**Prepared for:**

## Centers for Disease Control and Prevention

**Centers for Medicare & Medicaid Services 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 Implementation Guide

**For the Denver, Colorado Pilot (2019–2020)**

**Final Draft**

**Version 1.1**

**July 10, 2020**

The views, opinions, and/or findings contained in this report are those of The MITRE Corporation and should not be construed as official government position, policy, or decision unless so designated by other documentation. This guide may serve as a reference or framework for others implementing similar childhood obesity data solutions. However, the CODI Implementation Guide does not represent official views or guidance of CDC.

# Record of Changes

| Version | Date | Author / Owner | Description of Change |
|---|---|---|---|
| 1.0 | April 27, 2020 | A. Gregorowicz / Health FFRDC | Initial Version |
| 1.1 | July 10, 2020 | A. Gregorowicz / Health FFRDC | Revision based on feedback from expert determination and the implementation work group |

# Table of Contents

# List of Figures

# List of Tables

# 1. Introduction

As part of the Centers for Disease Control and Prevention's (CDC) efforts to promote health, prevent disease, injury, and disability, and prepare for emerging health threats, the Division of Nutrition, Physical Activity, and Obesity, and the Center for Surveillance, Epidemiology, and Laboratory Services partnered with the CMS Alliance to Modernize Healthcare federally funded research and development center (Health FFRDC) on the Childhood Obesity Data Initiative (CODI). CODI will expand the ability to capture, standardize, integrate, and query existing patient-level electronic health record (EHR) and community data. CODI uses privacy preserving record linkage (PPRL) to provide a way to gather an individual's information across clinical and community organizations, which facilitates a unified view of an individual for researchers to work with.

This document describes how to conduct the PPRL process using the selected technology. This involves different organizations in different roles working to build linkages while respecting individual privacy.

## 1.1 Background

Individuals are likely to have data at multiple organizations. To construct a complete picture of an individual for research purposes, it is critical to be able to link information gathered from different places into a longitudinal record.

PPRL is a process where organizations can create this linkage without directly sharing personally identifiable information (PII) with each other. Using PPRL, individuals are assigned an identifier, called a LINKID in the CODI project.

This PPRL solution is designed to operate in a distributed health data network (DHDN), in which data requests needed to answer researchers' queries are distributed across a number of clinical and non-clinical community partners. CODI relies on the Patient Centered Outcomes Research Network (PCORnet) or a PCORnet-compatible infrastructure. When responding to those distributed queries, organizations can include the LINKIDs. This allows for the construction of a longitudinally linked set of records.

This document is based upon several artifacts, including the CODI Data Architecture Gaps and Recommendations report[1], which was informed by the research question formulation, and the decision by the CODI Collaborative Work Group during an in-person meeting in December 2018 to adopt PPRL and a logical data warehouse query architecture.

The CODI Tools Landscape Analysis (TLA) subgroup examined several PPRL solutions and put forward recommendations in May 2019. The Health FFRDC performed a Goodness of Fit analysis on the recommended PPRL solutions. This analysis was delivered in December 2019 and concluded that the TLA recommended tool, *anonlink*, was suitable for use in CODI.

[1] https://3.basecamp.com/4113007/buckets/9652569/uploads/1749256123

Finally, the CODI Implementation Work Group has held discussions on matters relating to PPRL. The preferences of the group informed the development of this document.

## 1.2 Purpose

The purpose of this document is to provide the guidance necessary for participating organizations to implement PPRL. Toward that end, this document provides:

- A description of the PPRL process
- Descriptions of the roles for different participating organizations
- Specific guidance for each PPRL role
- Guidance for evaluating performance
- An appendix with content specific to the Denver pilot

## 1.3 Scope

This document provides implementation guidance for the PPRL process. It assumes that PII is stored in databases that conform to the CODI Record Linkage Data Model. The structure of this model and guidance for populating it can be found in the CODI Data Models Implementation Guide[2].

Some of the guidance provided in this document is implemented as open source software. The two particular software packages of interest are:

- Data Owner Tools[3]
- Linkage Agent Tools[4]

The CODI PPRL solution relies on *anonlink* for de-identification and matching. This document covers usage of *anonlink* at a high level. Further detail on *anonlink* configuration and operation can be found at:

- *clkhash*—the component of *anonlink* used to de-identify information
- *anonlink-entity-service*—the containerized version of *anonlink*

## 1.4 Audience

The primary audience for this document is the technical staff of organizations implementing a PPRL process. This document is written with CODI participating organizations as a primary focus, but it is applicable to other efforts seeking a PPRL solution. The secondary audience is those staff concerned with information security and privacy for participating organizations. Health services researchers may be interested in the PPRL process which is described in section 2 and section 3.

[2] https://3.basecamp.com/4113007/buckets/9652569/uploads/1986134733

[3] https://github.com/mitre/data-owner-tools

[4] https://github.com/mitre/linkage-agent-tools

## 1.5 Document Organization

This document is organized as follows:

- Section 2 – Background on CODI and the roles involved in PPRL
- Section 3 – An overview of PPRL and how it is implemented in CODI
- Section 4 – Data Owner guidance
- Section 5 – Data Partner guidance
- Section 6 – Linkage Agent guidance
- Section 7 – Key Escrow guidance
- Section 8 – Guidance for performance evaluation and implementation details
- Appendix A. – Guidance for the Denver Pilot

# 2. CODI Background

This section first summarizes the CODI record linkage process. It then defines several roles relevant to implementing the PPRL process.

## 2.1 Privacy Preserving Record Linkage Introduction

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 using 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.

PPRL is an alternative set of techniques to solve the issue of identity matching without exchanging PII directly. 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 of 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. The third party then compares the garbled information to establish linkages. Figure 2-1 illustrates this process.

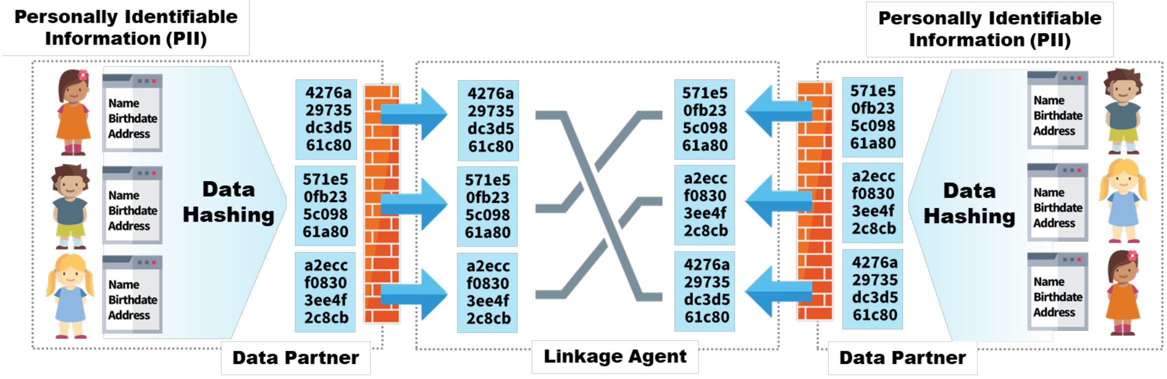

**Figure 2-1. Example of Privacy Preserving Record Linkage Performed by a Linkage Agent**

The third party conducting the matching assigns an identifier when a linkage is found and communicates the LINKID 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 are explored in section 3.1.

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 instead is working with garbled, de-identified data.

## 2.2 CODI Roles

A *data owner* is an organization that has data to contribute for research queries. A *data partner* is an organization that participates in the distributed network by hosting data. Many data partners will host their organization's own data, meaning they will be both a data owner and data partner. In other cases (e.g., non-clinical community health partners), the organization that contributes data will rely on an intermediary data partner to host their data.

A *linkage agent* is an organization that performs linkage on behalf of data partners. The linkage agent receives de-identified PII and produces globally unique LINKIDs used to construct longitudinal records. Ultimately, longitudinal records will be assembled by an organization in the *Data Coordinating Center (DCC)* role. The DCC distributes research queries to data owners, receives their responses, and conducts any analyses needed to meet researchers' requests.

A *key escrow* is an organization responsible for generating an encryption secret, called a "salt," that is used in the de-identification process. The key escrow will provide the salt value to data owners and data partners securely to ensure the security of the process.

The key escrow must be a separate entity from the linkage agent to ensure the privacy of the garbled information shared with the linkage agent. The key escrow also must be a separate entity from the DCC. The DCC and linkage roles can be filled by a single organization.

# 3. Privacy Preserving Record Linkage

PPRL is the process of matching individuals based on de-identified 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 linkage agent shares configuration information with each data partner. The key escrow provides a secret "salt" value to the data partners and data owners. The salt value will be the same for all data partners and data owners.

2. Each data partner creates a de-identified data set by:
   - Extracting PII from its operational database.
   - Passing the PII and salt value through a hashing process that will garble the information.
   - Sharing the garbled data with the linkage agent.

3. The linkage agent develops LINKIDs by:
   - Determining which de-identified values correspond to the same individual.
   - Establishing a unique LINKID for each individual.

4. The linkage agent shares the LINKIDs with each data partner.

Each data partner or data owner 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 linkage agent to perform matching. The following section describes the matching approach that will be used.

## 3.1 Process Overview

In order to handle variations in demographic information, this solution applies probabilistic matching. A key component of this is the use of 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.

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, the adversary can learn which data partners have information about a particular person if the adversary has access to the hashed data. To protect against this kind of attack, a "salt," or encryption secret, is added to the inputs before hashing.

For probabilistic 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 are then 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.

Bloom filters offer efficient storage of information and are 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 remainders of those operations are then used to set the positions in the Bloom filter to 1.

In our 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 3-1 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 3-1. 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 |

When two different inputs generate the same output, it is referred to as a collision. In the example calculations in Table 3-1, even though different hash values for the name fragment “oh” were created from the two encryption keys, the same bit was set when passing these through the modulo operation, which is the remainder after division. While Bloom filters are space efficient, situations where different inputs result in the same bit being set are possible.

The resultant Bloom filter from the Table 3-1 example is: 0000001001000000100000000000000000000010000000000100000000000000. This shows five unique cases when the value for the filter has been set to “1,” as indicated in Table 3-1. In our example, the first name fragment will set the value at the index position of 6, which is the 7th bit in the Bloom filter.

### 3.1.1 Comparing Bloom Filters

Bloom filters are often used to check for probabilistic set membership. The Bloom filter example 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. Because of the possibility of collisions, it is only likely—not definitive—that fragments “na,” “at,” “th,” “ha,” and “an” will report as false.

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, which results 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 (SDC), sometimes referred to as a Dice coefficient. Calculation of this metric starts by tabulating the values in Table 3-2.

**Table 3-2. Dice Coefficient**

| 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 3-1.

$$SDC = \frac{2TP}{2TP + FP + FN}$$

**Figure 3-1. Dice Coefficient Equation**

The Dice coefficient provides a value between 0 and 1. Comparing Bloom filters with the exact same inputs will result in a coefficient of 1. Comparing filters created from dissimilar inputs will result in a value closer or equal to 0. If a record has a given name of “John” and another record has a given name of “Johnathan,” Bloom filters derived from these different records can be compared using the Dice coefficient value. That value allows for a determination of whether the records likely represent the same individual.

Using this approach, data partners and data owners build Bloom filters based on individuals’ identity information. The hashing that is needed to create the Bloom filter uses the salt value provided by the key escrow.

Data partners and data owners transmit the Bloom filters they created to the linkage agent. These filters can then be compared between data partners. Filters that have a Dice coefficient above a threshold established for the matching process are considered a match.

### 3.1.2 Multiple Bloom Filters

Based on experiments conducted during the Goodness of Fit analysis, it was determined that the best approach for CODI is to develop multiple Bloom filters for each individual. Each filter is constructed from a different set of identity attributes. The matching process is performed for each set of Bloom filters and the results are combined to determine the final matches across individuals and data owners. An example of this is shown in Figure 3-2.

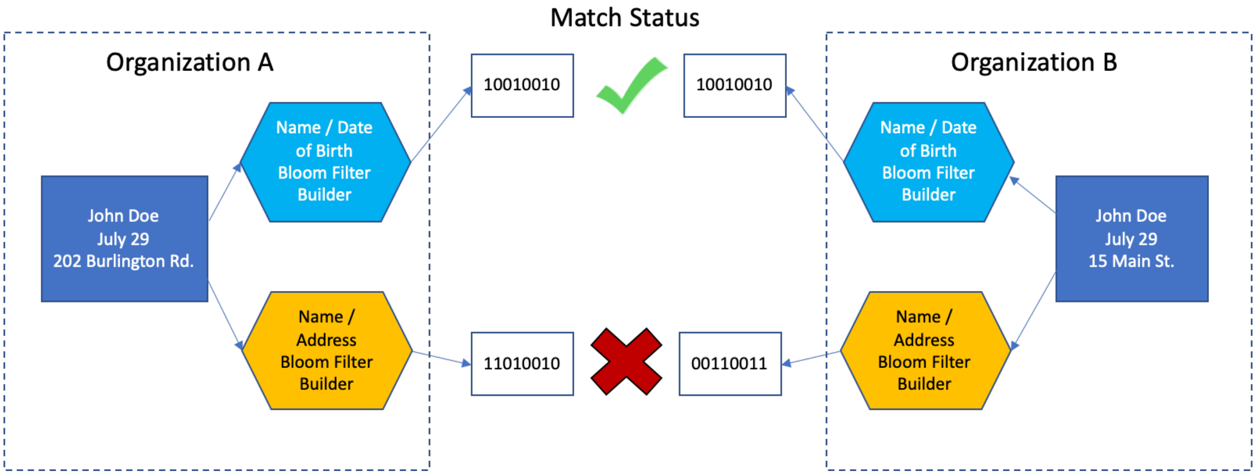
Match Status
Organization A
John Doe
July 29
202 Burlington Rd.
Name / Date of Birth Bloom Filter Builder
Name / Address Bloom Filter Builder
10010010
10010010
11010010
00110011
Organization B
Name / Date of Birth Bloom Filter Builder
Name / Address Bloom Filter Builder
John Doe
July 29
15 Main St.

**Figure 3-2. Matching with Multiple Bloom Filters**

In this example, the records for "John Doe" are used to create two separate Bloom filters. The first Bloom filter is based on name and date of birth. The second is created using name and address. When comparing the Bloom filters, the first set will identify a match, while the second will not due to the records having different addresses. Ultimately, the PPRL process will set a threshold for the number of Bloom filters needed for records to be considered a match.

## 3.2 Selected Technology

Our selected tool for implementing the PPRL process is the open source *anonlink* software package, which handles the construction of Bloom filters from PII. It also provides the capability to compare Bloom filters to determine record linkages.

Data owners and data partners shall use *anonlink*'s *clkutil*. This software accepts PII in comma separated values (CSV) format, along with a configuration file to output the de-identified information into a JSON file, which will be provided by the linkage agent. The software is called *clkutil* because the *anonlink* project refers to the Bloom filters created by the process as cryptographic long-term keys. This information will be passed to the linkage agent.

The linkage agent shall run the *anonlink-entity-service.* This tool offers a web service that accepts the de-identified information and performs matching. The service then returns groups of identifiers where the Bloom filters match above a supplied threshold.

There is no need to interact with these tools directly. Instead, data owners and data partners will use the open source *Data Owner Tools* package to work with *clkutil*. Linkage agents shall use the *Linkage Agent Tools* package to manage interactions with the *anonlink-entity-service*. We describe these tools in greater detail in the role-specific sections.

## 3.3 Assignment of Identifiers

To preserve individual privacy, *anonlink* does not assign identifiers, such as the PCORnet Common Data Model PATID, to the generated Bloom filters. When *clkutil* is used to de-identify information, the resulting Bloom filters are stored in a JSON array. *anonlink* uses the position in the JSON array as the identifier for the individual. The array position will correspond to the position of the PII in the CSV file generated by the data owners and data partners.

At the linkage agent, the *anonlink-entity-service* will provide a grouping of matched records as array positions in the files provided by the data owners and data partners. The linkage agent stores these array positions and performs deconfliction between groupings. The linkage agent then assigns a LINKID to these groupings.

The linkage agent shall ensure that every Bloom filter provided by data partners and owners is assigned a LINKID. Matching Bloom filters across organizations will be assigned the same LINKIDs. Bloom filters with no corresponding matches shall be assigned LINKIDs that are unique to the originating record at the single data owner or data partner. This ensures that all records supplied to the linkage agent are assigned a LINKID. This process is performed by *Linkage Agent Tools* and is detailed in section 6.3.

The linkage agent shall communicate the array position and associated LINKID back to data owners and data partners. Data owners and data partners use the CSV file containing PII behind their organizational firewall to translate the array position into a PATID, allowing an association of a LINKID to a PATID. This translation process is performed by *Data Owner Tools.*

LINKIDs shall be a Universally Unique Identifier (UUID) Version 1 as specified in RFC 4122.[5]

## 3.4 Process Frequency

It is recommended that the PPRL process be conducted annually. Data owners and data partners are expected to maintain LINKIDs in the LINK table of the CODI Record Linkage Data Model from previous years to facilitate reproduction of research. There is no expectation that the linkage agent will retain information from previous matching years. The entire PPRL process can be performed without any input of prior years' matching processes. There is no expectation that records will be assigned the same LINKID from year to year.

[5] https://tools.ietf.org/html/rfc4122

# 4. Guidance for Data Partners Hosting Their Own Data

The guidance in this section is provided for data owners who will be performing hashing, transmitting information to the linkage agent, and responding to research queries. Data owners who will be working with data partners to host information should refer to section 5.

The process of extracting information and preparing it for transmission to the linkage agent is illustrated in Figure 4-1.

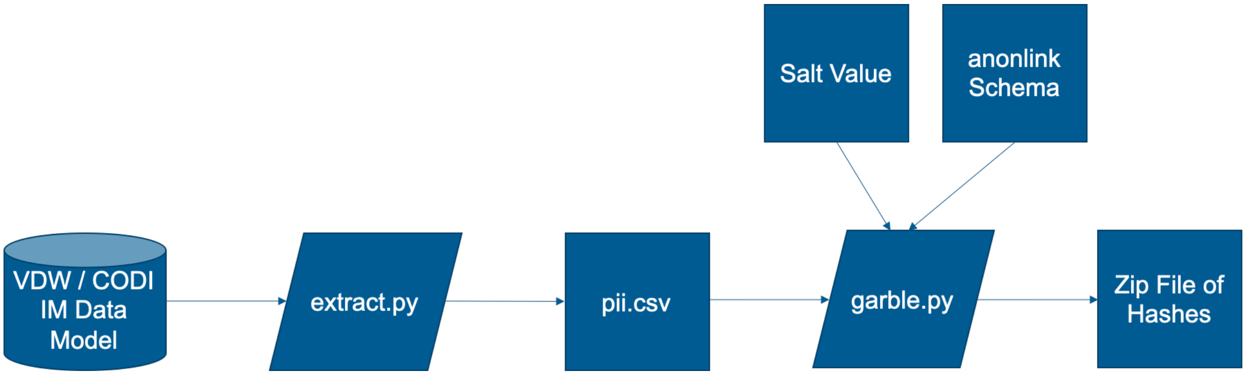

**Figure 4-1. Data Owner Information Flow**

## 4.1 Data Extraction

Data owners shall store PII to be used in the PPRL process in the IDENTIFIER table as specified in the CODI Record Linkage Data Model (RLDM). The *Data Owner Tools* software connects to this database to extract the information from this table. *Data Owner Tools* uses the SQLAlchemy library to connect to the database containing PII. Data owners must provide PII in a database compatible with SQLAlchemy, which includes PostgreSQL, MySQL, Microsoft SQL Server, or Oracle.

*Data Owner Tools* retrieves all rows in the IDENTIFIER table. Extraction is performed by the "extract.py" script provided in *Data Owner Tools*, which is highlighted in Figure 4-2.

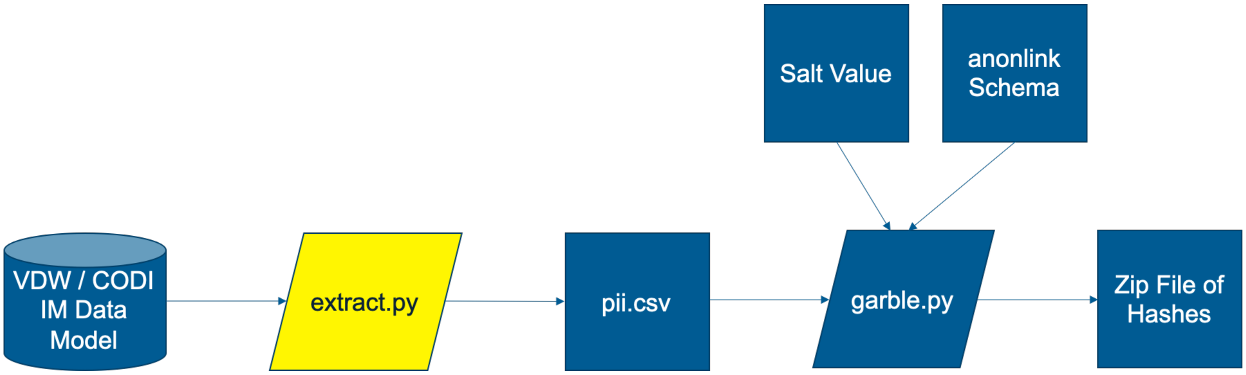

**Figure 4-2. Data Extraction, Validation and Cleaning**

## 4.2 Data Validation and Cleaning

Prior to de-identification, PII is cleaned to minimize differences in how data are handled between organizations. This cleaning is unlikely to have semantic significance.

**Table 4-1. Data Element Cleaning Process**

| Data Element(s) | Cleaning Process |
|---|---|
| Given Name<br>Family Name<br>Household Street Address<br>Parent Given Name<br>Parent Family Name<br>Parent Email | • Characters are converted to ASCII from UTF-8 using Normalization Form KD<br>• Leading and trailing whitespace characters are removed<br>• Characters are converted to uppercase |
| Household Phone | Extract only digit characters from original string |
| Household Zip | Leading and trailing whitespace characters are removed |
| Date of Birth | Converted to ISO 8601 date format |

Note that no cleaning is applied to the Date of Birth or Sex data elements because the CODI RLDM enforces constraints on these elements at the database level.

These cleaning procedures are performed by the “extract.py” script provided in *Data Owner Tools*, which is highlighted in Figure 4-2.

In addition to cleaning the data, *Data Owner Tools* provides a report on the extraction process. The report shows how many of the data elements were NULL, contained only whitespace, contained non-ASCII characters, or contained non-printable characters.

When extraction is complete, information is written to a file called “pii.csv,” which contains the cleaned PII in CSV format.

## 4.3 Obtaining and Maintaining Salt

Data owners will obtain the secret salt value, or encryption key, from the key escrow, which shall make the salt available via a secure transport mechanism. One potential approach is to make the salt available via Secure File Transfer Protocol (SFTP).  In this instance, the key escrow shall provide access credentials to each data owner.

Data owners shall provide to the key escrow a list of staff who have permission to access the secret salt value. Data owners are responsible for logging access to and usage of the secret salt value.

Data owners shall ensure that the secret salt value is encrypted when it is stored. This could be achieved by storing the salt on encrypted media or by using file- or folder-specific encryption.

Data owners should consult the National Institute of Standards and Technology (NIST) Special Publication 800-111[6] for guidance on selection and implementation of an encryption solution.

## 4.4 Generation of De-Identified Data for Matching

Data owners must de-identify PII before it can be transmitted to the linkage agent. To do this, data owners use *Data Owner Tools* to invoke the *anonlink clkutil* to build Bloom filters from PII extracted from the CODI RLDM.

Data owners need the secret salt value as well as *anonlink* schema files[7] to perform the de-identification. Schema files shall be obtained from the linkage agent.

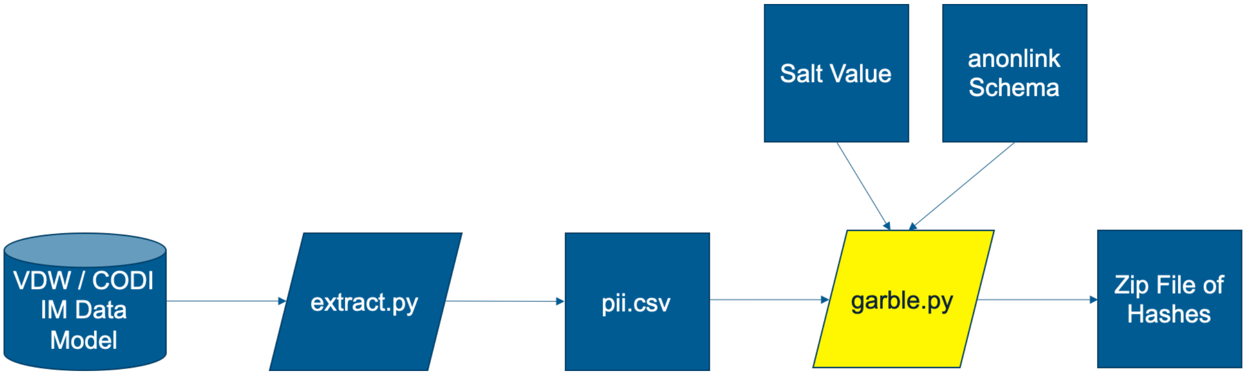

**Figure 4-3. Garbling PII**

Data owners run the "garble.py" script, highlighted in Figure 4-3, which requires the location of the schema, the location of the secret salt value, and the PII CSV file to be specified. Upon completion, the script creates a zip archive that contains the de-identified information to be transmitted to the linkage agent.

## 4.5 Destruction of Salt

When the data owner has generated the de-identified data, the data owner must destroy any copy of the secret salt value in their possession. Data owners may consult NIST Special Publication 800-88 Revision 1[8] for additional guidance on proper information disposal procedures.

Data owners shall provide an attestation of salt destruction to the key escrow.

## 4.6 Sending Information to the Linkage Agent

The linkage agent shall provide a secure transport mechanism to allow data owners to provide de-identified data; for example, the linkage agent could host an SFTP server where data owners can transmit their de-identified data. When operating an SFTP server, the linkage agent shall

[6] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-111.pdf

[7] https://clkhash.readthedocs.io/en/latest/schema.html

[8] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf

provide credentials to data owners for access. Data owners shall transmit the zip archive containing the de-identified information to the linkage agent.

## 4.7 Receiving and Incorporating LINKIDs

When the matching process is complete, the linkage agent notifies data owners that results are available. Data owners will be provided access to the match results via a secure transport mechanism provided by the linkage agent.

As described in section 3.3, the match results will include a LINKID mapped to a position of an individual in the generated PII CSV file. Data owners use the “linkidtopatid.py” script to generate a new CSV file that contains a mapping of LINKIDs to PATIDs. It is the responsibility of data owners to use this information to update the LINK table in the CODI RLDM.

Once data owners have updated the LINK table, they shall delete all records from the IDENTIFIER table. This process removes PII from systems that is no longer needed for the PPRL process.

# 5. Guidance for Data Partners Hosting Other Owners' Data

Data partners are organizations who host information on behalf of a data owner. As an example, a community organization might not have the capabilities needed to participate in a distributed health data network. In that situation, the data owner may transmit information to a data partner who is capable of responding to distributed queries.

Each relationship between a data owner and data partner will be unique. These arrangements will be determined by the types and qualities of data owner information, data owner priorities, and data owner technical capabilities.

When establishing a data partner-data owner relationship, the following factors should be considered:

- Identify the type of relationship (section 5.1)
- Determine how information will be shared (section 5.2)
- Determine how and what local identifiers will be shared (section 5.3)

## 5.1 Types of Data Partner and Data Owner Relationships

Relationships between data partners and data owners can be divided broadly into two separate categories: data partners who respond only to queries, and data partners who perform de-identification and respond to queries. See section 5.2 on considerations for information sharing between these partners.

- **Scenario 1: Query-only relationships.** For data partners who respond only to queries, the data owner is responsible for participating in the PPRL process. This includes extracting PII, obtaining and managing salt, performing de-identification, and receiving LINKIDs. This type of relationship allows the data owner to participate in CODI without disclosing client identifiers to an external data partner. The data partner will be able to respond to research queries using information stored in the CODI RDM combined with information from the LINK table to enable assembly of longitudinal records.

  In this arrangement, data owners will transmit all relevant CODI RDM information to data partners. However, the only information from the CODI RLDM that data owners will transmit is information related to LINK table. Specifically, information that would be placed in the CODI RLDM IDENTIFIER table will not be shared by the data owner.

- **Scenario 2: De-identification and query relationships**. In this arrangement, the data partner provides de-identification in addition to the query response. In this scenario, the data owner transmits information that would be placed in both the CODI RDM and CODI RLDM to the data partner. The data partner participates in the PPRL process on behalf of the data owner.

## 5.2 Receiving Information from a Data Owner

Regardless of the type of relationship between the data partner and data owner, the arrangement requires the transmission of sensitive information from one organization to another. Detailed guidance on exchange of information will be specific to a particular data partner-data owner relationship and is outside the scope of this document. Broadly, transmission of information must take place using secure communication protocols or be exchanged via encrypted media.

## 5.3 Management of Local Identifiers

Data owners working with data partners will have their own data models that they use to conduct their business. Specifics on translating information so that it can be represented in the CODI RDM or CODI RLDM are outside the scope of this document. However, there are aspects of the PPRL process that must be managed by data partners to ensure that data owner information can properly be integrated into a longitudinal record.

When translating data owner information into the CODI RDM, each individual will be assigned a PATID. Data partners and data owners must be able to establish a process to facilitate the assignment of a PATID. This may involve using an existing data owner identifier or developing an algorithm to create an identifier.

In the case of scenario 1 where data owners are performing their own de-identification, additional coordination may be required. Data partners and data owners should be clear on how identifiers are used in information intended for the CODI RDM and the CODI RLDM, and whether the data partner will be required to perform any identifier generation or translation.

# 6. Linkage Agent/Data Coordinating Center Guidance

The linkage agent is the organization responsible for performing matching using the de-identified information provided by data owners and data partners. The linkage agent creates LINKIDs and transmits them to data partners and data owners.

## 6.1 Receiving Information from Data Owners and Data Partners

The linkage agent shall provide a secure transport method that data owners and data partners will use to send their de-identified information as described in section 4.6. One possible approach is the use of an SFTP server. When using an SFTP server, the linkage agent shall create different authentication credentials for each data owner and data partner.

## 6.2 Matching

The matching process involves taking the de-identified information provided by data owners and data partners and comparing the Bloom filters to find identity linkages. The comparison process is performed by *anonlink*. Interactions with *anonlink* are managed by the *Linkage Agent Tools* software package.

*Linkage Agent Tools* operates using a configuration file. This file provides information about the data partners and data owners participating in PPRL, the *anonlink* schema being used, matching thresholds, and file locations. Consult the Configuration Section[9] of the *Linkage Agent Tools* "README.md" file for specific details on *Linkage Agent Tools* configuration.

The linkage agent first runs the "validate.py" script in *Linkage Agent Tools*. This script ensures that all required de-identified information is present and in the correct location. Next, the linkage agent runs "match.py." This script interacts with the *anonlink-entity-service* to conduct the matching process.

The CODI PPRL approach is to conduct multiple rounds of matching, each with different data elements. *Linkage Agent Tools* creates a new *anonlink* project[10] for each round of matching. Once the project has been created, *Linkage Agent Tools* sends the de-identified information to *anonlink* through the upload service.[11] Finally, *Linkage Agent Tools* creates an *anonlink* run,[12] which performs matching and provides a method for retrieving results.

*Linkage Agent Tools* stores the results from each separate *anonlink* project in a TinyDB[13] database. When all *anonlink* projects have completed, this database is accessed to generate LINKIDs.

[9] https://github.com/mitre/linkage-agent-tools/#configuration

[10] https://*anonlink*-entity-service.readthedocs.io/en/stable/api.html#operation/entityservice.views.project.projects_post

[11] https://*anonlink*-entity-service.readthedocs.io/en/stable/api.html#operation/entityservice.views.project.project_clks_post

[12] https://*anonlink*-entity-service.readthedocs.io/en/stable/api.html#operation/entityservice.views.run.list.post

[13] https://tinydb.readthedocs.io/en/latest/index.html

### 6.2.1 Development of *anonlink* Schema

The PPRL matching process is dependent on the development of sets of *anonlink* schema. These schemata determine which data elements will be used for matching in a particular project and what weights should be applied to the data elements. Linkage agents should consider the development of a synthetic population that can be used to test and tune *anonlink*. For the CODI Denver Pilot, a synthetic data set was created with associated synthetic longitudinal records to test and tune matching performance.

## 6.3 Generation of LINKIDs

*Linkage Agent Tools* provides a script to generate LINKIDs. The linkage agent executes the "linkids.py" script to generate a CSV file containing a full mapping of LINKIDs to data partners and data owners participating in the PPRL process.

*Linkage Agent Tools* follows these steps when assigning LINKIDs:

1. **Assign LINKIDs to non-conflicting individuals.** Using links identified across all *anonlink* projects, find the sets produced that contain a single record at a data owner or data provider. Assign each of these sets a LINKID.

2. **Handle linkage sets with conflicts.** A linkage set is considered to have a conflict if it identifies multiple records at the same data partner or data owner. For example, one *anonlink* project asserts a link between data owner A record 5 and data owner B record 7, and a second *anonlink* project asserts a link between data owner A record 5 and data owner B record 8. This is a conflict because each record at data owner B represents a unique individual. *Linkage Agent Tools* resolves the conflict by selecting the linkage identified by the plurality of *anonlink* projects. In the event of a tie, it will make a random selection. *Linkage Agent Tools* then generates a LINKID for the deconflicted set.

3. **Assign LINKIDs to unmatched records**. Identify all records that have not been included in a matching set. These represent individuals that have a record at a single data owner or provider. *Linkage Agent Tools* assigns a LINKID to each unmatched record.

As mentioned in section 3.3, LINKIDs are generated as UUIDs compliant with RFC 4122.

## 6.4 Making LINKIDs Available to Data Owners and Data Partners

Once LINKIDs are generated for all records for all data owners and data partners, the linkage agent must make these available to the appropriate parties.

The linkage agent executes the "dataownerids.py" script in *Linkage Agent Tools*. This script reads in the CSV file containing the mapping of LINKIDs to all records. It then creates a separate CSV file for each data owner or data partner. These files will be hosted by the linkage agent's SFTP server where they can be accessed by data owners and data partners.

## 6.5 Destruction of Matching Information

When all data partners and data owners have confirmed that they have obtained and integrated their linkage results, the linkage agent shall destroy the input and output information of the matching process. This includes:

- De-identified information provided by data owners and data partners
- “results.json” file created by TinyDB to store results of *anonlink* projects
- CSV file generated by the “linkids.py” script
- CSV files generated by the “dataownerids.py” script

The linkage agent may refer to NIST Special Publication 800-88 Revision 1 for additional guidance on proper information disposal procedures.

# 7. Key Escrow Guidance

The key escrow is the organization responsible for creating the secret salt value or encryption key that data owners and data providers will use in the de-identification process. Given that the de-identification process relies on the secret salt value remaining secure, it is critical that it be created and distributed appropriately.

One important aspect of this PPRL approach is that the linkage agent shall not have access to the salt value. The linkage agent can perform all of its matching work without access to that value.

The activities of the key escrow include:

- Generate secret salt value (section 7.1)
- Securely provide salt values to data owners and providers (section 7.2)
- Destroy salt value when it has been retrieved from all data owners and data partners (section 7.3)

## 7.1 Salt Generation

The key escrow is responsible for creating the salt value. The purpose of the salt is to introduce a random value into the de-identification process. As such, the key escrow must use a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) as the source for the salt value. The Open Web Application Security Project provides a Cryptographic Storage Cheat Sheet[14] with references to appropriate CSPRNGs.

The salt value shall comprise uppercase, lowercase, and digit characters, be 128 characters long, and be stored in an ASCII encoded text file. A script for generating appropriate values for the salt can be found on GitHub[15].

## 7.2 Providing Salt Values to Data Owners and Data Providers

The key escrow shall distribute the salt value to data owners and data providers via a secure transport method, such as SFTP. When using SFTP, the key escrow is responsible for distributing access credentials to data owners and data providers. If the transport mechanism allows, the key escrow shall log access to the salt value. The log should record which data owner or data partner accessed the salt value, and the date and time it was accessed.

## 7.3 Destruction of Salt

When all data partners and data owners have retrieved the salt value, the key escrow shall destroy the salt value. The key escrow may refer to NIST Special Publication 800-88 Revision 1 for additional guidance on proper information disposal procedures.

[14] https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cryptographic_Storage_Cheat_Sheet.md#secure-random-number-generation

[15] https://gist.github.com/eedrummer/bcfdc12db0e47a08c8033310a78c09d6

# 8. Deployment Concerns

Successful deployment of the CODI PPRL process involves coordination of effort across multiple organizations. This section describes performance evaluation and documentation approaches that can ensure successful execution of the PPRL process.

## 8.1 Performance Evaluation

The goal of this PPRL process is to identify instances where an individual has information stored at different organizations and establish a linkage that can be used to create a longitudinal record. Because this process uses a probabilistic matching process, there will be cases where records are linked incorrectly or where linkages are missed.

Performing an evaluation using traditional identity metrics is not possible because it would require knowledge of the actual record linkages which, if they existed, would eliminate the need for the PPRL process. However, there are some broad approaches that may be employed to estimate performance.

- **Gain insight into the false positive rate**. Researchers executing queries can monitor for discrepancies in individual sex and birth date. If there is disagreement between these values, it is not necessarily indicative of a false positive, but instead may be due to input or information processing errors. However, a high rate of disagreement in these values suggests that the linkage process is creating a high rate of false positives.

- **Manually validate linkages**. Depending on organizations' ability to share PII with one another, it may be possible to manually check linkages assigned by the CODI PPRL process for correctness.

## 8.2 Documentation of Implementation Details

In the implementation of the CODI PPRL process, organizations will need to share more concrete details of processes and systems configuration. Additionally, local conditions may necessitate that the PPRL implementation deviate from guidance offered in this document. Participants in a particular CODI PPRL instantiation should create artifacts to document these details and differences. These artifacts should be stored in a central location agreed upon by the participating organizations.

# Appendix A. Denver Pilot Specific Guidance

The CODI Denver Pilot has made the following implementation decisions in the instantiation of the PPRL process:

- The key escrow shall distribute salt values via secure email.
- The linkage agent shall operate a secure file transfer service, based on Egnyte[16], to receive de-identified information from data owners and data partners as well as to distribute LINKIDs.
- The PPRL process shall be conducted annually.

[16] https://www.egnyte.com/

# Acronyms

| | |
|---|---|
| **ASCII** | American Standard Code for Information Interchange |
| **BMI** | Body Mass Index |
| **CDC** | Centers for Disease Control and Prevention |
| **CDM** | Common Data Model |
| **CHORDS** | Colorado Health Observation Regional Data Service |
| **CODI** | Childhood Obesity Data Initiative |
| **CSPRNG** | Cryptographically Secure Pseudo-Random Number Generator |
| **CSV** | Comma Separated Values |
| **DCC** | Data Coordinating Center |
| **DHDN** | Distributed Health Data Network |
| **EHR** | Electronic Health Record |
| **ETL** | Extract–Transform–Load |
| **FFRDC** | Federally Funded Research and Development Center |
| **FN** | False Negative |
| **FP** | False Positive |
| **ISO** | International Organization for Standardization |
| **IT** | Information Technology |
| **JSON** | JavaScript Object Notation |
| **KD** | Compatibility Decomposition |
| **NIST** | National Institute of Standards and Technology |
| **OMOP** | Observational Medical Outcomes Partnership |
| **PATID** | Patient Identifier |
| **PCORnet** | Patient Centered Outcomes Research Network |
| **PII** | Personally Identifiable Information |
| **PPRL** | Privacy Preserving Record Linkage |
| **RDM** | Research Data Model |
| **RLDM** | Record Linkage Data Model |
| **SDC** | Sørensen–Dice coefficient |
| **SFTP** | Secure File Transfer Protocol |
| **TLA** | Tools Landscape Analysis |

| | |
|---|---|
| **TP** | True Positive |
| **UTF** | Unicode Transformation Format |
| **UUID** | Universally Unique Identifier |
| **VDW** | Virtual Data Warehouse |

# 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.

**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.

**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.

**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.

**Modulo** A mathematical operation that provides the remainder after division of one number by another.

**Positive Predictive Value** See Precision

**Precision** A ratio that provides the fraction of the identified matches that are correct.

**Recall** A ratio that provides the fraction of the correct possible answers that the system found.

**Salt** Random data applied to a hashing function. Salt prevents attackers from reversing a hashing process by guessing the input values.

**Sensitivity** See Recall

**Sørensen–Dice coefficient** A statistic that can be used to measure the performance of a matching algorithm. It is a combination of Precision and Recall. It is also called F1 Score.

**Specificity** A ratio that provides the fraction of the non-matches that were correctly identified as non-matches. Specificity was not used in CODI PPRL analysis.

# NOTICE

For further information, please contact The MITRE Corporation, Contracts Management Office, 7515 Colshire Drive, McLean, VA 22102-7539, (703) 983-6000.