A Cryptographic Perspective on California's Delete Request and Opt-out Platform

Aarushi Goel[1], Gabriel Kaptchuk[2], Peihan Miao[3],
Phuoc Van Long Pham[3], Satvinder Singh[4], Rachel E. Thomas[2]

[1]Rutgers University, aarushi.goel@rutgers.edu
[2]University of Maryland, {kaptchuk, rthomase}@umd.edu
[3]Brown University, {peihan_miao, phuoc_pham_van_long}@brown.edu
[4]Purdue University, sing1745@purdue.edu

Abstract

The California Consumer Privacy Act (CCPA) grants consumers the right to request deletion of personal data held by registered data brokers. California's 2023 **Delete Act** provides a new mechanism through which consumers can exercise this right by leveraging a centralized *Delete Request and Opt-out Platform (DROP)*, enabling users to submit a single request that must be periodically processed by all registered data brokers. Since this platform aggregates sensitive user information, it is intentionally designed with safeguards against information leakage and unauthorized disclosure. This legislation, and the system it introduces, are positioned to serve as a template for wider deployment, with copy-cat legislation already introduced in many US states.

In this document, we evaluate the privacy architecture of the first-generation DROP system from a cryptographic perspective. The proposed design relies on *hash-based record linkage*—a widely used technique that offers heuristic rather than rigorous privacy guarantees. We begin by analyzing the inherent risks of this approach and identifying simple, lightweight modifications that can modestly improve its privacy guarantees by making it more difficult for malicious actors to exploit these vulnerabilities.

We then contrast this approach with *Private Set Intersection (PSI)*, a class of provably secure cryptographic protocols that only reveal the intended matching records and nothing more. For future iterations of DROP, we propose a comprehensive PSI-based redesign, arguing that it would substantially reduce the risk of accidental information leakage while incurring only moderate additional computational overhead. Next, we highlight how DROP presents a compelling application for advanced variants of PSI with extended functionalities and stronger security guarantees, and we outline several research directions for the cryptography community, motivated by the unique requirements of such systems. Finally, we identify systemic risks that cannot be mitigated by PSI alone and discuss potential complementary approaches for addressing these broader challenges.

We hope this document serves as a common starting point for the cryptography and policy communities working to design future data deletion systems.

Contents

1 Introduction **3**

2 Analysis of the DROP System Design **5**
- 2.1 System Overview . . . 5
- 2.2 Hash-based Record Matching . . . 5
- 2.3 Privacy Risks . . . 8

3 Short-term Considerations: Immediately Actionable Options **10**

4 Medium-term Considerations: Using Private Set Intersection in DROP **11**
- 4.1 Overview of Private Set Intersection . . . 11
- 4.2 Use Cases of Private Set Intersection . . . 12
- 4.3 A Simple Protocol: The 2HashDH PSI Protocol for Record Linkage . . . 14
- 4.4 Deployment Considerations: Cost of Using Private Set Intersection . . . 15

5 Long-term Considerations: Extended Capabilities of Private Set Intersection **17**
- 5.1 Fuzzy PSI for Expanded Functionality . . . 17
- 5.2 PSI over Committed Inputs for Stronger Security . . . 19
- 5.3 Updatable PSI for Better Efficiency . . . 21

6 Synthesis: Putting Private Set Intersection in Context **21**
- 6.1 Client adding their data to the system . . . 23
- 6.2 Government processing client's requests . . . 24
- 6.3 Running the PSI . . . 25
- 6.4 Auditing the data broker . . . 26
- 6.5 Data brokers selling data . . . 28

7 Conclusion **29**

1 Introduction

The California Consumer Privacy Act of 2018 (CCPA) [1], as amended by the California Privacy Rights Act of 2020 (CPRA), requires data brokers operating in the State of California to register with the State. It also grants consumers the right to request that these data brokers delete personal information they hold about them. Under this Act, consumers could either submit individual deletion requests with each data broker or use commercial solutions such as DeleteMe [10], Incogni [92], and Aura [16] that offer to submit data deletion requests on behalf of consumers to data brokers.

In October 2023, California enacted *SB-362 Data Broker Registration: Accessible Delete Mechanism* (**the Delete Act**) which went into effect on January 1st, 2026 [7]. The Delete Act builds on CCPA by offering a centralized platform, operated by the California Privacy Protection Agency (CPPA), for processing these deletion requests, called the *Delete Request and Opt-out Platform (DROP)*. Consumers can submit a single deletion request to the DROP server, covering any information about them that may be collected by data brokers. Starting August 1st, 2026, registered data brokers are required to interact with the DROP server every 45 days [§7612(a)] to identify the data subjects who have submitted deletion requests to DROP and about whom they have collected information. The data brokers are required to delete all the information they have gathered about these data subjects. The rules for the DROP mechanism were finalized in late 2025 [8]. Following the introduction of this legislation, many other US states have introduced similar laws, including Connecticut (SB 4 – 2026), Illinois (HB4809 – 2026, HB2913 – 2025), Hawaii (HB2463 – 2026), Nebraska (LB602 – 2025), New Mexico (SB192 – 2026), New York (A9642 – 2026, S9088 – 2026), Rhode Island (S2766 – 2026), and Vermont (H211 – 2026).

Given that this system will process the data of many consumers—and many related systems may be deployed in other jurisdictions—it is critical to document and study the privacy properties of the proposal. In this document, we review the initial version [8] of the DROP mechanism from a cryptographic lens. Our review is motivated by the language of the Delete Act itself [7], which attempts to scope the privacy risks consumers assume when using the DROP server. In particular, the Act requires the DROP server to implement *"technical safeguards … to protect consumers' personal information from unauthorized use, disclosure, access, destruction, or modification"* (Section 6(a)(1)) and prevent the *"disclosure of any additional personal information when the data broker accesses the accessible deletion mechanism"* (Section 6(b)(3)). While these provisions are not technical specifications, they reflect an implicit understanding that the DROP server should be designed to minimize the risks associated with aggregating large volumes of consumer information.

High-Level Findings. The approach used in the current implementation of the DROP server is based on a standard set of techniques that we refer to as *hash-based record linkage*. While these techniques are also widely deployed in several other domains (e.g., linking administrative data or health data [15, 27, 74, 76, 77, 95]), they provide only heuristic privacy guarantees and do not offer the best-in-class security. We highlight the risks inherent in this approach and discuss how it fails to fully realize the privacy guarantees that the Delete Act envisions for the DROP server.

Bringing a cryptographic perspective to the DROP system, we then argue for adopting the paradigm of *provable security* rather than relying on heuristic guarantees for future systems. In the provable security paradigm, analysts are able to formally prove a set of properties that a system has, demonstrating that these properties will hold as long as some simple, well-understood axioms hold true. Unlike heuristic approaches, provably secure systems provide significantly stronger assurance that the design does not overlook subtle vulnerabilities or unintended information leakage.

Private Set Intersection. To this end, we advocate replacing the current approach with a well-studied class of "provably secure" cryptographic techniques known collectively as *private set intersection (PSI)*. PSI protocols enable two entities to compute only the matching records between their datasets, while revealing nothing else. These techniques have already been deployed at scale by major technology companies (including Apple [14], Google [56, 81], Microsoft [69], and Signal [87]), although they have seen more limited adoption in government and healthcare settings. Despite their comparatively shorter deployment history in these domains, incorporating PSI into DROP could substantially reduce the risk of accidental leakage of consumers' private information in DROP-like systems, incurring only moderate additional computational costs. We therefore strongly recommend that future iterations of the system consider transitioning to this technology.

This Document. The first version of DROP is already deployed, with over 200,000 Californians having already signed up for the system [12], and the rules governing this version of the system have already been finalized. As such, our analysis will not be able to significantly influence the initial deployment of the system. However, we are hopeful that versions of this high-impact regulation will be enacted in more jurisdictions—an instance of the "California Effect." As such, our goal is to advocate for the use of system architectures with stronger security properties in future versions of the DROP system and in other deployments. As part of this effort, we aim to clearly document the design and operation of the proposed DROP system in a manner that is accessible both to the cryptographic research community and to the policy community responsible for designing and deploying the system. We hope this document can serve as a starting point for deeper collaboration between these communities. The remainder of this document is organized as follows:

1. *Overview of the DROP System.* In Section 2, we begin with an overview of the current DROP system architecture and discuss the privacy and security risks inherent in the existing approach.

2. *Short-term Considerations.* We recognize that the first version of DROP has already been deployed, making substantial architectural changes difficult in the near term. Therefore, in Section 3, we first propose a set of lightweight modifications that should be relatively easy to adopt and can provide modest improvements to the system's privacy guarantees.

3. *Medium-term Considerations.* For the next iteration, in Section 4, we then discuss a more comprehensive redesign of the DROP system based on PSI, replacing the current hash-based record linkage mechanisms. While such an approach is expected to incur moderate computational overhead, it offers significantly stronger and provable security guarantees.

4. *Long-term Considerations.* DROP-like systems also present a unique opportunity for the cryptography community to investigate more advanced variants of PSI with extended functionalities and stronger security guarantees. Although standard PSI is highly efficient, existing techniques for these advanced PSI variants often remain computationally expensive, provide limited functionality, or fail to achieve sufficiently strong security guarantees. In Section 5, we highlight research directions and open problems motivated by the unique requirements of systems such as DROP.

5. *Summary and Contextualization.* Finally, in Section 6, we put our discussion regarding the use of PSI into context by situating it within the larger system components *beyond* the PSI. This serves to summarize the system and also provides an opportunity to highlight the limits of cryptographic interventions; cryptographic systems—even advanced ones—cannot prevent all possible malicious actions.

2 Analysis of the DROP System Design

In this section, we provide a review of the DROP system, including its intended functionality, current design, and potential privacy risks.

2.1 System Overview

The DROP system allows California residents to request deletion of their personal data held by registered data brokers by submitting a single request to a centralized platform, rather than contacting each broker individually. We illustrate the high-level design of the system in Figure 1.

Deletion Request Collection. As a first step, California residents may submit deletion requests through the DROP portal operated by the government. The system verifies residency and collects personal information such as name, date of birth, ZIP code, phone number, email address, and advertising identifiers. This collected information is then maintained in a centralized database held by the government.

Data Broker Deletion. Each registered data broker is required to periodically retrieve deletion requests from the DROP system and compare them with its local database. If a submitted request matches a record in the broker's database, the broker must delete the corresponding record, including all associated information about the resident. Importantly, this information includes inferences made about the resident which might not have been collected directly (e.g., they like cats, inferred from the websites they have visited). In addition, brokers are required to report the outcome of this matching process to the government (e.g., whether a match was found and deleted) and to apply the deletion requests to any newly collected data.

Status Codes. DROP represents the outcome of a consumer request using status codes. *Deleted* indicates that the broker successfully identified the consumer's records, deleted all non-exempt information associated with the request, and will no longer sell the consumer's data. *Opted-out* indicates that the broker could not confidently or uniquely identify the consumer using the submitted information. In this case, the broker may retain any matching records but is prohibited from selling or sharing the associated data. Finally, *Record not found* indicates that the broker could not locate records matching the submitted identifiers. This status reflects an inability to establish a match at the time of processing and does not necessarily imply the complete absence of consumer data. Additional statuses exist in the deployed DROP system and can be found in the public DROP documentation.

Privacy Considerations. In order to protect residents' personally identifiable information (PII), the government does not share raw data directly with data brokers. Instead, the DROP system adopts a hash-based approach for record matching, as specified in the final rules [8], inspired by hash-based record linkage. We describe this approach in more detail in the next subsection.

2.2 Hash-based Record Matching

The DROP system adopts a hash-based approach to matching deletion requests with data brokers' records. This approach is inspired by hash-based record linkage, variants of which have been used to link records across multiple data owners without sharing raw personal information, e.g., [15,27,74,76,77,95]. At a high level, this method first selects a fixed subset of fields as the *identifier* for all records. Then each data owner applies a *cryptographic hash function* to the selected fields for every record, producing a single hash value per record. Cryptographic hash functions are, informally, deterministic functions whose output *appears* random and are hard to invert. Instead of sharing raw data, data owners share these resulting hash values

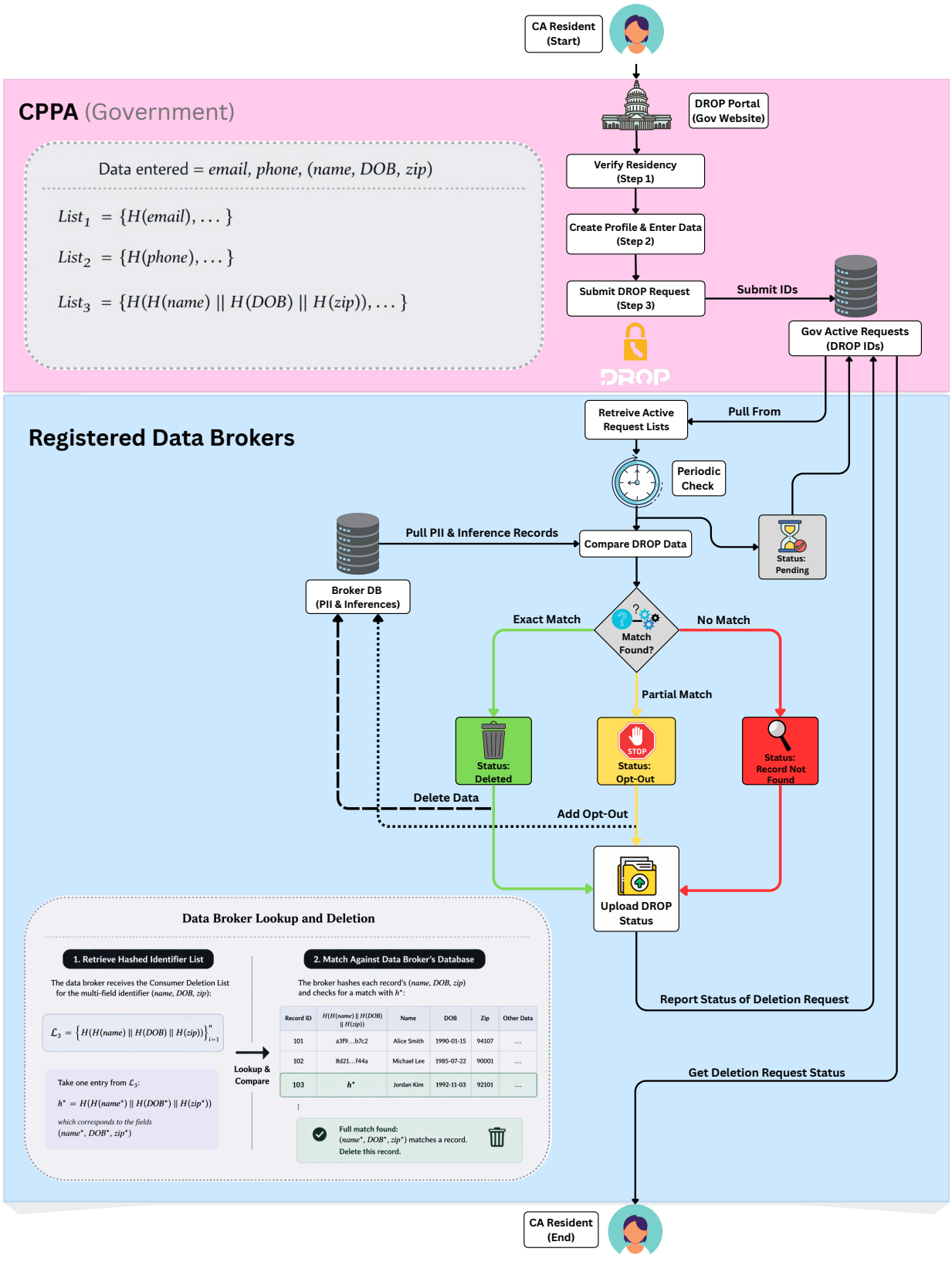

Figure 1: **Functional Architecture of the CPPA DROP Deletion Program.** Here, *inferences* denote attributes a broker predicts or derives about an individual (e.g., via machine-learning models) rather than directly collected PII.

and use these to identify matches. Although this approach is simple and scalable, it provides only heuristic privacy guarantees, as we discuss in the next subsection.

We now describe how the DROP system adapts this approach to the record matching setting. The government holds a dataset X with records $\{x_1, \ldots, x_n\}$, where each record x_i responds to a deletion request (i.e., a resident's data record submitted to DROP) and contains personal information such as the resident's name, date of birth, and phone number. Similarly, each data broker holds its own dataset $Y = \{y_1, \ldots, y_m\}$. Note that the fields in X and Y may differ, i.e., y_i may not contain certain information collected by the government (e.g., lacking phone number) or may contain additional information not collected by the government (e.g., brand loyalty, purchasing habits, or features inferred about individuals using custom machine learning models). The high-level goal is to enable the data broker to learn the matching entries between X and Y, i.e., $X \cap Y$, while hiding information about the entries in $X \setminus Y$.

In order to decide if an x_i matches some y_j, the DROP system considers some subset of the fields within X and Y to be *identifiers* which can be compared. Rather than selecting a single identifier, the DROP system selects multiple identifiers (e.g., $\mathsf{id}_1 = \mathsf{email}$, $\mathsf{id}_2 = \mathsf{phone}$). Furthermore, each identifier may contain more than one type of information (e.g., $\mathsf{id}_3 = (\mathsf{name}, \mathsf{date\text{-}of\text{-}birth}, \mathsf{zip\text{-}code})$). These identifiers are defined as "Consumer Deletion Lists" in [§7601(c)]. The data broker learns which entries in Y match some entry in X, in that $x_i[\mathsf{id}_t] = y_j[\mathsf{id}_t]$ for some record y_j and some identifier id_t.

The DROP system works as follows: the DROP server collects information from California residents that construct its data set X. The rules for the DROP system specify the identifiers $\mathsf{id}_1, \ldots, \mathsf{id}_k$. Thus, each record x_i in X contains several identifiers $(x_i[\mathsf{id}_1], \ldots, x_i[\mathsf{id}_k])$. For instance, $x_i[\mathsf{id}_1] = \mathsf{email}_i$, $x_i[\mathsf{id}_2] = \mathsf{phone}_i$, $x_i[\mathsf{id}_3] = (\mathsf{name}_i, \mathsf{DOB}_i, \mathsf{zip}_i)$.

The DROP server then constructs a "Consumer Deletion List" List_t for each identifier id_t by computing a cryptographic hash function H on the identifier, i.e., $H(x_i[\mathsf{id}_t])$, for all records. For example, the DROP server computes List_1 (for $\mathsf{id}_1 = \mathsf{email}$) and List_2 (for $\mathsf{id}_2 = \mathsf{phone}$) as follows:

$$\mathsf{List}_1 := \{H(x_i[\mathsf{id}_1])\}_{x_i \in X} = \{H(\mathsf{email}_1), H(\mathsf{email}_2), \ldots\}$$
$$\mathsf{List}_2 := \{H(x_i[\mathsf{id}_2])\}_{x_i \in X} = \{H(\mathsf{phone}_1), H(\mathsf{phone}_2), \ldots\}$$

If an identifier contains multiple fields, nested hashing is applied (specified in [§7613(a)(2)(A)]). For instance, List_3 for $\mathsf{id}_3 = (\mathsf{name}, \mathsf{date\text{-}of\text{-}birth}, \mathsf{zip\text{-}code})$ is computed as follows:

$$\mathsf{List}_3 := \{H(x_i[\mathsf{id}_3])\}_{x_i \in X} = \{H(H(\mathsf{name}_1) \| H(\mathsf{DOB}_1) \| H(\mathsf{zip}_1)), H(H(\mathsf{name}_2) \| H(\mathsf{DOB}_2) \| H(\mathsf{zip}_2)), \ldots\}$$

where $\|$ denotes concatenation.

Each data broker must then download all consumer deletion lists whose corresponding identifier is collected by the broker [§7610(a)(3)], and must do so at least every 45 days [§7612(a)]. For instance, if the data broker holds any record that contains email, they must download the corresponding consumer deletion list for email.

Recall that a cryptographic hash function H (e.g., SHA family [5, 6]) is a *deterministic* function (i.e., it always outputs the same value given the same input) that has the property that it is difficult to *invert* the function H (i.e., given an output of H, the best way to find an input that maps to this same output is to guess the same input). Therefore, the data broker can locally compute $H(y_i[\mathsf{id}_t])$ for each y_i and id_t in their dataset Y and use these hash values to find matches in List_t. Whenever $H(y_i[\mathsf{id}_t]) = H(x_j[\mathsf{id}_t])$ for some value $H(x_j[\mathsf{id}_t])$ on the consumer deletion list List_t, the data broker must delete the entirety of y_i, including all other fields in y_i [§7613(b)(1)]. The DROP rules then require the data broker to report a

"status" for each entry in the consumer deletion list, indicating if a corresponding y_i was found and deleted [§7614(b)(2)]. Additionally, the data broker is required to hold the consumer deletion list entries that did not match any record and check all newly acquired data against these entries, deleting any matching data in the same manner as above [§7613(c)].

The hash-based approach enjoys several practical advantages. It is simple to implement and deploy, making it easy to scale. Furthermore, it requires minimal interaction between the DROP server and data brokers, as brokers are responsible for retrieving relevant consumer deletion lists and performing necessary deletions, with no additional obligations on the government side beyond maintaining these lists. These properties likely contributed to the adoption of this approach; however, it comes with inherent privacy risks as we discuss below.

2.3 Privacy Risks

There are two subtly different privacy risks about which we should be concerned:

(1) leaking the independent identifiers uploaded to the DROP system, e.g., the information that *someone* with the phone number 555-555-5555 registered for DROP; and

(2) leaking the relationship between the identifiers, e.g., that the phone number 555-555-5555 is associated with example@example.com.

The current DROP design prioritizes (2), as different identifiers are hashed in separate consumer deletion lists, making these linkages difficult for someone to uncover. However, addressing leakage (1) requires more care.

As discussed above, the hash-based approach relies on mapping of identifiers to random looking values using a hash function, which can then be directly compared to check for matches by data brokers. Critically, while these identifiers *look* random, they are not—they are computed *deterministically* from the underlying identifiers and possibly some publicly chosen information, such as a key or a nonce (a number used only once), known to all parties performing the computation. This mapping process has the property that it is difficult to efficiently *invert*, i.e., given an output h, the only viable strategy for finding an x such that $H(x) = h$ is simply to guess possible values of x. Nevertheless, importantly, it *is* feasible to do this type of guess-and-check "attack," and this approach can be carried out in a reasonable amount of time when the number of possible x's is (relatively) small. Concretely, during the hash-based protocol, data brokers download a series of hashes $h_1 = H(x_1), \ldots, h_k = H(x_k)$ from the DROP system, at which point the data broker holds enough information to check if any given value $\hat{x}$ is in the DROP database (i.e., check if $\hat{x} \in \{x_1, \ldots, x_k\}$). This leakage is *inherent* to the approach.

This guess-and-check attack immediately facilitates leakage (1) listed above, as many of the identifiers come from (relatively) small spaces of options. However, the links between identifiers are not as easily recovered. The most direct way to recover these links is through lists containing multiple types of information (e.g., a list which hashes together name, date of birth, and zipcode); guessing the data in this list immediately provides linkage information. This means that care must be used when designing the set of lists. Additionally, linkages could potentially be extrapolated when the identifiers of a single user are added to multiple lists at the same time, e.g., during an update. In the worst case, one could imagine each consumer update list adding the information of only a single user, appending the hash of their phone number to one list, the hash of their name to another list, and their email to a third list. Because these updates were applied simultaneously, an attacker could infer that they pertain to the same person.

While it is important to consider the risks of a data broker attempting to extract from the consumer deletion lists, the biggest risk is that these lists may *leak*, at which point anyone who can get access to them will also be able to launch this attack without interacting with the DROP server. Documented uses of hash-based record linkage systems are generally a "one-off" event between two parties that have some sense of mutual trust or have signed carefully crafted non-disclosure agreements [15, 27, 74, 76, 77, 95]. Moreover, once the linkage has completed, the sender and receiver are free to delete any extraneous data generated during the process. The DROP system, on the other hand, will be running this protocol among hundreds of data brokers who may not be trustworthy, may not have implemented sufficiently high-quality security practices in order to guarantee that the consumer deletion lists remain confidential, and must continue to store copies of consumer deletion lists indefinitely. Generally speaking, we consider it *inevitable* that the list will be leaked and made accessible to malicious actors.

This risk is not without precedent; online services that use passwords to authenticate users must contend with the same problem. Specifically, it is well documented that users pick predictable passwords and regularly re-use passwords across multiple services (e.g., [23]). As such, best-practice for storing passwords is via hashing [93]. However, in the event that the list of hashed passwords is compromised (the equivalent of the consumer deletion list leaking in the DROP setting), using the guess-and-check attack (commonly called "offline dictionary attacks" when discussing passwords) is still possible.

It is worth being very clear about the impact of this attack, as it is easy to mis-calibrate an estimation of its risks—both to overestimate them and to underestimate them.

- *How expensive is guess-and-check?* When the space of possible inputs through which an attacker needs to iterate is too big, it makes carrying out guess-and-check impractical. Unfortunately, the space must be *very* big. To put this in context, a 2019 MacBook Pro performs $\approx 400,000$ calls to SHA256 (a widely used cryptographic hash function) per second on a single thread, meaning it can perform ≈ 6 million evaluations of SHA256 per second in total.[1] This means that it would be possible to enumerate the hashes of all possible social security numbers in less than 30 minutes and enumerating all gmail addresses with eight-character (alphanumeric) usernames would require approximately 6 days—even on outdated and non-specialized hardware. By contrast, enumerating all 16-character usernames would take on the order of 10^{10} years on the same hardware, illustrating how sharply the cost grows with input length. Enumerating all possible Apple advertising identifiers [70], on the other hand, would take approximately 10^{24} years on this computer. Even using specialized hardware and enlisting many computers, strategies that would make the attack significantly faster, this would still remain intractable. Even if a trillion such computers were enlisted to solve the problem, it would take them a trillion years to fully enumerate the space. As such, the risk corresponds to how many possible inputs there are, which is generally called the *entropy in the input* in technical parlance.

- *Extracting linkage.* As discussed above, leakage (2) is difficult to exploit since different identifiers are hashed into separate consumer deletion lists. However, when a consumer deletion list contains multiple types of information, it becomes possible to extract linkage between these fields, although the work required increases *multiplicatively*. For example, if an identifier contained both social security numbers and emails, it would take much longer than the times listed above for each identifier independently to fully enumerate the space—jumping to around 10^7 years of computation. However, if there were a small list of emails for which an attacker was interested in learning the social

[1] The MacBook Pro on which we bench-marked has 8 cores with multi-threading, enabling the simultaneous execution of 16 programs.

security number of the owner, this computation could be done in 30 minutes per target email. With access to more powerful hardware, this targeted form of linkage extraction becomes a non-trivial risk.

– *Run now, extract later.* Critically, the guess-and-check attack need not be run in the midst of executing the hash-based comparison protocol. When the consumer deletion lists leak, attackers can run these attacks invisibly and at their own leisure.

This attack strategy puts the important goals of DROP-like systems at risk. Even if only a small number of individuals are affected by these attacks, it could cause reputational damage, limiting the potential impact of this important project. Especially for projects run by governments, it is important to adhere to the highest level of security practice.

3 Short-term Considerations: Immediately Actionable Options

There are a number of extremely lightweight techniques that could be used in the current description of the DROP system without requiring significant changes to the system's architecture:

– *Use password hashing functions [Difficulty: low; Impact: high]:* A standard technique used in password hashing is the use of memory-hard or slow hash functions [93], like Argon2 [21] and PBKDF [64,96] respectively. These hash functions exploit the fact that an attacker needs to compute *many* more hashes than an honest receiver, as the attacker is attempting to brute force through many options, e.g., all 8 character gmail addresses. By making each hash computationally intensive or slow, this dramatically increases the cost of performing guess-and-check attacks without significantly slowing down honest use. Based on our current understanding of the final rules governing the first version of DROP [8], each consumer deletion list can specify the hash function to use (see [8, §7613 (a)(1)(B)]). In this case, switching to a password hashing function is possible without making any rule changes to the DROP system and will make guess-and-check attacks significantly more expensive to launch.

– *Hash salting [Difficulty: low; Impact: medium]:* A common technique in password hashing is to add some high entropy "salt" into each hash. That is, rather than compute $H(\text{email})$, a random value r (e.g., a random 128-bit string) is sampled and $(r, H(r, \text{email}))$ is sent to the receiver. In the context of password hashing, this prevents *precomputation* of guesses *before* the password hashes are compromised. We note, however, that salts are usually sampled uniquely for each password to ensure that guesses cannot be reused across credentials. In this context, sampling independent salt would require the honest receiver to do a quadratic number of hashes, which might be prohibitive. Thus, just using a single, global salt may be most appropriate. Integrating this change might be possible without changing the rules, as the use of salt can be considered just a particular choice of hash function.

– *Third party intermediary [Difficulty: high; Impact: medium]:* When using a vendor providing hash-based record linkage as a service, the vendor itself may do the computational work of finding the intersection. That is, the sender and receiver may each send the vendor a set of hashed identifiers, and the vendor may then tell the receiver the intersection of these sets. As such, any information leakage that occurs through the linkage process will be to the third party, rather than to the sender or receiver. Taking this approach requires finding an intermediary who will be responsible for keeping these

hashed lists confidential and executing the matching correctly. It is unlikely an organization would take on this responsibility for free; as such, this approach will likely increase the cost of deploying DROP or pose a significant logistical barrier. The main benefit is that copies of the consumer deletion lists would only need to be held by a small number of intermediaries, rather than by every data broker, reducing the risk of it leaking.

– *Trusted execution environments [Difficulty: high; Impact: high]:* Finally, some vendors or deployments of hash-based record linkage take advantage of an emerging set of secure computation environments generally called "trusted execution environments," e.g., Intel's SGX/TDX [4], AMD's Secure Encrypted Virtualization [2], and AWS's Nitro [3]. These environments make it difficult to observe the computation in progress; all data is transferred directly between the trusted execution environments and is not observable to even the organization that administers the computational resources. These resources are robust but not perfect; when attackers have physical access to the trusted execution environments, there is evidence that their defenses break down (e.g., [26, 57, 86, 100–103]). Given the anticipated technical sophistication of the likely attackers, this level of security may be appropriate for this application. Unfortunately, integrating trusted execution environments into software deployments requires some amount of technical expertise (either requiring the use of specialized cloud services or purchasing specific hardware), although recently the barrier to leveraging these technologies has been reducing [3].

Each of these approaches represents a *defense in depth*, that is, a partial solution that raises the cost of executing the attacks outlined in Section 2.3, while not completely eliminating the threat. The first two (salting and password-hashing functions) are virtually free to implement and are the most immediately actionable mitigations for future iterations of the system.

4 Medium-term Considerations: Using Private Set Intersection in DROP

Given the inherent risks associated with hash-based record linkage—even after some of the above additional protection mechanisms are integrated—it is natural to ask: *what alternatives exist that can simultaneously accomplish the goals of the DROP system while providing stronger security and privacy guarantees?* To answer this question, we explore using *private set intersection*, a cryptographic alternative with stronger guarantees that serves as a drop-in replacement to hash-based record linkage.

4.1 Overview of Private Set Intersection

Within the cryptographic literature, the problem of identifying common elements between two datasets while preserving the privacy of all non-common elements is formalized as the *Private Set Intersection* (PSI) problem. Viewed broadly, PSI has found numerous applications both in theory and in deployed systems. For example, two social media users may wish to determine whether they share mutual friends—or even compute the number of mutual friends they have—without revealing their entire friend lists to one another. Similarly, a password manager provider may wish to help users identify whether any of their passwords appear in publicly leaked password databases, without learning the users' actual passwords. PSI protocols facilitate this *exact* computation, revealing only the mutual friends or breached passwords, while keeping the other, "irrelevant" information from being revealed.

The key distinction between private set intersection and hash-based record linkage is that parties executing a PSI protocol cannot perform arbitrary after-the-fact queries on the other party's dataset. In the context of the DROP system, this property would prevent an adversary who gains access to the consumer deletion lists from mounting the guess-and-check attack described in Section 2.3, since every guess would require an explicit *interaction* with the DROP server. Requiring interaction also creates an opportunity to *authenticate* requests, ensuring that only registered data brokers can query the lists. Moreover, the DROP server could monitor the volume and pattern of queries issued by each broker, potentially detecting attempts to "extract" information from the system.

Over the more than 40 years since PSI was first formalized [73], it has received significant interest from the research community. The interest results in a suite of PSI protocols having not only depth (with efficiency being pushed to the limit) but also width (supporting enhanced functionalities).

Depth: Protocols with good performance. State-of-the-art PSI protocols [20, 82, 84] achieve high practical efficiency. Specifically, communication is the dominating cost of these protocols, with each side sending ≈ 1 hash output per element in the input set, resulting in performance comparable with hash-based record linkage. In Section 4.3, we explain another PSI paradigm based on the Diffie-Hellman assumption [60, 73], which we find better suited to the use case of the DROP system. The main advantage of the Diffie-Hellman-based protocol is in terms of updatability, which is crucial when the two sets (the government's list and the data broker's list) evolve over time. We further implement this protocol and show its scalability for the current size of California's population, with communication cost of only one elliptic curve group element (roughly the size of a hash output) per element each way; see Section 4.4 for more details.

Width: Protocols with enhanced functionality. While the plain PSI functionality suffices for the current usage of the DROP system, we provide more discussion about other functionalities that a PSI protocol can support. In terms of the protocol output, various aggregate statistics about the intersection could be computed without leaking the intersection itself, such as associated sum [38, 62, 63, 75], cardinality [44, 48, 52, 67], or more generically, a secret share of each element in the intersection [35, 59, 79, 84][2]. Further upgraded functionalities, such as PSI on fuzzy inputs [29, 50, 51, 53, 54, 88, 98, 99], PSI on committed inputs [19, 47, 55, 91], and updatable PSI [11, 17, 18, 71, 72] have also been extensively studied. We revisit them as future direction for strengthening the security and expanding the usage of the DROP system in Section 5.

In the following subsections, we discuss two examples of PSI deployment in Section 4.2, then propose a PSI solution for the DROP system in Section 4.3, along with deployment considerations in Section 4.4.

4.2 Use Cases of Private Set Intersection

Private set intersection is more than an academic proposal: it comprises a set of techniques that have seen real-world deployment at scale and is regularly considered for additional deployments. To illustrate the maturity of this technology, we briefly describe some of these deployments.

Use Case I: Breached Credential Notifications in Password Managers. Private Set Intersection has emerged as a widely deployed cryptographic primitive in large-scale consumer systems, particularly in password monitoring services offered by companies such as Apple, Google, and Microsoft [14, 36, 37, 40, 69, 81, 94]. Modern password managers are designed so that user passwords remain hidden even from the

[2] This variant of PSI is called Circuit-PSI.

developers operating the service. While this architecture substantially improves privacy and security, it creates a challenge: developers still need a way to notify users when one of their passwords appears in a database of breached credentials (e.g., credentials leaked or posted on the dark web). At the same time, developers do not want to broadly distribute databases of breached credentials, since doing so could further amplify the risk of malicious misuse.

To address this tension, several password managers have adopted PSI-based protocols. In these systems, the client-side password manager (typically integrated into a browser or stand-alone application) and a server operated by the password manager provider engage in a PSI protocol. The client inputs the user's credentials, while the server inputs the database of breached credentials, and the client learns only the intersection. The password manager can then notify the user if any stored credentials have been compromised.

This application highlights two key properties of PSI that are directly relevant to implementation challenges under the California Delete Act.

- PSI is trusted by leading technology companies to process highly sensitive data. Passwords are widely regarded as among the most sensitive categories of personal information, since they enable access to nearly every aspect of a user's digital life. Nevertheless, PSI enables intersection computation without requiring either party to reveal its full dataset.
- PSI operates efficiently at real-world scale, demonstrating that privacy-preserving set comparison is not merely a theoretical concept, but a practically deployable technology in large consumer systems.

In fact, Google's Password Checkup tool scanned more than 21 million username–password pairs within the first month of deployment and flagged approximately 316,000 credentials as compromised [65]. This large-scale deployment illustrates both the scalability of PSI-based approaches and their practical effectiveness in helping users identify compromised credentials while preserving privacy and security.

Use Case II: Apple CSAM. In 2021, Apple announced a system aimed at combating the distribution of Child Sexual Abuse Material (CSAM) on its platform [61]. The proposed system relied on a perceptual hashing algorithm called *NeuralHash*, which maps each image to a fixed-length hash value. Unlike traditional cryptographic hash functions, perceptual hashes are designed so that visually similar images produce similar outputs. This property enables the detection of CSAM images even when the images have been slightly modified or distorted.

In Apple's design, each image stored on a user's device was associated with a NeuralHash value, and the client maintained a local set of these hash outputs. Apple, in turn, maintained a database of NeuralHash values corresponding to known CSAM images provided by organizations such as the National Center for Missing and Exploited Children (NCMEC). A PSI protocol would then be used to determine whether the client's set of image hashes intersected with Apple's CSAM hash database, without revealing the full contents of either set.

Apple was prepared to deploy this system—and publicly announced it as an upcoming feature—further underscoring that PSI is a deployment-ready technology. The proposal, however, received substantial criticism from the research community [9, 58]. Importantly, these concerns were not primarily about the correctness or efficiency of PSI itself, but rather about the broader deployment setting and surrounding infrastructure. In particular, researchers raised concerns that the system lacked sufficient safeguards to prevent the underlying infrastructure from being repurposed to scan for non-CSAM content, such as material related to law enforcement investigations or politically motivated surveillance. Given the highly sensitive nature of the underlying data—namely, users' personal photo libraries—many researchers argued that even the additional privacy protections provided by PSI were insufficient in this context.

We argue that these concerns are specific to Apple's infrastructure and deployment setting, both of which are naturally resolved in our setting. In the DROP system, the data brokers are the ones receiving output and the number of queries they make to the government dataset can easily be upper-bounded. As such, the same concerns are unlikely to apply. The main lesson to be learned from this case study is that another leading technology company was ready to use PSI even on the most sensitive data types.

4.3 A Simple Protocol: The 2HashDH PSI Protocol for Record Linkage

The core problem with hash-based record linkage is that hash functions are both *deterministic* and *locally computable*, thereby enabling the guess-and-check attacks described in Section 2.3. Many private set intersection protocols address this issue by replacing the hash function with an *Oblivious Pseudorandom Function* (OPRF) [33], denoted by f. An OPRF behaves similarly to a hash function, except that evaluating it requires access to a secret key k. Moreover, OPRFs support an interactive, "oblivious" evaluation protocol involving another party, often called the *input holder*, who provides an input x. At the conclusion of the protocol, the following guarantees hold:

- The input holder receives $f_k(x)$ and does not gain any information about the key k;
- The key holder does not gain information about the input x.

Given access to an OPRF, the remainder of the PSI protocol proceeds similarly to hash-based record linkage. The sender begins with an input set $X = \{x_1, \ldots, x_n\}$, samples a random secret key k, and sends the receiver the set of OPRF evaluations $\{f_k(x_1), \ldots, f_k(x_n)\}$. The receiver then uses the interactive OPRF protocol described above to obtain evaluations of the same function on its own input set $Y = \{y_1, \ldots, y_m\}$, resulting in $\{f_k(y_1), \ldots, f_k(y_m)\}$. Finally, the receiver can locally compare these two sets of OPRF outputs to determine the intersection.

Crucially, because the receiver never learns the secret key k, it cannot compute additional evaluations of f_k beyond those obtained through the interactive protocol. This prevents the receiver from performing the type of after-the-fact guess-and-check attacks that arise in hash-based record linkage systems.

The 2HashDH OPRF. The above approach reduces the problem of designing a private set intersection protocol to designing a suitable OPRF. This task has been central to modern PSI research, resulting in several possible constructions, each of which have their individual strengths and weaknesses. Rather than enumerating over all possible OPRFs, we instead illustrate one technique based on foundational work of Diffie and Hellman [45], first formalized in [60, 73]. Our description of this technique requires some familiarity with number theory, which we briefly review along the way.

We describe here an OPRF construction, which is based on elliptic curves and provides provable security while requiring only relatively small OPRF outputs. The reader only needs to know the following basic fact about elliptic curves: given a base point G on the curve and an arbitrary scalar k, one can efficiently compute the point $k \cdot G$. The security of this construction relies on a well-studied assumption in both mathematics and computer science: for appropriate classes of elliptic curves, given only the base point G and a random point $k \cdot G$, it is computationally infeasible to recover the scalar k. This is commonly known as the elliptic curve *discrete logarithm* assumption.[3]

The OPRF would then consist of two components: a secret key k that corresponds to the hidden scalar and a standard cryptographic hash function H that maps arbitrary input strings to points on the elliptic

[3] More precisely, the OPRF protocol relies on the elliptic curve *decisional Diffie-Hellman* assumption in the random oracle model, which is a related yet stronger assumption than discrete logarithm.

curve. To evaluate the OPRF on an input x using the secret key k, one first hashes x to a curve point $H(x)$ and then multiplies the resulting point by k, obtaining $k \cdot H(x)$. Now consider another party that wishes to evaluate the OPRF on an input x without learning the secret key k. Such a party can interact with the key holder via the following protocol:

1. The input holder first hashes its input x to the curve point $H(x)$ and masks this value using a randomly chosen scalar r, obtaining $r \cdot H(x)$. The masked value is then sent to the key holder.
2. Using its secret key k, the key holder multiplies it with the received value to obtain $k \cdot (r \cdot H(x)) = (r \cdot k) \cdot H(x)$, and sends this masked OPRF evaluation back to the input holder.
3. Finally, the input holder removes the masking factor using the inverse scalar r^{-1}, recovering $r^{-1} \cdot (r \cdot k) \cdot H(x) = k \cdot H(x)$, which is precisely the desired OPRF output.

From a security perspective, this 2HashDH OPRF construction dominates hash-based record linkage. As in the hash-based approach, the server (i.e., key holder) computes a deterministic function on each item in its dataset and distributes the outputs to all data brokers. The OPRF preserves the property that the same function (i.e., same key k) can be reused across multiple interactions without compromising secrecy, enabling the server to reuse computation across invocations. The difference from hash-based record linkage is that the deterministic function is keyed by a secret k that data brokers never learn, preventing guess-and-check attacks. To check a candidate input, an attacker must query the server through the oblivious evaluation protocol, turning the offline attack into an online one that the server can monitor and rate-limit. Furthermore, since OPRF still behaves like a hash function even given k, compromising the key only downgrades security to the hash-based baseline.

4.4 Deployment Considerations: Cost of Using Private Set Intersection

To estimate the practical costs of deploying PSI in a DROP-like system, we implement the 2HashDH PSI protocol and benchmark it at the scale of California's population (approximately 39 million residents as of July 2025 [30]). Our open-source implementation[4] is run on a cloud server[5] equipped with a single NVIDIA A10 GPU, which, at the time of writing, can be rented for approximately $1.212 per hour.

Table 1 shows three important aspects of the performance: (1) the *bandwidth consumption*, (2) the *throughput*, and (3) the *monetary cost*. Another question we want to answer is about the *scaling trend* of the cost as the set size grows; hence we run the benchmark on three different set sizes, including the two common set sizes in the literature (1 million and 16 million) and a set size that reflects the current California's population (39 million). We observe a *linear* scaling trend along all three aspects (communication, computation, and monetary) of the cost, which is on par with the hash-based record linkage approach, with further analysis in the following paragraphs.

Bandwidth consumption. Bandwidth consumption, or *communication cost*, measures how many bytes of data a party sends through the network to run the protocol. Minimizing the communication cost is important, since this cost is not easy to parallelize. In this protocol, the server only needs to publish the set of OPRF evaluations once, with the cost being 32 bytes per set element. Any data broker can then download this set of OPRF values and use them in the same way as the hashes in the current system. In order for the data brokers to evaluate their own inputs, the data broker sends and receives a total of 64 bytes per element.

[4] Code available at https://github.com/phuocchubeo123/dh-oprf-gpu.git
[5] AWS EC2 g5.2xlarge.

Throughput. Even on a low-cost GPU, the protocol shows high throughput. The total amount of time needed to evaluate the server's own set of inputs is less than 1 second per million inputs. The end-to-end runtime for a data broker to query a set of a million inputs is 2 seconds. At the scale of California's population (39 million inputs), the server's one-time evaluation cost is less than 30 seconds, and the data broker obtains the OPRF evaluations in approximately 1 minute. This cost scales linearly with the set size and is highly parallelizable.

Monetary Cost. We also estimate the monetary cost for running the 2HashDH PSI protocol. For the largest set size that we consider (39 million), the end-to-end cost to rent a cloud server and run the PSI protocol is approximately 1.6 cents. This cost is, again, linear in the set size and is therefore easy to predict for larger inputs. For example, when scaling up to 350 million inputs (approximately the US population), the cost is about $0.14.

Set Size	Server One-time Cost		Data Broker PSI Cost			
	OPRF Comp. (s)	Publish (MB)	Upload (MB)	Download (MB)	Time over LAN (s)	Monetary Cost ($)
10^6	0.95	32.0	32.0	32.0	2.05	0.00055
16×10^6	11.3	512	512	512	26.6	0.0066
39×10^6	27.0	1,248	1,248	1,248	64.1	0.0158

Table 1: Experimental results for 2HashDH PSI on GPU.

These results suggest that modern PSI protocols can scale to support DROP-like systems in practice. We note that the input sizes considered in our benchmarks are on the order of the entire population of California and therefore likely overestimate the cost of a real deployment, since it is unlikely that every Californian would register with the system.

Considering other PSI systems. We remark that state-of-the-art PSI constructions [20, 82, 84] based on vector oblivious linear evaluation (VOLE) [24, 25] are concretely more efficient than the 2HashDH construction. They rely mostly on symmetric-key operations, avoiding the expensive per-element public-key operations in 2HashDH. Their overall communication cost is also slightly lower than that of 2HashDH. However, there are other practical trade-offs when considering them for deployment. These VOLE-based PSI protocols are more complex to implement, with many different components and parameters to choose, and they require some heuristics to estimate the concrete security level (e.g., parameters for the learning parity with noise (LPN) assumptions). As a nearer-term deployment option, the 2HashDH protocol is much simpler to implement — it relies on a well-studied assumption (Diffie-Hellman), the 2HashDH OPRF is specified in an IETF RFC [43], and its computation is highly parallelizable.

Software Quality, Standardization, and Interoperability. One clear advantage of hash-based solutions is the availability of commercial, off-the-shelf offerings [42], as well as deployment guidance from well-established organizations, e.g., [78]. As a result, hash-based approaches may be easier to deploy and maintain in practice. It may even be possible to transfer some of the operational or legal risks associated with data leakage to a third-party vendor.[6]

[6]We have not reviewed any specific contractual terms; we merely note that such arrangements may be possible.

By contrast, to our knowledge, there are currently no commercial PSI vendors or standardized software libraries that provide PSI functionality.[7] This is likely due, at least in part, to the fact that PSI remains an active and rapidly evolving area of research. More broadly, we are unaware of widely adopted standards for either approach, which may further hinder deployability and interoperability.

5 Long-term Considerations: Extended Capabilities of Private Set Intersection

In the previous section, we discussed "standard" PSI techniques. We note, however, that there are several important variants of PSI that offer expanded functionality (e.g., Fuzzy PSI), stronger security guarantees (e.g., PSI on committed inputs), and improved efficiency (e.g., Updatable PSI). These directions remain active areas of research and are likely to continue evolving rapidly over time. Nevertheless, we believe it is important to discuss them, as they may significantly influence the design and operation of future DROP-like systems.

5.1 Fuzzy PSI for Expanded Functionality

Early drafts of the DROP system proposed using a *threshold*-based matching rule; for example, if 50% of the identifiers associated with a consumer matched, the system would treat the records as corresponding to the same individual [13]. This requirement was ultimately removed because it was viewed as overly complex and potentially prone to erroneous deletions. While stricter thresholds (e.g., 90%) or requiring certain fields to match exactly (such as advertising identifiers) in addition to satisfying the threshold could likely mitigate many erroneous deletions, such approaches would only further increase the complexity of the system.

A somewhat unusual residual effect of transitioning from threshold-based logic to exact matching is that the current regulations include a "partial match" category that is itself not entirely satisfactory. In particular, when *multiple* data broker records partially match a single opt-out request, this ambiguity effectively prevents the broker from selling *any* of the partially matching records, while simultaneously not triggering a deletion event because it is unclear which specific record should be deleted.

Standard PSI inherently enforces exact matching on identifiers. Consider, for example, an identifier containing the name "Björn O'Connor-López." A standard PSI protocol would fail to identify a match if the data broker stored a slightly misspelled version of the name (e.g., "Conor" with a single "n") or only partial information (e.g., "Björn López"), even after both parties locally standardized their encodings (for example, replacing "ö" with "o"). Similarly, if an identifier consists of multiple fields—such as an email address, date of birth, and zip code—standard PSI would require every field to match exactly.

Fuzzy PSI is a variant of PSI that instead outputs elements that satisfy some notion of "approximate" matching, where the similarity metric is built directly into the protocol design. For example, the matching criterion could be based on geometric distance (e.g., Hamming or Minkowski), textual similarity (e.g., edit distance), or threshold-style agreement across multiple identifiers. In recent years, fuzzy PSI has been the subject of extensive research [22, 28, 29, 34, 39, 41, 50, 51, 53, 54, 68, 80, 83, 88, 89, 97–99, 104, 105], leading to practically efficient constructions for a variety of notions of approximation. Despite substantial progress, some similarity notions are not yet mature enough for real-world deployment, particularly in adversarial settings where malicious parties may deviate from the protocol to extract additional information.

[7] There have been efforts to standardize certain OPRFs [43], but these have not been extended to full PSI protocols.

At the same time, fuzzy PSI offers a promising long-term approach to addressing the complexity-flexibility trade-offs that contributed to the removal of threshold-based matching logic from the DROP system. In particular, the fuzzy matching logic could be implemented once within a carefully designed and extensively audited cryptographic protocol, eliminating the need for individual data brokers to independently implement and maintain their own custom matching heuristics.

Open Problems. For cryptographers interested in tackling open problems directly relevant to the DROP system, we provide a more technical description of the problem statement here. We first briefly recall the goal of fuzzy PSI: Two parties holding sets X and Y respectively want to find all $x \in X$ such that $\mathsf{Match}(x, y) = 1$ for some $y \in Y$, where $\mathsf{Match}(\cdot, \cdot) \in \{0, 1\}$ is a *matching predicate* and $x \in X, y \in Y$ are points in d dimensions. In the fuzzy PSI literature, the matching predicate is typically defined as a distance metric, i.e., $\mathsf{Match}(x, y) = 1$ iff $\mathsf{Dist}(x, y) \leq \delta$ for some distance metric Dist and threshold δ. As we describe below, the DROP system requires a more general predicate that cannot be expressed as a threshold on a single distance metric.

- **Matching Predicate** $\mathsf{Match}(\cdot, \cdot)$**.** Recall that each input data point consists of the identifiers corresponding to a single person (name, email, phone number, date of birth, etc.). We require a two-layer fuzzy PSI protocol, unlike any seen in the literature so far.

  1. *First layer of matching (per identifier):* For every identifier, we specify a (potentially fuzzy) matching rule. For example, two names name_1 and name_2 match if $\mathsf{EditDistance}(\mathsf{name}_1, \mathsf{name}_2) \leq \delta$ for some appropriate constant δ. Each identifier can have a different metric. For example, two dates of birth DOB_1 and DOB_2 match if $\mathsf{DOB}_1 = \mathsf{DOB}_2$.
  2. *Second layer of matching (across identifiers):* Two different points, say $(\mathsf{name}_1, \mathsf{DOB}_1, \mathsf{email}_1, \mathsf{phone}_1)$ and $(\mathsf{name}_2, \mathsf{DOB}_2, \mathsf{email}_2, \mathsf{phone}_2)$ match if at least t of their identifiers match under the per-identifier rules above, where t is some parameter.

  Combined, as an example, we specify a matching predicate like the following: We say that two inputs $(\mathsf{name}_1, \mathsf{DOB}_1, \mathsf{email}_1, \mathsf{phone}_1, \mathsf{ZIP}_1)$ and $(\mathsf{name}_2, \mathsf{DOB}_2, \mathsf{email}_2, \mathsf{phone}_2, \mathsf{ZIP}_2)$ match if at least 3 of the following 5 conditions hold:

  - $\mathsf{EditDistance}(\mathsf{name}_1, \mathsf{name}_2) \leq 5$
  - $\mathsf{DOB}_1 = \mathsf{DOB}_2$
  - $\mathsf{EditDistance}(\mathsf{email}_1, \mathsf{email}_2) \leq 2$
  - $\mathsf{HammingDistance}(\mathsf{phone}_1, \mathsf{phone}_2) \leq 1$
  - $\mathsf{ZIP}_1 = \mathsf{ZIP}_2$

- **Distributional Assumptions on Inputs.** To achieve better efficiency, for instance to avoid exponential dependence on the dimension or quadratic dependence on the set sizes, many fuzzy PSI protocols make assumptions on the distribution of the input sets, such as requiring one party's points to be well-separated. Some of these assumptions do not hold for real-world inputs, so protocols relying on them may not be applicable to the DROP system. Some works observe this and either avoid such assumptions [22, 89] or compile existing protocols to work for real-world datasets [88]. An open problem here is to design fuzzy PSI protocols that are efficient under assumptions realistic for the input distributions in the DROP setting.

– **Security.** Most existing works on fuzzy PSI are only secure against semi-honest adversaries. Extending security to malicious adversaries remains a challenging open problem.

While a fuzzy PSI protocol satisfying all of the above properties would be ideal, we note that progress on even one of these aspects (e.g., achieving malicious security, or fuzzy PSI for edit distance) is already valuable.

5.2 PSI over Committed Inputs for Stronger Security

While replacing hash-based record linkage with PSI mitigates the risk of guess-and-check attacks, there remain several additional security properties that may be desirable in a DROP-like system but are not provided by either standard PSI or traditional hash-based approaches. In particular, consider the following questions:

1. How can a consumer verify that their data was correctly included in the deletion lists maintained by the DROP system?
2. How can the DROP system operator verify that data brokers submit their *actual* datasets to the system?
3. How can downstream entities verify that data purchased from a data broker has been properly sanitized in accordance with DROP deletion requests?

In the current design, the first concern is addressed primarily through trust in the government, both to correctly include consumer data and to maintain a robust and error-free registration process. The second concern is handled through occasional auditing of data brokers. The third concern, however, is not directly addressed by the existing architecture. In particular, enabling third parties to validate sanitization would seemingly require wider access to the consumer deletion lists, thereby increasing the risk of guess-and-check attacks.

While cryptography cannot perfectly resolve all of these challenges (see discussion below), more advanced cryptographic techniques may help partially address them. For example, such techniques could enable client-side verification capabilities, support the creation of robust audit trails, and allow third-party organizations to validate that sanitization was performed correctly without directly exposing the deletion lists themselves. Designing a system that simultaneously provides these guarantees while also meeting the efficiency requirements of large-scale deployments remains highly non-trivial, but may become practically feasible within the next few years.

The core cryptographic primitive needed to realize this vision is *PSI over committed inputs* [19,31,46,47,55,66,90,91], a recently proposed variant of PSI in which parties first cryptographically bind themselves to their inputs before executing the protocol. In cryptographic terminology, this binding process is referred to as a *commitment*. A commitment has two key properties: it is *hiding*, meaning it reveals nothing about the underlying input, and *binding*, meaning the committing party cannot later open it to a different input than the committed one. Commitments can serve as the "glue" connecting the PSI protocol with additional cryptographic mechanisms that provide external verifiability and accountability guarantees. In particular, PSI over committed inputs could be leveraged to address the concerns outlined above in the following ways:

1. Since commitments are hiding, the DROP system can publish the commitment to its collected data without creating any privacy risk. Users could then request that the DROP system demonstrate

that their data is included in the committed dataset via a *partial de-commitment*. This capability follows directly from well-studied cryptographic commitment techniques. In this way, users would no longer need to blindly trust that the DROP system is error-free, although they would still need to trust the DROP system to correctly execute the PSI protocol on the committed inputs.[8]

2. Symmetrically, the data brokers' inputs are also committed, so the government could maintain a commitment to each broker's input for each execution of the PSI protocol and require the brokers to provide zero-knowledge consistency proofs across executions. Such proofs would demonstrate, for example, that records identified in prior executions were subsequently deleted. These proofs can be generated efficiently, particularly in settings where the PSI output is revealed to both parties. This approach would also simplify the auditing process, since auditors would no longer need to verify the correctness of an entire database during every audit; instead, it would suffice to verify the correctness of the initial committed database together with the validity of incremental updates across executions.

3. Finally, if one could design PSI protocols with *output commitments*, then downstream customers purchasing data from a data broker could use the same partial de-commitment techniques to verify that the purchased data appeared as input to at least one execution of the PSI protocol, while simultaneously confirming that it did not appear in the corresponding PSI output.

Open Problems. For cryptographers interested in tackling open problems directly relevant to the DROP system, we provide a more technical description of the problem statement here. We first briefly recall the goal of PSI over committed inputs: Two parties holding private inputs X and Y respectively along with public commitments $\mathsf{Commit}(X), \mathsf{Commit}(Y)$ want to compute $X \cap Y$ if the public commitments are consistent with the private inputs. We now discuss some interesting open problems:

- **Output Commitments.** Current works show how to efficiently bind the *input* of a PSI protocol to an existing commitment. This leaves open the question of binding the *output* of the protocol to a commitment: having both parties learn $\mathsf{Commit}(X \cap Y)$. Ideally, it should be publicly verifiable that the commitment contains the result of the PSI, but this can be easily bootstrapped by having the sender verify that the commitment is well-formed and then sign the commitment itself. From an efficiency perspective, it is important to design an approach to this problem that makes black-box use of cryptography.

- **Efficient Zero-knowledge Proofs.** Two distinct types of proofs are required:

  1. **Auditing Proofs:** A proof given by the data broker to the government proving that the input used by the data broker in the current PSI execution excludes the output of the previous execution, i.e., that the previously matched records were deleted on the broker's end.
  2. **Compliance Proofs:** A proof given by the government to the data broker, which the broker can use to prove to any third-party customer that the data being sold is compliant with the DROP deletion requirements.

  Constructing efficient zk-proofs for both remains a challenging open problem. Output commitments as discussed above are one potential way of solving these problems. An alternative approach could be via Dual-Output PSI with committed inputs, where the PSI output is revealed to both parties. Constructing this variant itself is a challenging and interesting open problem.

[8] Technically, this final property could also be verified by honest data brokers.

5.3 Updatable PSI for Better Efficiency

As required by the current regulations, each data broker must access the DROP system every 45 calendar days to download the latest consumer deletion lists. In a PSI-based design, this would require the government and each data broker to execute the PSI protocol on a regular basis with periodically updated sets. Notably, the current hash-based approach minimizes the government's computational overhead across repeated executions, since the government only needs to compute hashes corresponding to newly submitted deletion requests. This type of incremental efficiency is not inherent to standard PSI protocols.

Updatable PSI [11, 17, 18, 72] is a recently proposed variant of PSI in which the work required across repeated executions is proportional only to the number of entries that were added or deleted relative to the previous execution. As a result, updatable PSI can provide dramatic efficiency improvements in settings where the protocol must be run frequently. Nevertheless, further research is needed to ensure that these protocols can provide similarly strong privacy guarantees in adversarial settings where protocol participants may deviate from the prescribed protocol in an attempt to extract additional information.

Open Problems. For cryptographers interested in tackling open problems directly relevant to the DROP system, we provide a more technical description of the problem statement here. We first briefly recall the goal of updatable PSI: Two parties hold private sets X and Y respectively. These sets update over time as $X_0, X_1, X_2, \ldots$ and $Y_0, Y_1, Y_2, \ldots$, where $X_0 = X$ and $Y_0 = Y$. Parties wish to compute $X_i \cap Y_i$ for all i. The goal is to compute $X_{i+1} \cap Y_{i+1}$ more efficiently than standard PSI given (i) the previous result $X_i \cap Y_i$ and (ii) the fact that X_{i+1} and Y_{i+1} only differ slightly from X_i and Y_i, i.e., $|(X_{i+1} \setminus X_i) \cup (X_i \setminus X_{i+1})| \ll |X_i|$, meaning the number of items added to and deleted from X_i are much smaller than the set size $|X_i|$ (and similarly for Y_i). We now discuss some interesting problems:

- **Malicious Security.** No existing updatable PSI protocol achieves security against malicious adversaries. An important stepping stone can be achieving security against a malicious receiver (if we model the government as a semi-honest sender).
- **Compatibility.** Making existing fuzzy or committed PSI protocols compatible with the notion of updatability is another interesting problem. This is particularly relevant in practice because a real deployment would only use a single PSI protocol, and hence the desired properties must coexist (the same challenge applies to making fuzzy and committed PSI compatible with each other).

6 Synthesis: Putting Private Set Intersection in Context

In this section, we take a step back to summarize the impact of instantiating DROP-like systems with a PSI protocol. We take this opportunity to think about the overall system, highlighting ways in which more advanced cryptographic techniques can improve security of the system and also reflect on the inherent limitations of cryptography.

Mapping a PSI-based Deployment. To facilitate this discussion, we leverage the visual language recently developed by Canetti, Ha, and Kaptchuk [32]. Their framework is intended to support contextualizing cryptographic proposals within the deployment context for the purposes of threat modeling. While we do not leverage their full threat modeling process, the framework helps us discuss the interplay between technical and non-technical components of deployment.

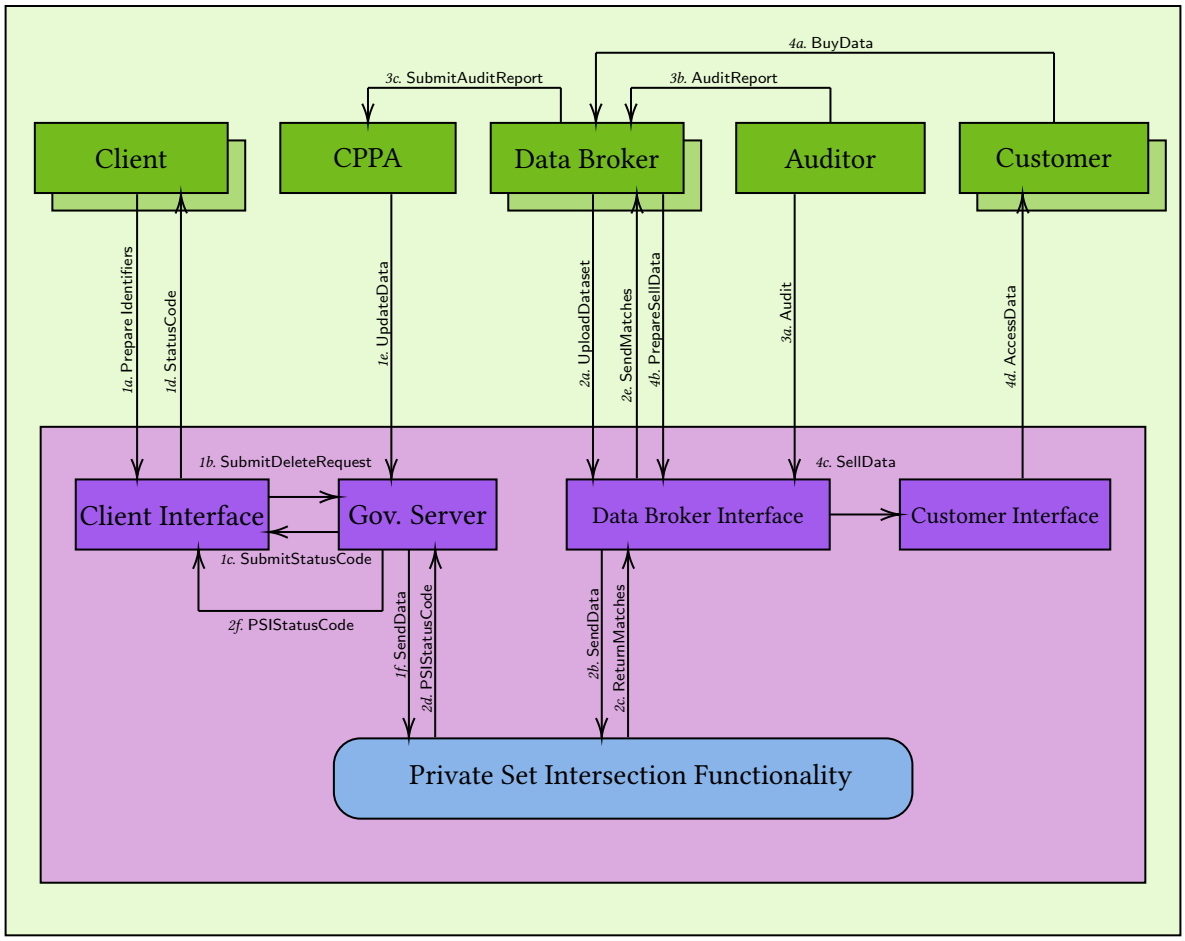

Figure 2: A visualization of the DROP system instantiated with the basic PSI protocol discussed in Section 4. We divide the system into three parts: cryptographic components (blue), non-cryptographic technical components (purple), and human, organizational, and social components (green). We further divide the information flows in the system into four stages: (1) clients submitting opt-out requests and initializing the PSI protocol, (2) data brokers interacting with the PSI to identify records that should be deleted, (3) a periodic auditing process to ensure data brokers are in compliance, and (4) customers purchasing data from data brokers.

In Figure 2, we render the DROP system instantiated using a PSI protocol into this framework. Specifically, we visualize a DROP system instantiated with the basic PSI system described at the beginning of Section 4. We intentionally expand the scope of our rendering to include two parties that we have, up until this point, not discussed: the *auditor* and a *customer* who purchases data from a data broker.

We divide the process of running a DROP-like system into three conceptual categories: (1) cryptographic parts of the system (shaded in blue), (2) non-cryptographic technical parts of the system (shaded in purple), e.g., databases or web-browsers, which are responsible for feeding inputs to and gathering outputs from the cryptographic parts of the system, and (3) human, organizational, and social parts of the system (shaded in green). Dividing the overall system in this way allows us to more clearly

understand the impact of using cryptographic tooling, highlighting both its positive impacts and its limits. By separating the cryptographic parts of the system from the non-cryptographic parts of the system, the guarantees offered by cryptography are more clear and opportunities to expand those guarantees are easy to find. On the other hand, it is difficult to design cryptography which can replace—or even fully reason about—person-to-person interactions; including the social parts of the system within our visualization allows us to reason more clearly about the limits of cryptography.

In the following subsections, we walk through this rendering step-by-step and highlight the ways in which cryptography can or cannot interact with the data flows between the various components of this expanded system. As part of our discussion, we show how the advanced features described in Section 5 augment this basic functionality.

6.1 Client adding their data to the system

The left-hand side of Figure 2 depicts the interaction between the client and the government-operated deletion infrastructure. The client interacts with the system through the *Client Interface*, which serves as the primary entry point into DROP-like systems. We briefly walk through the client-relevant data flows, before discussing it further.

1. **Client → Client Interface:** In Step *1a*, the client first prepares a collection of identifiers (e.g., name, e-mail address, phone number, or other personally identifiable information) and fills out an opt-out or deletion request through the client interface. The client accesses the interface through an identity gateway that authenticates the request prior to submission.

2. **Client Interface → Gov. Server:** In Step *1b*, the client interface forwards the authenticated opt-out or deletion request to the government server for further processing and inclusion in the deletion workflow. The government server then responds in Step *1c* with a status code, indicating if the authentication process succeeded and the information was processed correctly.

3. **Client Interface → Client:** Upon receiving confirmation, Step *1d* shows that the client interface returns a status code indicating whether the request was accepted or rejected.

Discussion. From a cryptographic perspective, this information flow is "far away" from the private set intersection—and, indeed, the client never directly interacts with the cryptographic components of the system. But, this flow touches on authentication, a process which merits further discussion.

- *Authenticating requests.* Opt-out requests must be authenticated, but the manner in which they are authenticated is beyond the scope of the actual system design. In the deployed system, this is managed through a third-party (but still governmental) identity management system like login.gov or the California Identity Gateway. This authentication checks that the client has the right to submit an opt-out request, but does not authenticate the contents of the request itself. From a cryptographic perspective, there are emerging techniques to authenticate with better privacy properties e.g., [49, 85], which could potentially also be extended to allow a client to prove that other information they provide is authentic.

  It is not immediately clear if stronger, cryptographic authentication is necessary in this setting or if stronger privacy properties make sense—especially because authenticated clients then provide plaintext information about themselves to the server. However, we note that this leaves the possibility that clients attempt to manipulate the way the system functions by submitting bad inputs

or data brokers themselves intentionally inject inputs into the system. It seems unlikely that the impact of such actions would be significant.

6.2 Government processing client's requests

The government component, represented by the *CPPA* and the *Gov. Server*, acts as the central coordinator of the DROP system. It is responsible for collecting deletion requests, maintaining commitments, initiating PSI executions, and coordinating auditing procedures.

1. **Gov. Server → Client Interface:** After receiving a deletion request from the client interface, the government server validates and processes the request in Step *1c*, which is then relayed to the user in Step *1d*.

2. **CPPA → Gov. Server:** As shown by Step *1e*, the CPPA periodically initiates execution of the system by instructing the government server to incorporate all newly received requests.

3. **Gov. Server → PSI Protocol:** During Step *1f*, the government server provides the PSI protocol with the set of deletion requests.

Discussion. There is nothing dictating when the CPPA needs to start including new opt-out requests into the system, so we model this decision as stemming from within the social component of the system, triggering new information going into the PSI functionality. However, we should observe that guaranteeing the *right* inputs are fed into the private set intersection is not naturally guaranteed by the base protocol.

- *Verifying inclusion.* In the basic PSI protocol outlined in Section 4, there is no cryptographic way for a client to verify that their information was properly incorporated into the input set. While it is unlikely that the government server would *intentionally* omit client requests, having the ability to audit the system provides confidence that there isn't a bug. As discussed in Section 5.2, we can add verification into the system using committed PSI. This would change the modeling to allow the client interface to query the Private Set Intersection Functionality directly on their inputs.

- *Verifying authenticity of inclusions.* We might also be interested in providing data brokers with the ability to verify that each record in the government input really was supplied by a real person. This could, in principle, prevent the server from injecting additional records unilaterally. While this is unlikely to happen in practice, it could similarly prevent programming errors or suspicion. In principle, this is cryptographically possible, as the server could use cryptographic techniques to convince data brokers that all inputs were generated by someone with access to a valid government credential. Unfortunately, this likely fails to provide any meaningful security against the most malicious form of government adversary, as the government itself—albeit different parts of the government—would be responsible for issuing these credentials.

- *Verifying consistency of inputs.* One final property that one might want from such a system is a guarantee that the DROP system uses the same opt-out lists for all data brokers. This means that an internal actor who can update opt-out lists cannot target a specific data broker with additional requests—any malicious action taken must affect all data brokers, making it potentially easier to identify. As discussed in Section 4 and Section 5.2, this *can* be provided by PSI systems, but is not inherent. The simple protocol described in Section 4.3 allows the server's message in the PSI

protocol to be *reused*, making it straightforward to enforce that the same inputs are used for all data brokers.[9] This protocol, however, is significantly slower than state-of-the-art PSI systems, which do not provide a direct mechanism for enforcing this property; in order to ensure consistency across different invocations of the PSI when using state-of-the-art techniques, one could leverage the committed inputs approach discussed in Section 5.2.

6.3 Running the PSI

The center of Figure 2 captures the data flows relevant to running the PSI itself. The data broker has two components, a social component that decides when to run the system and possibly interfaces with other technical systems that they administer (e.g., an external database) and a technical component, Data Broker Interface, which drives the cryptographic components. There are six main data flows to consider:

1. **Data Broker → Data Broker Interface:** As illustrated by Step *2a*, the data broker triggers their software (represented by the Data Broker Interface) to begin a new iteration of the PSI protocol.
2. **Data Broker Interface → PSI Protocol:** During Step *2b*, the data broker interface provides all their data into the PSI protocol and receives as output the entries of the database that need to be deleted.
3. **PSI Protocol → Data Broker Interface:** In Step *2c*, the private set intersection functionality outputs the results of the PSI to the data broker interface.
4. **PSI Protocol → Gov. Server:** In Step *2d*, the private set intersection additionally informs the government server about the result of the protocol, as the rules require that clients be able to see when their data is deleted.
5. **Data Broker Interface → Data Broker:** Once the set of matches has been received, the data broker interface forwards to the data broker the output of the PSI protocol, namely the records requiring deletion. This is demonstrated in Step *2e*.
6. **Gov. Server → Client Interface:** In Step *2f*, the government server provides status reports corresponding to deletion requests identified through the PSI execution, allowing users to understand when their data was identified as a candidate for deletion.

Discussion. These data flows capture the core of the cryptographic components of the system we outline, in which the PSI protocol is run on each data broker's data. This leads to several important observations:

- *Defining a match.* As discussed in Section 5.1, PSI allows for some degree of configurability when it comes to defining a match through fuzziness. While more research must be done in order to expand the types of fuzziness that are supported, this offers an important way to expand the functionality offered in this part of the system.
- *Efficiency.* State-of-the-art PSI protocols are very fast, but inherently require work proportional to the size of the input sets. When sets are large, this could lead to significant processing times. As discussed in Section 5.3, there are emerging PSI protocols that recognize that intersection is likely to be computed repeatedly over periodically changing sets—just as happens with the DROP system—which reduces the runtime of the protocol.

[9] We note that this protocol lacks malicious security guarantees, which would be required, possibly using the techniques discussed in Section 5.2.

– *Enforcing deletion.* In the modeling we do above, we don't capture the database system the data broker is using to store their information; instead, we pass the results of the PSI up to the social layer of the system, such that the data broker can appropriately relay that information to the right subsystem. This also highlights an important problem: there is no way to verify that the matching records will actually be deleted. In fact, there is no good cryptographic way to do this at all, as the data broker can always keep a second copy of their database offline. This is inherent in the approach, and motivates the need for the auditor (discussed below).

– *Two-sided output.* In order to fulfill the requirements of the current rules, the PSI protocol would be required to provide the results to both the government server and the data broker interface. This is necessary if we want to allow the government server to share a deletion status code with the client (message 2f). Unfortunately, this runs into the well-known fairness impossibility result. As such, it would always be possible for a malicious party to refuse to complete the protocol, preventing the other party from learning the output. When combined with auditing, this may not be a real-world concern, but it is a limitation on what cryptography can reasonably accomplish.

– *Building an interface.* Finally, it is worthwhile to pause to think about who will hold responsibility for building the data broker interface. There are three obvious choices: (1) the government agency running the system, (2) a third party, or (3) the data brokers themselves. Recall that there are steep penalties for failing to delete data and that data brokers all might have their own way of storing data. Thus, while options (1) and (2) seem to be much more natural solutions from an engineering perspective, the existing deployment instead chooses option (3). This ensures that the data brokers cannot escape their legal responsibilities because of a bug introduced by one of the other parties—or, worse, that the data brokers are not accidentally assessed massive fines due to a bug introduced by one of the other parties. Moreover, this helps support interoperability with the various database systems used by data brokers. On the other hand, this makes it significantly more difficult to deploy PSI in practice, as it requires each data broker to implement a non-trivial cryptographic protocol on their own, increasing the cost of compliance, risk of implementation bugs, and complexity of auditing.

6.4 Auditing the data broker

One component of the system that we have yet to discuss in detail is *auditing*, a process mandated by the Delete Act. Specifically, the Delete Act requires all registered data brokers to undergo an independent audit to assess compliance with the law at least once every three years. The auditor acts as an independent oversight entity responsible for evaluating whether data brokers are complying with their statutory deletion obligations and whether the broker's implementation of the DROP framework is operating correctly. Most importantly, this is the only place in the existing approach that validates that a data broker actually deletes the appropriate data.

We model the auditing process in three steps:

1. **Auditor → Data Broker Interface:** Step *3a* models the auditor invoking some software functionality associated with the data broker interface. Presumably this would include access to the system, the logs, and potentially any databases. If the database isn't integrated with the data broker interface, then the auditor might also need to similarly query that system. The auditor might also want to inspect source code, internal policies, database schema, historical compliance reports, or other information.

2. **Auditor → Data Broker:** After completing the audit, the auditor generates a formal audit report summarizing the results of the evaluation. We assume that this report is most likely to take the form of a human-readable document, rather than a machine readable one. As represented by Step *3b*, the report is delivered to people within the data broker organization.

3. **Data Broker → CPPA:** The Delete Act further requires data brokers to retain audit reports and related materials for a minimum of six years and to provide such materials to the CPPA within five business days upon written request. We capture this process in Step *3c*.

Discussion. The auditing process is the least well-defined operation in the current system design, and so we are forced to speculate both about how it will work in practice and how cryptography could support stronger audits. While there are ways in which cryptographic tools could help simplify or improve the auditing process, auditing is fundamentally a non-cryptographic process and cryptographic guarantees alone are insufficient to establish full compliance. For instance, an auditor would need to verify that there are no auxiliary storage systems or backup databases maintained by the broker that hold data that should be deleted, but finding these systems depends upon non-technical, social factors about which cryptography cannot reason.

We imagine that an audit would involve the auditor checking at least the following:

(i) the data broker has software that correctly participates in the PSI protocol,

(ii) there exists code or guides to delete data from operational databases based on the results of the PSI protocol

(iii) deletion workflows execute within the legally mandated time periods,

(iv) the broker is not maintaining undeclared replicas or secondary copies of deleted data.

As discussed below, these fall short of actually checking that the data broker is fully compliant.

– *Timing of deletion.* Full compliance would require that a data broker *immediately* delete the appropriate data, but auditing need only happen once every three years and only checks the state of the system at audit time. This leaves a significant gap of time in which the data broker can arbitrarily deviate from the protocol. It also leaves open the question of assessing appropriate fines: the auditor has no way to pinpoint exactly how long a data broker has been in violation of the rules, which determines the value of the fine.

  Weaving commitments into the input and output of the PSI offers an opportunity to get a clearer snapshot of the timing of the system over time, which could be valuable data to the auditors. Specifically, this would create an audit trail when certain records were identified by the PSI protocol for deletion. Of course, commitments alone would be insufficient, as this would require the data broker to continue to hold the plaintext within the commitments in order to open them to the auditor. Encrypting this information under an auditing public key would be a straightforward way to resolve this tension.

  Of course, even integrating these cryptographic tools would still not let an auditor learn that data was *actually* deleted at the appropriate time. As noted above, this task is likely beyond the reach of cryptographic interventions.

– *Auditing inputs.* In addition to checking that data was deleted at the appropriate time, it might be valuable for an auditor to check that the *right* data has been fed into the PSI protocol, i.e., the inputs to the PSI really do reflect the contents of the data broker's database. Here, again, we run into a timing issue: an auditor only comes infrequently (e.g., once every three years), making it difficult for them to check what inputs were used in the past or the contents of the databases at that time.

  As above, it is possible that leveraging commitments and encryption could make this form of auditing easier, in that an auditor could at least learn what inputs were provided to each invocation of the PSI protocol. Of course, this is not sufficient to check that these inputs match perfectly the data broker's database.

– *Data requirements for auditing.* In addition to reflecting on the limits of auditing, we also observe that auditing appears to come with privacy risks. Namely, how should the auditor determine if there is any data that *should* be deleted? Clearly, the auditor could run the PSI protocol on the data broker's data, finding any intersection, but this approach makes it difficult to find any bugs in the data broker interface; if some records are not being identified correctly because of the bug, they will not appear in the audit. Thus, it seems like the most effective approach to auditing would require that auditors be able to access the raw client opt-out requests, which of course comes with significant privacy risks.

6.5 Data brokers selling data

Finally, Figure 2 models the interaction between data brokers and downstream customers who purchase data. While this might seem like it is "far away" from the cryptography, modeling this interaction gives us an opportunity to discuss additional opportunities to leverage cryptographic components.

We model this in the most basic system as an interaction between the social components of the customer and the data broker. Clearly, each party will have some software components in their system (a database or an email client, at minimum), so we capture these components within the purple part of the figure. There are four information flows:

1. **Customer → Data Broker:** Step *4a* models a customer initiating a request to purchase data from a broker.

2. **Data Broker → Data Broker Interface:** As represented by Step *4b*, the data broker prepares the requested dataset for sale by interacting with the software components of their system.

3. **Data Broker Interface → Customer Interface:** In Step *4c*, the data broker interface sends the customer interface the purchased data.

4. **Customer Interface → Customer:** Finally, in Step *4d*, the customer accesses the purchased data.

Discussion. In this basic approach, the customer need not be aware that the data broker is running their information through the PSI system at all. While this is convenient, it misses a key opportunity for cryptography to improve the guarantees of the system. Specifically, the customer can—if the system is designed with the right cryptographic tooling—act as an auditor.

– *Verifying sanitization.* As discussed above, there are hard limits on what auditing can hope to accomplish. We can augment this by allowing customers to act as auditors for the data they purchase. Ideally, buyers would verify cryptographic evidence establishing that this data was derived from a

database properly processed through a PSI execution. The most natural approach would be to simply allow each customer to run the PSI with the government server themselves and verify that there is no intersection. While elegant and simple, this dramatically increases the set of people who can interact with the government servers, which poses an innate cybersecurity risk. Moreover, there is no guarantee that these customers only input data they have purchased into the PSI protocol; there is the risk that they use their access to try and launch very targeted guess-and-check attacks—recall that PSI only prevents *offline* guess-and-check attacks. Finally, this approach would put a non-trivial computational load on customers.

An alternative approach would be to leverage a proof system, which allows a prover (e.g., a data broker) to convince a verifier (e.g., the customer) that some data has some property. There has been dramatic progress improving the performance of these proof systems, which allow a verifier to check an arbitrary property in milliseconds. Realizing such a system remains technically challenging, as it dramatically increases the protocol's scope and requires integrating both input and output commitments (see Section 5.2). While modern proof systems continue to improve in efficiency, developing practical, end-to-end frameworks that capture the entire data lifecycle remains an active area of research. Ultimately, this scenario highlights an opportunity to expand cryptographic utility: while current techniques excel at guaranteeing isolated computations, future frameworks must adapt to secure broader, real-world organizational processes and downstream data flows.

7 Conclusion

In this document, we have examined the privacy architecture of the first-generation DROP system through a cryptographic lens. Hash-based record linkage provides only heuristic privacy guarantees, leaving the consumer deletion lists exposed to guess-and-check attacks—a risk that grows especially acute once these lists are replicated across hundreds of data brokers and retained indefinitely. We then outlined a spectrum of remedies calibrated to different time horizons: lightweight, immediately deployable mitigations such as password-hashing functions and salting; a medium-term redesign built on private set intersection, which we show is practical at the scale of California's population; and a set of longer-term research directions—fuzzy, committed-input, and updatable PSI—motivated directly by the requirements of DROP-like systems.

At the same time, our analysis underscores that cryptography is not a panacea. Even an idealized PSI deployment cannot, on its own, guarantee that a data broker actually deletes matched records, that periodic audits capture violations occurring between them, or that the surrounding organizational and social processes behave honestly. Situating PSI within the full deployment context makes both its strengths and its limits legible. Because copy-cat legislation is already spreading to other jurisdictions, we hope this document can serve as a shared starting point for the cryptography and policy communities looking to build future deletion systems.

References

[1] Title 1.81.5: California Consumer Privacy Act of 2018. Available at https://leginfo.legislature.ca.gov/faces/codesdisplayText.xhtml?division=3.&part=4.&lawCode=CIV&title=1.81.5.

[2] Amd secure encrypted virtualization (sev). Available at https://www.amd.com/en/developer/sev.html.

[3] Aws nitro system. Available at https://aws.amazon.com/ec2/nitro/.

[4] Intel trust domain extensions. Available at https://www.intel.com/content/www/us/en/products/docs/accelerator-engines/trust-domain-extensions.html.

[5] Federal information processing standards publication 180-4: Secure hash function. Available at https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.180-4.pdf, August 2015.

[6] Federal information processing standards publication 202: Sha-3 standard: Permutation-based hash and extendable-output functions. Available at https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, August 2015.

[7] California delete act (sb 362), 2023.

[8] *Data Broker Registration and Accessible Deletion Mechanism.* 2025. Accessed on 28 December 2025 via Westlaw at https://govt.westlaw.com/calregs/Browse/Home/California/CaliforniaCodeofRegulations?guid=I492E4D90BFC111F081779CCDE22850D9.

[9] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Jon Callas, Whitfield Diffie, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Vanessa Teague, and Carmela Troncoso. Bugs in our pockets: The risks of client-side scanning, 2021.

[10] Abine. Deleteme: Remove your personal info from search results. https://joindeleteme.com/.

[11] Archita Agarwal, David Cash, Marilyn George, Seny Kamara, Tarik Moataz, and Jaspal Singh. Updatable private set intersection from structured encryption. *CiC*, 2(4):25, 2026.

[12] California Privacy Protection Agency. Californians take control of personal data with DROP platform. https://www.linkedin.com/posts/calprivacymore-than-200000-californians-have-used-activity-7423136473274695680-liWF.

[13] California Privacy Protection Agency. Final statement of reasons and update to informative digest. Available at https://cppa.ca.gov/regulations/pdf/dropfosr.pdf#::text=In%20subsection%20(a)(2)(A)%2C%20the%20Agency%20added%20specific,data%20broker%20must%20process%20a%20deletion%20request., November 2025.

[14] Apple Platform Security. Password Monitoring – Apple Platform Security. https://support.apple.com/en-al/guide/security/sec78e79fc3b/web. Accessed: 2026-03-05.

[15] NORC at the University of Chicago. Pprl2-23-n02: Final report. Available at https://www.americasdatahub.org/wp-content/uploads/2026/03/PPRL2-Final-Report.pdf, January 2026.

[16] Aura. Aura: All-in-one digital safety. https://buy.aura.com/.

[17] Saikrishna Badrinarayanan, Peihan Miao, Xinyi Shi, Max Tromanhauser, and Ruida Zeng. Updatable private set intersection revisited: Extended functionalities, deletion, and worst-case complexity. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 200–233. Springer, Singapore, December 2024.

[18] Saikrishna Badrinarayanan, Peihan Miao, and Tiancheng Xie. Updatable private set intersection. *PoPETs*, 2022(2):378–406, April 2022.

[19] James Bartusek, Sanjam Garg, Abhishek Jain, and Guru-Vamsi Policharla. Laconic PSI on authenticated inputs and applications. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 262–293. Springer, Singapore, December 2025.

[20] Alexander Bienstock, Sarvar Patel, Joon Young Seo, and Kevin Yeo. Near-optimal oblivious key-value stores for efficient psi, PSU and volume-hiding multi-maps. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 301–318. USENIX Association, 2023.

[21] Alex Biryukov, Daniel Dinu, Dmitry Khovratovich, and Simon Josefsson. Rcc 9106: Argon2 memory-hard function for password hashing and proof-of-work applications. Available at https://datatracker.ietf.org/doc/rfc9106/, September 2021.

[22] Erik-Oliver Blass and Guevara Noubir. Assumption-free fuzzy PSI via predicate encryption. Cryptology ePrint Archive, Paper 2025/217, 2025.

[23] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552. IEEE Computer Society Press, May 2012.

[24] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

[25] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Cham, August 2019.

[26] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: SGX cache attacks are practical. In *USENIX Workshop on Offensive Technologies, WOOT*, 2017.

[27] Adrian P Brown, Anna M Ferrante, Sean M Randall, James H Boyd, and James B Semmens. Ensuring privacy when integrating patient-based datasets: new methods and developments in record linkage. *Frontiers in public health*, 5:34, 2017.

[28] Dung Bui and Kelong Cong. Efficient fuzzy labeled psi from vector ring-ole. In *Cryptology and Network Security: 24th International Conference, CANS 2025, Osaka, Japan, November 17–20, 2025, Proceedings*, page 173–198, Berlin, Heidelberg, 2025. Springer-Verlag.

[29] Dung Bui, Gayathri Garimella, Peihan Miao, and Van Long Phuoc Pham. New framework for structure-aware PSI from distributed function secret sharing. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part V*, volume 16249 of *LNCS*, pages 294–326. Springer, Singapore, December 2025.

[30] California Department of Finance. E-6. population estimates and components of change by county — july 1, 2020–2025. https://dof.ca.gov/forecasting/demographics/estimates/E-6/, 2025.

[31] Jan Camenisch and Gregory M. Zaverucha. Private intersection of certified sets. In Roger Dingledine and Philippe Golle, editors, *FC 2009*, volume 5628 of *LNCS*, pages 108–127. Springer, Berlin, Heidelberg, February 2009.

[32] Ran Canetti, Julie Ha, and Gabriel Kaptchuk. Analyzing cryptography in context: A cryptography-native approach to threat modeling. In Benjamin Stock and Elissa M. Redmiles, editors, *USENIX Security 2026*. USENIX Association, August 2026.

[33] Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. SoK: Oblivious pseudorandom functions. In *2022 IEEE European Symposium on Security and Privacy*, pages 625–646. IEEE Computer Society Press, June 2022.

[34] Anrin Chakraborti, Giulia Fanti, and Michael K. Reiter. Distance-aware private set intersection. In Joseph A. Calandrino and Carmela Troncoso, editors, *USENIX Security 2023*, pages 319–336. USENIX Association, August 2023.

[35] Nishanth Chandran, Divya Gupta, and Akash Shah. Circuit-PSI with linear complexity via relaxed batch OPPRF. *PoPETs*, 2022(1):353–372, January 2022.

[36] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled PSI from fully homomorphic encryption with malicious security. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1223–1237. ACM Press, October 2018.

[37] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1243–1255. ACM Press, October / November 2017.

[38] Yu Chen, Min Zhang, Cong Zhang, Minglang Dong, and Weiran Liu. Private set operations from multi-query reverse private membership test. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14603 of *LNCS*, pages 387–416. Springer, Cham, April 2024.

[39] Wutichai Chongchitmate, Steve Lu, and Rafail Ostrovsky. Approximate PSI with near-linear communication. Cryptology ePrint Archive, Paper 2024/682, 2024.

[40] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from homomorphic encryption with reduced computation and communication. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 1135–1150. ACM Press, November 2021.

[41] Chengrui Dang, Xv Zhou, and Bei Liang. Efficient fuzzy PSI based on prefix representation. Cryptology ePrint Archive, Paper 2025/1796, 2025.

[42] Datavant. Datavant: The data collaboration platform trusted for healthcare. Available at https://www.datavant.com/.

[43] Alex Davidson, Armando Faz-Hernández, Nick Sullivan, and Christopher A. Wood. Oblivious pseudorandom functions (OPRFs) using prime-order groups. https://datatracker.ietf.org/doc/rfc9497/, 2023.

[44] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and private computation of cardinality of set intersection and union. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *CANS 12*, volume 7712 of *LNCS*, pages 218–231. Springer, Berlin, Heidelberg, December 2012.

[45] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 1976.

[46] Sky Faber, Ronald Petrlic, and Gene Tsudik. Unlinked: Private proximity-based off-line OSN interaction. In *ACM WPES*, pages 121–131. ACM, 2015.

[47] Francesca Falzon and Evangelia Anna Markatou. Re-visiting authorized private set intersection: A new privacy-preserving variant and two protocols. *Proceedings on Privacy Enhancing Technologies*, 2025.

[48] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Berlin, Heidelberg, May 2004.

[49] Matteo Frigo and abhi shelat. Anonymous credentials from ECDSA. Cryptology ePrint Archive, Report 2024/2010, 2024.

[50] Ying Gao, Lin Qi, Xiang Liu, Yuanchao Luo, and Longxin Wang. Efficient fuzzy private set intersection from fuzzy mapping. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part VI*, volume 15489 of *LNCS*, pages 36–68. Springer, Singapore, December 2024.

[51] Gayathri Garimella, Benjamin Goff, and Peihan Miao. Computation efficient structure-aware PSI from incremental function secret sharing. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VIII*, volume 14927 of *LNCS*, pages 309–345. Springer, Cham, August 2024.

[52] Gayathri Garimella, Payman Mohassel, Mike Rosulek, Saeed Sadeghian, and Jaspal Singh. Private set operations from oblivious switching. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 591–617. Springer, Cham, May 2021.

[53] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Structure-aware private set intersection, with applications to fuzzy matching. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 323–352. Springer, Cham, August 2022.

[54] Gayathri Garimella, Mike Rosulek, and Jaspal Singh. Malicious secure, structure-aware private set intersection. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 577–610. Springer, Cham, August 2023.

[55] Aarushi Goel, Peihan Miao, Phuoc Van Long Pham, and Satvinder Singh. PICS: Private intersection over committed (and reusable) sets. Cryptology ePrint Archive, Paper 2025/1071, 2025.

[56] Google Research. Private Intersection-Sum Protocols with Applications to Attributing Aggregate Ad Conversions. https://research.google/pubs/pub51026/. Accessed: 2026-03-05.

[57] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on intel SGX. In *EUROSEC*, 2017.

[58] Matthew Green, Vanessa Teague, Bruce Schneier, Alex Stamos, and Carmela Troncoso. An evaluation of the risks of client-side scanning. Real World Crypto 2022 (RWC 23). Recording available at https://www.youtube.com/live/AGlwtmfgMjk?si=l6nnmGFRckmdOwxj&t=1520.

[59] Yan Huang, David Evans, and Jonathan Katz. Private set intersection: Are garbled circuits better than custom protocols? In *NDSS 2012*. The Internet Society, February 2012.

[60] Bernardo A. Huberman, Matthew K. Franklin, and Tad Hogg. Enhancing privacy and trust in electronic communities. In Stuart I. Feldman and Michael P. Wellman, editors, *Proceedings of the First ACM Conference on Electronic Commerce (EC-99), Denver, CO, USA, November 3-5, 1999*, pages 78–86. ACM, 1999.

[61] Apple Inc. Apple PSI System Security Protocol and Analysis. PDF Document, aug 2021. Accessed on 18 December 2023. Since initial publication, Apple has modified the contents of the webpage. The initial announcement can be viewed at https://web.archive.org/web/20210805191006/https://www.apple.com/child-safety/pdf/ApplePSISystemSecurityProtocolandAnalysis.pdf.

[62] Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, Mariana Raykova, David Shanahan, and Moti Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *2020 IEEE European Symposium on Security and Privacy*, pages 370–389. IEEE Computer Society Press, September 2020.

[63] Mihaela Ion, Ben Kreuter, Erhan Nergiz, Sarvar Patel, Shobhit Saxena, Karn Seth, David Shanahan, and Moti Yung. Private intersection-sum protocol with applications to attributing aggregate ad conversions. Cryptology ePrint Archive, Report 2017/738, 2017.

[64] Burt Kaliski. Rfc 2898: Pkcs #5: Password-based cryptography specification, version 2.0. Available at https://datatracker.ietf.org/doc/html/rfc2898, September 2000.

[65] Michael Kan. Google finds 316,000 accounts using logins from past breaches. *PCMag*, August 2019. Accessed: 2026-03-13.

[66] Florian Kerschbaum. Outsourced private set intersection using homomorphic encryption. In Heung Youl Youm and Yoojae Won, editors, *ASIACCS 12*, pages 85–86. ACM Press, May 2012.

[67] Lea Kissner and Dawn Xiaodong Song. Privacy-preserving set operations. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer, Berlin, Heidelberg, August 2005.

[68] Anunay Kulshrestha and Jonathan R. Mayer. Identifying harmful media in end-to-end encrypted communication: Efficient private membership computation. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 893–910. USENIX Association, August 2021.

[69] Kristin Lauter, Sreekanth Kannepalli, Kim Laine, and Radames Cruz Moreno. Password monitor: Safeguarding passwords in microsoft edge. https://www.microsoft.com/en-us/research/blog/password-monitor-safeguarding-passwords-in-microsoft-edge/, January 2021. Microsoft Research Blog. Accessed: 2026-03-05.

[70] P. Leach, M. Mealling, and R. Salz. RFC4122: A Universally Unique IDentifier (UUID) URN Namespace. Available at https://datatracker.ietf.org/doc/html/rfc4122, 2005.

[71] Guowei Ling, Peng Tang, Shi-Feng Sun, and Weidong Qiu. Efficient updatable psi from asymmetric psi and psu. *IEEE Transactions on Information Forensics and Security*, 2026.

[72] Junxin Liu, Peihan Miao, Mike Rosulek, Xinyi Shi, and Jifeng Wang. Updatable private set intersection from symmetric-key techniques. EUROCRYPT, 2026.

[73] Catherine Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *1986 IEEE Symposium on Security and Privacy*, pages 134–134. IEEE, 1986.

[74] Jason Meyer, Chris Frederick, Rick Edwards, Rui Wang, Honoka Suzuki, Alex Bohl, and David Jones. Utilizing privacy preserving record linkage (pprl) to link data from two federal statistical agencies: Final project summary report. Available at https://www.americasdatahub.org/wp-content/uploads/2025/09/NCSES-NCHS-PPRL-Final-Project-Summary-Report.pdf, August 2025.

[75] Peihan Miao, Sarvar Patel, Mariana Raykova, Karn Seth, and Moti Yung. Two-sided malicious security for private intersection-sum with cardinality. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 3–33. Springer, Cham, August 2020.

[76] Lisa B Mirel, Dean M Resnick, Jonathan Aram, and Christine S Cox. A methodological assessment of privacy preserving record linkage using survey and administrative data. *Statistical Journal of the IAOS*, 38(2):413–421, 2022.

[77] Aditi Pathak, Laina Serrer, Daniela Zapata, Raymond King, Lisa B Mirel, Thomas Sukalac, Arunkumar Srinivasan, Patrick Baier, Meera Bhalla, Corinne David-Ferdon, et al. Privacy preserving record linkage for public health action: opportunities and challenges. *Journal of the American Medical Informatics Association*, 31(11):2605–2612, 2024.

[78] Stacy J. Petersen, Robert D. Lieberthal, Keith J. Miller, and Niyati H. Vakil. Privacy preserving record linkage (pprl) strategy and recommendations: Pprl linkage strategies report. Available at https://www.nia.nih.gov/sites/default/files/2023-08/pprl-linkage-strategies-preliminary-report.pdf, January 2023.

[79] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Cham, May 2019.

[80] Lucas Piske, Jaspal Singh, Ni Trieu, Vladimir Kolesnikov, and Vassilis Zikas. Distance-aware OT with application to fuzzy PSI. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 4679–4691. ACM Press, October 2025.

[81] Jennifer Pullman, Kurt Thomas, and Elie Bursztein. Protect your accounts from data breaches with password checkup. Available at https://security.googleblog.com/2019/02/protect-your-accounts-from-data.html, February 2019.

[82] Srinivasan Raghuraman and Peter Rindal. Blazing fast psi from improved okvs and subfield vole. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 2505–2517, 2022.

[83] David Richardson, Mike Rosulek, and Jiayu Xu. Fuzzy PSI via oblivious protocol routing. Cryptology ePrint Archive, Paper 2024/1642, 2024.

[84] Peter Rindal and Phillipp Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 901–930. Springer, Cham, October 2021.

[85] Michael Rosenberg, Jacob D. White, Christina Garman, and Ian Miers. zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In *2023 IEEE Symposium on Security and Privacy*, pages 790–808. IEEE Computer Society Press, May 2023.

[86] Alex Seto, Oytun Kuday Duran, Samy Amer, Jalen Chuang, Stephan van Schaik, Daniel Genkin, and Christina Garman. Wiretap: Breaking server sgx via dram bus interposition. In *2025 SIGSAC Conference on Computer and Communications Security (CCS '25)*. Association for Computing Machinery, 2025.

[87] Signal. Technology preview: Private contact discovery for Signal. https://signal.org/blog/private-contact-discovery/. Accessed: 2026-03-05.

[88] Satvinder Singh, Yanxue Jia, and Aniket Kate. Fuzzy private set intersection for real-world datasets. Cryptology ePrint Archive, Paper 2026/441, 2026.

[89] Hyunjung Son, Seunghun Paik, Yunki Kim, Sunpill Kim, Heewon Chung, and Jae Hong Seo. Doubly efficient fuzzy private set intersection for high-dimensional data with cosine similarity. Cryptology ePrint Archive, Paper 2025/054, 2025.

[90] Yunqing Sun, Jonathan Katz, Mariana Raykova, Phillipp Schoppmann, and Xiao Wang. Actively secure private set intersection in the client-server setting. In *ACM CCS*, pages 1478–1492, 2024.

[91] Yunqing Sun, Hanlin Liu, Kang Yang, Yu Yu, Xiao Wang, and Chenkai Weng. Committed vector oblivious linear evaluation and its applications. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *ACM CCS 2025*, pages 3635–3648. ACM Press, October 2025.

[92] Surfshark. Incogni: Automatic personal data removal. https://incogni.com/.

[93] David Temoshok, James L. Fenton, Yee-Yin Choong, Naomi Lefkovitz, Andrew Regenscheid, Ryan Galluzzo, and Justin P. Richer. Nist special publication 800-63b-4: Digital identity guidelines; authentication and authenticator management. Available at https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63B-4.pdf, July 2025.

[94] Kurt Thomas, Jennifer Pullman, Kevin Yeo, Ananth Raghunathan, Patrick Gage Kelley, Luca Invernizzi, Borbala Benko, Tadek Pietraszek, Sarvar Patel, Dan Boneh, and Elie Bursztein. Protecting accounts from credential stuffing with password breach alerting. In Nadia Heninger and Patrick Traynor, editors, *USENIX Security 2019*, pages 1556–1571. USENIX Association, August 2019.

[95] William E Trick, Fred Rachman, Keiki Hinami, Jennifer C Hill, Craig Conover, Lisa Diep, Howard S Gordon, Abel Kho, David O Meltzer, Raj C Shah, et al. Variability in comorbidites and health services use across homeless typologies: multicenter data linkage between healthcare and homeless systems. *BMC Public Health*, 21(1):917, 2021.

[96] Meltem Sonmez Turan, Elaine Barker, William Burr, and Lily Chen. Nist special publication 800-132: Recommendation for password-based key derivation (part 1: Storage applications). Available at https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-132.pdf, December 2010.

[97] Erkam Uzun, Simon P. Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy labeled private set intersection with applications to private real-time biometric search. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 911–928. USENIX Association, August 2021.

[98] Aron van Baarsen and Sihang Pu. Fuzzy private set intersection with large hyperballs. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part V*, volume 14655 of *LNCS*, pages 340–369. Springer, Cham, May 2024.

[99] Aron van Baarsen and Sihang Pu. Fuzzy private set intersection from vole. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 327–360. Springer, 2025.

[100] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution. In *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, August 2018.

[101] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lipp, Marina Minkin, Daniel Genkin, Yarom Yuval, Berk Sunar, Daniel Gruss, and Frank Piessens. LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection. In *41th IEEE Symposium on Security and Privacy (S&P'20)*, 2020.

[102] Stephan van Schaik, Andrew Kwong, Daniel Genkin, and Yuval Yarom. SGAxe: How SGX fails in practice. https://sgaxeattack.com/, 2020.

[103] Stephan van Schaik, Alex Seto, Thomas Yurek, Adam Batori, Bader AlBassam, Christina Garman, Daniel Genkin, Andrew Miller, Eyal Ronen, and Yuval Yarom. SoK: SGX.Fail: How stuff get eXposed. 2022.

[104] Xinpeng Yang, Meng Hao, Chenkai Weng, Robert H. Deng, Yonggang Wen, and Tianwei Zhang. Efficient fuzzy private set intersection from secret-shared oprf, 2026.

[105] Cong Zhang, Yu Chen, Yang Cao, Yujie Bai, Shuaishuai Li, Juntong Lin, Anyu Wang, and Xiaoyun Wang. Fast fuzzy PSI from symmetric-key techniques. Cryptology ePrint Archive, Paper 2025/885, 2025.