralized data management within the healthcare sector and demonstrate the feasibility of building a secure non-trivial application which uses Google Drive as a data store. Currently, data management in the healthcare space is inadequate; information is often fragmented, dispersed, and lacks proper security measures. A specific challenge involves the transfer of medical records to new healthcare providers, ensuring data privacy during the transfer process, and establishing the right to delete information once a patient has discontinued services with a provider. Public confidence in organizations responsible for storing and sharing health information has reached an all-time low [1]; to improve this process it is crucial to introduce greater transparency and user control.

GDP provides a centralized place to store user data with fine-tuned data-sharing abilities. The GDP serves as a simple interface between healthcare providers' and patients' personal health records. Using the GDP Interface users host all their medical data in a Google Drive folder and can fine-tune the sharing permissions associated with data points within the interface. Each medical provider is given a unique Google Drive link with access to their specially tailored medical data in the form of a JSON. To demonstrate functionality in this proof of concept, we store and retrieve a set of general health data formatted in a common health format, and show that this data is both secure and theoretically interoperable with any medical system.

1. Introduction and Background

1.1. Motivation: Solid

The Solid paper introduces the concept of decoupling user data from applications, presenting a unique approach to data ownership and interoperability. Solid is a framework for social web applications that introduces the idea of storing data in a pod as opposed to directly in the web applications themselves. The primary goal of the Solid framework is to put users back in control of their data and to break down the data silos that currently exist.

However, Solid has not seen widespread adoption. From the user perspective, a significant challenge arises due to the limited number of available POD providers, all of which provide subpar usability and access control. Additionally, despite privacy being a main goal of Solid, the current protocol has gaps in its security and privacy methods that make Solid incompatible with high-security data such as medical information. Specific shortcomings of the Solid protocol include lack of fine-grained access controls, and a lack of correct and effective use of cryptography to ensure "confidentiality, authenticity and/or integrity of data when stored and/or exchanged." [2]

1.2. Our Solution: Leveraging Google Drive

We propose the viability of using consumer-grade cloud storage as an alternative for POD providers, hoping to alleviate the shortcomings of pods. Google already has a strong reputation surrounding data privacy and fine-grained access control, which can be leveraged to create a user-friendly "POD" secure enough to be used for users' medical data.

1.2.1 The Google Drive API

Firstly, Google Drive's high-quality access control and rich API make it ideal for this purpose. In GDP we leverage both the general Google Drive API and the Google Picker API. Upon logging into the GDP, a new folder named Google Drive Pod is created in the user's Google Drive. Using specific developer scopes we ensure that our application can "See, edit, create, and delete only the specific Google Drive files you use with this app." This ensures that our application only has access to data that the user provides and the user is made aware of this upon logging into the application.

1.2.2 HIPAA Compliance for Medical Data

The Health Insurance Portability and Accountability Act of 1996 (HIPAA) is a federal law that requires the creation of national standards to protect sensitive patient health information from being disclosed without the patient's consent or knowledge. All health apps that handle or process personal health data must be HIPPA compliant. The second reason why Google Drive is ideal for this purpose is that it can support HIPAA compliance. Our proof of concept does not include the HIPPA compliance functionality; however, in future work, this is a feature that would need to be implemented before integrating our application with medical providers.

1.2.2 GDPR Compliance

The General Data Protection Regulation (GDPR) is a set of regulations designed to protect the privacy and security of personal data, including personal health records. The global reach of Google has mandated Google Cloud, functioning as a data processor, to align with GDPR regulations. Google commits to processing all customer data in compliance with GDPR and provides additional security features and documentation that help both developers and individuals protect their personal data.

Article 9(A): Processing of Special Categories of Personal Data

Article 9 prohibits the processing of highly personal data data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person’s sex life or sexual orientation unless explicit consent is provided by the data subject.

Article 17: Right to Erase

Article 17 expresses the data subject's right to be forgotten, stating that a data subject has the right to request that a data controller erase all personal data attached to the subject. Deletion abilities must be implemented in modern day applications to be in compliance with this article.

Article 20: Right to Data Portability

The concept of data portability aligns with the GDPR's objective to shift control of personal data from businesses to the data subject. The right of data portability ensures that individuals have the right to extract the information an organization has collected on them to simply obtain or reuse for their own purposes.

1.2.4 Users Trust

Another reason why Google Drive is ideal for this situation is the pre-existing trust that users have in Google; Most users already trust Google with a significant portion of their digital data, relying on services like Gmail, Google Calendar, and Google Photos. The seamless integration of these services and the convenience they offer have contributed to a sense of reliability and dependability among users. This established trust becomes a compelling reason for users to consider Google Drive as a preferred option when it comes to storing and managing their medical data.

Additionally, Google's commitment to robust security measures, continuous updates, and user privacy enhances the overall credibility of Google Drive.

1.4 Key Limitations

The key limitations of this work stem largely from its position as a first draft proof of concept for the more general concept of storing sensitive data in an application-managed segment of a user’s Google Drive. Notably, the personalized sharing link that is given to the medical provider currently has the ‘anyone can access’ permission attached to it – This means that if the link is leaked, privacy is not preserved. Future development should convert this link to one which is restricted to a specific set of email addresses, further relying on Google’s robust security systems to enforce privacy. Another limitation is that the sharing links produced do not automatically update when a patient’s master record changes, and therefore new data must be proactively shared with every medical provider.

2. Google Drive Pod Design

Google Drive Pod is designed to use Google Drive’s robust permissions system in order to enforce privacy. Google Drive’s privacy and security have been deployed and iterated upon for over a decade, which imputes a high degree of confidence that they are correct. GDP first creates a folder structure within the user’s Google Drive, and as medical data is imported, creates JSON-formatted files within this structure to store both the medical data itself – formatted in the common FHIR JSON format – and metadata about sharing permissions.

When data is requested to be shared with a medical provider, GDP creates a separate file in the user’s Google Drive which contains only the selected subset of information. Creating a separate file simplifies privacy, since if the file creation operation is correct (i.e. does not include information into the file that should not be there), then all subsequent operations are guaranteed to use only the selected subset of information since they rely on the newly-generated file.

The Google Drive Pod interface is a web-based application which accesses the Google Drive API directly and performs all computations locally – This contributes to the privacy of GDP, as local data processing eliminates the server as a potential point of failure.

This application is broken down into two core pieces; a patient view and a doctor view. For demo purposes, users can toggle between these two views using the nav bar across the top of the application. These views are entirely isolated from each other in terms of data sharing: the only data the doctor view can access is that which is provided through a sharing link.

2.1. Patient View

The patient view consists of a single page featuring three primary components. The leftmost panel holds six user action buttons. From top to bottom, they are Google *Login, Load from Drive, Load Specific File*, *Load from URL*, *Save to Drive*, and *Delete Data*. These buttons interface with Google Drive and allow users to upload data to their accounts. The middle panel, titled *My Health Data*, serves as a visual tool summarizing and presenting the user's medical information as text. The rightmost panel, titled *My Data Sharing*, allows users to selectively choose which data attributes from their uploaded medical data they want to share with each of their medical providers. This panel has a *Doctor Name* text box, a *Select Data to Share* dropdown, and a *Share with Provider* button.

The components in each panel work together to provide a seamless user experience.

2.1.1 General Login & Account

The top button, *Google Login,* interfaces with Google Auth, allowing the user to log into their account using their Gmail. During this login process, the user will be prompted with the security implications of linking their Google account with GDP; the GDP can "See, edit, create, and delete only the specific Google Drive files you use with this app." If Google authentication is successful a new Google Drive folder titled DrivePod will be automatically created in the user's Google Drive. When the user logs in to GDP Upon creation this Google folder will be empty. Generally speaking, this folder will hold all of the user's medical data that is accessed within the GDP in the form of JSON files. The next section will discuss in detail how data is uploaded and stored within GDP.

2.1.2 View and Upload Personal Health Data

The second, third, and fourth buttons in the left panel are responsible for viewing and uploading personal health data into the GDP application. The third button, *Load Specific File* opens the Google Picker interface, allowing the user to import an existing medical record from anywhere in their drive. The fourth button, *Load from URL,* allows a user to load from an external hosted JSON link from somewhere on the web. In our Proof of Concept, we use git to host sample FHIR JSON files. After a file is uploaded into the application using either *Load from Drive* or *Load from URL* a summary of the data uploaded is shown to the user in the Middle panel, *My Health Data.* The buttons upload the medical data into the application but do not save the record to the user's account. The fourth button, *Save to Drive,* enables this functionality. If the user wants to save the health record to their account for future use, they must click the *Save to Drive* button after uploading. Under the hood in the DrivePod folder, anytime a new file is uploaded and saved back to drive, the information is added to a master record.json file that holds all of the user's medical data that is stored in the GDP. The second button, *Load from Drive* allows users to see a summary of the medical data that they have stored in GDP and even showcases to users what data has been recently updated to the user.

2.1.3 Data Sharing Options

The rightmost panel in the patient view takes care of the Data Sharing capabilities. After the user logs in and uploads some data, the *Select Data to Share* dropdown will auto-populate to represent each data attribute that exists in the user's master record.json file. From here users simply enter a health provider's name in the *Doctor Name* text box and select exactly what data attributes the user wants to share with that specific doctor. As an example, let's say my master records have my Name, Address, Blood Type, Dental X-ray Results, and Foot X-ray results. These are all in my master record.json file and come up as options to share with my Dentist but I can select that my Dentist can only see the Name, Address, Blood Type, and Dental X-ray Results. Once the attributes are chosen the user clicks the *Share with Provider* button and the user is provided with a link to share with that specific provider, in my example above the dentist. This link will point to a newly created JSON file in the DrivePod folder that contains the chosen subset of the master record.json data. If the user changes their mind about sharing a specific attribute, they can re-enter the doctor's name and change the selected attributes, the link will remain the same and the data in the file will automatically be updated.

2.1.4 The Right to Delete

The Google Drive pod interface makes it easy to exercise the right to delete. Firstly, the bottom button on the left panel, *Delete Data* will clear the data in memory and subsequently clicking *Save to Drive,* deletes all of the user's medical data stored within GDP, meaning it wipes the master record.json file and any tailored doctor JSON files stored within the DrivePod folder. Currently, the interface lacks more granular deletion capabilities. However, users can still manage this functionality directly within Google Drive. If a user wants to revoke data access to a specific provider, the user can delete that specific JSON record directly from the DrivePod folder. For example, continuing from our dentist example above, if the user changes dentists, she can delete the dentist.json file to revoke data access from her old provider.

2.2. Doctor View

The doctor's view is very simple, having only one functionality point. There is a search bar that allows the doctor to search for their users' personalized data records. Pasting in the link provided by a patient allows the doctor to view the data points shared.

3. Implementation

Our project was implemented using the next.js framework using Javascript and CSS. The frontend of our patient view is in patientPage.js file and the frontend of our doctor view is in the doctorPage.js file.

Our core backend functionality of our implementation consists of two primary components, the integration with Google Drive and the data parsing capabilities. To test our implementation we use a test set of medical data.

3.1. Google Drive Integration

To integrate Google Drive into our application, we first had to create a new Google Project in the Google Developer Console. To enable Google Authentication within our app, we created a new Oauth Client, giving us a Client ID to use in our application when implementing authentication services. Next, we enabled both the Google Drive API and Google Picker API, providing us with an API key to use when implementing the file uploading/downloading from drive functionalities.

The gdrive.js file holds all of the functions that interface with Google Drive. We use the Axios get, post, and fetch requests to retrieve and send data back and forth between Google Drive.

3.2 Data Parsing

The data parsing functionality is implemented in the parsing.js file. This file is responsible for parsing, merging, summarizing, and creating subsets of the FHIR-formatted JSON data stored in Google Drive.

3.3 Dummy Medical Records

In order to test the functionality of our implementation, we hosted sample FHIR medical data on a GitHub pages site to use for testing purposes. There are three primary files that we used for functionality testing: FHIR_Reference, OnlyMeds, and WithMeds. FHIR_Reference is taken from the HL7 specification website [3], and OnlyMeds and WithMeds are medication data and a union of medication data plus FHIR_Reference, respectively.

4. Evaluation

4.1 Functionality

In terms of functionality, our proof of concept met expectations; we were able to create an application performing a secure, non-trivial computation with Google Drive as a data store, while delivering better privacy and access control than Solid Pods.

4.2 Usability: Security

Security for Google Drive Pods is largely offloaded to Google's competent access control system – by storing data as files in a user's drive, Google's robust access control mechanisms can protect them (both from modification and also from unauthorized access by impersonation of the account owner).

Additionally, security measures were implemented to ensure compliance with the GDPR data protection laws. Figure 1 illustrates which interface features support each GDPR Articles mentioned.

GDPR Article	Article 9(A)	Article 17	Article 20
Supporting Features	*Select Data to Share* dropdown	*Delete Data + Save to Drive OR direct deletion in DrivePod Folder*	masterRecord.json feature

Figure 1

4.2 Usability: Runtime Analysis

To get an overview of the runtime of the system, we performed runtime analysis on select user-facing action buttons. We evaluated the runtime performance of user elements that have status load indicators; *Save to Drive, Load From Drive,* and *Share with Provider.* The runtime for each button for three different scenarios, each scenario was tested 4 times. The average for each scenario (e1, e2 & e3 on the graph) is shown on the graphs. Scenario 1 (labeled as e1 on the graphs) tests the runtime of the buttons when no data is loaded into the application, serving as a base case or error case scenario. Scenario 2 (e2), evaluates the runtime while uploading, saving a user's initial health record, in this case, the FHIR_Reference dataset. For the sharing data with the Provider graphs, e2 represents making the first doctor record. Scenario 3 represents adding additional data to the initial record, in this example we added the Meds record to the existing FHIR_Refrence data. For the sharing data with the Provider graphs, e3 represents adding these additional attributes to the doctor's existing record.

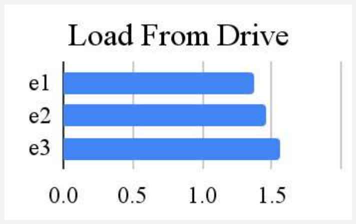

Figure 2

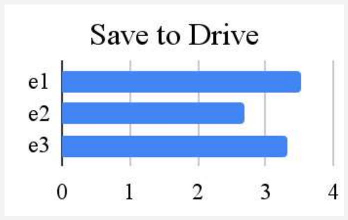

Figure 3

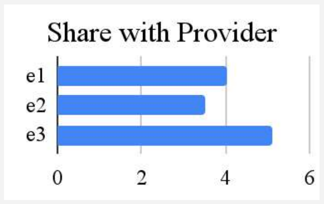

Figure 4

From this analysis, it can be concluded that further optimization of this system needs to be conducted before launching to production. Loading from Drive takes over a second in each scenario and Saving to Drive and Sharing with Provider both take over 2 seconds in each scenario. For each button, the runtime sees a noticeable increase from scenario 2 to 3, indicating that as the data stored in GDP increases the runtime of these actions will also increase. Given the added Meds records only added one additional data point to the overall record, this causes some concern for the scalability of this system.

Individual Contributions

Overall, a good partnership.

Oren: Took the lead on the data parsing and merging section. Collaborated on Google Drive Integration.

Lauren: Took the lead on the overall design, UI/UX, and Button functionality. Collaborated on Google Drive Integration.

References

[1] Platt JE, Jacobson PD, Kardia SLR. Public Trust in Health Information Sharing: A Measure of System Trust. Health Serv Res. 2018 Apr;53(2):824-845. doi: 10.1111/1475-6773.12654. Epub 2017 Jan 18. PMID: 28097657; PMCID: PMC5867170.

[2] Esposito, C.; Horne, R.; Robaldo, L.; Buelens, B.; Goesaert, E. Assessing the Solid Protocol in Relation to Security and Privacy Obligations. Information 2023, 14, 411. https://doi.org/10.3390/info14070411

[3] https://build.fhir.org/patient-example.json.html