CODI 1.0 was supported by the Centers for Disease Control and Prevention (CDC) and the Office of the Assistant Secretary for Planning and Evaluation (ASPE) of the U.S. Department of Health and Human Services (HHS). The content and conclusions of this document are those of the author(s) and do not necessarily represent the official views or endorsement of the CDC, ASPE or HHS.

Prepared for:

Centers for Disease Control and Prevention

CMS Alliance to Modernize Healthcare (Health FFRDC)
A Federally Funded Research and Development Center

Childhood Obesity Data Initiative

Task Order No. 200-2016-F-89363

CODI Privacy-Preserving Record Linkage Goodness of Fit Analysis

Final

Version 1.0

December 16, 2019

MITRE
7515 Colshire Drive
McLean, Virginia 22102

Record of Changes

Version	Date	Author / Owner	Description of Change
0.1	13-Nov-19	A. Gregorowicz / Health FFRDC	Initial draft
1.0	16-Dec-19	A. Gregorowicz / Health FFRDC	Incorporated feedback from CDC

Executive Summary

The CMS Alliance for Modernizing Healthcare Federally Funded Research and Development Center (Health FFRDC) partnered with the Centers for Disease Control and Prevention (CDC) to support the Childhood Obesity Data Initiative (CODI), the goal of which is to enhance data capacity for research, evaluation, and public health surveillance. CODI is structured to enhance existing infrastructure, rather than build new. As such, CODI is designed to operate within distributed data networks (DDNs). A weakness of current DDNs is that information for patients shared across organizations is not linked, limiting researchers' ability to monitor individuals' activities and outcomes over time and across settings.

To address this limitation, the CODI Collaborative Work Group (CCWG) chose to use privacy-preserving record linkage (PPRL). PPRL allows for deidentified record linkage across participating organizations (i.e., data partners). The PPRL process includes each data partner garbling patients' identifiable information in a secure way and then sharing that deidentified information with a third party called a linkage agent. The linkage agent identifies shared patients across the institutions based on similar garbled information and then returns to each data partner a unique identifier that is reported by all data partners in response to future research queries.

There are a number of open source and commercial PPRL packages. The Health FFRDC evaluated a subset of these in order to make recommendations on the PPRL package that best fit CODI. A CDC Fellow, supported by CCWG members, selected the subset of tools tested by the Health FFRDC. These included open source anonlink and the R PPRL package, as well as a commercial product called CURL (Colorado University Record Linkage), developed by the University of Colorado.

Evaluation Process

The Health FFRDC team developed three test scenarios and created synthetic patient records to support each scenario.

- **Scenario 1:** Examined each package's ability to match individuals across three synthetic data partners.
- **Scenario 2:** Explored each package's performance on the more complex process of correctly matching records when sibling information is present. This is a common challenge with pediatric record linkage. The linkage system must correctly match records for an individual while not incorrectly linking that person to the records of their brothers or sisters. In this scenario, data elements such as given name, sex, address, and/or date of birth have been varied. Some data elements are the same for the test case, such as family name or parent contact information.
- **Scenario 3:** Assessed the packages' performance on a large set of records with no matches to determine how the linkage process performed at scale.

The Health FFRDC team also documented the level of difficulty associated with package installation and package modifications, taking into account the differences in developer support provided with open source and commercial software.

Findings

- **anonlink:** This package performed better than CURL on Scenario 1 but performed poorly on Scenarios 2 and 3. Installation was simple, although the Health FFRDC team had to create software to automate some processes to conduct testing.
- **CURL:** This package performed better on Scenario 2 than anonlink but was unable to complete the matching job for the third scenario. There were some components of the software received from the developer that initially did not work and required updates. This package does not offer an Application Programming Interface so supplemental software to automate some processes was not feasible.
- **R PPRL package:** The Health FFRDC team was unable to get this package to perform hash functions with the Scenario 1 test data set. The evaluation was not attempted for Scenario 2 or Scenario 3.

The results described above used anonlink in a situation where it made a single attempt at matching using all available data elements. Because none of the packages performed well across all scenarios, the Health FFRDC changed the anonlink approach, running it multiple times, matching smaller sets of identifiers in a sequential pattern. With these adjustments, anonlink out-performed CURL's best performance in Scenario 2 and it performed adequately in Scenario 3.

Recommendations

Based on these findings, the Health FFRDC recommends that CDC:

- Select anonlink to support CODI record linkage.
- Conduct additional experiments to identify the optimal configuration of anonlink to support CODI.

Table of Contents

List of Figures

List of Tables

1. Introduction

Research that assesses childhood obesity interventions is limited because researchers cannot easily link pediatric health-related data stored across different health information systems to assess interventions' effectiveness. One of the goals of the Childhood Obesity Data Initiative (CODI) is to establish longitudinal records for children across systems and settings that can be used to answer research questions.

In order to create longitudinal records, individuals must be matched across organizations. Further, CODI seeks a solution to matching individuals that can scale to a national level. To do this, the CODI Collaborative Work Group selected a privacy-preserving record linkage (PPRL) approach, which allows matching to take place without disclosing personally identifiable information (PII) outside of an organization's boundaries.

The Centers for Disease Control and Prevention (CDC) partnered with the Health FFRDC, operated by The MITRE Corporation, to conduct an evaluation of selected PPRL tools and to make recommendations about which tool(s) would best fit CODI infrastructure and research requirements. This document describes the process for evaluating PPRL tools. It describes the requirements, information gathering process, and testing of selected tools. The document concludes with a tool recommendation and next steps that can be taken to prepare for PPRL solution deployment.

1.1 Problem

The process of matching individuals, in the absence of a shared, unique identifier, often requires organizations to exchange information with each other or a third party to participate in a matching process. Matching occurs by comparing that shared PII to see if there are similarities in demographic attributes such as name, sex, date of birth, or address.

Although this approach to matching works, it has its drawbacks. First, there is always increased risk of privacy breaches when PII is shared outside organizations' firewalls. Second, this approach does not scale well: while a small number of partners may agree to share information with each other, it is unlikely that large numbers of organizations would be willing to exchange PII nationally, outside of a national mandate. It is similarly unlikely that consolidating PII by a nationwide third-party matcher would be appealing. In order to conduct matching at scale, there must be an approach that does not involve exchanging PII beyond organizational boundaries.

1.2 Privacy-Preserving Record Linkage Solution

Alternative techniques exist to solve the issue of identity matching without exchanging PII directly. These techniques are categorized as privacy-preserving record linkage. The basis for this class of solutions is that the PII is obfuscated, or garbled, prior to transmission beyond an organizational boundary for matching. The garbling of information takes place through a series or prescribed steps that makes it nearly impossible for an outside party to recover the PII, but still allows for the establishment of links across organizations.

PPRL solutions allow for "blind" matching. In this case, the third party is provided access to garbled data, but is unable to view PII. This party then compares the obfuscated information to establish linkages. Figure 1-1 illustrates this process.

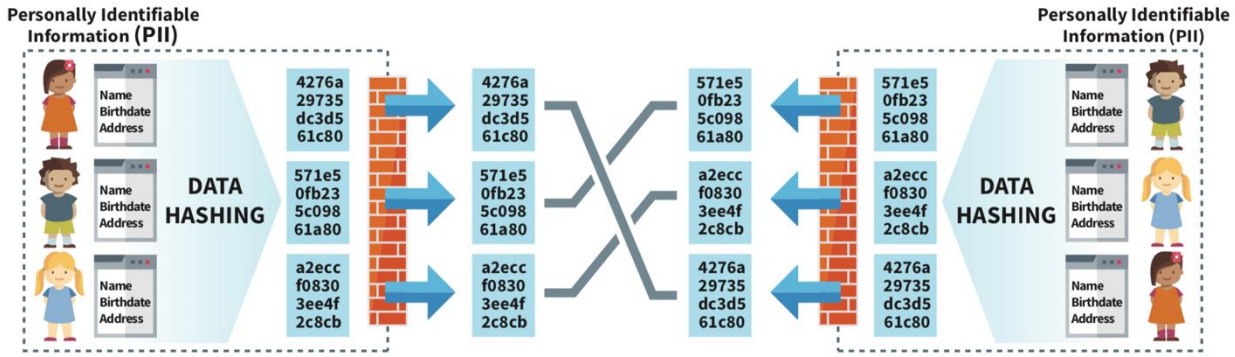

Figure 1-1. Blind Matching with PPRL

The third party conducting the matching assigns an identifier when a linkage is found and communicates the identifier back to the participating organizations for use in establishing longitudinal records.

The blind matching process can vary in sophistication. A simple approach requires exact matches on the garbled information. This technique is of limited usefulness when working with real-world data as it is unable to handle variations in information such as typos or nicknames. More sophisticated techniques allow for partial matches by examining the similarities in the garbled information. Both approaches will be explored in Section 2.

CODI will use PPRL to establish linkages across organizations without sharing PII. This approach can be deployed at a greater scale. With PPRL, the third-party matching organization is not a large warehouse of PII, but is instead working with garbled, deidentified data.

1.3 PPRL Roles and Responsibilities

There are several key roles to support the PPRL process within CODI.

- **Data partners:** Organizations participating in CODI that own the data that will be shared; these organizations will conduct hashing, share encrypted garbled data for record linkage, and incorporate the unique identifiers derived from the record linkage process into their data warehouses.
- **Trusted third party (TTP):** This organization must be trusted by all data partners to act in their best interests; this is usually supported through a legal agreement between the data partners and the TTP. This organization's role is to share the hashing configuration files (see Section 2 for more details) and encryption key with the data partners.
- **Semi-trusted third party (STTP):** This organization is provided with encrypted hashed values from the data partners, conducts the record linkage process, and shares the unique linked identifiers back to each data partner. To ensure the hashed values remain encrypted, the STTP must not have access to the encryption keys.

1.4 Goals for Selected Tool

There are many PPRL solutions available on the market, as both open source and commercial software packages. When selecting a tool for usage in CODI, the ideal solution must meet several goals.

- **Sufficient PII garbling:** CODI prioritizes preserving the privacy of the children studied by CODI research. The tool must be able to perform reasonable identity matching, without direct access to PII, by working on the garbled data.
- **Open source:** CDC's ideal PPRL solution is open source, with no licensing costs. This would allow for national expansion of CODI infrastructure without the expense of software licensing.
- **Deployable with minimal Information Technology (IT) support:** Tools requiring extensive IT support limit scalability. In addition, the non-clinical community organizations that CODI wishes to reach may not have dedicated IT resources. If a tool requires substantial installation and maintenance effort, it is unlikely to achieve adoption with these partners.
- **Functions well with a pediatric population:** Because CODI is focused on supporting childhood obesity research, the PPRL tool must perform well on a pediatric population. Identity matching is more difficult in a population where there is a greater presence of child siblings who often share a significant portion of their demographic information, such as family name, parent, and contact information. Additionally, techniques for matching adults, such as referencing information from credit bureaus, do not apply in this domain.

2. Privacy-Preserving Record Linkage Approaches

PPRL is the process of matching individuals based on garbled information. Matched records are assigned a globally unique LINKID, which can be used to link individuals across organizations.

The matching process typically involves the following steps:

1. A TTP shares configuration information with each data partner. The TTP also provides encryption keys to the data partners.
2. Each data partner creates a garbled data set by:
    - Extracting PII from its operational database.
    - Using the encryption keys provided by the TTP, passing the PII through a hashing process that will garble the information.
    - Sharing the garbled data with the STTP.
        - It is important that the STTP performs only matching and does not have access to the encryption keys. This arrangement prevents any party from having access to all of the PII being used in the matching process.
3. The STTP develops LINKIDs by:
    - Determining which garbled values correspond to the same individual.
    - Establishing a unique LINKID for each individual.
4. The STTP sharing the LINKIDs with each data partner.

Each data partner stores the LINKIDs for future research queries.

A key aspect of PPRL is the method used to garble the PII, which impacts the capabilities of the STTP to perform matching. The following sections describe different matching approaches that may be used.

2.1 Deterministic Matching

With deterministic matching, an STTP performs exact matching based on information that is garbled through hashing by the data partner. The inputs to the hashing process are usually combined with a random value, called a salt, to prevent a class of attacks that can be used to reveal the identities.

2.1.1 Hashing

Hashing is a mathematical function with two key properties. First, the same inputs always produce the same hashed (i.e., garbled) output. Second, given the output, it is nearly impossible to determine which inputs were used.

Hashing is an integral component of PPRL because if two hash values are identical, then the inputs that produced those hash values must also be identical. Thus, if two data partners have information about John Doe, they will hash John Doe to the same value. The STTP can therefore establish a globally unique LINKID for John Doe without receiving any PII for John Doe.

One weakness of hashing is that an adversary can independently create hash values for an individual. For example, by hashing every person in the phone book (including John Doe), the adversary can learn which data partners have information about John Doe if the adversary had access to the hashed data. To protect against this kind of attack, a "salt," or encryption key, is added to the inputs before hashing.

2.1.2 Use of an Encryption Key to Mitigate Attacks

A salt, or encryption key, is a randomly generated value provided as an extra input to the hashing function. The addition of this value causes a change in the output of the hashing function and prevents attacks, like the one described in Section 2.1.1, because the adversary's hashed phone book values will no longer yield the same output as the value with the encryption key.

The encryption key must be kept secret to prevent a phone book style attack. To do this, the encryption key is typically generated by a TTP and shared via a secure channel with data providers.

When using an encryption key, it is important that all data partners use the same encryption key value as an input to the hashing functions. If organizations used different encryption keys, no matches would be identified.

2.1.3 Hashing Example

PPRL techniques often generate multiple hash values for a given record. These hash values are created by combining different data elements from the patient's record and hashing the resulting values. Figure 2-1 presents an example of what hashed values derived from a synthetic individual record might look like. Different data elements will be combined to generate multiple hashes for the individual. In this example, three hashes were generated for this record.

Sample Individual Record

Data Element	Value
First Name	John
Last Name	Doe
Date of Birth	7/4/2005
State	MA
Insurance Number	23-0009876

Hash Combination Elements	Combination Value	Hashed Value
First Name + State + Insurance Number	JohnMA23-0009876	1bf6e725d482e28d774a987688c59e4faca0213f
Last Name + Date of Birth + State	Doe7/4/2005MA	ccc20f4d42210c781e1be805ce6162a8c718059c
First Name + Last Name + Date of Birth	JohnDoe7/4/2005	0a49bf0326dbdfd3e29ec18a50bc71701c8c42d2

Figure 2-1. Hashing Example from Sample Individual Record Without an Encryption Key

As described in Section 2.1.1, these hash values could be calculated by an attacker with knowledge of PII. To prevent these attacks, an encryption key value is added as an input to the

hashing function to prevent an attack. Table 2-1 presents a random set of characters representing a sample encryption key.

Table 2-1. Example Encryption Key Value

Salt	tm0eoRWdkW

Applying the encryption key to the hashes displayed in Figure 2-1 would now yield the results presented in Table 2-2.

Table 2-2. Example Hash Values with Encryption Key

Data Elements	Value	Hashed Value
First Name + State + Insurance Number + Encryption Key	JohnMA23-0009876tm0eoRWdkW	99f04beb9494b66ba0530b1c6c4eae91c8ac45dc
Last Name + Date of Birth + State + Encryption Key	Doe7/4/2005MAtm0eoRWdkW	6759019832dbdfbe300fc257f0d0dbbc57378d5b
First Name + Last Name + Date of Birth + Encryption Key	JohnDoe7/4/2005tm0eoRWdkW	9b4d8ebae857caef82949a6fd87f4522afcb496b

Use of the encryption key changes the hash values. This prevents an attacker from determining whether a patient has information stored at a given organization, based the attacker's own hash values generated from known PII.

2.1.4 Limitations of Deterministic Matching

Following this process of matching, it is possible to identify exact matches in information. For example, if Organization A has a record for "Jon Doe" and Organization B has a record for "Jon Doe," it will create a match. This approach does not handle any variation in the information. Referring to the previous example, if Organization B has a record for "Jo**h**n Doe," this subtle change will cause the match to be missed. Because scenarios like this are probable in real-world information systems, a more robust approach to matching is required.

2.2 Probabilistic Matching

In order to handle variations in demographic information, an alternate method for garbling information is necessary, called probabilistic matching. Hashing with an encryption key is still used for this technique. In contrast to deterministic matching, PII is fragmented prior to being salted and input into the hashing algorithm. The outputs of the salted and hashed fragments are then used to construct one or more data structures called Bloom filters. These Bloom filters can then be compared to determine if there is a match. The comparison of Bloom filters does not require an exact match, allowing for variation in the underlying data.

2.2.1 Building Bloom Filters

A Bloom filter is a data structure that offers efficient storage of information and is often used for probabilistically testing set membership. A Bloom filter starts as an array of bits at a specified

length, with all bits set to 0. An item is added to a Bloom filter by passing it through multiple hashing functions, or through a single hash function with multiple encryption key values. This results in multiple output values. The output values are each divided by the length of the Bloom filter and the remainder of those operations are then used to set the positions in the Bloom filter to 1.

In a PPRL process, the information is fragmented prior to being input to a hashing function. For example, the name "John" could be fragmented into "Jo", "oh" and "hn". Table 2-3 presents an example of how a Bloom filter could be created for the name "John." In this case, two encryption key values are used. The example Bloom filter is 64 bits long.

Table 2-3. Example Bloom Filter Construction

Name Fragment	Salt	Combined Value	Hash Value	Bit to Set (Hash mod 64)
Jo	tm0eoRWdkW	Jotm0eoRWdkW	f8c6c76e3d4f69b42ed2d233591212fe0187c106	6
Jo	sLJp9wvfpy	JosLJp9wvfpy	177cfa71b1826df0968d343410bd88a199969731	49
oh	tm0eoRWdkW	ohtm0eoRWdkW	2b21bb44fff52320149bddd35266bfbbf1680ba6	38
oh	sLJp9wvfpy	ohsLJp9wvfpy	d7dd3104605c9680f6091c1c27f4736d5673fee6	38
hn	tm0eoRWdkW	hntm0eoRWdkW	3906a4eb6bbc4edd938e57895a657f5d39ba6c90	16
hn	sLJp9wvfpy	hnsLJp9wvfpy	be4d1c934e4d74b2139f8f4a55fc8ec0ec4e2689	9

It should be noted in the example calculations in Table 2-3, even though two different encryption keys were used for the name fragment "oh", which resulted in two different hash values, passing these through the modulo operation resulted in setting the same bit. While Bloom filters are space efficient, these kinds of situations where different inputs result in the same bit being set are possible. When two different inputs generate the same output, it is referred to as a collision.

The resultant Bloom filter from the Table 2-3 example is: 0000001001000000100000000000000000000100000000001000000000000000. This shows five cases when the value for the filter has been set to "1", as indicated in Table 2-3. Also, the position of each bit as based on a zero-index. In this example, the first name fragment will set the value at the index position of 6, which is the 7th bit in the Bloom filter.

2.2.2 Comparing Bloom Filters

As mentioned previously, Bloom filters are often used to check for probabilistic set membership. The Bloom filter generated in the previous section allows for checking for the presence of certain name fragments. As an example, the name "Johnathan" could be broken down into fragments, and following the same encryption key and hashing procedure, the presence of the "Jo", "oh" and "hn" fragments would be reported as true. However, fragments "na," "at," "th," "ha," and "an" will likely report as false. The reason that it is not possible to definitively state that these fragments will return false is due to the possibility of collisions, as discussed in the previous section.

Instead of testing for the presence of name fragments for "Johnathan," it is possible to construct a separate Bloom filter from this name using the same two encryption keys that result in two different 64-bit arrays. The array generated for "John" and the array generated for "Johnathan"

can be compared by calculating a Sørensen–Dice coefficient, sometimes referred to as a Dice coefficient or F1 Score. Calculation of this metric starts by tabulating the values in Table 2-4.

Table 2-4. Dice Coefficient/F1 Terms

Value	Definition
True Positive (TP)	The bit at a given position in the first Bloom filter is set to 1 and the corresponding bit in the second Bloom filter is also set to 1
False Positive (FP)	The bit at a given position in the first Bloom filter is set to 0 and the corresponding bit in the second Bloom filter is set to 1
False Negative (FN)	The bit at a given position in the first Bloom filter is set to 1 and the corresponding bit in the second Bloom filter is set to 0

These terms can then be used in the equation in Figure 2-2.

$$DSC = \frac{2TP}{2TP + FP + FN}.$$

Figure 2-2. Dice Coefficient Equation

DSC provides a value between 0 and 1. Comparing Bloom filters with the exact same inputs would result in a coefficient of 1. Comparing filters created from dissimilar inputs will result in a value closer or equal to 0. Returning to the previous example, if a record has a given name of "John" and another record has a given name of "Johnathan", Bloom filters can be created for these two separate records. These can be compared, and the resulting Dice coefficient value can be used to determine if the records match the same individual.

Using this approach, data partners can build Bloom filters based on individuals' identity information. The Bloom filter approach can be accomplished through a single Bloom filter that includes all identity information or with multiple Bloom filters for each individual. In the latter approach, filters are constructed for a subset of data elements or even a single data element.

Data partners transmit the Bloom filters they created to the STTP. These filters can then be compared between data partners. Filters that have a Dice coefficient above a particular number established for the matching process are considered a match.

2.2.3 Limitations of Probabilistic Matching

Several issues can arise when using probabilistic matching, most of which are related to the use of Bloom filters. First, the storage space efficiency provided by Bloom filters is obtained through the loss of information. As an example, if a child and parent share the same family name, a Bloom filter created using the fragmenting approach will be the same for just the child's family name and a filter build using the child and parent's family name. This can lead to situations where increasing the number of data elements into the matching process leads to little or no benefit.

Secondly, the size of the Bloom filter has an impact on matching performance. Adding fragments into the Bloom filter changes bits from 0 to 1. There is no process that changes those bits back to

0. If too much information is placed into the filter, there is the potential that many or all bits will be set to 1. Referring back to the example, 64-bit Bloom filter created in Section 2.2.1, it is possible to also create fragments for a family name, date of birth, and address and pass them through the same salting, hashing, and modulo operation. When these additional data elements are introduced to the Bloom filter, many more bits are set to 1. Given enough data, all bits for the Bloom filter will be 1.

When too many bits are set to 1 on a Bloom filter, there is an abundance of false positives, as the Bloom filters for all records appear similar. A solution to this issue is to increase the size of the Bloom filter. However, increasing the size of the Bloom filter increases (1) the size of information that must be transmitted between the data partner and STTP and (2) the computing resources needed by the STTP to compare the filters.

Finally, there is a trade-off in the combination of data elements used when building Bloom filters. Placing a large number of data elements into a single Bloom filter increases system recall and reduces precision. An example of recall is when a Bloom filter is constructed based on given and family name: a record that swaps those names will still result in the same Bloom filter. As an example of precision loss, the combination of city of residence with name information can lead to false positives because individuals living in Austin, TX, will have a greater likelihood of matching individuals with the given name Austin.

2.3 Combined Matching Approaches

A matching system can combine deterministic and probabilistic matching in successive rounds to improve match recall and precision. An STTP may execute a deterministic matching phase to identify exact matches across data partners. The STTP may employ multiple rounds of deterministic matching to compare hashes constructed from different data elements. The identities paired across data partners in these rounds can then be removed from the potential pool of matches in later rounds of the process.

After deterministic matches have been found, the STTP could employ a probabilistic approach to find non-exact matches. Like deterministic matching, there can be several rounds of probabilistic matching using Bloom filters that have been constructed from different combinations of data elements. Multiple Bloom filters may be considered in a matching process, combining the resulting Dice coefficients of comparisons through a weighting formula.

3. Identity Matching Tools Landscape Analysis

Prior to conducting the Goodness of Fit (GoF) analysis, the CODI Collaborative Working Group (CCWG) convened a subgroup to evaluate identity matching and deduplication (i.e., information merging) tools for their ability to provision encryption keys and hash methods to support PPRL. The overall goal of this tools landscape analysis (TLA) was to make recommendations about which tool(s) should be evaluated for a potential CODI solution.

The CCWG created a subgroup to perform the TLA, which was led by CDC Fellow Pradeep Podila. It involved surveying individuals participating in CODI, literature, and the public Internet to identify PPRL solutions. Once solutions were identified, they were graded against a checklist developed by the TLA subgroup. Upon completion of the assessment, the group discussed the results and recommended three tools for evaluation:

- anonlink (open source).
- R PPRL package (open source): GNU Public License (GPL) v3 licensing might pose an issue with the intended use of the tool.
- Colorado University Record Linkage (CURL) (commercial, closed source).

Further details on the TLA are available in Appendix B.

4. Goodness of Fit Process

The three tools identified by the TLA subgroup were subjected to an evaluation process. The goal of this process was to assess the tools on data sets to simulate operation in the CODI pilot environment in the Denver metro area. This process included:

- Development of test scenarios
- Creation of a testing data set
- Executing the tests

4.1 Test Scenarios

The MITRE team created three test scenarios to test the selected PPRL tools. The first scenario replicated deployment of the tool at a small scale. Tests in this situation involved three synthetic healthcare providers, or data partners. These synthetic data partners were referred to as "System A," "System B," and "System C." Each synthetic partner possessed a set of synthetic demographic information, representing individuals who sought care within the data partner's system.

Synthetic demographic information from one data partner may or may not match to information on one or more synthetic individuals within other partners' systems. As an example, an individual in System A may match to an individual in System C, simulating an individual who received care at both locations. This represents an individual who spanned two synthetic data partners. Synthetic individuals were created who spanned all three simulated data partners. Additionally, synthetic individuals were created within each synthetic data partner who did not have matches in other systems.

Each synthetic data partner was assigned 250 records, for a total of 750 for the scenario. In this set, there were 255 correct links between records. If a simulated individual had records at each data partner, this represented three links: System A to System B, System B to System C, and System A to System C.

The second test scenario added sibling records into the data set developed for the first scenario. This test added 60 records to the data set, for a total of 810 records. This set also introduced 10 new true matches to test the ability of a matching system to correctly identify when families have records for more than one child in multiple systems.

The third scenario stressed the performance of the PPRL tools. It consisted of two synthetic data partners, System A and System B. In this case, both partners were assigned 150,000 records, for a total of 300,000 records. There were no patient matches between the data partners, meaning that any matches returned by the PPRL tools would be false positives. Knowing the potential limitations of the Bloom filter-based approaches employed by the tools, this scenario was designed to highlight any false positive rate issues that may be encountered at scale.

4.1.1 Simulation of Data Partners

Each data partner was assigned a set of synthetic records representing individuals who would seek care within the data partner's system. Each data partner's synthetic individuals were stored in separate files. Files were formatted to accommodate the PPRL tool being tested.

Depending on the PPRL tool, the file for a given data provider would be supplied to the appropriate tool-specific software for garbling the records. The garbled records would again be stored in separate files, each representing a data partner.

4.1.2 Simulation of the Data Coordinating Center

The CODI Data Coordinating Center (DCC) is responsible for executing the matching process. The collection of garbled files from each of the data providers was supplied to the appropriate tool-specific software. The PPRL tool then produced one or more files containing the matching results and generated network IDs for global use; these network IDs uniquely identified individuals' records at multiple data partners. In production operation, the network IDs would be communicated back to the data partners as LINKIDs. For evaluation purposes, the record linkages represented by the network IDs were scored for correctness.

All testing was performed on a single computer. Out of scope for this evaluation were the transmission of garbled files from data partners to the DCC, and communication of the network IDs from the DCC to data partners. Transmission of this data in a production environment is likely to take place outside of the selected PPRL tool.

4.2 Creation of a Testing Data Set

To perform the evaluation, synthetic individuals' data must be created and assigned to data partners. For the evaluation to be reflective of real-world performance, the synthetic individuals must mimic information as it is represented in real-world systems. To meet this requirement, testing data used in the evaluation was seeded with information obtained from a direct mailing information vendor. This allowed for the generation of records that conformed to the CODI Record Linkage Data Model.[1]

4.2.1 Data Elements

Table 4-1 lists the data elements populated by synthetic data generation and used for testing.

Table 4-1. CODI Identity Management Data Elements Used in the Evaluation

Element Name	Description
Birth Date	Date of birth
Sex	Sex assigned at birth
Given Name	A given name for the child
Family Name	A family name for the child
Parent Given Name	A given name for a parent of the child

[1] The CODI Record Linkage Data Model is described in the CODI Implementation Guide.

Element Name	Description
Parent Family Name	A family name for a parent of the child
Household Street Address	An address for the child, including number/name/unit (i.e., the information sometimes referred to as street line 1 and street line 2)
Household ZIP	A ZIP code for the child
Household Phone	A phone number for the child
Household Email	An email address for the child

4.2.2 Obtaining Realistic Demographic Information

This evaluation used real-world data obtained from a direct mailing list vendor. MITRE worked with InfoUSA to purchase 100,000 records for use in testing. Individuals were sampled from selected Colorado counties (Adams, Arapahoe, Boulder, Broomfield, Denver, Douglas, and Jefferson). These counties will be covered by the CODI pilot. Individuals were selected using random sampling: 20,000 records were randomly selected for households where InfoUSA believed the individual was between the ages of 18 and 22; 80,000 households were selected where the family had a child between the ages of 2 and 17. The purchased information included the following data elements that were used in the evaluation:

- Title
- First Name
- Middle Initial
- Last Name
- Street Address
- City
- State
- Zip Code
- Phone Number
- County
- Ethnicity
- Gender

For this data set, Ethnicity is a label applied to a given household by InfoUSA. Example values include English, Italian, Danish, African American, and Jewish.

Not all data elements provided are listed, as they were not used in the evaluation. The data listed above was the basis for generation of the test data set.

4.2.3 Introducing Variation

The evaluation did not directly use the raw direct mailing information received from InfoUSA. Instead, new records were created that contained real-world demographic data, but were not reflective of an actual individual. The steps to generate the 750 records for the first test scenario were as follows:

1. Create the child record by:
   a. Randomly selecting a name from the 100,000-record set. The name included the given name, middle initial, and family name.
   b. Replacing the middle initial for the individual with a culture- and gender-appropriate name. The MITRE Identity Matching Lab maintains a list of names that are common for individuals who identify with a given ethnicity or culture. This allows for the selection of a middle name that would be of the same cultural or ethnic use as the family name.
   c. Generating a random date of birth such that the individual is not more than 19 years old.
2. Generate parent information by:
   a. Selecting a second record from the 100,000 set, where the family name of the selected record is not the same as the family name of the child. Select some cases to have the same culture-appropriate information as the child, but not all.
   b. Replacing middle initial with middle name using the same method as the child.
   c. Using the child's family name for the parent's family name 80% of the time; retaining the original name 20% of the time.
3. Populate the child's address by selecting an address from the 100,000 set that is not from the records selected for either the parent or child.
4. Populate the child's phone number by selecting a phone number from the 100,000 set that is not from the records selected for either the parent or child.
5. Generate the parents' email addresses from corpus of email addresses scraped from the Twitter website by a previous MITRE project, then use a matching tool developed by MITRE to score the likelihood that someone with the generated parent name would have an email address in the corpus. Select the email address with the highest likelihood score. This will ensure that the email address is realistic for the parent information.
6. Create true positive synthetic matches for child records: Manually generate two or three synthetic records based on the original child record where the synthetic record varies on one or more characteristics. This creates synthetic records that are true matches of the original child record. These manual variations mimic data errors observed in the real world, including given and family name swaps, nicknames, typographical errors, truncations, place holders, and missing fields.
7. Repeat Steps 1–4 to generate records without matches until all three simulated systems contain 250 records.

4.2.4 Generating Sibling Variations

Sixty additional synthetic sibling records were created based on a random record selection from the 750 children generated to support the second test scenario. For each false positive synthetic "match," a subset of the data elements was used to create a new record that should not match the original record. The most difficult sibling cases created represented same-sex twins. In this

situation, the records shared all data elements except given name. Other cases introduced variation into date of birth, sex, address, and parent information.

4.2.5 Stressing the System

In order to determine if the systems could perform to scale, a data set of 300,000 individuals was generated to stress the matching systems. This data set should not contain any matches. It was generated using the following steps:

1. Randomly select an ethnicity from the InfoUSA data set.
2. Determine the number of individuals to generate for that ethnicity based on the distribution observed in the InfoUSA data.
3. For each record being generated:
    a. Create the child record by selecting a seed name from the Social Security Administration Death Master File stored in a MITRE Identity Matching Lab Database and replacing the middle name with a culture- and gender-appropriate name.
    b. Populate the parent information by repeating Step a for the parent name, then changing the parent family name to match the child's family name 80% of the time.
    c. Populate the remaining child data by randomly selecting a date of birth with the selection weighted by birth rate data from the years 2009 to 2014; randomly selecting from the InfoUSA data an address and phone number; and finally, randomly selecting an email address from the pool of email addresses obtained from Twitter.

This procedure was used to generate a set of 300,000 records that should include no true matches. For testing, these records were split into two sets to be assigned to two different synthetic data providers.

4.2.6 Ground Truth

In order to test the PPRL tools with the data sets, the set of true matches must be known. This is referred to as "ground truth." For the 3 sets of 250 records, the Health FFRDC maintained a list of true matches. Each of the child records was assigned an identifier. The ground truth file contained pairs of identifiers representing records that identified the same synthetic individual. For individuals with records across all three of the simulated data partners, there were three corresponding pairs of identifiers in the ground truth file. If a pair of records was not listed in the ground truth file, then the records were not considered a match.

The ground truth file was created manually. The individual generating the variations on the record was responsible for recording the linkages between the records.

4.3 Evaluation Metrics

The evaluation examined the performance of PPRL tools using common metrics for examining identity matching tools. The metrics used to conduct the evaluation are described in Table 4-2.

Table 4-2. Definitions of Evaluation Metrics

Metric	Description	Calculation	Value set
Precision	A ratio that provides the fraction of the identified matches that are correct	TP/(TP+FP)	A system that returns only correct answers will have a Precision of 1. A system that returns only False Positives will have a Precision of 0.
Recall	A ratio that provides the fraction of the correct possible answers that the system found	TP/(TP+FN)	If all possible matches are identified, a system will have a Recall of 1. If no correct matches are found, a system will have a Recall of 0.
F1 Score	Harmonic mean of Precision and Recall	2*(Precision * Recall) / (Precision + Recall)	A number between 0 and 1 that represents a combination of Precision and Recall
False Positives	A count of False Positives returned by the system	N/A	>=0
Sibling False Positives	A special case of False Positives, providing a count of False Positives that were caused by the system identifying records that belong to synthetic siblings as matching the same individual	N/A	>=0

Notes: TP=True Positive; FP=False Positive; FN=False Negative

This evaluation constructed an assessment of PPRL performance by looking across these metrics. This is important, as tools can be tuned to perform well on a single metric at the expense of others, ultimately impacting real-world performance. An example of this is extreme tuning for precision: a system can often achieve a precision of 1 by returning only the single match it is most confident in. This system would not be useful in practice and would also have a very poor Recall and F1 Score. Conversely, a system could return all possible matches for a given data set. This would achieve a Recall of 1 but would similarly be of no real value in actual application.

Finally, the evaluation does not present any thresholds or acceptable ranges for Precision, Recall, or F1 Score. Matching algorithm performance is influenced by data set quality, making the establishment of generic performance thresholds difficult. Solutions will be compared relatively and to results observed by the Health FFRDC in other patient matching settings.

4.4 Comparing Commercial and Open Source Solutions

This evaluation involved comparing open source and commercial tools. The same performance metrics can be used to assess matching ability between tools; however, the differences between tool licensing and the support that typically accompanies those licensing fees results in different approaches to installing, configuring, and tuning the systems.

4.4.1 Service Offering Differences

Open source software provides access to an application's source code without a licensing fee. This access is typically provided without any guarantee on the operation of the software, or any requirement that the software authors provide support. Support for open source software can sometimes be obtained from volunteer communities or through fee-based options provided by the authors or related consultants.

In contrast, commercial products can require a licensing fee for usage. The purchased license usually provides the purchaser with assurance of proper software performance and support. The level of support may vary, ranging from basic troubleshooting to advice on optimal use of the product in the purchaser's environment.

4.4.2 Comparison Strategies

Due to the differences between open source and commercial software, the evaluation employed different strategies when working with the PPRL tools depending on license type.

The CDC stated a preference for a PPRL package with an open source license. One reason for this preference is a desire to minimize deployment costs. As such, it would be unreasonable to assume that the CODI effort would wish to establish a support contract for a particular open source tool. Because of this restriction, the evaluation did not seek any support when evaluating open source tools. Only publicly available information was used to install, configure, and tune open source PPRL tools.

The evaluation employed a different strategy when examining the commercial tool. Since this tool requires a licensing agreement for deployment, there is an expectation of support. Following this expectation, the evaluation involved contacting the tool vendor for troubleshooting issues as well as advice on configuring and tuning.

These two strategies were used to arrive at an impartial assessment of the amount of effort that would be required to prepare a tool for usage as well as set expectations for tool performance, given the time and staffing allocated to the evaluation.

5. anonlink Evaluation

As an open source project, anonlink was evaluated without the expectation of support. A member of the evaluation team filed an issue on GitHub[2] to inform the software developers that their public mailing list was improperly restricted. This was the only contact between the evaluation and developer teams.

5.1 Underlying PPRL Technology

anonlink uses a single Bloom filter to implement PPRL. Data elements go through the fragmentation, hashing, and modulo operations with the resultant bits being set on a single Bloom filter created for the individual.

Bloom filters are compared using the Dice coefficient. The user selects a threshold, and records that meet or exceed the threshold are considered a match.

Users have the ability to restrict the maximum number of bits that a data element may use in a Bloom filter. This provides a mechanism for assigning a weight to a data element's contribution to the overall match score.

5.2 Installation

anonlink is written in the Python programming language. The evaluation team used a computer running macOS 10.14.6 with Python 3.7.4 and Package Installer for Python (pip) 19.1.1.

The tool for taking PII and performing garbling in the anonlink software suite is called clkhash. Version 0.14.0 of clkhash was installed via pip, by executing "pip install clkhash" within the macOS Terminal. Installation instructions were available on the clkhash GitHub page[3] and were executed without issue.

The anonlink project provides a server that can be used to perform matching called anonlink-entity-service. This server can be deployed via the Docker container platform. The same computer used to install clkhash already had Docker Desktop 2.1.0.4 installed. The anonlink-entity-service was cloned from its GitHub repository.[4] The build and run instructions supplied on the project's GitHub page[5] were executed without issue.

5.3 Configuration

anonlink is configured with a JavaScript Object Notation (JSON) file supplied to the clkhash tool as well as the anonlink-entity-service. The format for the JSON file is described in the clkhash documentation.[6] The evaluation used Version 2 of the configuration file format.

[2] https://github.com/data61/anonlink/issues/219
[3] https://github.com/data61/clkhash#installation
[4] https://github.com/data61/anonlink-entity-service at commit b48937a54d12653e6fc07153f8681984772b42e6
[5] https://github.com/data61/anonlink-entity-service#build
[6] https://clkhash.readthedocs.io/en/latest/schema.html

This configuration file specifies the data elements that will be used to construct the Bloom filter for each individual. The configuration file provides the following capabilities:

- The ability to set the size of the Bloom filter
- Specification of the data elements to use to construct the Bloom filter
  - Specification of the size of the fragments when breaking up the data elements
  - Optional rules that can be applied to validate input
  - Optional restriction on the maximum number of bits that a particular data element may set when creating a Bloom filter

5.3.1 Adjusting Configuration

Creating or changing anonlink configuration requires manual creation of JSON files. The evaluation team used the open source Visual Studio Code application, simply as a text editor, to create and edit the anonlink JSON files used in the evaluation.

The JSON format allows configuration to easily be stored in a version control system, such as Git. It also allows for changes to be made rapidly, using only a text editor. Additionally, it is possible to programmatically create a configuration file. This could be leveraged to create a system that generates a series of configurations with different settings and then tests matching performance in an automated set-up. This could be used to build an automated tuning system.

While configuration of a tool through a JSON file has benefits, it also has limitations. anonlink does not provide any tooling to edit the configuration file. As such, users who are unfamiliar with JSON or other machine consumable formats will not be able to configure the system. Addition and removal of data elements can be a tedious process, as the user is responsible for managing the structure of the file when making changes.

5.3.2 Distributing Configuration

anonlink does not provide any features to support the distribution of configuration information from the DCC to data partners. Transport of the JSON file must be provided by the user.

Salt values are supplied to the clkhash tool via command line arguments. As with the JSON configuration file, users will be responsible for securing the salt values and distributing them to data partners.

5.4 Testing Preparations and Execution

Testing of anonlink, where it makes a single attempt at matching by using all available data elements, started by creating a configuration file that specified the data elements for testing. The configuration file was passed to the clkhash tool to garble the synthetic PII used in testing. This process was repeated for each synthetic data provider in the test. The process was automated with a 20-line JavaScript-based script, created by the Health FFRDC.

The garbled files were then uploaded, along with the configuration to the anonlink-entity-service for matching.

5.4.1 Conducting a Matching Run

The process of performing matching consisted of interacting with web services provided by the anonlink-entity-service. The Health FFRDC developed a set of tools in JavaScript to handle all aspects of the process. Ultimately, a matching run could be executed in a single command line instruction with the developed tool.

Software development was necessary as a part of this evaluation due to limitations of anonlink. Currently, the software supports only pairwise matching. It does not support the matching of records across three or more parties. To compensate for this, the developed software ran anonlink multiple times on pairs of the synthetic data partners. The results of these multiple runs were combined and used to generate network IDs.

In the process of creating network IDs, conflicts may arise. The testing environment assumes that each record at a synthetic data provider represents a single individual and there are no intra-provider duplicates. Pairwise matching may lead to situations where linkages across data partners violate that assumption. An example of how this happens is provided in Table 5-1.

Table 5-1. Example of a Conflict Generated Through Pairwise Matching

Matching Pair	System A ID	System B ID	System C ID
System A to System B	MRN-5A	MRN-13B	N/A
System B to System C	N/A	MRN-13B	MRN-45C
System A to System C	MRN-10A	N/A	MRN-45C

As this example shows, the matching process resulted in a linkage that would involve two separate records from System A. To address this issue, the Health FFRDC employed a rudimentary approach to conflict resolution. When conflicts were detected, the software would order the links by the score assigned by anonlink-entity-service. The link with the lowest score was removed and the tool again checked for conflicts. This process was repeated until conflicts were resolved.

In total, 349 lines of JavaScript were created by the Health FFRDC and used to automate interaction with the anonlink-entity-service. The tools perform the following operations:

1. Create a new project within the anonlink-entity-service, which involves uploading the JSON configuration file and setting a scoring match threshold. This represents a pairwise matching operation.
2. Upload the garbled files for the two synthetic data providers in the pair.
3. Request that the system initiate matching.
4. Poll the system until matching is complete and download results.
5. Repeat Steps 1 through 4 for all synthetic data provider pairs.
6. Create network IDs, deconflicting where necessary.
7. Write network IDs to a comma-separated values (CSV) file.

The resultant CSV file containing network IDs was compared to the ground truth to calculate the necessary evaluation metrics.

5.4.2 Making Adjustments

Changing the configuration of anonlink between matching runs involved editing the JSON configuration file or adjusting the scoring threshold in the JavaScript tooling. After that, the process of garbling the PII and interacting with the anonlink-entity-service was repeated. Rapid testing was possible because this process was automated. On the data set of 750 records, a matching configuration could be fully tested in under one minute.

5.5 Results

The Health FFRDC applied three test scenarios to the anonlink software suite, which included:

- **Test Scenario 1:** Examined each package's ability to match individuals across three synthetic data partners. This is a data set that included 750 synthetic children with true matches across three synthetic data partners.
- **Test Scenario 2:** Explored each package's performance on the more complex process of correctly matching records when sibling information is present. This is a common challenge with pediatric record linkage. The linkage system must correctly match records for an individual while not incorrectly linking that person to the records of their brothers or sisters. In this scenario, data elements such as given name, sex, address, and/or date of birth have been varied. Some data elements are the same for the test case, such as family name or parent contact information. This is the data set from Test Scenario 1, but with 60 additional records.
- **Test Scenario 3:** Assessed the packages' performance on a large set of records with no matches to determine how the linkage process performed at scale. This is the data set with 300,000 records.

5.5.1 Test Scenario 1

The first scenario tested involved three synthetic data providers, each with 250 records for a total of 750 records. Ten different configurations were attempted and the results for the best configuration are shown in Table 5-2.

Table 5-2. anonlink Best Configuration Results from Scenario 1

Metric	Result
Precision	1.0
Recall	0.937
F1 Score	0.968
False Positives	0

anonlink performed very well on this data set. It is unlikely that the performance observed on this data set could be replicated in production usage. anonlink returned 0 false positives. This is likely due to a coincidence of the characteristics of the data set being favorable to anonlink's

approach. The data set records varied in ways that required consideration of all fields in the record, yet the data set did not include close, but not correct matching records. As the results of the next test scenario will show, introduction of records with similar data elements (siblings) has a dramatic impact on performance.

5.5.2 Test Scenario 2

The second test used the augmented data file that included the 750 records from the first scenario plus 60 sibling records. Table 5-3 presents the best configuration results from this test.

Table 5-3. anonlink Best Configuration Results from Scenario 2

Metric	Result
Precision	0.794
Recall	0.887
F1 Score	0.838
False Positives	59
Sibling False Positives	59

anonlink performed poorly when siblings were added into the data set. One possible explanation for this result is that the single Bloom filter approach allows anonlink to handle a wide class of variations well, including given and family name swaps. This could, however, lead to anonlink incorrectly matching siblings.

5.5.3 Test Scenario 3

The final scenario involved the data set of 300,000 records split between two synthetic data providers, such that each has 150,000 records. There were no correct matches between the providers, so any identified matches were false positives.

In this test, anonlink returned 736,093 matches, all of which were false positives. Because of the nature of this test, deconfliction was not applied in this scenario, which is how the matched number is larger than the total of possible matches if the single record per individual assumption was held to be true. Regardless, the magnitude of the result shows that anonlink's approach of a single Bloom filter with the number of data elements used for testing does not scale well.

5.6 Other Considerations

While the default configuration of anonlink uses a single Bloom filter to perform matching, it is possible to run anonlink with different configurations, using different smaller sets of data elements in each anonlink-entity-service project. The matching can be performed for each project and the results assembled together to come up with an ultimate matching result.

The Health FFRDC implemented this approach using the previously described JavaScript tools and extending them with an additional 72 lines of code to perform the multiple rounds of matching and merging of results. The previously described conflict resolution logic was used across all matching projects, as opposed to on a single project. The evaluation team tested Test Scenario 2, the sibling scenario, with the following anonlink projects:

- Project 1: Use the data elements given name, family name, sex, date of birth, and ZIP code.
- Project 2: Use the data elements given name, family name, sex, date of birth, and phone number.
- Project 3: Use the data elements given name, family name, sex, date of birth, and street address.

This test yielded the results shown in Table 5-4.

Table 5-4. Multi-Project anonlink Results

Metric	Result
Precision	0.917
Recall	0.778
F1 Score	0.842
False Positives	18
Sibling False Positives	18

The team also evaluated this approach in Scenario 3, where multi-project anonlink generated 14,223 false positives. Testing in Scenario 3 resulted in a rate of 0.047 false positives per record. This is close to the rate observed when testing CURL, in Section 6.5.3.

Given that CODI is interested in conducting research and the matching solution, false positives are not as great of a concern as they would be in a point of care setting. Future evaluations will examine the impact on false positives relative to the correctness of answers to research questions.

6. CURL Evaluation

CURL is a commercial offering created by the University of Colorado. To facilitate this evaluation, the Health FFRDC was provided with a no-cost license for CURL. The evaluation team had many interactions with the CURL team, including email, teleconferencing, screen sharing, and a shared file space hosted by the Health FFRDC.

6.1 Underlying PPRL Technology

CURL offers multiple approaches to PPRL. These can be configured by the user with a graphical interface to obtain a matching approach for a specific scenario. CURL refers to a matching scenario as a matching job.

CURL performs deterministic matching using garbled data. Users select data elements to use to create hashed values. CURL can also process data elements prior to hashing. As an example, CURL can build a hash using the first three letters of given and family name.

CURL also provides Bloom filter technology for use in probabilistic matching. Bloom filters can be constructed to contain as many or as few data elements as desired. CURL can generate a Bloom filter per data element, then allows users to assign weights to each Bloom filter. The distances between filters can be used with the weighting to determine an overall match score.

CURL allows for the use of multiple approaches simultaneously. A CURL matching job might first attempt deterministic matching and any pairs identified through that process are removed from the pool of potential matches. The next round of matching may be another round of deterministic matching, or it could be a probabilistic round using a Bloom filter-based approach. This would repeat until all matching rounds are complete.

6.2 Installation

CURL is written in the Java programming language. The University of Colorado provided the software to the Health FFRDC in two forms via a shared file space. The first form allowed for deployment as Docker containers. The software was also provided as a Java Archive (JAR) file.

CURL is composed of four components. The first is CURL Center, which is a website that users interact with to configure their matching jobs. This site is operated by the University of Colorado, so no installation effort was required.

The second component is CURL Keymaster. This application generates salt values used for a matching job. Initial attempts to deploy this application with Docker Desktop 2.1.0.4 on macOS 10.14.6 failed. The application was run successfully using the JAR file on Oracle Java 1.8.0_202. Attempts to run the application on the more recent OpenJDK Java 11.0.2 were unsuccessful. The inability to run all CURL applications with Docker as provided by the team would lead to greater administrative burden in a production environment.

The third component is CURL Site. This application garbles data partners' PII. This successfully ran on Docker Desktop 2.1.0.4 on macOS 10.14.6.

The final component is CURL Honest Broker. This is an application that the DCC will run to compare garbled information from the data partners and perform matching. This application failed to run on Docker Desktop 2.1.0.4 running on macOS 10.14.6. It was also run using Oracle Java 1.8.0_202. In addition to the Java-based application, CURL Honest Broker requires a database to operate. PostgreSQL 11.4 was used as a database. The CURL application provided a SQL-based set-up script that the Health FFRDC staff executed. The Health FFRDC team also modified the provided configuration file to contain the correct connection information for that database.

6.3 Configuration

Configuration of CURL takes place through the CURL Center web application hosted by the CURL team. This is the application a CURL user interacts with to create, edit, and manage a matching job (see 6.3.1 for additional details). Figure 6-1 displays the portion of the application used to set up linkage methods.

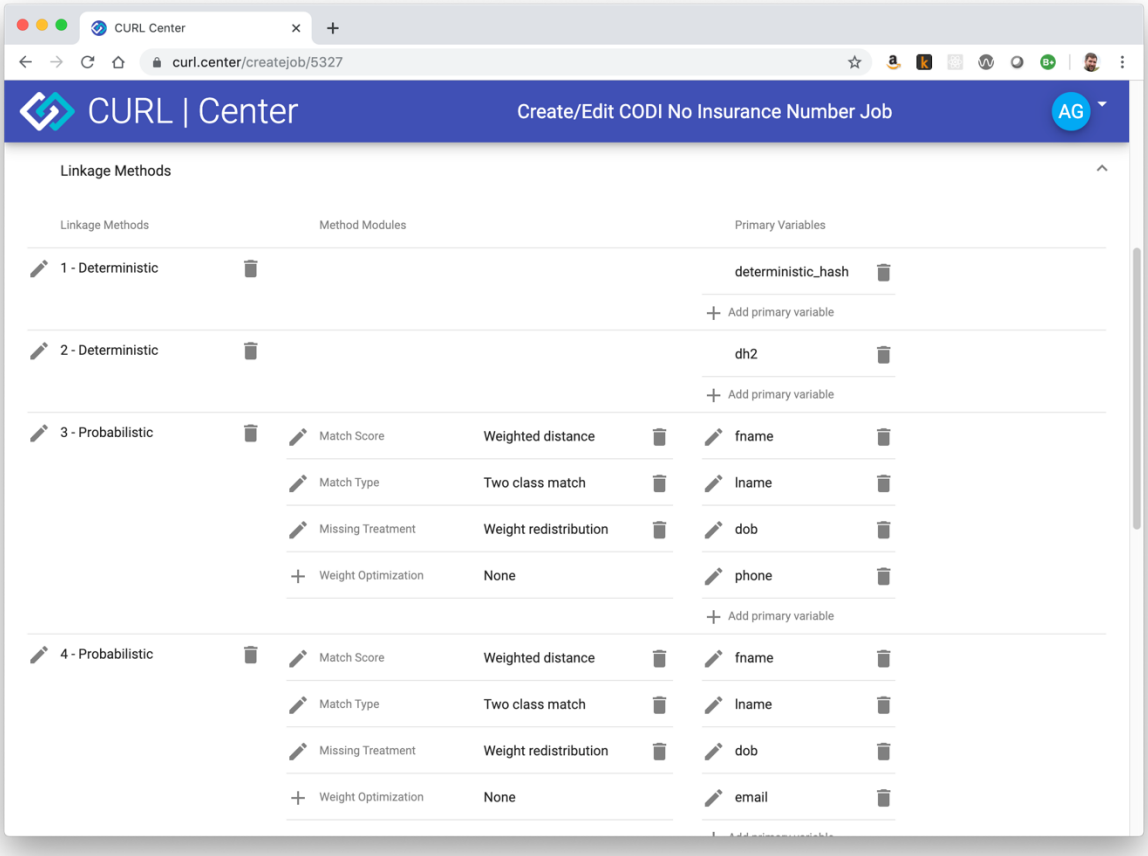

Figure 6-1. CURL Center User Interface for Configuring Linkage

All aspects of working with CURL are specified through this user interface. Data elements are identified in the Source Variable section. Data elements can then be preprocessed in the Normalization Variables section. It is possible to simply use the data elements as they were

supplied from the data partner, or this functionality can be used to trim whitespace, perform case conversions, or other cleaning routines specific to the data element.

The next section of the CURL interface is Custom Variables. This is where data elements may be transformed prior to garbling. Example transformations that were used in the evaluation include:

- Left: Select a user-specified number of characters from the start of a data element.
- Concatenate: Join two or more data elements together into the same variable.
- Soundex: Pass the data element through the Soundex algorithm, which will transform the text into its phonetic spelling. This can often be used to address typos or close misspellings.

The Obfuscation Variables section follows creation of the Normalization Variables and Customization Variables. This allows the users to select a variable for garbling. For exact matching on hash values, a custom variable is used that concatenates data elements together. Bloom filters can be created from single data elements or from multiple using the concatenation approach.

Finally, CURL Center provides a Linkage Methods section. This is where a user will select Obfuscation Variables to compare for matching. Different matching methods can be created and CURL will execute them in order.

Once configuration is complete, CURL generates a set of configuration files that are used with each of CURL's suite of tools. These files are JSON but are not editable by users. The CURL application employs cryptographic technology to sign the configuration files. Other tools in the CURL suite validate the signature when loading the configuration file to ensure that it has not been tampered with during transmission.

6.3.1 Adjusting Configuration

Changes in configuration are made through the CURL Center interface. This allows users without software development skills to create and adjust matching configuration. An example of editing a Custom Variable in CURL Center is shown in Figure 6-2.

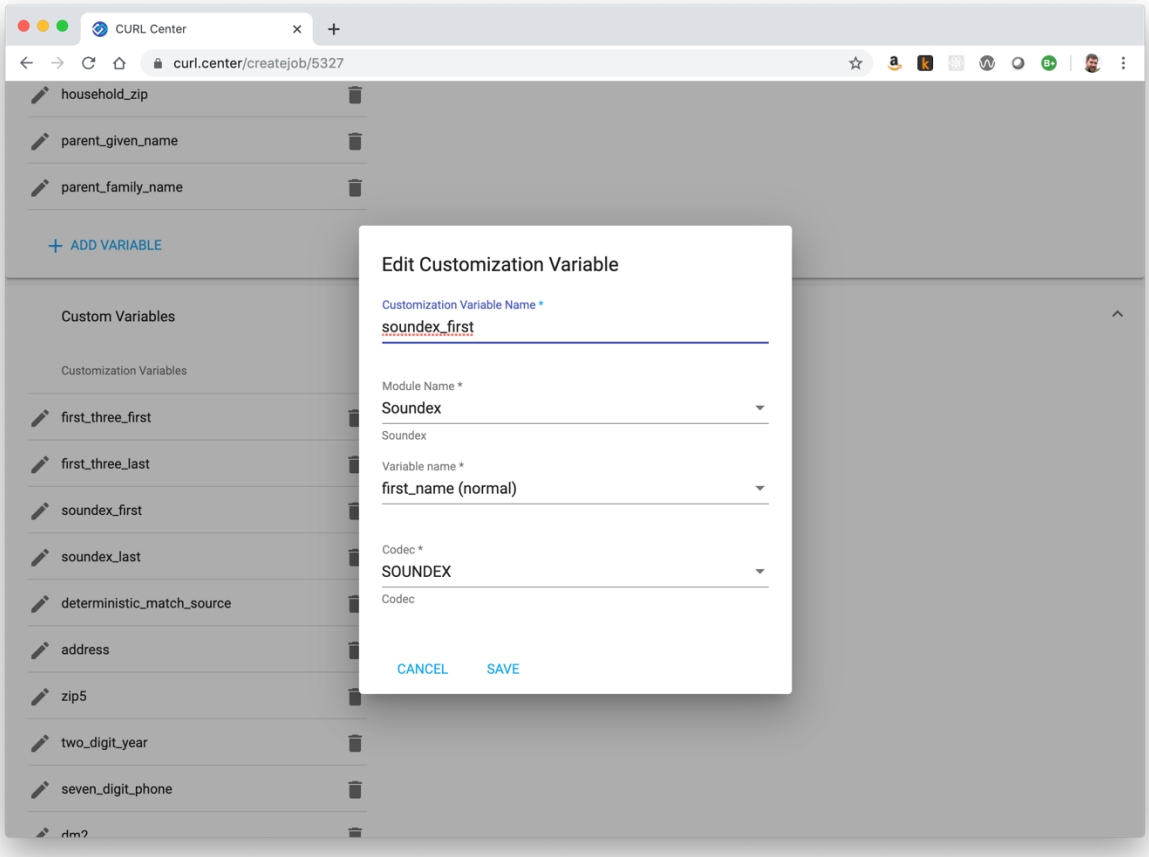

Figure 6-2. Editing a Custom Variable in CURL Center

The user interface did pose some challenges for the Health FFRDC team. For example, there is a difference in how thresholds and weighting fields are represented in the tool. Matching thresholds are represented using an integer between 0 and 100, while weighting thresholds are represented as decimal values between 0 and 1. This caused confusion for the Health FFRDC team as the match threshold was initially set to a decimal value, 0.65, and the tool treated the entry as an extremely low matching threshold. After leveraging CURL support through discussions and sharing of configuration files, this discrepancy was discovered and additional documentation was inserted into the tool.

CURL does not currently offer an Application Programming Interface (API). This is a hinderance for users making small changes in the configuration and re-running the tool to evaluate performance. When conducting the evaluation, for each time the configuration changed the team must:

1. Make the changes in CURL Center.
2. Download the CURL Keymaster configuration file.
3. Upload the configuration to CURL Keymaster and regenerate the salt.
4. Download the new salt files.

5. Download the configuration for CURL Site for each simulated data partner. In the case where 750 records were used, all CURL Site operations were repeated three times.
6. Upload the simulated PII data to be garbled, configuration file, and salt file to CURL Site.
7. Garble the data.
8. Download the CURL Honest Broker file.
9. Upload the configuration file and garbled data files to CURL Honest Broker.
10. Perform the matching job.

Performing a full matching job required interacting with four separate tools in different applications. While the process was straightforward, it was time consuming and typically took between 5 and 10 minutes of time, during which the user was fully engaged in the process.

6.3.2 Distributing Configuration

All CURL suite tools require configuration files for use. CURL Center provides a mechanism for distributing the files. Users can create accounts in CURL Center and then be provisioned access to the appropriate files depending on their role. CURL Center also maintains information on the state of the matching job. In production use, CURL Center could be used to see which data partners have garbled their data and transmitted their information to the DCC. Because the Health FFRDC team played the roles of data partner and DCC, this functionality was not evaluated. A view of the participants and configuration file information in CURL Center can be seen in Figure 6-3.

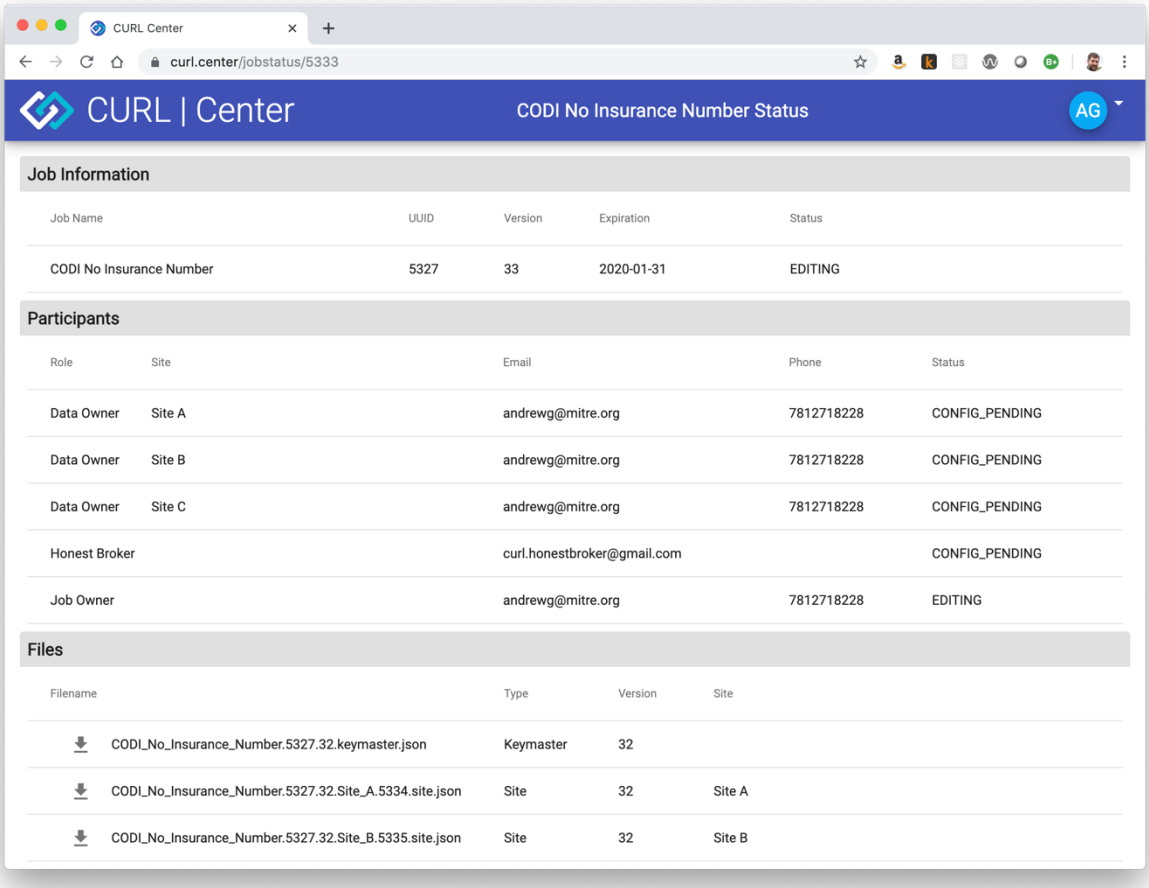

Figure 6-3. CURL Center Participant and Configuration Information

6.4 Test Preparations

There were no specific preparations required when working with CURL. Because CURL tools are web-based and no API is available, it was not possible to write code to automate otherwise manual processes.

6.4.1 Conducting a Matching Run

The process of conducting a matching run involved working with the suite of CURL tools. These tools are web based, so most of the interactions involved uploading and downloading files.

The process starts with downloading the configuration for the CURL Keymaster. The configuration file is then uploaded into the Keymaster application. The evaluation team would then request the generation of new keys, or salt from the application. This interface can be seen in Figure 6-4.

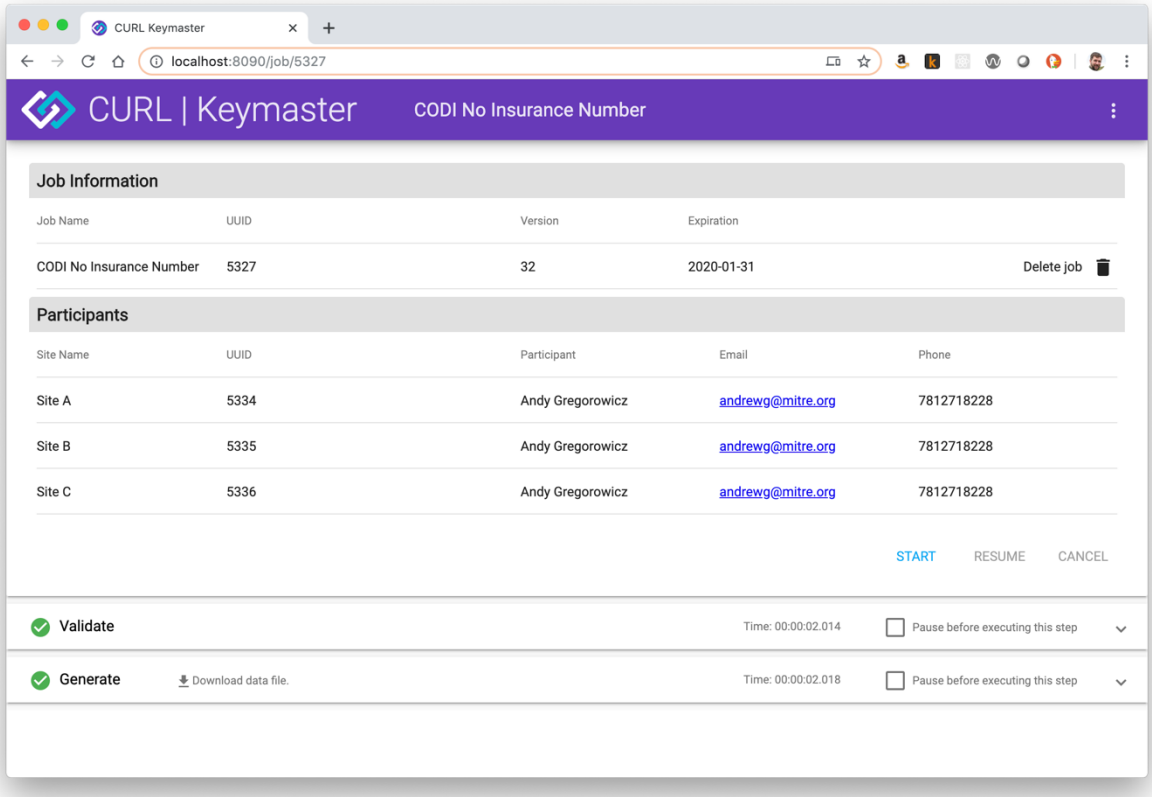

Figure 6-4. Generating Keys (Salt) with CURL Keymaster

With the necessary information gathered from CURL Keymaster, the evaluation team can start the process of garbling the synthetic PII through CURL Site. A single instance of CURL Site was used in evaluation, with different configuration files uploaded to represent different data partners. The user interface for starting the garbling process can be seen in Figure 6-5.

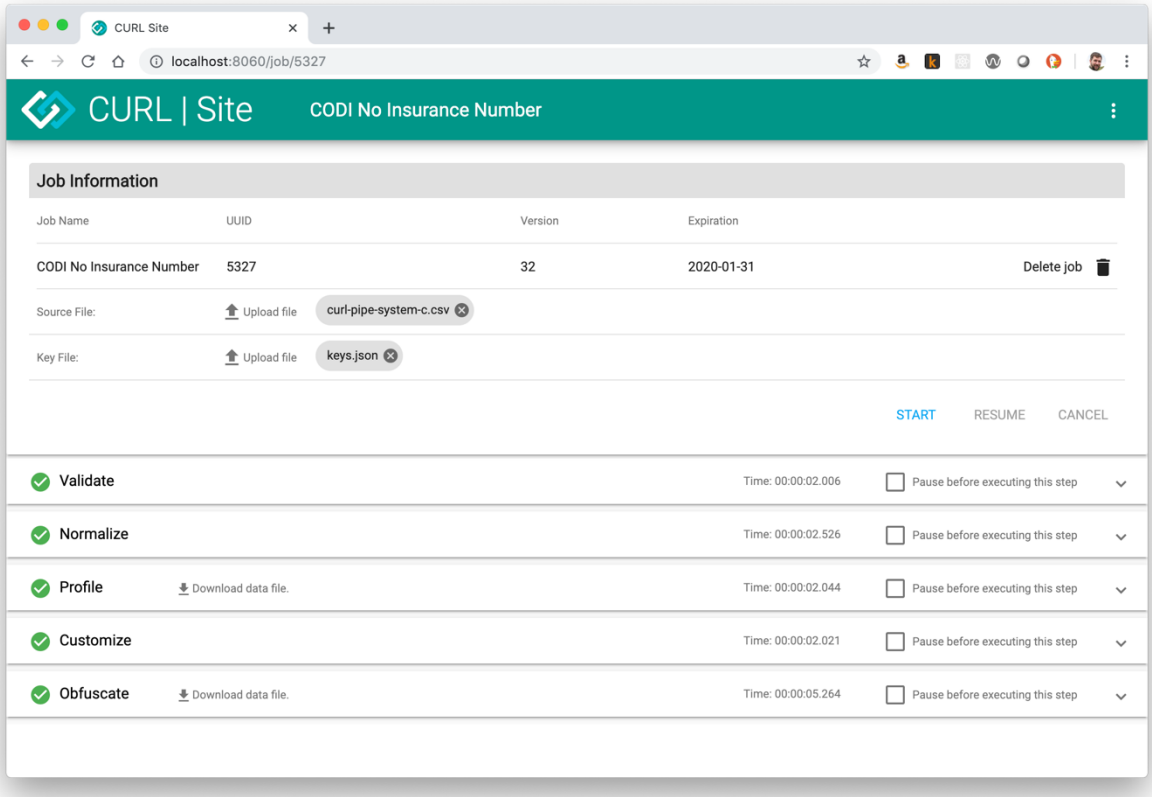

Figure 6-5. Starting the Garbling of PII with CURL Site

After data has been garbled, it is ready for matching with CURL Honest Broker. The files from the synthetic data providers are uploaded into CURL Honest Broker and matching is started through its user interface, which can be seen in Figure 6-6.

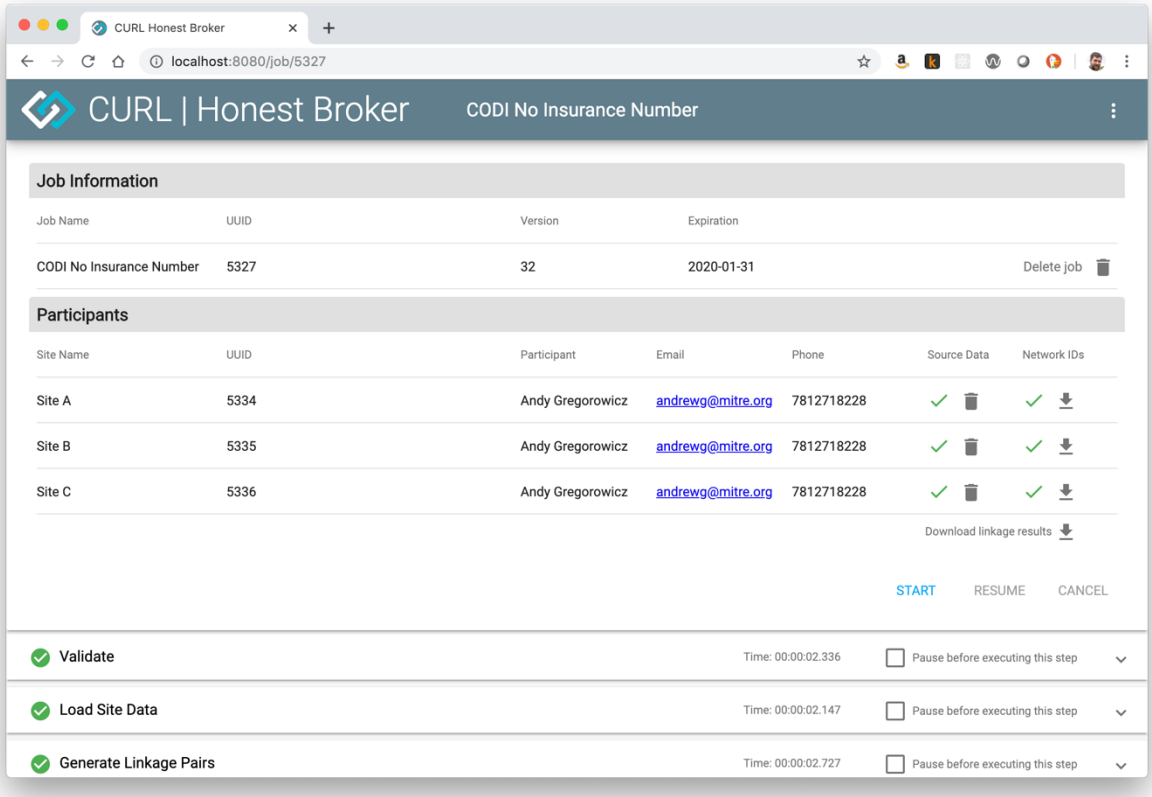

Figure 6-6. Starting Matching in CURL Honest Broker

Once matching is complete in CURL Honest Broker, a CSV file of network IDs can be downloaded. This is what the evaluation team used for scoring CURL performance.

6.4.2 Making Adjustments

Making adjustments in matching configuration involves using CURL Center to make changes, regenerating the configuration files, and repeating the matching process.

The evaluation team shared matching performance results with the CURL team and sought advice on how performance could be improved. The CURL team uses a spreadsheet to plan its rounds of matching. It shared this spreadsheet with the evaluation team and created a draft plan that was evaluated.

Evaluation of the draft plan showed that the CURL configuration was performing poorly on cases where given and family name had been swapped. The CURL team suggested an approach of constructing a Bloom filter based on a concatenation of given name, family name, and given name again. This change improved performance.

Based on the advice of the CURL team and testing, the final configuration for Goodness of Fit was used:

- Round 1: Exact hash matching on first three letters of first name, first three letters of given name, soundex of first name, soundex of last name, sex, and date of birth
- Round 2: Exact hash matching on sex, five-digit ZIP code, last two digits of birth year, last seven digits of phone number, and parent family name
- Round 3: Blocking on sex, matching separate Bloom filters based on given name, family name, date of birth, and phone number
- Round 4: Blocking on sex, matching separate Bloom filters based on given name, family name, date of birth, and parent email
- Round 5: Blocking on sex, matching separate Bloom filters based on given name, family name, date of birth, and street address concatenated with ZIP code
- Round 6: Blocking on sex, matching separate Bloom filters based on given name, family name, date of birth, parent given name, and parent family name
- Round 7: Blocking on sex, matching separate Bloom filters based on concatenation of given name, family name and given name, date of birth, and ZIP code

6.5 Results

The Health FFRDC applied three test scenarios to the CURL software suite, which included:

- **Scenario 1:** Original data set that included 750 synthetic children with true matches across three synthetic data partners
- **Scenario 2:** Original data set with addition of synthetic false positives designed to assess CURL's ability to differentiate between siblings
- **Scenario 3:** Large data set with no true matches, designed to assess CURL's ability to perform at scale

6.5.1 Test Scenario 1

The first scenario tested involved three synthetic data providers, each with 250 records for a total of 750 records. Nine different configurations were attempted and the results for the best configuration are shown in Table 6-1.

Table 6-1. CURL Best Configuration Results for Scenario 1

Metric	Result
Precision	0.877
Recall	0.843
F1 Score	0.86
False Positives	30

CURL produced acceptable results in Scenario 1. While nine different configurations were tested completely, a substantially larger number of configurations were attempted and failed. When the evaluation team first attempted CURL usage, it was unable to obtain reasonable results. Upon consulting with the CURL team via email, teleconference, screen sharing, and sharing of

configuration files, it was determined that there was a defect in CURL Center. This defect caused all the configuration files to contain erroneous information that prevented proper operation.

From communication between the evaluation team and the CURL team, it appears that troubleshooting this issue was complicated by the fact that the CURL team was working from a development version of the CURL Center application. In total, the evaluation team spent approximately 40 staff hours in troubleshooting or generating configurations that would not be able to work due to this issue. Although the problem was resolved, it shows that future use of CURL will likely require direct involvement with the CURL team.

6.5.2 Test Scenario 2

The second test used the augmented data file that included the 750 records from the first scenario plus 60 sibling records. Table 6-2 presents the best configuration results from this test.

Table 6-2. CURL Best Configuration Results from Scenario 2

Metric	Result
Precision	0.79
Recall	0.805
F1 Score	0.798
False Positives	55
Sibling False Positives	25

CURL performance declined as compared to Scenario 1 when siblings were introduced into the test set. However, the number of false positives generated was similar to multi-project anonlink. Further testing and configuration would likely be able to reduce the number of false positives.

6.5.3 Test Scenario 3

The final scenario involved the data set of 300,000 records split between two synthetic data providers, such that each has 150,000 records. There were no correct matches between the providers, so any identified matches were false positives.

CURL was unable to complete the matching job. Initially, this test was performed on the macOS-based computer. This machine had approximately 330GB of free space, which CURL filled attempting to complete the matching job. To complete testing, the job was shifted to MITRE's cloud environment, where it was tested on a server provisioned with 1TB of disk space and eventually tested with 2TB of disk space. In all cases, CURL filled all available disk space and failed to fully complete the job. It is likely that there is a defect in CURL's code that creates network IDs that causes it to consume disk space in an unbounded manner.

However, it was possible to ascertain the number of matches that CURL had identified. In this test, CURL identified 9,232 matches, all of which were false positives. In this configuration CURL had a rate of 0.031 false positives per record. This result reflects positively on CURL's internal matching capability, as it performed better in this scenario than anonlink. Further, CURL checks for intra-site matches—cases where there are matches within the same synthetic provider. It has the possibility of returning even more matches than tools that do not have this feature. This

fact needs to be considered in light of CURL being unable to successfully complete the test scenario.

7. R PPRL Package Evaluation

As an open source project, the R PPRL package was evaluated without the expectation of support from anyone involved with the project. There was no contact between the Health FFRDC and project development teams.

The R PPRL package provides "a toolbox for deterministic, probabilistic and privacy-preserving record linkage techniques."[7] Installation consists of running the install.packages method from within an R environment (such as R Studio).

The Health FFRDC team was able to garble a sample data set bundled with the R package. Once data was hashed, the Health FFRDC team was unable to find any documented method to perform matching on the hashed values. By examining the source code, the Health FFRDC team discovered some potential undocumented capabilities for matching hashed values but was not able to get the undocumented capabilities to function properly.

The Health FFRDC team was not able to garble the test data set developed for the GoF analysis. The team tried reducing the number of records in that data set and tried populating all missing data elements. These efforts were to determine if there were undocumented data requirements or size restrictions of the R PPRL package. Neither approach resulted in successful garbling of the test scenario data. After all reasonable approaches identified by the Health FFRDC team were attempted, the evaluation was stopped.

[7] https://cran.r-project.org/web/packages/PPRL/index.html

8. Recommendations for Selected Tool and Next Steps

8.1 Tool Recommendation

Recommendation 1: The Health FFRDC recommends the use of anonlink for record linkage in CODI. We make this recommendation based on the following findings:

- anonlink is open source and released using the Apache License 2.0, which has no limitations relevant to CODI. Organizations opting to use anonlink may use, distribute, and modify the software at no cost.
- The build and run instructions for anonlink were easy to follow, and the Health FFRDC team executed them without issue.
- anonlink uses state-of-the-art privacy-preserving techniques.
- anonlink can be easily configured by local data partners to tune its behavior for optimal performance. The tool developers provide documentation for the configuration file.
- The Precision and Recall measurements recorded by the Health FFRDC team are comparable to those observed by a different Health FFRDC team conducting similar work on behalf of the Office of the National Coordinator for Health IT. Results from anonlink were at least as good as the other tools analyzed, and the Health FFRDC team believes they are sufficient to recommend moving forward with anonlink.

The Health FFRDC acknowledges the following limitations of anonlink, which we believe are outweighed by its benefits:

- anonlink produced excess false positives when given a large population containing no actual matches.
- No commercial technical support exists for anonlink. Any technical support will need to be provided by CDC, the Health FFRDC, or local implementers.
- anonlink does not provide a means to distribute the configuration file used by the tool to compute hash values. Each CODI network must determine how best to disseminate the configuration file so that all data partners compute their hash values identically.

8.2 Recommendations for Next Steps

The Health FFRDC recognizes that additional work must be done to optimize the use of anonlink in CODI.

As shown in Section 5.5, anonlink was unable handle large volumes of data or synthetic sibling records without generating an excess of false positives. Preliminary work by the Health FFRDC team indicates that the false positive rate can be mitigated by (1) increasing the size of the Bloom filter to minimize accidental hash collisions and (2) performing multiple matching iterations with different sets of identifiers. For example, a first pass might identify all records for which full name and date of birth suffices to match the records. A subsequent pass might also consider address, phone number, or insurance number. Preliminary testing shown in Section 5.6 provides support for this approach.

Recommendation 2: The Health FFRDC recommends that additional experiments be conducted to identify the optimal configuration of anonlink to support CODI.

The Health FFRDC noted that commercial technical support does not currently exist for anonlink. Built into the CODI timeline are two mitigating components. First, the Health FFRDC will develop an Implementation Guide for Privacy-Preserving Record Linkage. This document will provide future implementers with guidance on how to obtain, install, and configure the PPRL tool selected by CDC. The Implementation Guide provides a way to augment existing anonlink documentation with additional information specific to CODI. Second, the Health FFRDC will provide technical assistance to CODI pilot sites for several months in 2020. The Health FFRDC believes that these mitigations address the risk of no commercial technical support.

Finally, the Health FFRDC noted that anonlink does not provide a means to distribute the configuration file (whereas CURL does). We believe that dissemination of the configuration file is a task best performed by each CODI network's linkage agent. The Health FFRDC team can provide a candidate configuration for the pilot networks.

Appendix A. Identity Matching Tools

The following tools were identified by the CODI Tools Landscape Analysis subgroup for consideration in the project. The list was quickly reduced to a smaller set of tools because many of these are identity matching solutions that do not support PPRL.

- anonlink
- BigMatch
- CURL
- D-Dupe or Dedupe
- Datavant
- fast Probability Record Linkage
- Freely Extensible Biomedical Record Linkage (FEBRL)
- Fast Probabilistic Record Linkage (FRIL)
- FuzzyMatcher
- GRLC/PPRL R Package combines Merge ToolBox functionality
- Health Data Link (HDL)
- Java gEneric DAta Integration (JedAI) Toolkit
- LinkIT/FRIL
- LinkPlus
- Merge ToolBox
- MERLIN
- MITRE-developed Patient Matching tools
- OYSTER Entity Resolution
- Point-of-contact Interactive Record Linkage
- PRIVATEER
- Python Record Linkage Toolkit
- R RecordLinkage/CRAN
- REcord Linkage At Istat (RELAIS)
- ReMaDDer
- SecondString
- SILK
- Similarity Metric Library (SimMetrics)
- TAILOR (RecOrd LinkAge Toolbox)
- The Link King
- WHIRL

Appendix B. Identity Matching Tools Landscape Analysis

Prior to conducting the Goodness of Fit (GoF) analysis, the Childhood Obesity Data Initiative (CODI) Collaborative Working Group (CCWG) convened a subgroup to evaluate identity matching and deduplication (i.e., information merging) tools for their ability to provision encryption keys and hash methods to support privacy-preserving record linkage (PPRL). The overall goal of this tools landscape analysis (TLA) was to make recommendations about which tool(s) should be evaluated for a potential CODI solution.

B.1 Analysis Process

The CCWG convened a subgroup to conduct the TLA. This subgroup collectively arrived at the decision to utilize publicly available information (i.e., without any direct contact with software developers/vendors) to identify the identity matching tools and make recommendations on the tool(s) to be included in the GoF analysis. This provided a level playing field between open source solutions, which may not have a team to field the subgroup's questions, and commercial solutions. Additionally, some CCWG members are involved in the development of PPRL solutions. In order to avoid conflicts of interest, those participants were not involved in the TLA subgroup, nor were they contacted in matters concerning the TLA.

The TLA subgroup was led by Pradeep Podila, a Centers for Disease Control and Prevention (CDC) Fellow deployed at Denver Health. The subgroup membership is listed in Table B-1.

Table B-1. TLA Subgroup Membership

Member	Organization
Tom Carton	Louisana Public Health Institute/REACHnet
Nedra Garrett	CDC
Andy Gregorowicz	MITRE
Dawn Heisey-Grove	MITRE
Ray King	CDC
Keith Miller	MITRE
Kris Mork	MITRE
Pradeep Podila	CDC/Denver Health
Ken Scott	Denver Health
Carmen Smiley	Office of the National Coordinator for Health IT

B.1.1 Preferred Tool Attributes

PPRL tools were evaluated for the TLA based on a checklist developed by the subgroup. Table B-2 lists the attributes included in the checklist.

Table B-2. PPRL Tool Checklist

Attribute	Details
Most current update	Determine when the tool was last updated. The goal of this attribute is to ensure that the tool has an active community that is maintain the software package.
Availability of technical support	Discover whether there are forums, mailing lists, issue trackers, or other mechanisms by which support can be obtained for the software package.
Availability of product documentation	List whether documentation has been published for the tool.
License	Determine the tool software license and what impact that may have on the project.
Programming language	The programming language used to create the tool.
Operating system support	List the operating systems that the tool supports.
Data matching	Probabilistic, deterministic, etc.
Pre-processing	The ability of the tool to clean or otherwise interact with the source identity data prior to matching.
Blocking	The ability of the tool to split potential matches into groups to reduce the amount of comparisons that must be made to complete the matching process
Machine learning	List whether the tool supports any machine learning based matching techniques.
Input data types	File types accepted by the matching tool, such as comma-separated values (CSV), XML, JSON, etc.
Hardware requirements	List any hardware requirements, such as processor capability, RAM, or disk space requirements.
Additional features	List features that may not be necessary to the process, but can add benefit, such as a Graphical User Interface or Application Programming Interface (API).
Scalability	List any evidence discovered that shows the tool being used to match large populations of individuals.
Software limitations	List any know limitations of the software, such as the ability to match patient populations under a certain size.

B.1.2 Information Gathering

To populate the attributes listed in Table B-2, the subgroup consulted public sources of information. This included examining publications, white papers, and online presentations, as well as information from GitHub and Wikipedia.

The subgroup referenced several books for greater detail on the matching process and to identify additional PPRL tools for consideration. These included:

- Data Quality and Record Linkage Techniques (2007) by Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler
- Entity Resolution and Information Quality (2010) by John R. Talburt

- Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection (2012) by Peter Christen
- Methodological Developments in Record Linkage (2015) by Katie Harron, Harvey Goldstein, and Chris Dibben

B.1.3 Information Review

The TLA subgroup met between March 22, 2019, and May 17, 2019. During that time, the group reviewed artifacts developed by the subgroup lead. Feedback was provided to the subgroup lead during weekly meetings.

The subgroup lead generated a list of PPRL tools for further investigation (Appendix A.). Based on the list, the lead conducted research to populate the tool checklist. The populated checklist can be found in Appendix C. . The subgroup discussed that information, and the list of recommended tools for inclusion in the GoF was finalized on May 17, 2019 and presented to the CCWG on May 27, 2019.

B.2 Selected Tools

The subgroup made a consensus recommendation that the following tools be included in the GoF analysis:

- anonlink (open source)
- R PPRL package (open source)
- Colorado University Record Linkage (CURL) (commercial, closed source)

B.2.1 anonlink

anonlink is an open source PPRL package created and maintained by Data 61, a division of Commonwealth Scientific and Industrial Research Organisation, Australia's national science research agency. Source code is available on GitHub and published under the Apache License, Version 2.0.

anonlink provides a series of software repositories on GitHub with content that can be used to facilitate PPRL. Most of the software in the anonlink repositories uses the Python programming language for implementation, with portions written in C++ to improve performance.

One of the anonlink software repositories is called clkhash. This repository contains the application that is responsible for generating garbled information from PII. Matching logic is handled in a Python library called anonlink. There is also a software repository that contains a matching solution called anonlink-entity-service, which packages anonlink into a server using Docker, a containerization technology. The Docker server provides an API that allows users to create new matching scenarios, upload the garbled information, and obtain matching results. These tools are updated frequently, with the latest version of clkhash being published on November 14, 2019.

B.2.1.1 Rationale for anonlink Selection

anonlink met several of the criteria outlined in the checklist. It is released as open source under a license that is compatible with use in CODI. anonlink has published documentation, and the publicly available GitHub source code repositories appear to receive regular updates.

From the perspective of deployment and operation, anonlink also fits the criteria. The clkhash application operates on comma-separated values (CSV) files, which are typically easy for systems to generate. No operating system requirements were listed, but anonlink's Python implementation suggests that it can be deployed on Microsoft Windows, macOS, or Linux systems. No hardware requirements were listed nor were any limits on the size of scale of data provided. Literature provided by the tool stated that it was capable of matching two data sets, each containing 100,000 records, in under five minutes.[8]

B.2.2 CURL

CURL is a closed source PPRL package developed by University of Colorado (CU), Anschutz Medical Campus.

The CURL website describes the tool as follows[9]:

> CU Record Linkage (CURL) system is a platform to perform record linkage operations including linkage configuration and management, data normalization, data encryption and hashing, data linkage, data de-duplication, and linked data dissemination. CURL platform supports both centralized and distributed record linkage. CURL implements modular software architecture which allows each module in CURL to be updated, added or removed without the need to change the source code of the core platform.

B.2.2.1 Rationale for CURL Selection

While CURL did not meet the open source criteria, it satisfied many other requirements. The tool is actively maintained by a team at CU, which offers technical support. It is written in the Java programming language, allowing it to be deployed on Microsoft Windows, macOS, and Linux systems. The tool operates on pipe "|" delimited files, which are similar to CSV files. The product did not state any hardware requirements or limits on data size or scale. CURL's use on prior projects was a positive sign that it could handle the size and scale of the CODI Pilot population.

B.2.3 R PPRL Package

The R PPRL Package is an open source tool developed by the German Record Linkage Center. It is provided under the GPL V3 open source license.

The R PPRL Package website describes the tool as "A toolbox for deterministic, probabilistic and privacy-preserving record linkage techniques."[10]

B.2.3.1 Rationale for R PPRL Package Selection

[8] https://github.com/data61/anonlink#benchmark
[9] http://www.ucdenver.edu/academics/colleges/medicalschool/programs/d2V/tools/Pages/CURL.aspx
[10] https://cran.r-project.org/web/packages/PPRL/

The R PPRL Package met the open source criteria; however, there were concerns about license. R PPRL is licensed under GPL V3, which requires users to disclose all source code for the system that the package is used in when the system is distributed between parties. Distribution of the R PPRL package to data partners may necessitate the disclosure of source code for other software involved in the matching process.

The tool's most recent version was published on January 8, 2019, along with documentation, indicating active work. It is written in the R programming language, which allows it to be deployed on Microsoft Windows, macOS, and Linux systems. The tool operates on R data frames, which can be created from CSV files. The product did not state any hardware requirements or limits on data size or scale.

Appendix C. PPRL Tool Checklist

The following tables contain the PPRL Tool descriptions, evaluation grid, tool abilities and requirements, and software information generated by the CDC Fellow as a part of the tools landscape analysis.

Table C-1. PPRL Tool Descriptions

Tool	Tool Description	Organization	Developer/ Owner	Contact Information for Sales	Contact Information for Technical
ANONLINK	Anonymous linkage using cryptographic hashes and bloom filters. A Python (and optimized C++) implementation of anonymous linkage using cryptographic linkage keys. Use clkhash to create cryptographic linkage keys from PII.	Australian National University [Data61/CSIRO]	Brian Thorne, Stephen Hardy, Jakub Nabaglo, Wilko Henecka, Hamish Ivy-Law		Brian.Thorne@data61.csiro.au
PPRL Package	A toolbox for deterministic, probabilistic and privacy-preserving record linkage techniques. Combines the functionality of the 'Merge ToolBox' (<http://record-linkage.de>) with current privacy-preserving techniques.	German Record Linkage Center (GRLC)	Rainer Schnell		Dorothea Rukasz: mitarbeiter.schnell@uni-due.de
FRIL + LinkIT	LinkIT (open-source tool) implements novel algorithms that support data transformations for linking sensitive attributes, and is designed to work with our previously developed tool, FRIL (Fine-grained Record Integration and Linkage), to provide a complete record linkage solution. LinkIT can be also used as a stand-alone secure transformation tool to link string records.	Emory University	Luca Bonomi, Li Xiong, James J. Lu		
SOEMPI	Secure Enterprise Master Patient Index is a Scientific record linkage tool for performing privacy preserving and non-privacy preserving record linkage. It is based on	Vanderbilt University	Csaba Toth		support@sysnetint.com

Tool	Tool Description	Organization	Developer/ Owner	Contact Information for Sales	Contact Information for Technical
	OpenEMPI and the research and the work is conducted by the Vanderbilt University.				
CURL	Colorado University Record Linkage (CURL) system is a platform to conduct record linkage operations like data normalization, encryption and hashing by using a user-friendly graphical user interface and incorporating secure hashing methods and a secure network firewall.	UC, Denver	Toan Ong	CUInnovations@ucdenver.edu	curl.support@ucdenver.edu
Health Data Link	SAFE, SECURE, & ACCURATE DATA LINKAGES	Health Data Link	Jacob Plummer (CEO)/ Satyendra Goel (TO)		Online leave a message
Datavant	Connecting the World's Health Data: Datavant helps organizations protect, link, and exchange their health data	Datavant	Travis May (CEO)	bob@datavant.com	support@datavant.com

Table C-2. PPRL Tools Landscape Analysis Evaluation Grid

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Country of origin	Australia	Germany	USA	USA	USA	USA	USA
Year of development/ release	No direct information available	March 20 2018	October 21 2008	No direct information available	No direct information available	No direct information available	No direct information available
Current version (# revisions), Release Date	0.6.0 (Nov 27, 2017); 0.11.2 (3/15/2019)	January 8 2019	November 12 2011	August 4 2014	No direct information available	No direct information available	No direct information available

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Where is the software source code hosted?	Github	CRAN R Project	Sourceforge	HIP Lab Vanderbilt/ Github	Commercial	Commercial	Commercial
Availability of ongoing maintenance	Yes	Yes	No	No	Yes	Yes	Yes
Has the tool been used in healthcare world?	Yes	No direct information available	No direct information available	No direct information available	Yes	Yes	Yes
Known users?	No direct information available	No direct information available	No direct information available	No direct information available	PEDSnet, PCORnet	CDRNs, Virtual inter-state disease registries, Virtual patient record locators	PCORnet
Application of the tool	No direct information available	No direct information available	Educational	No direct information available	Education, Research	Research, Healthcare	Research, Healthcare
# Records the tool could process	1 million x 1 million in under 5 mins	No direct information available	No direct information available	10,000 x 10,000 records in 2 mins	No direct information available	Generate linkages within Hours	15 billion patients records across 200 data sources
Contact information of known users	No direct information available	No direct information available	No direct information available	No direct information available	PEDSnet, PCORnet	CDRNs, Virtual inter-state disease registries, Virtual patient record locators	PCORnet
Scalability**	Yes	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Availability of support staff	No direct information available	No direct information available	No	No direct information available	Yes	Yes	Yes
Type of availability	No direct information available	No direct information available	No	No direct information available	Email Support	No direct information available	Yes. Active Blog
Open source	Yes	Yes	Yes	Yes	No	No	No
Open source - license details	Apache 2.0	GPL-3	No direct information available	Apache 2.0	No	No	No
Cost ($) associated with licensing	No	No	No	No	$	No direct information available	No direct information available
Compatibility with CODI Query Architecture	Yes	Yes	No direct information available	No direct information available	Yes	No direct information available	No direct information available
How does it communicate	REST APIs only limited parameters	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available
Operating system	Windows, Mac OS X, Linux	Windows, Linux	Windows, Linux	Windows, Linux	No direct information available	No direct information available	No direct information available
Programming language	Python	R	No direct information available	Enterprise Java Beans (EJB)	Python	No direct information available	No direct information available
Style of operation	Standalone, and Cloud-ready	Standalone	No direct information available	No direct information available	Standalone	No direct information available	No direct information available
Command line tool	Yes	Yes	No direct information available	No	No direct information available	No direct information available	No direct information available
Graphical User Interface (GUI) or Step-by-step wizard	No	No	Yes	Yes	Yes	No direct information available	No direct information available

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Research prototype (Programs/ Codes only)	No	No	Yes	No	No	No	No
Database dependencies	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available
Analytic dependencies	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available
Availability of extensive user documentation	Yes	Yes	Yes	Yes	No direct information available	No direct information available	No direct information available
Availability of example programs	Yes	Yes	Yes	Yes	No direct information available	No direct information available	No direct information available

Table C-3. PPRL Tool Abilities

Ability to Work With:	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Comma separated values (csv) files	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available
XML documents	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available
JSON files	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available

Table C-4. PPRL Tool Requirements

Requirements of IM Tool	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Accommodate fields	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	Yes
Handle fields for pre-processing	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	Yes
Normalize data	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available
Customize salts	No direct information available	No direct information available	No direct information available	Yes	Yes	AnoX Signature Generator	No direct information available
Hash	Yes	Yes	Data Transformations	No direct information available	Yes	AnoX Signature Generator	Yes
Type of hash	SHA2/ Cryptographic	Cryptographic Linkage Keys	Data Transformations	No direct information available	SHA2 /Cryptographic	AnoX Signature Generator	irreversible hash function
Does the hash method support probabilistic or deterministic linkage	Probabilistic	Both	No direct information available	No direct information available	Both	No direct information available	Probabilistic
Filtering and Blocking	Yes	Yes	No direct information available	No direct information available	Yes	No direct information available	No direct information available
Type of filtering and blocking	Bloom Filter	Bloom Filter	No direct information available	Bloom Filter	Bloom Filter	No direct information available	No direct information available
Filtering with multibit trees	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available
Prevent Cryptographic attacks (in-built measures)	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available

Requirements of IM Tool	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Parallel process	Yes	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available
Parallel record linkage approaches	Yes	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available	No direct information available

Table C-5. PPRL Tools Export Abilities

Ability to export results of record linkage into:	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Comma separated values (csv) files	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available
XML documents	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available
JSON files	No direct information available	No direct information available	No direct information available	No direct information available	Yes	No direct information available	No direct information available

Table C-6. PPRL Tool Software Downloads and Limitations

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Software download link#1	https://dmm.anu.edu.au/DLforum/viewtopic.php?t=27	https://cran.r-project.org/web/packages/PPRL/index.html	http://fril.sourceforge.net/download.html	https://github.com/MrCsabaToth/SOEMPI	-	-	-

Evaluation Criteria	ANONLINK	PPRL Package	FRIL + LinkIT	SOEMPI	CURL	Health Data Link	Datavant
Software download link#2	https://github.com/data61/anonlink	http://grlc.german-microsimulation.de/services/privacy-preserving-record-linkage-r-package-pprl/index.html		http://hiplab.mc.vanderbilt.edu/projects/soempi/	-	-	-
Software download link#3	-	-	-	https://hiplab.mc.vanderbilt.edu/projects/soempi/user_setup.html	-	-	-
Additional details (includes Limitations)	The linkage process has order n^2 time complexity - although algorithms exist to significantly speed this up. Several possible speedups are described in http://dbs.uni-leipzig.de/file/P4Join-BTW2015.pdf	GPL License	No download link available for "LinkIT" open-source tool. Note: Luca Bonomi is currently with UCSD (lbonomi@ucsd.edu)	Extended the OPENEMPI into SOEMPI. Other instances can be inherited from OPENEMPI.	-	https://www.healthdata.link/	https://datavant.com/

Appendix D. PPRL Frequently Asked Questions

D.1 How does CODI link individuals' records across settings and systems while ensuring their privacy is maintained?

Through CODI, data coordinating centers (DCC) can link an individual's record across heath information systems and non-clinical community-based programs while protecting individuals' personally identifiable information (PII) using privacy-preserving record linkage (PPRL). PPRL involves a process of hashing completed by each data partner in the CODI network, which garbles an individual's PII behind the organizations' firewalls. Those hashed values are considered de-identified data and are shared with the DCC, which conducts probability matching across organizations' data. The output of that matching process are LINKIDs that are shared with the data partners and used in future CODI queries for creating linked longitudinal records for each individual.

D.2 What is privacy-preserving record linkage? How does privacy-preserving record linkage work?

Privacy-preserving record linkage relies on garbling individuals' personally identifiable information (PII) data in a process called hashing. This term comes from the common use of the word "hash" and means that the process makes a mess of the data. Hashing prevents access to individuals' PII while allowing records to be linked across organizations based on an encoded identifier. Links can be established across organizations without disclosing PII outside the originating organization.

Each organization first applies the hashing function to garble the PII and then shares the garbled data with a STTP. Data are garbled in the same way in the different organizations. That way, although the STTP does not have access to individuals' PII, it knows that records exist for the same child at both organizations because the garbled values are the same from both organizations.

D.3 How are individuals' identifiers garbled to preserve privacy?

Individuals' personally identifiable information (PII) are passed through a hashing function that:

- Garbles the data.
- Is "one-way." It is very difficult, if not impossible, to derive the original PII from the garbled output.
- Is deterministic. The same input will always generate the same output.

This process works when there is an exact record match. It can also be used to make approximate matches—for example, to account for nicknames or keystroke errors.

D.4 What else does CODI use to protect individuals' private data?

One weakness of hashing is that an adversary can independently create hash values for an individual. To protect against this attack, an encryption key is added to the inputs before hashing. The encryption key is a randomly generated value provided as an extra input to the hashing

function. The encryption key is shared through a secure data exchange to all data partners. As long as the encryption key is kept secret, hashing is safe from this kind of attack.

D.5 Why are the data sent out by CODI organizations for record linkage considered de-identified data?

The garbled data (or hash values) shared by CODI data partners for record linkage are considered de-identified data because any identifying characteristics are removed through the hashing process. HIPAA does not have data sharing restrictions for de-identified data because any identifying health information has been removed. CODI will engage data experts to provide expert determination that the hashed values are ‘de-identified’; this is a process in which the expert validates that the data are statistically unlikely to be re-identified by a potential adversary.

Acronyms

API	Application Programming Interface
CCWG	CODI Collaborative Work Group
CDC	Centers for Disease Control and Prevention
CODI	Childhood Obesity Data Initiative
CSV	Comma-Separated Values
CURL	Colorado University Record Linkage
DCC	Data Coordinating Center
DDN	Distributed Data Network
FFRDC	Federally Funded Research and Development Center
GoF	Goodness of Fit
GPL	GNU Public License
IT	Information Technology
JAR	Java Archive
JSON	JavaScript Object Notation
PII	Personally Identifiable Information
pip	Package Installer for Python
PPRL	Privacy-Preserving Record Linkage
STTP	Semi-Trusted Third Party
TLA	Tools Landscape Analysis
TTP	Trusted Third Party

Glossary

Bloom Filter	A data structure that is often used to probabilistically test the presence of an element within a set. Bloom filters are space efficient, meaning that they allow for the testing of presence in a set without needing to have access to the entire set. This space efficiency is achieved by a process that can allow for false positives to be provided when testing for element presence.
Hashing	Hashing is a mathematical function with two key properties. First, the same inputs always produce the same output. Second, given the output, it is nearly impossible to determine which inputs were used. Hashing transforms input data by shuffling and mixing up the information it is given.
Encryption Key	An encryption key is typically a random string of bits generated specifically to scramble data. Encryption keys are created with algorithms designed to ensure that each key is unique and unpredictable. Salt values are examples of encryption keys.
Salt	Random data that is applied to a Hashing function. Salt prevents attackers from reversing a hashing process by guessing the input values.
Modulo	A mathematical operation that provides the remainder after division of one number by another.
Information Garbling	The process of transforming information so that it cannot be easily reconstructed by an unauthorized party. Some forms of garbling are reversible given an encryption key, such as symmetric encryption. Other forms of garbling, such as the Bloom filters constructed using cryptographic hashes, are intended for one-way usage.

Notice