This work was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305A220451 to The Regents of the University of California — Berkeley. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

Hashed Linkages for Administrative Datasets

A Technical How-to Guide

SAMANTHA FU, CHARLES DAVIS, JESSE ROTHSTEIN, APARNA RAMESH, AND EVAN WHITE

CALIFORNIA POLICY LAB

MARCH 2022

Introduction

Linking data together can be a powerful way for governments and researchers alike to tackle vexing public policy research problems. However, for researchers, finding ways to link data directly between two departments can often be more challenging than even obtaining the data in the first place. Even when a researcher develops the necessary relationships and trust with multiple government agencies, traditional linking requires each agency to share identified data, sometimes resulting in privacy and security concerns, and also requires the agencies to work together in ways that are not always easy to accomplish. We discuss these issues at greater length in another report that covers the logistical, statutory, and technological considerations for implementing privacy-preserving linkages. This how-to guide focuses on the linkage process itself, and aims to serve as a technical handbook for parties interested in linking datasets that have been de-identified using cryptographic hashing methods.

We begin with a brief discussion of privacy-preserving linkage, before moving on to each of the essential steps in the hashed linkage process. We expect that this guide will be most useful to the data analysts or researchers performing the linkage (hereafter referred to as the "linkage team"), whether they are affiliated with the agencies whose data are being hashed (hereafter referred to as the "data contributors") or a third-party organization. Some sections may also be of use to data contributors, regardless of whether they are performing the linkage or not. The table below outlines the sections most applicable to the different parties involved.

STEP	RESPONSIBLE PARTY
1. Compiling data requests	Linkage team
2. Picking a "salt"	Data contributors
3. Hashing PII	Linkage team *and* data contributors
4. Transferring data	Data contributors
5. Linking datasets	Linkage team
6. Evaluating and refining linkage quality	Linkage team
7. Using linked administrative data	Linkage team

What is privacy-preserving linkage?

A privacy-preserving linkage method is one that does not require the transfer of personally identifiable information (PII) such as names and Social Security numbers. Privacy-preserving linkage methods generally rely on masked or otherwise obfuscated PII, and thus do not require the exchange of sensitive information between data contributors. There are a number of possible ways to achieve this, but this guide focuses on the use of cryptographic hashing, arguably one of the more straightforward methods.[1]

Using this method, data contributors encrypt PII fields using a hash function, which converts a piece of data of any length into a hexadecimal string of a fixed length (e.g., 64 characters). This transformation cannot be reversed — there is no decryption key that allows outputs to be converted back to inputs. It would be possible to determine which outputs correspond to which inputs by repeating the hashing process if one knew the hashing algorithm used and had (or could guess) original inputs. For this reason, data security is strengthened by adding a string of characters known as a "salt" to each PII field prior to hashing. This "salt" is agreed on between the data contributors, and does not need to be provided to the party conducting the linkage, thereby ensuring that third parties will never be able to determine which rows of data correspond to which individuals.

To link hashed PII across datasets, the inputs to the hashing function must be exactly the same and data contributors must perform the hashing in the exact same manner. While the latter can be ensured by providing hashing code to each contributor, the former cannot be guaranteed, due to differences in data collection, errors in entry, and so forth. We discuss strategies to minimize differences across datasets below. Despite best efforts, however, some inputs may still differ, and the resulting hashed outputs will also differ. For instance, the hashed results of "Jonathon" and "Jonathan" will bear no resemblance to one another. This prevents researchers from using string-comparison measures such as those introduced by Jaro (1989) and Winkler (1990), which are common when performing linkages that use unhashed PII. Another limitation of privacy-preserving linkage methods is that researchers without access to underlying PII are not able to fully validate the accuracy of matches. This is true of many other linkage methods without a source of "ground truth." This guide will discuss options for partially mitigating both these issues.

1 For a review of privacy-preserving linkage more generally, users may refer to Vatsalan et al. (2017).

1. Compiling data requests (Linkage team)

Before embarking on a hashed linkage, it is important to verify that there is sufficient overlapping PII between the two datasets that need to be linked. If the only overlapping PII are first and last names, for instance, it may be difficult to perform linkages with any level of confidence, as many people share the same names. PII variables such as Social Security numbers, which are meant to uniquely identify individuals across the US population, are ideal for linkage purposes. If these are lacking, linkages may still be performed using some combination of uniquely-identifying variables, such as first and last names, birthdates, and addresses. Demographic characteristics like race or gender may also be useful for linking in some circumstances.

Once the linkage team has verified that there is sufficient overlapping PII[2] between the datasets that need to be linked, it is best to put together a comprehensive data request specifying what exact data is needed from each data contributor. This is helpful to both parties, as the data contributor will know exactly what data to extract, and the linkage team will know exactly what data to expect.

Some agencies may have formal data-request procedures for exactly this purpose. Some agencies may also have publicly available data documentation from which to craft a request, though many do not. In the latter case, it is usually worthwhile to ask the agencies from which you are requesting data for a list of the tables and variables available in their systems. This list should include variable types (i.e., numeric, character, or boolean) if available.

Generally speaking, data requests should include at least the following:

- Time period (e.g., 2010 to 2020)
- Unit of time (e.g., monthly, quarterly, annual)
- Data tables (e.g., enrollment in a program such as Supplemental Nutrition Assistance Program, or SNAP)
- Variables needed in each table (e.g., SSN, date of enrollment)
    - The initial request should include all PII elements that will be used for linkage. These elements will later be hashed and the raw versions will be dropped from the final dataset(s) to be shared
- Data universe (e.g., all individuals enrolled in SNAP at any point between 2010 and 2020)

2 The PII must also be of high quality and reliability, otherwise "garbage in, garbage out."

2. Picking a salt (Data contributors)

Once data requests have been agreed upon by all parties, the data contributors should also decide on a "salt." This salt should **not** be shared with any third parties, including the linkage team if applicable. A salt is a case-sensitive string, similar to a password, that is used as an additional input to the hashing function. While most hashing functions are designed to be irreversible, such that the original input cannot ever be derived from the hashed output, using a salt adds an additional layer of security against re-engineering outputs, because the hashed outputs of "John" and "John#this is a salt#" will be completely different.

To illustrate the need for a salt: if data is hashed without adding a salt, it would be possible for a hacker to feed a list of common names into a hashing algorithm, and re-engineer the inputs by comparing the hashed outputs. Thus, the attacker could feasibly identify all instances of "John" in the hashed data by rehashing the input "John". However, if a salt is added to the data before hashing, the attacker would no longer be able to do so, as trying to guess the salt would be computationally impossible if the salt was strong enough. Thus, so long as the salt is never shared with third parties, no one would be able to re-engineer the raw PII inputs from the hashed outputs.

A salt can theoretically include any combination of letters, numbers, and symbols. We recommend avoiding symbols or special characters, because they may be treated differently by the two computer systems conducting the hashing. The longer a salt is, the harder it is to crack, and thus a general rule would be to choose a salt with 20 or more characters. Password managers are usually able to generate the random combinations of letters and numbers required.

Once the data contributors agree upon a salt, each should record the exact combination of characters in a secure location. If those two data contributors wish to conduct another hashed linkage in the future, that salt will be needed again.

3. Hashing PII (Linkage team and data contributors)

The next step in the process is to encrypt the raw PII using a hash function. This involves two distinct stages. First, writing the code that will perform the hash, which should generally be done by the party who will later perform the linkage, with input from the data contributors. And second, running the hashing code, which should be done by the agency whose data is being hashed, so that identifiable data does not have to be shared with other parties.

Writing the hashing code (Linkage team, with help from data contributors)

The hashing process itself is simple, often just one line of code. The more difficult part of the hashing code is cleaning and normalizing the data from each data contributor, so that the input strings are exactly the same.

The PII fields that will be included in each agency's data should have been determined when compiling the data requests (step 1). One initial determination is which of those fields needs to be hashed. Sometimes fields like ZIP codes, year or month of birth, race, or gender may not require hashing and can be linked unhashed, which is easier.

The most time-consuming portion of the code is cleaning and standardizing the input strings to be hashed from each dataset. A different version of the code will be required for each dataset. Fields will usually have different names and data is often stored differently on each side. For example names may be in as little as one field or as many as six or more (e.g., prefix, first, middle, last, suffix, generation). Data may also be stored in different formats, for example with dates stored as strings, integers, or date objects.

Observations that differ by even one character (e.g., Sarah and Sara) produce completely different hashes, and thus the code must first standardize all PII fields before hashing. The linkage team should consider a few common standardization strategies:

- Converting all strings to the same case (lower or upper)
- Removing all spaces and punctuation
- Standardizing the use of special characters (e.g., by replacing "ñ" with "n")
- Removing all numbers, if appropriate (e.g., for variables containing names)
- Removing common prefixes and suffixes (e.g., "Mr", "Dr", "Jr")[3]
- Ensuring consistency in abbreviations (e.g., "Ave" versus "Avenue" in address fields)

Some PII fields pose more challenges than others when it comes to standardization. SSNs, for instance, are designed to be nine digits long, and do not include letters. Thus, it is easy to identify observations that do not conform to this format and either exclude them or attempt to standardize them.

3 Removing "Jr" may lead to a false match between a parent and child, but keeping it in could lead to a missed match between two of the son's records, one of which is missing the "Jr". Choosing whether false negatives or false positives is more detrimental is one of the decisions that is specific to each project and dataset.

Addresses, on the other hand, may not conform to any standard formatting. To illustrate: the same address could plausibly be written as “1671 Seventh Street, Apt 1A”, “1671 7th St, Unit 1A”, or “1671 Seventh St, #1A”, and may be stored in one field or several fields (e.g., number, street, unit, predirection, postdirection). The number of possible abbreviations or other differences that might have to be accounted for can be very large, but it may nonetheless be worth attempting, depending on the quality of the data and the availability of PII fields to link on. There are also existing programming tools that can help standardize addresses.

Once the inputs are standardized on each side, we recommend dividing each PII field up into smaller substrings before hashing. This allows for “fuzzy” linkages using inexact matches (e.g., a transposition of a number, or matching “Jon” with “John”). The linkage team’s chosen linkage strategy (see Step 5) should dictate how many substrings to create. More substrings allows for more fuzziness in the matching, but can create larger hash files and longer runtimes for the linkage. A few examples of substrings we use: the first two letters of first and last names, phonetic versions of first and last names (using soundex), and various substrings of the SSN, birthdate, or address.

Once the data has been cleaned, standardized, and subsetted, all that remains is to pick a hash function to use. There are several possible functions, but for purposes of linkage, it is important to pick one that does not allow for the possibility of “collision,” meaning that the function should not allow for two different inputs to produce the same output. We use the SHA256 function, the cryptographic industry standard.

Sample hashing code is available on our github: https://github.com/californiapolicylab/hashed-linkage.

Running the hashing code (Data contributors)

Because the success of the linkage depends on being able to match hashed fields exactly, it is important to test the hashing code to ensure that it is working correctly. One way to do this is to send each agency a dataset containing synthetic data for each PII field that will be hashed. The agency should run the hashing code on this test dataset, appending the salt that was agreed upon previously to every observation, and send back the hashed output. The linkage team should then compare the hashed output for all agencies to ensure that they match exactly, which serves as a check on both the hashing code and the input of the salt.

The analysts writing the hashing code are often doing so "blind" — without access to the underlying data or operating environment. (In most settings it is important that the analysts not have access to the salt, as this could enable them to re-identify the data.) As a result, developing the hashing code is usually an iterative process that may involve several rounds of testing and modification. Some issues that have arisen in past linkages include:

- **Reading in data incorrectly:** Variables that an analyst expects to be numeric might actually be strings, and vice versa, or string variables might be truncated accidentally.[4] It's important for the data contributors running the hashing code to note any errors at this stage and send back log files to the linkage team.
- **Parsing the salt:** In one case, an agency's test output was not matching the second agency's, and this was due to how a special character in the salt was being treated in the different agency's operating environments.
- **Memory or storage issues:** Hashing can result in very large datasets as each hashed observation, even a short field like a first initial, becomes a 64-character string. Some data contributors have run out of memory or storage on their local desktops when attempting to hash large datasets. The files can easily reach tens or hundreds of gigabytes.

Once the hashing code has been written and tested, each data contributor should run the code on all the datasets requested that contain PII fields.

4. Transferring data (Data contributors)

After the PII has been hashed, it should be transferred to the linkage team alongside the clear-text (i.e., unhashed) analytical data. This transfer process can take place in any number of secure manners, depending upon the data security protocols of the relevant agencies. Secure file transfer protocol (SFTP) is a common method and secure online dropboxes are another popular option. Others prefer to transfer the data via encrypted hard drives so as to prevent transmission over the internet.

Whatever the method, once the data has been transferred to the linkage team, it should undergo initial hygiene checks. In particular, analysts should confirm that no PII is present in the data and that the number of rows and columns corresponds to what was expected. Best practice would then be to follow the "separation principle" often used with identified data, and to separate the hashes from the analytical data, retaining a crosswalk between each contributing dataset that can be used to link analytical data together once the linkage is complete.

4 For example, the default version of SAS's proc import function automatically truncates strings to the maximum length observed in the first 20 rows.

Depending on the volume of hashes, it may also make sense to replace the 64-character hashes with integers that take up less disk space, being careful of course to ensure that like hashes are converted to like integers.

5. Linking datasets (Linkage team)

There are many possible strategies for linking data once it has been obtained. The choice of strategy often depends heavily on the fields available for linkage, the quality of the data in those fields, the availability of technical expertise, the availability of computational resources, and other factors.

In a white paper on data linkage previously published by CPL, Augustine et al. (2018) describe three different types of linkage methods: rules-based linking, supervised machine learning, and unsupervised machine learning. Rules-based methods, which use a predetermined set of rules to decide whether any two records are a match, can often be the simplest to implement. However, deciding on the appropriate rules requires an in-depth knowledge of the data, and defining and checking the rules can also take some time. The resulting rules are also unlikely to generalize to other datasets and linkage problems.

Supervised machine-learning methods can be faster to implement, and may involve less human input than the rules-based methods above. While human input is required to initially train the algorithm (i.e., classify some number of pairs as matches or not), once the algorithm has been trained, it then infers its own set of rules that can best reproduce decisions made by the human trainer, and applies these rules to classify the remaining pairs in the dataset. Unsupervised machine-learning methods, on the other hand, require no training, and are able to infer matching rules from the data alone. However, while both machine-learning approaches generally require less human input, the rules developed by the algorithms may be complex enough that they are hard to understand, let alone explain. Additionally, human time and technical expertise is still required to prepare the data and set up the algorithm for use. The remainder of this section focuses on rules-based linkage, and outlines the rules used in one example linkage which matched hashed data from a state tax agency to a state social services agency.

Regardless of choice of method, the fundamental questions involved in linking datasets are (1) how to identify the same person both within and across datasets, and (2) what level of confidence is required. Before thinking about identifying the same person across datasets, the linkage team must first decide on how to identify the same person *within* datasets. Most datasets are likely to contain multiple observations for the same person, whether because they are longitudinal and cover a span of years, because a person can receive multiple services from the same agency, or for any number of other reasons. This can introduce additional complexity, as some PII fields can reasonably change over time (names, for instance). PII can also be recorded incorrectly in some observations, and correctly in others. Linkage strategies should ideally account for these errors and potential differences.

Datasets often have unique identifiers that are meant to identify the same person across records. However, depending on how these identifiers are assigned, they may not always accurately capture this information. For instance, if an agency assigns unique identifiers based on a person's SSN, and that SSN is recorded incorrectly in one year, then that person will have two different unique identifiers in the agency's data. Thus, the first decision that researchers need to make is whether or not to trust an agency's identifier, or to create their own identifier using some combination of the available PII (e.g., SSN, SSN and last name, SSN and date of birth, etc.).

If a decision is made to trust the agency's unique identifier, and that identifier links to multiple different sets of PII within a dataset, then an additional decision needs to be made on whether all available sets of PII should be used for matching to the second agency, and, if not, which set should be used (whether the most recent, the most complete, or otherwise).

Once researchers are reasonably confident that they have been able to identify records belonging to the same people in each separate dataset, they can move on to trying to match these people across datasets. In the rules-based example we discuss below, this primarily involves comparing different sets of available PII. The PII used for hashing in this case were SSN, first name (FN), last name (LN), and date of birth (DOB). A number of substrings based on these fields were also created, including first two letters and first four letters of first name and last name, birth day, birth month, birth year, and more. Table 1 below contains the full list of all hashed variables created for this particular linkage.

TABLE 1. List of hashed variables.

VARIABLE
First Name
First Name (first two letters only)
First Name (first four letters only)
First Name (soundex [phonetic spelling])
Last Name
Last Name (first two letters only)
Last Name (soundex [phonetic spelling])
SSN (excluding digits 8 and 9)
SSN (excluding digits 7 and 8)
SSN (excluding digits 6 and 7)
SSN (excluding digits 5 and 6)
SSN (excluding digits 4 and 5)
SSN (excluding digits 3 and 4)
SSN (excluding digits 2 and 3)
SSN (excluding digits 1 and 2)
Birth day
Birth month
Birth year

In cases where higher levels of confidence are required (for instance, when the linked data may be used for targeted outreach or other activities related to government administration, as opposed to aggregated estimates for research purposes only), researchers might decide to rely only on exact matches across all PII fields, or require that SSNs, if available, always match perfectly. In cases where lower levels of confidence might suffice, however, researchers may decide to be more inclusive, and allow for partial matches on all or any of the available PII fields (see, for instance, rounds 5–7 below). This is very much dependent on both the purpose of the linkage and the quality of the data being linked.

In the following example, eight rounds of rules-based linkage were conducted, with each subsequent round using less restrictive rules than the prior round. Once a set of PII was matched in one round, it was removed from the universe of potential matches for the next round. This was done so that only sets that could not be matched in the earlier, higher-quality rounds were considered for matching in the later, lower-quality rounds.

Using successive rounds was also a form of "blocking" (see Augustine et al. (2018) for a more comprehensive discussion), which reduced the number of PII pairs that needed to be compared in each round, thus making the problem more computationally manageable. The criteria used for matching in each round are outlined below, but are meant to be purely illustrative. We do not recommend that the same rounds are used across all linkages. Rather, rounds need to be tailored to each specific linkage and the idiosyncrasies in the datasets being linked. In another hashed linkage conducted by CPL involving higher-education, safety-net, and financial-aid data, many of the rounds below were discarded as they produced poor quality matches.

- Round 1: Perfect match on SSN, FN, LN, and DOB
- Round 2: Perfect match on SSN, and perfect match on two out of FN, LN, DOB
- Round 3: Perfect match on FN, LN, DOB, and partial match on SSN
- Round 4: Perfect match on SSN, and perfect match on one out of FN, LN, or DOB
- Round 5: Perfect match on FN and LN, and partial match on SSN and DOB
- Round 6: Perfect match on LN and DOB, and partial match on SSN and FN
- Round 7: Perfect match on FN and DOB, and partial match on SSN and LN
- Round 8: Perfect match on FN, LN, and DOB

Returning to the example above, within each round, scores were assigned for each set of observations that match using that round's criteria, and the pair with the highest score was kept. To illustrate using round 3, which requires a perfect match on first name, last name, and birthdate, and a partial match on SSN: there are two observations in Dataset B that match perfectly with the observation in Dataset A on first name, last name, and birthdate. One observation has an SSN that differs by just 1 digit, while the other has an SSN that differs by 2 digits. The first observation is assigned a higher score than the second, as we assume that an SSN with 1 digit differing is more likely to be a match than an SSN with 2 digits differing. In the end, the linkage team may decide that all three observations correspond to the same individual.

DATASET A

First Name	Last Name	Birthdate	SSN
John	Smith	12/2/1978	123456789

DATASET B

First Name	Last Name	Birthdate	SSN
John	Smith	12/2/1978	123455789
John	Smith	12/2/1978	123455589

Likewise with rounds, the value of the scores assigned should also be tailored to the datasets in question, and specifically the likelihood of certain types of errors in each PII field. In the above linkage, for instance, higher scores were given to last name matches than first name matches, on the assumption that last names were more likely to be unique.

6. Evaluating and refining linkage quality (Linkage team)

As noted above, one key limitation of hashed linkages is that researchers are not able to validate the accuracy of matches. This is true of many real-world linkage problems, hashed or otherwise, but there are still a number of options for researchers to attempt to evaluate and refine the quality of the linkage. One way might be to look at the distribution of matches that are found in each round. If, for instance, a given set of PII in the first agency's data matches to one person in the second agency's data in an earlier round (using perfect matches), and a second person in a later round (using partial matches), it might be an indication that the later round is picking up spurious matches, and should be discarded or refined.[5]

Another way of evaluating linkage quality could be to use non-PII variables that represent the same (or similar) information across both datasets. This might involve looking at an early round requiring perfect matches on all PII variables, which we can be fairly confident about, and calculating the match rate of the validation variable in that round. In the example linkage above using tax and safety-net data, wage information was available in both datasets. Though not from the exact same source, the data were similar enough to facilitate a comparison. We found that 46% of matches in the first round had the exact same annual wages (and 71% had annual wages within a $1,000 margin of error).

5 This requires using all available sets of PII in each round and not removing matched sets in subsequent rounds, as described in the example in section E.

This match rate gave us a base of comparison to diagnose the approximate false positive rate in later rounds. For instance, we found that rounds 2 and 4 above had very similar exact match rates (46% and 44% respectively), but that the remaining rounds all had exact match rates of less than 10%. If we assume that all the first-round matches are true positives, this allows us to calculate the approximate false positive rate of later rounds as the difference in the match rates between the later round and round 1, divided by the match rate in round 1. In round 3, for instance, we saw only 9% of matches with the same annual wages. This implies a high false positive rate of ~80% [(0.46-0.09)/0.46], suggesting that round 3 is producing matches of fairly low quality and might need to be discarded.

Both of these strategies are iterative, in the sense that preliminary linkage results are used to refine the choice of rules and rounds, and both also focus on the reduction of false positives.

Yet another strategy that we took for diagnosing false positives in the social-services and tax data linkage was to examine family relationships. Although SNAP (food stamps) cases are conceptually distinct from tax units — a SNAP case consists of people who prepare food together, who may or may not be married or in tax dependency relationships with each other — we expect that most of the time when two people show up on the same SNAP case they should also appear on the same tax return. We found that when we were able to match one person from a multi-person SNAP case to tax data, someone else from the case matched to a different tax return only about one-fifth of the time. The true rate of multiple returns is not zero, as some SNAP cases consist of multi-generational or blended households, so we interpreted this as an indication that the overall false positive rate was fairly low. In a somewhat larger share of cases, (about 30%), one or more members of the SNAP case did not match to any tax return, perhaps indicating that false negatives are more prevalent here.

Unfortunately, strategies to minimize false negatives in hashed linkages are not obvious. False negatives can be minimized by setting looser matching rules, but that may increase false positives. One option is to use very coarse blocking (such as a partial match on any PII field), identify all the pairs within that blocking that match on some other non-PII validation variable, and then use either of the machine-learning methods described above to identify which of the pairs are in fact the same person.

7. Using linked administrative data (Linkage team)

Once data from two or more agencies have been successfully linked, analysts and researchers can begin to answer questions of interest. Linking data across previously siloed departments has great potential to break down programmatic silos between departments. For example, linking data related to health, education, safety net, and criminal justice can help further policy goals related to caring for the whole person. Linking data across different safety-net programs can help departments understand how to improve program enrollment. Linking program data with outcome data can help departments understand how successful a program is in achieving its outcomes, and how to continue to improve the program.

CPL has conducted a number of projects utilizing privacy-preserving linkage to answer these questions and further these policy goals. Examples include:

- Measuring the California Earned Income Tax Credit take-up gap amongst CalFresh (food stamp) enrollees
- Estimating the number of California college students who are eligible for but not enrolled in safety-net programs
- Analyzing the characteristics of the California children at risk of not receiving the federal stimulus payments and expanded Child Tax Credit

Conclusion

Hashed linkage is a powerful tool for unlocking the potential of linked administrative data, for research or for practice. But hashed linkages are complex to orchestrate, and require practice to be able to execute efficiently. We hope that this guide offers interested parties a clear roadmap for executing hashed linkages, as well as code to get them started.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant #2024283. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. All errors should be attributed to the authors.

We are very grateful to the staff at the California Department of Social Services and the California Franchise Tax Board for their thoughtful feedback.

Any findings reported herein were performed with the permission of the California Department of Social Services and the California Franchise Tax Board. The opinions and conclusions expressed herein are solely those of the authors and should not be considered as representing the policy of the collaborating department, agency, or any department or agency of the California government.