Supporting Private and Efficient Consumer Requests to Data Brokers

Elina van Kempen
University of California, Irvine
Irvine, California, USA
evankemp@uci.edu

Pavel Frolikov
University of California, Irvine
Irvine, California, USA
pavel@uci.edu

Chloe Georgiou
University of California, Irvine
Irvine, California, USA
cgeorgio@uci.edu

Gene Tsudik
University of California, Irvine
Irvine, California, USA
gtsudik@ics.uci.edu

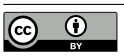
Abstract

Under data privacy regulations, such as GDPR and CCPA, consumers are legally entitled to access, delete, or opt-out from collection of, personal information that so-called data brokers collect about them. However, the process of exercising these rights remains non-uniform, inefficient, and privacy-invasive. A consumer must submit individual requests to hundreds of data brokers, each requiring different – and often extensive – identity verification procedures. In January 2026, California launched a centralized facility that allows consumers to formulate a single (unified) request to all registered data brokers. While this simplifies the process, it raises significant privacy, scalability, and security concerns. Current commercial services (e.g., DeleteMe and Incogni) automate such requests for a fee, although they require consumers to provide all personal information upfront to yet another entity of uncertain trustworthiness.

Motivated by the above, this paper presents PIVOT, an efficient cryptographic scheme that enables privacy-agile data access, deletion, and opt-out requests through an untrusted intermediary. PIVOT combines proxy re-encryption with additively homomorphic encryption so that data brokers can verify consumer identity without the intermediary learning any personally identifiable information (PII). A consumer encrypts their PII once, and the intermediary facilitates consumer identity verification with all data brokers. Data brokers learn only whether their databases contain matching records corresponding to a given consumer, without acquiring any new PII beyond what they already possess. PIVOT ensures both consumer and broker confidentiality, as well as verification soundness, requiring no user interaction after the initial request.

CCS Concepts

• **Security and privacy → Privacy-preserving protocols**; **Public key encryption**; **Privacy protections**.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WPES '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3026-9/2026/11
https://doi.org/10.1145/3847192.3847365

Keywords

Data brokers, Consumer privacy, Proxy re-encryption, Additively homomorphic encryption, Private list intersection, Identity verification, CCPA, GDPR

ACM Reference Format:
Elina van Kempen, Pavel Frolikov, Chloe Georgiou, and Gene Tsudik. 2026. Supporting Private and Efficient Consumer Requests to Data Brokers. In *25th Workshop on Privacy in the Electronic Society (WPES '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3847192.3847365

1 Introduction

Prominent current data privacy regulations, such as the European Union's General Data Protection Regulation (GDPR) [17], California Consumer Privacy Act (CCPA) [11] and Brazilian Lei Geral de Proteção de Dados Pessoais (LGPD) [20], grant consumers the right to access, delete, and correct their personal information collected by entities such as website operators and other providers. Under these regulations, the latter must verify consumer identity before processing such requests in order to prevent unauthorized disclosure of information or other potentially harmful actions.

For entities with which consumers have accounts (e.g., social media platforms or retailers), identity verification is straightforward: consumers are authenticated using their normal account credentials [11, 28]. However, whenever consumers lack accounts, e.g., those who merely browse websites or whose data is collected through third-party tracking, verification becomes much more difficult. Without account-based authentication, entities rely on matching consumer-provided Personally Identifiable Information (**PII**) against their records [9, 31, 38]. This approach varies widely across business entities, resulting in awkward and burdensome verification procedures that are fragmented and privacy-invasive for consumers.

A data broker is a very particular business entity type which collects, aggregates, and sells consumer information, usually without any direct consumer interaction. The data broker ecosystem includes both small and obscure operators as well as established (mainly US-based) credit-rating agencies, such as Experian, Transunion, Equifax, Lexis, and Acxiom [12]. They compile profiles from social media, public records (e.g., court cases, property ownerships changes, as well as car and drivers' license registrations), and even

from other brokers.[1] All of that is usually done **without** consumers' knowledge or consent [6]. Though some brokers support user accounts (e.g., Experian), the overwhelming majority does not. Also, historically, brokers are mostly an American phenomenon with roots going back to pre-Internet years – 1980-s [18].

In the context of most business types, consumers knowingly use offered services. In contrast, consumers typically do not know which brokers have their PII. Moreover, even for those brokers that do, consumers do not know the nature of that PII. Therefore, in order to comprehensively exercise their privacy rights, a consumer must submit a separate request to each broker, which can be numerous, e.g., California maintains a broker registry with well over 500 brokers [12]. Alarmingly, each request requires supplying a potentially distinct set of PII to each broker so that they can perform required-by-law identity verification. This holds even for those brokers that have no PII about the consumer. This is because a consumer has no way of knowing *a priori* whether a given broker has any PII about them.

A recent study on CCPA compliance of brokers [38] reported that one broker, after receiving extensive consumer PII for verification purposes, responded with:

"...[W]e never had this information until receiving it in your email below..."

Thus, the act of exercising privacy rights can itself lead to new data collection by entities that previously held no PII about the consumer. Furthermore, this is not a one-time burden. Brokers continuously acquire new consumer PII. Each request provides only an ephemeral snapshot of the data that a given broker possesses. Privacy-conscious consumers must periodically repeat this burdensome process with hundreds of brokers to maintain any lasting control over their PII [38]. Several commercial services, including DeleteMe [3], Incogni [34], and Aura [8], offer to submit data deletion requests on behalf of consumers to hundreds of brokers, for a monthly or yearly fee. However, these services require consumers to provide all of their PII upfront to yet another entity of uncertain trustworthiness. Storing all consumer PII at such a single intermediary elevates the risk of eventual leakage. Also, a recent study shows that these services have inconsistent broker coverage and underwhelming efficacy [21].

Since 2020, California Privacy Protection Agency (CPPA, recently renamed CalPrivacy) has been the California state government body tasked with CCPA enforcement. Recently (in 2023), it proposed the *Delete Request and Opt-out Platform* (DROP) [14, 16]. This is a free state-run platform that just launched in January 2026. It allows consumers to submit a single deletion or opt-out request and requires all registered California brokers to process such requests. While DROP appreciably improves consumer accessibility, several concerns remain. **First,** DROP only handles deletion and opt-out requests, while information access requests are not supported. **Second,** DROP treats (stores, etc.) consumer identifiers as hashes, which represents insufficient privacy protection for low-entropy consumer data, such as names, dates-of-birth, as well as physical and email addresses. **Third,** neither DROP nor aforementioned commercial services address one fundamental problem: during identity verification, brokers may learn new PII even about consumers whose data they do not possess.

As long as brokers continue to operate at scale, consumers need an effective, efficient, and privacy-agile means of exercising their legal privacy rights without incurring heavy burden. An ideal solution would minimize the amount of time and effort consumers would spend navigating complex verification procedures. To ensure widespread adoption and practicality, such a system must also avoid imposing excessive operational overhead on brokers.

Motivated by the above, this paper introduces PIVOT, a cryptographic scheme that enables privacy-preserving identity verification for accountless consumer data requests. Recall that the overwhelming majority of Verifiable Consumer Requests (VCR-s) in the context of brokers correspond to accountless consumers.

PIVOT provides a general framework adaptable to any jurisdiction requiring identity verification for data access, deletion, or opt-out requests. In it, a consumer makes a single request to an intermediary which, in turn, facilitates identity verification with all registered brokers. Notably, this intermediary learns no PII about the consumer, while brokers learn no new PII beyond what they already possess. PIVOT supports all three types of consumer requests: access, deletion, and opt-out. At the core of PIVOT, one important feature is a novel combination of (surprisingly practical) proxy re-encryption and additively homomorphic encryption techniques.

Anticipated Contributions:

- **Unified Consumer Effort:** Consumers provide their PII to the intermediary **once**. This process supports any number of brokers without any further interactions involving the consumer.
- **Strong Privacy Guarantees:** PIVOT ensures that: (1) the intermediary learns no consumer PII during the verification process, and (2) brokers learn only whether their databases contain matching records. No new PII is acquired beyond what they already possess.
- **Comprehensive Request Support:** PIVOT supports all three request types: delete, opt-out, and information access.
- **Open-Source Implementation:** A publicly available implementation of PIVOT [39], evaluated in Section 8.

Organization: Section 2 provides the background and overviews related work. Section 3 introduces our cryptographic building blocks. Section 4 describes PIVOT preliminaries, followed by Section 5 which presents the complete scheme. Next, Section 6 provides security analysis. Section 7 describes the implementation, Section 8 addresses its performance, and Section 9 discusses extensions and variants.

2 Background & Related Work

2.1 Privacy-invasive Identity Verification

Recent work demonstrates that the identity verification process in handling data requests by business entities is usually privacy-invasive [9, 10, 22, 27, 31, 33, 35, 36, 38, 40], consisting in matching consumer-provided PII against their records. In particular, when making data requests to popular website operators, sensitive PII is often required, such as copies of a government-issued ID or a

[1] Hereafter we use "brokers " instead of "data brokers" for brevity.

signed affidavit [9, 10, 22, 31, 40]. The same kind of PII is requested when dealing with smartphone applications [22, 27, 33].

A recent study evaluated the data request process of People Search Websites [35], a subclass of brokers that make people's PII easily searchable online. It was found that brokers often require consumers to provide additional information for identification, including phone numbers as well as email and physical addresses. On a similar note, [38] made CCPA data access requests to all 500+ California-registered brokers and discovered that, while various consumer PII elements are required for identity verification, very few brokers actually possess data about the requesting consumer. Thus, most brokers in fact learned new PII as a result of a consumer request. In addition, many brokers never responded to requests [38], in blatant violation of the CCPA. All this increases the threats of subsequent data misuse by untrustworthy and/or non-compliant brokers.

2.2 Commercial and Government Solutions

As mentioned in Section 1, consumers currently need to make individual requests to all brokers – on a regular basis – in order to stay in control of brokers' sharing of their PII. Since this is an extremely time-consuming and burdensome task, several commercial services emerged, e.g., DeleteMe [3], Incogni [34], and Aura [8]. They market a seemingly simple and appealing approach: a consumer supplies their PII just once to this service, which in turn promises to submit deletion requests to all brokers, on that consumer's behalf.

In practice, these removal services require plaintext PII, such as: full name, address, email, phone, date of birth, and copies of identity documents. They store this PII and forward it to brokers, where verification is, as usual, performed by matching the given PII elements against a specific broker's records. While this is a convenient (albeit not free and potentially expensive) solution for consumers, it is still very privacy-invasive since all PII is available to the removal service, which becomes a single point of failure, i.e., an attack could leak PII elements of great multitudes of consumers.

In 2023, California voted to create DROP, a state-run broker opt-out and removal platform, following the Delete Act [14]. This platform, just launched in January 2026, is free and only requires the consumer to provide various identifiers once. Each registered broker has to access the platform with a frequency of at most 45 days in order to retrieve a list of consumers to remove from (or opt out of) its databases. The Delete Act states [14]:

"...[T]he accessible deletion mechanism shall permit a consumer to securely submit information in one or more privacy-protecting ways..."

Nonetheless some important questions remain: (1) *will PII be stored privately?* (2) *how will identity verification be performed?* and (3) *how will consumer PII be shared with brokers?* According to proposed regulations, lists of hashed identifiers will be stored and shared with brokers [16]. However, no details are currently provided regarding the identity verification process by the platform, or the hash function(s) to be used [16]. If no *salt* is used in hashing, even hashed identifiers are easily guessable due to their low entropy [32].

2.3 Privacy-Enhancing Techniques

Prior research yielded some privacy-preserving methods to mitigate privacy risks of current identity verification methods. For example, Adhatarao et al. [4] attempted to identify themselves using only an IP address and no other PII. However, all such requests were denied.

Both Boniface et al. [9] and Jordan et al. [25] proposed using consumer-generated "cookies" during web sessions. In the former, a cookie includes the consumer's email address and public key, which are then used by a server during identity verification time to send a challenge. The latter proposes a private-public key-pair generated at each web session by a trusted execution environment (TEE) on the consumer device. The public key is given to the server at the beginning of the session, and the private key is used to sign future data requests. However, both methods do not work for entities with which the consumer has no interaction history, such as brokers.

Another method proposed by Hwang et al. [24] enables privacy-preserving identity verification for accountless consumers using Private List Intersection (PLI). PLI is a variant of Private Set Intersection (PSI): it allows two parties to compute the intersection of their respective labeled and ordered lists without learning elements outside of the intersection. A consumer executes PLI with each broker which can verify a consumer's identity using the PII elements it already has in its database, while learning no new PII about the consumer. Although [24] provides privacy-preserving authentication, it is inconvenient for the consumer who must perform the same PLI-based process with every broker.

Although PIVOT also uses PLI, it provides a streamlined and privacy-preserving identity verification process which is significantly less burdensome for consumers.

Delegated PSI is a class of PSI protocols where the "cloud" privately and obliviously computes the intersection of two parties' sets. This way, computation overhead for the two parties is greatly reduced, as compared to regular PSI.

It appears that [26] is the first to propose a delegated PSI protocol, based on RSA encryption with quadratic complexity. [29] presented one based on symmetric key encryption, where both parties and the cloud must be online throughout. More recent work uses Bloom filters [1, 2], polynomial encodings [23], or PRF-s [41]. These protocols either lack in efficiency [23], or require users to directly communicate with each other [1, 2, 41].

[42] is the approach most similar to the one in this paper. It uses proxy re-encryption to securely compute the PSI. However, that scheme is limited in its privacy properties, since it makes plaintext guessing attacks easy. In the context of relatively low-entropy consumer PII, guessing attacks must be mitigated. Also, the setting of [42] is the case when two peer parties want to compute a PSI with each other, aided by the cloud. In contrast, we focus on a setting where a consumer wants to securely and efficiently compute **many PSI instances with many brokers**.

3 Building Blocks

Throughout the paper, we denote $x^{(\alpha)}$ and $x^{(\beta)}$ the first and the second value of a tuple $x = (a, b)$, i.e. $x^{(\alpha)} = a$ and $x^{(\beta)} = b$.

3.1 Bilinear pairings

A map $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map if $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups with the same prime order q and if it satisfies the following properties:
(1)Bilinearity: $\forall\ a, b \in \mathbb{Z}_q$ and $g \in \mathbb{G}_1, e(g^a, g^b) = e(g, g)^{ab}$.
(2)Non-degeneracy: With g a generator of $\mathbb{G}_1$, $e(g, g) \neq 1_{\mathbb{G}_2}$.
(3)Computability: The pairing operation e is efficient.

3.2 Private List Intersection (PLI)

PLI is a variant of Private Set Intersection (PSI) where values are labeled and ordered [24]. Since in PLI one-to-one comparison of list elements is possible, computation of the intersection is more straightforward than in PSI. The ideal functionality $\mathcal{F}_{\mathsf{PLI}}$ with inputs from a sender S (in this context, the consumer) and a receiver R (in this context, a broker) is defined in Figure 1.

Ideal Functionality of PLI
Parameters: List size n
Ideal Functionality of PLI:
nosep Wait for inputs $Y = (y_1, y_2, ..., y_n)$ from receiver R (in PIVOT, a broker) and $X = (x_1, x_2, ..., x_n)$ from sender S (in PIVOT, a consumer)
nosep Compute the list intersection $\text{out}_R = X \cap Y$
nosep Send $\perp$ to S and send out_R to R

Figure 1: Idealized PLI Functionality: $\mathcal{F}_{\mathsf{PLI}}$ from [24]

3.3 Proxy re-encryption (PRE)

Proxy re-encryption (PRE) is a cryptographic primitive that allows a designated proxy to convert a ciphertext encrypted for Alice to a ciphertext encrypted for Bob, given a special PRE key provided by Alice. During PRE, the proxy does not learn the plaintext. A unidirectional PRE scheme ensures that PRE can only be done from Alice to Bob – and not from Bob to Alice – with the same PRE key.

PIVOT builds on the unidirectional PRE scheme from [7], which is based on the bilinear Diffie-Hellman assumption. The original scheme is presented below, with modifications described in Definition 5.1 to provide additive homomorphism.

Definition 3.1 (Unidirectional Proxy re-encryption [7])**.** This unidirectional PRE scheme consists of a tuple of six algorithms:
$(\mathsf{KeyGen}, \mathsf{ReEncKeyGen}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{ReEnc}, \mathsf{Dec})$
Global parameters are (g, Z) where $g \in \mathbb{G}_1$ and $Z = e(g, g)$:
[PRE$_1$] $\mathsf{KeyGen}\ (g, Z) \rightarrow (sk_A, pk_A)$: A user's A secret key is of the form $sk_a = (a_1, a_2)$, and its corresponding public key is $pk_A = (Z^{a_1}, g^{a_2})$.
[PRE$_2$] $\mathsf{ReEncKeyGen}\ (a_1, g^{b_2}) \rightarrow rk_{A \rightarrow B}$: A user A delegates decryption to a user B by publishing the PRE key $rk_{A \rightarrow B} = (g^{b_2})^{a_1} = g^{a_1 b_2}$, computed from B's public key and A's secret key.
[PRE$_3$] $\mathsf{Enc}_1\ (pk_A, m) \rightarrow c_{a,1}$: To encrypt a message $m \in \mathbb{G}_2$ such that it can only be decrypted by user A (first-level encryption), pick random $r \in \mathbb{Z}_q$ and output ciphertext $c_{a,1} = (Z^{a_1 r}, mZ^r)$. To achieve proxy invisibility, i.e. it is not possible to determine whether the ciphertext was re-encrypted, output $c_{a,2} = (Z^{a_2 r}, mZ^r)$ at no security loss.
[PRE$_4$] $\mathsf{Enc}_2\ (Z^{a_1}, m) \rightarrow c_{a,R}$: To encrypt a message $m \in \mathbb{G}_2$ such that it can re-encrypted to A's delegates (second-level encryption), pick random $r \in \mathbb{Z}_q$ and output ciphertext $c_{a,R} = (g^r, m(Z^{a_1})^r) = (g^r, mZ^{a_1 r})$.
[PRE$_5$] $\mathsf{ReEnc}\ (c_{a,R}, rk_{A \rightarrow B}) \rightarrow c_{b,1}$: Given $rk_{A \rightarrow B}$, one can change a second-level ciphertext destined to A into a first-level ciphertext destined to B. From $c_{a,R} = (g^r, mZ^{ar})$, compute $e(g^r, rk_{A \rightarrow B}) = Z^{r a_1 b_2}$. Output $c_{b,1} = (Z^{r a_1 b_2}, mZ^{a_1 r}) = (Z^{b_2 r'}, mZ^{r'})$ where $r' = a_1 r$.
[PRE$_6$] $\mathsf{Dec}\ (sk_a, c_{a,i}) \rightarrow m$: To decrypt a first-level ciphertext $c_{a,i} = (c_{a,i}^{(\alpha)}, c_{a,i}^{(\beta)})$ where $i \in [1, 2]$, compute $m = \frac{c_{a,i}^{(\beta)}}{(c_{a,i}^{(\alpha)})^{1/a_i}}$, to decrypt a second-level ciphertext $c_{a,R} = (c_{a,R}^{(\alpha)}, c_{a,R}^{(\beta)})$, compute $m = \frac{c_{a,R}^{(\beta)}}{e(c_{a,R}^{(\alpha)}, g)^{a_1}}$.

Definition 3.2 (Extended Decisional Bilinear Diffie–Hellman (eDBDH) assumption)**.** Let $\mathbb{G}_1$ be a cyclic group of prime order q with generator g, $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ bilinear, and $Z := e(g, g)$. Choose $a, b, c, d \leftarrow \mathbb{Z}_q$ uniformly at random. Given: $\left(g,\ g^a,\ g^b,\ g^c,\ Z^{bc^2},\ Z^d\right)$, decide whether $d \equiv abc \pmod q$. The eDBDH assumption says that no PPT adversary can distinguish $(g, g^a, g^b, g^c, Z^{bc^2}, Z^{abc})$ from $(g, g^a, g^b, g^c, Z^{bc^2}, Z^t)$ for uniform $t \in \mathbb{Z}_q$ with non-negligible advantage [7].

Theorem 3.3 (PRE security)**.** *Under the eDBDH assumption in $(\mathbb{G}_1, \mathbb{G}_2, e)$, re-encrypted ciphertexts produced by the scheme are secure. Moreover, under the Discrete Log assumption in G, recovering a user's secret key from delegations is infeasible. Hence, the scheme is secure as long as eDBDH and Discrete Log assumptions for the secret key hold.*

3.4 Additively Homomorphic Encryption

An additively homomorphic encryption (AHE) scheme supports arithmetic addition of multiple ciphertexts, i.e., an AHE scheme has the property that $\mathsf{Enc}(k, x) \oplus \mathsf{Enc}(k, y) = \mathsf{Enc}(k, x + y)$, where $\mathsf{Enc}()$ is the encryption function, $\oplus$ is some operator, and k is an encryption key.

4 System and Threat Models

Notation used in the rest of the paper is summarized in Table 1.

4.1 Three-Party Setting

Prior PLI methods assume direct communication between a consumer (C) and a broker ($\mathcal{BR}_j$). PIVOT extends this to a three-party setting where an intermediary ($\mathcal{MED}$) mediates between C and brokers. C submits their PII once to $\mathcal{MED}$, and the latter orchestrates identity verification with every broker on C's behalf, while learning no C PII.

4.2 Participants and Threat Model

All brokers and $\mathcal{MED}$ are assumed to be honest-but-curious, i.e., they follow the rules, though they may attempt to learn PII beyond what is necessary. This assumption stems from real-world constraints: $\mathcal{MED}$, whether a government-provided service or a paid service, is trusted by the consumers, and it is impossible to

Table 1: Notation Summary

Notation	Description
Parties	
C	Consumer
$\mathcal{MED}$	Intermediary
$\mathcal{BR}$	Set of all registered brokers
$\mathcal{BR}_j$	A specific broker j
Consumer Identifiers and Data	
ID_C	C's public identifier (e.g., email, name+DOB)
$\mathcal{P}_C$	C's complete list of PII attributes $(x_1, x_2, \ldots, x_N)$
$\mathcal{P}_{C_j}$	C's list of PII attributes $(x_1, x_2, \ldots, x_n)$ for IV with $\mathcal{BR}_j$
Broker Data	
$\mathcal{P}_{\mathcal{BR}_j}$	$\mathcal{BR}_j$'s list of PII attributes $(y_1, y_2, \ldots, y_n)$ for a C
Labels and Metadata	
$\mathsf{L}_{\mathcal{BR}}$	Union of all PII field labels across all $\mathcal{BR}$
$\mathsf{L}_{\mathcal{BR}_j}$	Set of PII field labels required by $\mathcal{BR}_j$ for IV
ℓ_i	Label of PII element i
Size Parameters	
m	Size of $\mathcal{BR}$ (total # of brokers)
N	Size of $\mathsf{L}_{\mathcal{BR}}$ (# of unique PII labels across $\mathcal{BR}$)
n	Size of $\mathcal{P}_{\mathcal{BR}_j}$ (number of PII elements in a $\mathcal{BR}_j$'s list)
$[x]$	Set of integers $\{1, 2, \ldots, x\}$
Cryptographic Primitives	
g	Generator of cyclic group $\mathbb{G}_1$
Z	Equals $e(g, g)$ where e is the bilinear pairing
e	Bilinear pairing map
Keys	
sk_A	Secret key of party A; format: (a_1, a_2)
pk_A	Public key of party A; format: (Z^{a_1}, g^{a_2})
$rk_{A\to B}$	PRE key from party A to party B
Ciphertexts	
c_i	C's ciphertext of $-x_i$ encrypted under consumer's key
$\tilde{c}_i$	$\mathcal{BR}_j$'s ciphertext of y_i encrypted under $\mathcal{BR}_j$'s key
a_i	Combined and blinded ciphertext sent to $\mathcal{BR}_j$ by $\mathcal{MED}$
d_i	$\mathcal{BR}_j$'s partially decrypted result; equals $1_{\mathbb{G}_2}$ if $x_i = y_i$

verify whether a broker is telling the truth without audits. We consider collusion to be out of scope since entities can communicate consumer data outside PIVOT. An extension of PIVOT that tolerates malicious brokers and $\mathcal{MED}$ is sketched out in Section 9.3. On the other hand, it is reasonable to assume that C might be malicious and attempt to bypass identity verification by providing fake or manipulated PII. In more detail, the parties play the following roles:

(C) Consumer: wants to submit a data request (of type: delete, opt-out, or access) to every broker.[2] For simplicity's sake, we assume that C makes the same type of request to all brokers. C interacts only with the intermediary, and never contacts brokers directly. C is assumed to have a labeled list of their own PII $\mathcal{P}_C$ and a unique public identifier ID_C. Sharing ID_C in plaintext is a functional requirement rather than a design choice: a verifiable consumer request must identify the individual it concerns, and a broker must locate candidate records before any verification can take place (Section 4.3) [11]. Current practices reflect this, e.g., DROP shares consumer identifiers with all registered brokers as hashes [16], which, given their low entropy, offer marginal protection over plaintext [32]. The privacy downside is that a broker that has no records for C Alice nonetheless learns ID_C. It thus learns that Alice exists and is exercising her privacy rights. Section 9.2 presents a variant of PIVOT that avoids this leakage by ensuring that only brokers that have ID_C receive the request.

PIVOT ensures that C learns nothing about broker datasets beyond what brokers explicitly return. Identity theft, whereby an adversary impersonates C using stolen PII, is out of scope: it defeats *any* verification scheme based on PII matching (including direct requests, commercial removal services, and DROP), and PIVOT neither helps nor hurts such an adversary.

($\mathcal{MED}$) Intermediary: acts as a proxy between consumers and brokers. Its responsibilities include receiving deletion, opt-out, or access requests from C and contacting brokers on C's behalf. $\mathcal{MED}$ handles all identity verification procedures for C. $\mathcal{MED}$ learns ID_C and, for each broker, only a single bit: whether that broker's database contains any record for ID_C (plus the final status of each request). Beyond ID_C, no consumer PII is ever revealed to $\mathcal{MED}$. This role resembles current commercial services, such as DeleteMe or Incogni, which submit multiple deletion requests on consumers' behalf. As an honest-but-curious entity, $\mathcal{MED}$ may attempt to learn consumer PII during interactions with C and brokers.

($\mathcal{BR}$) Brokers: We use $\mathcal{BR}$ to denote the set of all brokers of size m, while $\mathcal{BR}_j$ is a specific broker $\in \mathcal{BR}$. A given $\mathcal{BR}_j$ has a database of consumer PII. We denote $\mathcal{P}_{\mathcal{BR}_j}$ the list of PII elements corresponding to a specific C. A $\mathcal{BR}_j$ must declare the exact types of PII elements $\mathsf{L}_{\mathcal{BR}_j}$ that it requires for consumer identity verification purposes.

Upon receiving a request from $\mathcal{MED}$ for C, $\mathcal{BR}_j$ performs identity verification. If the identity verification succeeds, $\mathcal{BR}_j$ processes C's request:

- *Deletion:* remove C's data.
- *Opt-out:* prevent future collection or sale of C's data.
- *Access:* return all consumer data currently held.

Brokers may attempt to learn new consumer PII beyond what they already possess.

4.2.1 Scope. We do not consider Denial-of-Service (DoS) attacks whereby $\mathcal{MED}$ refuses to forward requests or brokers ignore them. We also do not attempt to enforce the veracity of broker responses. A broker can always claim that ID_C is not in its database, or report an empty intersection during identity verification. Ensuring that brokers faithfully process requests is therefore a compliance and enforcement concern. Our focus is on C privacy and minimizing C interaction when exercising their privacy rights.

4.3 Assumptions

Communication Channels: All communication between participants occurs over authenticated and encrypted channels, e.g., TLS/HTTPS.

Broker Registration: A list of brokers is publicly available, e.g., by legally mandated registration, e.g., as in CCPA. Each broker declares which PII elements are needed to verify consumer identity and publishes this along with its public key.

Public Identifier: Similar to the proposed CPPA DROP system, each C has a public identifier ID_C (at least based on their name and email address) that is shared in plaintext. $\mathcal{MED}$ forwards ID_C to brokers, who use it to search their databases before performing

[2] Consumers retain the ability to contact brokers individually through traditional request mechanisms.

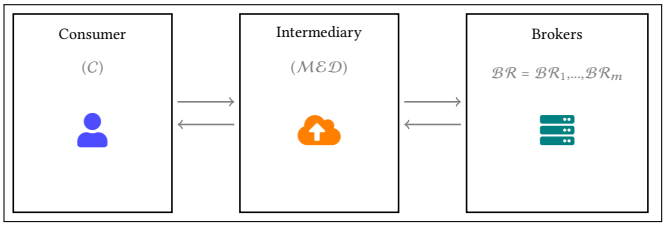


Figure 2: PIVOT **scheme Participants.**

identity verification. In Section 9, we provide a protocol extension to prevent a broker from learning ID_C if C is not in broker's database.

Asynchronous Operation: $\mathcal{MED}$ maintains state for each C, tracking the status of their request with each broker. The timing of brokers' responses will vary, e.g., CCPA and GDPR allow up to 90 days (including extensions) to process consumer requests. For **Access** requests, $\mathcal{MED}$ also stores encrypted data returned by brokers until the consumer retrieves it.[3]

Identity Verification Requirements: Under CCPA, entities must verify that a consumer making a data request is the individual about whom the business has collected PII:

"...[A] request made by a consumer ..., and that the business can verify, using commercially reasonable methods, ... to be the consumer about whom the business has collected personal information.." (1798.140(ak)) [11].

In practice, verification procedures vary widely across brokers and, notably, do not involve strong identification mechanisms, such as verification of government-issued IDs. Many brokers require minimal effort: a consumer provides basic PII (e.g., name and email address) with no attempts to verify that they correspond to the individual they claim to be.

The proposed CCPA DROP system validates California residency, as stated in [16]:

"...[C]onsumers will have their California residency verified by the Agency prior to submitting a deletion request ..." (§7620(a))

Verification of California residency is separate and distinct from verifying consumer identity. Since opt-out requests do not require identity verification [28], DROP only verifies eligibility either through Login.gov [30] or by asking name, date of birth, California address, and either a phone number or an email address [15].

Identity verification remains the responsibility of each broker. Similarly, PIVOT does not prescribe or enforce any particular verification policy. $\mathcal{MED}$ can verify C's possession of its identifiers (e.g., email address, phone number) by sending an email or SMS challenge.

4.4 System Requirements

PIVOT must satisfy the following requirements:

Correctness. A legitimate (well-formed) consumer identity that corresponds to at least one record in a brokers' database must result in a successful identity verification.

[3] $\mathcal{MED}$ determines its own data retention policy based on scalability and privacy considerations. A bounded retention period (e.g., 90-180 days) balances practical deployment constraints with minimizing the period during which encrypted consumer data resides with $\mathcal{MED}$.

Consumer Efficiency. A consumer must be able to submit data requests to all brokers in one effort, ideally with a single (logical) message to the intermediary.

Privacy. Two privacy requirements must be satisfied, based on the adversary model in Section 4.2.

(1) **Consumer Privacy:**
- The intermediary does not learn consumer PII from the identity verification process.
- Brokers do not learn new consumer PII beyond what they already store in their databases.

(2) **Broker Privacy:** Both the consumer and the intermediary learn nothing about the content or characteristics of brokers's databases, aside from the content sent in responses to legitimate requests.

5 PIVOT Description

5.1 Consumer Experience

From C's perspective, PIVOT is straightforward and requires minimal effort. C begins by visiting $\mathcal{MED}$'s web portal using HTTPS, where they are presented with a form containing labeled fields for various PII elements, e.g., name, email address, phone number, date of birth, etc. This form includes the union of all PII elements that all registered brokers require for identity verification. C fills in as many fields as they choose – and/or are able – to provide and selects the desired request type: **Delete, Access**, or **Opt-out**.

Once C clicks submit, their browser performs all cryptographic operations locally. Each PII value is hashed and blinded client-side before leaving their device. The browser simultaneously generates PRE keys for each registered broker. Afterwards, the browser sends the encrypted data and all PRE keys to $\mathcal{MED}$. At this point, C's work is done. The identity verification process and all communication with brokers happens entirely through $\mathcal{MED}$. C simply waits for notifications as broker responses trickle in.

Each notification indicates whether a particular broker successfully verified C's identity and processed the request. For **Delete** and **Opt-out** requests, the notification confirms whether the action was completed or informs of verification failure. For **Access** requests, brokers encrypt C's data under C's public key and send it to $\mathcal{MED}$, where $\mathcal{MED}$ is accordingly unable to read this encrypted data. When C visits $\mathcal{MED}$'s portal to retrieve their data, their browser automatically decrypts it.

5.2 PIVOT Details

PLI enables privacy-preserving identity verification: $\mathcal{BR}_j$ can verify C's identity by checking which PII elements received match those it possesses, without learning any new PII elements [24].

To achieve PLI with the help of an oblivious $\mathcal{MED}$, we use additively homomorphic PRE. $\mathcal{MED}$ can re-encrypt ciphertexts received from C to $\mathcal{BR}_j$'s public key, allowing $\mathcal{BR}_j$ to decrypt them without $\mathcal{MED}$ learning any plaintext value. The PRE scheme proposed by Ateniese et al. [7] does not allow additive homomorphic operations, and thus is unsuitable for PIVOT. Therefore, we modify it slightly, as follows:

Definition 5.1 (Additively homomorphic proxy re-encryption)**.** This additively homomorphic unidirectional PRE scheme is a tuple of seven algorithms:

$$(\mathsf{KeyGen}, \mathsf{ReEncKeyGen}, \mathsf{Enc}_1, \mathsf{Enc}_2, \mathsf{ReEnc}, \mathsf{Add}, \mathsf{ParDec})$$

Global parameters (g, Z) where $g \in \mathbb{G}_1$ and $Z = e(g, g)$:

[$\mathbf{A}_1$] KeyGen $(g, Z) \rightarrow (sk_A, pk_A)$: A user's A secret key is of the form $sk_a = (a_1, a_2)$, and its corresponding public key is $pk_A = (Z^{a_1}, g^{a_2})$.

[$\mathbf{A}_2$] ReEncKeyGen $(a_1, g^{b_2}) \rightarrow rk_{A\rightarrow B}$: A user A delegates decryption to a user B by publishing the PRE key $rk_{A\rightarrow B} = (g^{b_2})^{a_1} = g^{a_1 b_2}$, computed from B's public key and A's secret key.

[$\mathbf{A}_3$] Enc_1 $(pk_A, m) \rightarrow c_{a,1}$: To encrypt a message $m \in \mathbb{Z}_q$ such that it can only be decrypted by user A, pick random $r \in \mathbb{Z}_q$ and output ciphertext $c_{a,1} = (Z^{a_1 r}, Z^m Z^r)$. To achieve proxy invisibility, output $c_{a,2} = (Z^{a_2 r}, Z^m Z^r)$ at no security loss.

[$\mathbf{A}_4$] Enc_2 $(Z^{a_1}, m) \rightarrow c_{a,R}$: To encrypt a message $m \in \mathbb{Z}_q$ such that it can re-encrypted to A's delegates, pick random $r \in \mathbb{Z}_q$ and output ciphertext $c_{a,R} = (g^r, Z^m (Z^{a_1})^r) = (g^r, Z^m Z^{a_1 r})$.

[$\mathbf{A}_5$] ReEnc $(c_{a,R}, rk_{A\rightarrow B}) \rightarrow c_{b,1}$: Given $rk_{A\rightarrow B}$, one can change a second-level ciphertext destined to A into a first-level ciphertext destined to B. From $c_{a,R} = (g^r, Z^m Z^{ar})$, compute $e(g^r, rk_{A\rightarrow B}) = Z^{r a_1 b_2}$. Output $c_{b,1} = (Z^{r a_1 b_2}, Z^m Z^{a_1 r}) = (Z^{b_2 r'}, Z^m Z^{r'})$ where $r' = a_1 r$.

[$\mathbf{A}_6$] Add $(c_1, c_2) \rightarrow (c_{12})$: To add two first-level ciphertexts encrypted under the same key[4] $c_1 = (c_1^{(\alpha)}, c_1^{(\beta)})$ and $c_2 = (c_2^{(\alpha)}, c_2^{(\beta)})$, compute $c_{12} = (c_1^{(\alpha)} \cdot c_2^{(\alpha)}, c_1^{(\beta)} \cdot c_2^{(\beta)})$.

[$\mathbf{A}_7$] ParDec $(sk_a, c_{a,i}) \rightarrow m$: To partially decrypt a first level ciphertext $c_{a,1} = (c_{a,1}^{(\alpha)}, c_{a,1}^{(\beta)})$, compute $Z^m = \frac{c_{a,1}^{(\beta)}}{(c_{a,1}^{(\alpha)})^{1/a_1}}$, to partially decrypt a first-level proxy-invisible or re-encrypted ciphertext to user A: $c_{a,2} = (c_{a,2}^{(\alpha)}, c_{a,2}^{(\beta)})$, compute $Z^m = \frac{c_{a,2}^{(\beta)}}{(c_{a,2}^{(\alpha)})^{1/a_2}}$, and to partially decrypt a second-level ciphertext $c_{a,R} = (c_{a,R}^{(\alpha)}, c_{a,R}^{(\beta)})$, compute $Z^m = \frac{c_{a,R}^{(\beta)}}{e(c_{a,R}^{(\alpha)}, g)^{a_1}}$.

Remark. Our introduction of an additively homomorphic operation (**[$\mathbf{A}_6$]** Add), whereby the plaintext becomes the exponent of Z, changes decryption semantics. Direct decryption becomes computationally infeasible, since it would require solving an instance of the bilinear discrete logarithm problem. However, testing for zero (i.e., whether the encrypted plaintext $m = 0$) is efficient, doable by simply checking whether $Z^m = 1_{\mathbb{G}_2}$. PIVOT takes advantage of this exact property.

5.2.1 Technical Overview. $\mathcal{C}$ contacts $\mathcal{MED}$ to request service and sends its plaintext identifier ID_C. After receiving ID_C, $\mathcal{MED}$ composes a union $\mathcal{U}$ of all PII types required by $\mathcal{BR}$ in their identity verification processes.

Since every possible needed PII element is included in $\mathcal{U}$, $\mathcal{C}$ can provide all needed PII only once. Along with the PII, $\mathcal{MED}$ provides a list of all $\mathcal{BR}$ public keys. $\mathcal{C}$ encrypts (Enc_2) the PII elements under *its own* public key, and computes a PRE key for each $\mathcal{BR}_j$. All these values are sent to $\mathcal{MED}$ in a single logical message.

We note that computing a PRE key for each $\mathcal{BR}_j$ represents a non-negligible burden for $\mathcal{C}$, though it is much more efficient than encrypting all PII values under each brokers key: $O(nm)$ vs $O(n+m)$. This overhead is quantified in Section 8.

At this point, $\mathcal{MED}$ has all $\mathcal{C}$'s PII elements as second-level ciphertexts (which it cannot decrypt) and PRE keys from $\mathcal{C}$ for each $\mathcal{BR}_j$ in $\mathcal{BR}$. $\mathcal{MED}$ stores these values and proceeds with contacting each $\mathcal{BR}_j$ on $\mathcal{C}$'s behalf. After checking that a specific $\mathcal{BR}_j$ has the specific ID_C in its database, $\mathcal{MED}$ re-encrypts $\mathcal{C}$'s ciphertexts under $\mathcal{BR}_j$'s public key. Next, $\mathcal{MED}$ and $\mathcal{BR}_j$ proceed with PLI computation.

PLI operates as follows:

(1) $\mathcal{BR}_j$ encrypts every encrypted PII element that it has for $\mathcal{C}$ under its own public key $pk_{\mathcal{BR}_j}$, and sends the resulting ciphertext vector to $\mathcal{MED}$.
(2) $\mathcal{MED}$ homomorphically subtracts each PII element encrypted by $\mathcal{BR}_j$ from the corresponding $\mathcal{C}$ ciphertext. Recall that the latter has been already proxy re-encrypted for this $\mathcal{BR}_j$. If the plaintext of the two respective PII elements match, the result is an encryption of zero under $pk_{\mathcal{BR}_j}$, which $\mathcal{BR}_j$ can easily test.
(3) To prevent $\mathcal{BR}_j$ from solving a discrete logarithm problem (e.g., for small-value plaintexts) and thus learning non-matching PII elements of $\mathcal{C}$, $\mathcal{MED}$ re-randomizes each encrypted subtraction by exponentiating it to a unique random value. $\mathcal{MED}$ then sends a vector of all re-randomized ciphertexts to $\mathcal{BR}_j$.
(4) $\mathcal{BR}_j$ computes the PLI result by testing whether each ciphertext is an encryption of 0. It thus learns the intersection of its list of PII elements (corresponding to $\mathcal{C}$) with the list provided by $\mathcal{C}$. $\mathcal{BR}_j$ can not decrypt any non-matching elements. Thus, it obtains no PII about $\mathcal{C}$ beyond what it already possesses. Finally, $\mathcal{BR}_j$ uses the intersection to verify $\mathcal{C}$'s identity, according to its own policy.

5.3 PRE Scheme Description

We now describe using the additively homomorphic PRE scheme corresponding to Definition 5.1 above. It is also summarized in Figure 3.

Setup: Each $\mathcal{BR}_j$ must register with a relevant authority and provides a public key $pk_{\mathcal{BR}_j} \leftarrow \mathsf{KeyGen}(g, Z)$. $\mathcal{MED}$ accesses the registry and retrieves all public keys. $\mathcal{MED}$ also retrieves from each $\mathcal{BR}_j$ the list of PII elements $\mathsf{L}_{\mathcal{BR}_j}$ it needs for consumer identity verification purposes and computes its union $\mathsf{L}_{\mathcal{BR}} = \bigcup_{\mathcal{BR}_j \in \mathcal{BR}} \mathsf{L}_{\mathcal{BR}_j}$. $\mathcal{MED}$ publishes both $\{pk_{\mathcal{BR}_j}\}_{\mathcal{BR}_j \in \mathcal{BR}}$ and $\mathsf{L}_{\mathcal{BR}}$ on its website.

Phase 1: $\mathcal{C}$ One-time Data Transmission:

[P1.1]: $\mathcal{C}$ retrieves $\{pk_{\mathcal{BR}_j}\}_{\mathcal{BR}_j \in \mathcal{BR}}$, and $\mathsf{L}_{\mathcal{BR}}$ from $\mathcal{MED}$ (e.g., through HTTP GET).

[P1.2]: For each label $\ell_i \in \mathsf{L}_{\mathcal{BR}}$, $\mathcal{C}$ encrypts each corresponding PII element x_i using the second-level encryption algorithm Enc_2 such that $c_i = \mathsf{Enc}_2(Z^{a_1}, -x_i) = (g^{r_i}, Z^{-x_i} Z^{a_1 r_i})$ where $r_i \xleftarrow{\$} \mathbb{Z}_q$ for all $i \in [N]$ with $((a_1, a_2), (Z^{a_1}, g^{a_2})) = (sk_C, pk_C) \leftarrow \mathsf{KeyGen}(g, Z)$ and where N is the size of $\mathsf{L}_{\mathcal{BR}}$. $\mathcal{C}$ also prepares zero-knowledge proofs $\pi_{c_i} = PoK\{r_i, -x_i : c_i = (g^{r_i}, Z^{-x_i} Z^{a_1 r_i})\}$. $\mathcal{C}$ computes PRE keys $rk_{\mathcal{C}\rightarrow\mathcal{BR}_j} \leftarrow \mathsf{ReEncKeyGen}(sk_C^{(\alpha)}, pk_{\mathcal{BR}_j}^{(\beta)}) = (pk_{\mathcal{BR}_j}^{(\beta)})^{sk_C^{(\alpha)}}$ for each $\mathcal{BR}_j \in \mathcal{BR}$, along with proofs $\pi_{\mathcal{BR}_j} =$

[4] E.g., a re-encrypted ciphertext can be added to a proxy-invisible, first-level ciphertext.

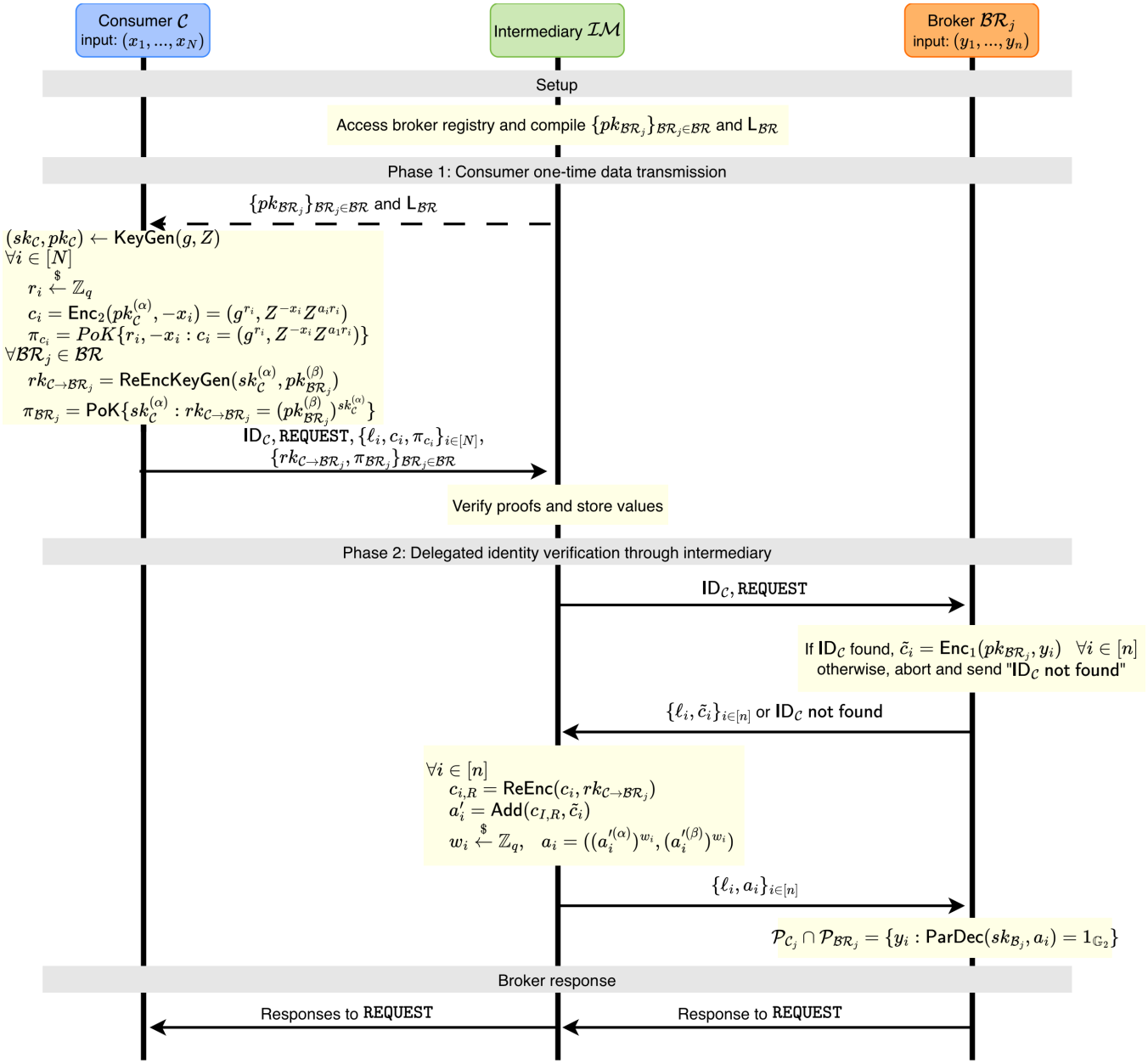


Figure 3: PIVOT **Scheme.**

$PoK\{sk_{\mathcal{C}}^{(\alpha)} : rk_{\mathcal{C} \to \mathcal{BR}_j} = (pk_{\mathcal{BR}_j}^{(\beta)})^{sk_{\mathcal{C}}^{(\alpha)}}\}$. $\mathcal{C}$ selects its request type `REQUEST` to be either `Access`, `Opt-out` or `Delete`, and sends $(\mathsf{ID}_{\mathcal{C}}, \mathsf{REQUEST}, \{\ell_i, c_i, \pi_{c_i}\}_{i \in [N]}, \{rk_{\mathcal{C} \to \mathcal{BR}_j}, \pi_{\mathcal{BR}_j}\}_{\mathcal{BR}_j \in \mathcal{BR}})$ to $\mathcal{MED}$. Thus, $\mathcal{MED}$ is unable to decrypt and learn anything about $\mathcal{C}$'s elements, yet it can later re-encrypt the elements under $\mathcal{BR}$' keys. We name the set of $\mathcal{C}$'s PII $\mathcal{P}_{\mathcal{C}} = \{\ell_i, c_i\}_{i \in [N]}$.

[P1.3]: $\mathcal{MED}$ verifies each proof and aborts if any proof does not verify. Otherwise, it stores all received values.

At this point, $\mathcal{C}$ has completed sending its input. $\mathcal{MED}$ will communicate on its behalf with all $\mathcal{BR}_j \in \mathcal{BR}$. The following steps are repeated for each $\mathcal{BR}_j \in \mathcal{BR}$.

Phase 2: Delegated Identity Verification:

[P2.1]: $\mathcal{MED}$ contacts $\mathcal{BR}_j$ and sends $\mathsf{ID}_{\mathcal{C}}$ and the request type: "`Access`", "`Delete`", or "`Opt-out`".

[P2.2]: If $\mathcal{BR}_j$ does not find $\mathsf{ID}_{\mathcal{C}}$ in its database, it sends "$\mathsf{ID}_{\mathcal{C}}$ `not found`" to $\mathcal{MED}$ and aborts. If $\mathsf{ID}_{\mathcal{C}}$ is in its database, $\mathcal{BR}_j$ proceeds with identity verification as follows. $\mathcal{BR}_j$ encrypts all data elements $y_i \in \mathcal{P}_{\mathcal{BR}_j}$ under its own public key such that

$\tilde{c}_i \leftarrow \mathsf{Enc}_1(pk_{\mathcal{BR}_j}, y_i) = (Z^{b_2\tilde{r}_i}, Z^{y_i}Z^{\tilde{r}_i})$ where $\tilde{r}_i \xleftarrow{\$} \mathbb{Z}_q$ for all $i \in [n]$ where n is the size of $\mathcal{P}_{\mathcal{BR}_j}$ and $Z^{b_2} = pk^{(\alpha)}_{\mathcal{BR}_j}$. $\mathcal{BR}_j$ sends $\{\ell_i, \tilde{c}_i\}_{i\in[n]}$ to $\mathcal{MED}$.

[P2.3]: $\mathcal{MED}$ receives $\{\ell_i, \tilde{c}_i\}_{i\in[n]}$ from $\mathcal{BR}_j$. $\mathcal{MED}$ retrieves $\mathcal{P}_{C_j} \subseteq \mathcal{P}_C$ such that $\{\ell_i\} \in \mathcal{P}_{C_j} = \{\ell_i\} \in \mathcal{P}_{\mathcal{BR}_j}$. $\mathcal{MED}$ re-encrypts the values provided by C under $\mathcal{BR}_j$'s key, using $rk_{C\to\mathcal{BR}_j}$, such that $c_{i,R} = \mathsf{ReEnc}(c_i, rk_{C\to\mathcal{BR}_j})$. Now, $\mathcal{MED}$ computes $a'_i = \mathsf{Add}(c_{i,R}, \tilde{c}_i) = \mathsf{Enc}_1(pk_{\mathcal{BR}_j}, -x_i + y_i)$. If a'_i is sent to $\mathcal{BR}_j$, then $\mathcal{BR}_j$ can easily check whether $x_i = y_i$ by executing ParDec and checking whether $Z^m = 1_{\mathbb{G}_2}$. However, it may also learn values x_i when $x_i \neq y_i$ after solving for the discrete logarithm and subtracting y_i, which is feasible for small exponents. To prevent this, $\mathcal{MED}$ selects a random value $w_i \in \mathbb{Z}_q$ for each element, and re-randomizes the ciphertexts: $\mathcal{MED}$ computes $a_i = ((a'^{(\alpha)}_i)^{w_i}, (a'^{(\beta)}_i)^{w_i})$. Values $\{\ell_i, a_i\}_{i\in[n]}$ are sent to $\mathcal{BR}_j$.

[P2.4]: $\mathcal{BR}_j$ receives the encrypted values and for each a_i computes $d_i = \mathsf{ParDec}(sk_{\mathcal{BR}_j}, a_i)$. For each element, if $x_i = y_i$, then $d_i = 1_{\mathbb{G}_2}$. Otherwise, d_i is random and thus $x_i \neq y_i$. $\mathcal{BR}_j$ outputs $\{y_i : d_i = 1_{\mathbb{G}_2}\}_{i\in[n]} = \mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$.

$\mathcal{BR}_j$ Response: $\mathcal{BR}_j$ uses $\mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$ to verify C's identity. $\mathcal{BR}_j$ responds to the data request as follows:

- "`Cannot verify identity`": $\mathcal{BR}_j$ sends this to $\mathcal{MED}$ if the identity verification was not successful, otherwise:
- "`Data deleted`": in response to a data deletion request.
- "`Opted-out`": in response to an opt-out request.
- In response to a data access request, $\mathcal{BR}_j$ encrypts C's data under pk_C and sends it to $\mathcal{MED}$. $\mathcal{MED}$ sends a notification to C to retrieve the data.

6 Security Analysis

PIVOT satisfies correctness, C privacy, and $\mathcal{BR}_j$ privacy during identity verification phases 1 and 2. We follow simulation-based definitions for privacy in the presence of honest-but-curious and malicious adversaries, as in [19, 24].

6.1 Correctness

PIVOT correctly executes if $\mathcal{BR}_j$ outputs the intersection of $\mathcal{MED}$'s list and $\mathcal{BR}_j$'s list $\mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$.

6.2 C Privacy

PIVOT realizes C privacy in the presence of honest-but-curious adversaries corrupting $\mathcal{MED}$ or $\mathcal{BR}_j$ if there exists a probabilistic polynomial-time simulator SIM such that [19, 24]:

$$\{\mathsf{SIM}_{\mathcal{MED},\mathcal{BR}_j}(1^\lambda, \mathcal{P}_{\mathcal{BR}_j}, \mathcal{F}_{\mathsf{PLI}}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j}))\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda}$$
$$\overset{c}{\equiv} \{view^{\Pi}_{\mathcal{MED},\mathcal{BR}_j}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j}, \lambda), output^{\Pi}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j})\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda}$$

Proof of Correctness

Assume an execution of PIVOT with honest C, $\mathcal{MED}$ and $\mathcal{BR}_j$. $\mathcal{BR}_j$ computes $d_i = \mathsf{ParDec}(sk_{\mathcal{BR}_j}, a_i)$, which is equal to $Z^{(-x_i+y_i)w_i}$ as follows:

$$\begin{aligned} d_i &= \frac{(Z^{-x_i}Z^{r_i} \cdot Z^{y_i}Z^{\tilde{r}_i})^{w_i}}{((Z^{b_2 r_i} \cdot Z^{b_2\tilde{r}_i})^{w_i})^{1/b_2}} \\ &= \frac{(Z^{-x_i+y_i}Z^{r'_i})^{w_i}}{(Z^{b_2 r'_i})^{w_i/b_2}} \qquad \text{where } r'_i = r_i + \tilde{r}_i \\ &= \frac{Z^{(-x_i+y_i)w_i}Z^{r'_i w_i}}{Z^{r'_i w_i}} \\ &= Z^{(-x_i+y_i)w_i} \end{aligned}$$

If $x_i = y_i$, then the result becomes $Z^0 = 1_{\mathbb{G}_2}$. Otherwise, the result looks random to $\mathcal{BR}_j$ as $\mathcal{BR}_j$ does not know w_i. Thus, $\mathcal{BR}_j$ correctly outputs $\mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$ by checking whether $d_i = 1_{\mathbb{G}_2}$. □

Proof of C Privacy

For honest-but-curious $\mathcal{MED}$ and $\mathcal{BR}_j$, a simulator $\mathsf{SIM}_{\mathcal{MED},\mathcal{BR}_j}$ can be constructed as follows:

Given Y and $\mathcal{F}_{\mathsf{PLI}}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j}) = \mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$, the simulator $\mathsf{SIM}_{\mathcal{MED},\mathcal{BR}_j}$:

- Generates key pair: $(sk_S, pk_S) \leftarrow \mathsf{KeyGen}(g, Z)$ where $sk_S = (s_1, s_2)$
- Receives $pk_{\mathcal{BR}_j}$ from $\mathcal{MED}$ and computes PRE key: $rk_{S\to B} \leftarrow \mathsf{ReEncKeyGen}(s_1, g^{b_2})$
- For all $i \in [n]$:
  - If $y_i \in \mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$: sets $z_i = -y_i$
  - Otherwise: sets $z_i = r$ where $r \xleftarrow{\$} \mathbb{Z}_q$
- Computes ciphertexts: $C_i = \mathsf{Enc}_2(Z^{s_1}, z_i)\ \forall i \in [n]$
- Sends $\{C_i\}_{i\in[n]}$ and $rk_{S\to B}$ to $\mathcal{MED}$

$\mathcal{MED}$'s view of the interaction with $\mathsf{SIM}_{\mathcal{MED},\mathcal{BR}_j}$ is the same as its view of its interaction with a real-world honest C, because of the security of the proxy re-encryption scheme under the eDBDH and discrete logarithm assumptions. $\mathcal{BR}_j$'s view of the interaction with $\mathsf{SIM}_{\mathcal{MED},\mathcal{BR}_j}$ is the same as its view of its interaction with a real-world honest C: in both cases, $\mathcal{BR}_j$ receives an encryption of a random value when $y_i \notin \mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$, and an encryption of 0 when $y_i \in \mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}$. □

6.3 $\mathcal{BR}_j$ Privacy

PIVOT realizes $\mathcal{BR}_j$ privacy in the presence of an honest-but-curious $\mathcal{MED}$ if there exists a probabilistic polynomial-time simulator SIM such that [19]:

$$\{\mathsf{SIM}_{\mathcal{MED}}(1^\lambda, \perp, \perp)\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda}$$
$$\overset{c}{\equiv} \{view^{\Pi}_{\mathcal{MED}}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j}, \lambda), output^{\Pi}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j})\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda}$$

and in the presence of a malicious C if, following the real/ideal paradigm of [19], for every real world adversary $\mathcal{A}$ there exists a probabilistic polynomial-time simulator SIM such that for any

input $\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j}$

$$\{\mathsf{IDEAL}_{\mathcal{F}_{\mathsf{PLI}},\mathsf{SIM}}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j})\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda} \stackrel{c}{\equiv} \{\mathsf{REAL}_{\pi,\mathcal{A}}(\mathcal{P}_{C_j}, \mathcal{P}_{\mathcal{BR}_j})\}_{\mathcal{P}_{C_j},\mathcal{P}_{\mathcal{BR}_j},\lambda} \quad (1)$$

Proof of $\mathcal{BR}_j$ Privacy

For an honest-but-curious $\mathcal{MED}$, a simulator $\mathsf{SIM}_{\mathcal{MED}}$ can be constructed as follows: $\mathsf{SIM}_{\mathcal{MED}}$ chooses n random values $(z_1, ...z_n)$ and encrypts them under pk_b, i.e. $\mathsf{SIM}_{\mathcal{MED}}$ sets $\tilde{c}_i = \mathsf{Enc}_1(Z^{b_2}, z_i)$ for all i, and sends $(\tilde{c}_1, ..., \tilde{c}_n)$ to $\mathcal{MED}$. Because of the semantic security of the proxy re-encryption scheme under the hardness assumption of eDBDH and discrete logarithm, the ciphertexts produced by $\mathsf{SIM}_{\mathcal{MED}}$ are indistinguishable from the ones produced by $\mathcal{MED}$ in the real execution.

For a malicious C, a simulator SIM is constructed as:

- SIM interacts with C and receives the values $\{c_i\}_{x_i \in X}$, $rk_{C \to \mathcal{BR}_j}$ and the proofs $\{\pi_{c_i}\}_{i \in [n]}$ and $\pi_{\mathcal{BR}_j}$. If any proof does not verify, SIM aborts.
- SIM runs the extractor algorithm and extracts $-x_i, r_i$ and $sk_C^{(\alpha)}$. SIM checks whether $sk_C^{(\alpha)}$ was used when computing $\{c_i\}_{x_i \in X}$ by verifying whether $Z^{-x_i} Z^{sk_C^{(\alpha)} r_i} = c_i^{(\beta)}$ (i.e., $a_i = sk_C^{(\alpha)}$). If the verification is successful, SIM forwards $\{x_i\}$ to the ideal functionality. Otherwise, SIM sends random values.

It is easy to see that the output of the honest $\mathcal{BR}_j$ in the real world interacting with the malicious C is the same as the output of the ideal world $\mathcal{B}\tilde{\mathcal{R}}_j$ with SIM interacting with the ideal functionality, due to the soundness of the proof of knowledge. □

7 Implementation

This section describes the implementation of PIVOT, which is publicly available at [39].

7.1 Configuration and Setup

PIVOT is implemented in Python 3.13.3 with three participants: C, $\mathcal{MED}$, and $\mathcal{BR}_j$. Communication occurs over TLS-secured TCP sockets. All communication uses Python's `socket` and `ssl` modules with mutual TLS authentication via X.509v3 certificates[5].

A separate setup script (`setup.py`) performs one-time parameter generation, creating global parameters and key pairs, which are serialized to binary files. A shared configuration module (`config.py`) loads these at runtime across all participants.

The implementation runs on a MacBook Pro with an M4 processor and 24 GB of RAM, communicating over localhost.

7.2 Cryptographic Primitives

PIVOT uses the Charm-Crypto library [5] for cryptographic primitives. Charm offers native implementations of bilinear pairings and group operations, along with built-in serialization support for cryptographic objects. Charm enables the implementation of the PRE scheme (Definition 3.1) with additively homomorphic modifications (Definition 5.1). It uses the SS512 symmetric pairing group `PairingGroup('SS512')`. For PII hashing, Charm's hash-to-group function `group.hash(pii, ZR)` is used.

[5]See https://www.ietf.org/rfc/rfc5280.txt

7.3 Consumer C

`consumer.py` (honest-but-curious) loads PII as label-value pairs (e.g., `email:user@example.com`). It encrypts each PII element via `encrypt(params, pk_c, -group.hash(pii, ZR))` and computes PRE keys using `rkeygen(params, pk_c, sk_c, pk_b)`. After establishing a TLS connection to $\mathcal{MED}$, it transmits encrypted data and PRE keys over the socket, closes the connection, and terminates.

7.4 Intermediary $\mathcal{MED}$

`intermediary.py` (honest-but-curious) orchestrates communication between consumers and brokers. Its core operations use `re_encrypt` for PRE and `add` for homomorphic ciphertext combination. Random blinding factors `group.random(ZR)` are applied to each ciphertext before transmission to brokers.

7.5 Broker $\mathcal{BR}_j$

Upon receiving the consumer identifier, `broker.py` (honest-but-curious) encrypts its PII elements using `fencrypt` and sends them to $\mathcal{MED}$. After receiving blinded ciphertexts, it performs partial decryption via `decrypt` and tests the identity element in G2.

8 Evaluation

PIVOT's performance is evaluated across all three participants. All experiments were conducted using the implementation described in Section 7. Each experiment was run 10 times and results were averaged.

Parameters: The union of PII elements N is configured to 30. A recent study of CCPA compliance [38] found that the total union of PII elements requested by all California-registered brokers is 23; we round up to overestimate and allow headroom for the innovative demands of brokers. For individual broker PII requirements, we set $n = 10$ elements per $\mathcal{BR}_j$, rounding up from the observed maximum of 8 [38]. California currently maintains a registry of 541 brokers [12]. We vary the number of registered $\mathcal{BR} = m \in \{100i : i \in [10]\}$.

Table 2: Performance with Variable # of brokers

	Execution time (ms)			Bandwidth sent/received (kB)		
$\mathcal{BR}$	100	500	1000	100	500	1000
Consumer	79	288	577	27.4/0	80.9/0	147.6/0
Intermediary	6	6	6	6.0/33.3	6.0/86.9	6.0/153.6

8.1 Consumer Performance

We first analyze C execution time by comparing PIVOT against a baseline whereby C directly encrypts PII values for each $\mathcal{BR}_j$ individually, without using $\mathcal{MED}$. Figure 4 shows C execution time as the number of $\mathcal{BR}$ varies from 200 to 1000. For 600 brokers, PIVOT achieves a **14.2**× speedup, over the naïve $O(nm)$ approach, completing in 0.34 seconds compared to 4.85 seconds. Table 2 shows

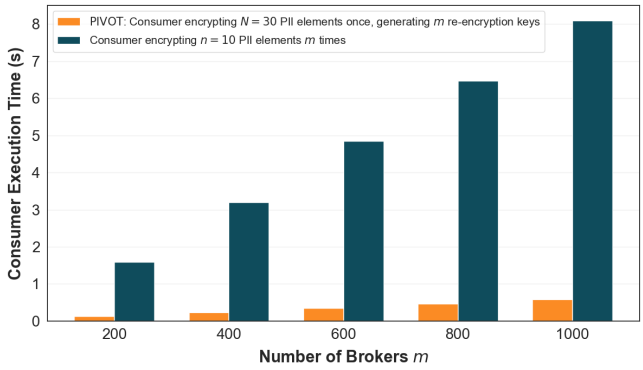


Figure 4: Runtime for consumers with PIVOT, compared to individually contacting brokers

	Execution time (ms)			Bandwidth sent/received (kB)		
# PII elements/$\mathcal{BR}_j$	10	20	30	10	20	30
Intermediary	6	12	18	6.1/86.9	11.6/92.5	17.2/98.0
Broker	8	15	23	6.0/6.1	11.6/11.6	17.1/17.2

Table 3: Performance with Variable # of PII Elements

that C execution time scales linearly with m, while analysis determined that the overhead of adding one broker is only 0.55 ms. With 500 brokers, C completes all operations in 288 ms and transmits 80.9 Kbytes. At an upload speed of 3 Mbps[6], this transmission takes approximately 0.2 seconds.

8.2 Intermediary Performance

For each C-$\mathcal{BR}_j$ pair, $\mathcal{MED}$'s processing time scales linearly with the number of PII elements requested by $\mathcal{BR}_j$: Tables 2 and 3 show 6 ms for $n = 10$, 12 ms for $n = 20$, and 18 ms for $n = 30$ regardless of the total number of registered brokers m. Since each interaction is independent, these operations can be parallelized.

Storage: $\mathcal{MED}$ must store encrypted C data. This amounts to approximately 81 Kbytes per C. Scaling to the entire US adult population of approximately 267 million [37] would require roughly 21 terabytes of storage.

Broker Performance: Similar to $\mathcal{MED}$, $\mathcal{BR}_j$ execution time scales linearly with the number of PII elements requested for identity verification. Bandwidth also scales linearly, from 6.0/6.1 Kbytes sent/received for $n = 10$ to 17.1/17.2 Kbytes for $n = 30$.

9 Extensions & Variants

This section presents three PIVOT extensions, each strengthening privacy or security guarantees at some additional cost. Each extension is independent and can be applied on its own, depending on the deployment's threat model.

[6]A conservative estimate based on California's broadband adoption data showing that the majority of households meet at least the 25/3 Mbps standard [13].

9.1 Better Privacy via PLI Variants

For better C privacy, we can use PLI variants. For instance, [24] introduces PLI-Cardinality (PLI-CA), which only outputs the cardinality of list intersection instead of matching elements. Using PLI-CA, $\mathcal{BR}_j$ would learn only the number of matching elements, thus preventing it from confirming whether its consumer PII database and data collection methods are accurate.

PLI-CA can be easily implemented in PIVOT by having $\mathcal{MED}$ remove labels and shuffle the list $\{a_i\}$ before sending its values to $\mathcal{BR}_j$. In more detail, after computing $\{\ell_i, a_i\}_{i\in[n]}$, $\mathcal{MED}$ randomly permutes $\{a_i\}_{i\in[n]}$ and does not send labels $\{\ell_i\}_{i\in[n]}$, to hide any correspondence between the output values and their order in the list. Then, $\mathcal{BR}_j$ simply outputs $|\{i : \mathrm{ParDec}(sk_{\mathcal{BR}_j}, a_i) = 1_{\mathbb{G}_2}\}| = |\{\mathcal{P}_{C_j} \cap \mathcal{P}_{\mathcal{BR}_j}\}|$ and obtains the total number of matching elements.

This approach improves privacy in scenarios where knowing which PII elements matched is not necessary for fulfilling a request, and having a high number of matching PII elements is enough to verify C's identity.

9.2 Partial C Anonymity

Recall that $\mathcal{MED}$ first sends ID_C to $\mathcal{BR}_j$, who checks whether it possesses ID_C in its database. However, in the case when $\mathcal{BR}_j$ does not, then the existence of C is revealed to $\mathcal{BR}_j$ by sharing ID_C. Preventing this would provide partial anonymity to C: only a $\mathcal{BR}_j$ that already knows ID_C will receive the data request.

CPPA's proposed DROP system [16] hashes identifiers to ensure C privacy. However, since identifiers are low entropy (e.g., names, and physical or email addresses), C privacy could be easily compromised.

Instead, we envision the following interaction between $\mathcal{MED}$ and $\mathcal{BR}_j$, starting in Phase 2 of PIVOT. $\mathcal{MED}$ compiles a list $\mathcal{L}_{\mathsf{id}} = \{\mathsf{ID}_C\}_{C\in Cs}$, where Cs are consumers using $\mathcal{MED}$'s services. $\mathcal{BR}_j$ similarly compiles a list $\mathcal{L}'_{\mathsf{id}} = \{\mathsf{ID}_{C'}\}_{C'\in Cs'}$, where Cs' are all the consumers in $\mathcal{BR}_j$'s database. $\mathcal{MED}$ and $\mathcal{BR}_j$ run a standard PSI protocol, with inputs $\mathcal{L}_{\mathsf{id}}$ and $\mathcal{L}'_{\mathsf{id}}$. The PSI protocol returns $\mathcal{L}_{\mathsf{id}} \cap \mathcal{L}'_{\mathsf{id}}$ to $\mathcal{MED}$. Thus, $\mathcal{MED}$ will only make data requests on behalf of $\{\mathsf{ID}_C\}_{C\in\mathcal{L}_{\mathsf{id}}\cap\mathcal{L}'_{\mathsf{id}}}$ to $\mathcal{BR}_j$, such that $\mathcal{BR}_j$ does not learn ID_C if C is not in its database.

Since the databases of both $\mathcal{MED}$ and $\mathcal{BR}_j$ may be very large, a single PSI execution can incur high computation and communication overhead. To lower costs, instead of doing it in real time, this process can be performed asynchronously and periodically, e.g., every day, week, or month. This follows current and proposed laws, e.g., the California DROP regulation stipulates that brokers must consult the state-run platform every 45 days.

9.3 Malicious $\mathcal{MED}$ & $\mathcal{BR}_j$

PIVOT can be easily amended to tolerate malicious $\mathcal{MED}$ or $\mathcal{BR}_j$. We define malicious behavior as any deviation from the scheme. As mentioned in Section 4.2, collusion between $\mathcal{BR}_j$ and $\mathcal{MED}$ as well as veracity of $\mathcal{BR}_j$'s final response are out of scope.

C privacy against malicious $\mathcal{MED}$. Since C only retrieves $\mathcal{BR}$ information from $\mathcal{MED}$, it suffices to include certificates along with public keys. C can v erify these certificates before proceeding.

$\mathcal{BR}_j$ privacy against malicious $\mathcal{MED}$. $\mathcal{MED}$ computes proofs of the form:

$$\pi_{a_i} = PoK\{w_i : a_i^{(\alpha)} = (Z^{b_2 r_i} Z^{b_2 \tilde{r}_i})^{w_i}\} \; \forall \; i \in [1, n]$$

and appends them to its second message to $\mathcal{BR}_j$.

C privacy against malicious $\mathcal{BR}_j$. $\mathcal{BR}_j$ appends zero-knowledge proofs of the form:

$$\pi_{\tilde{c}_i} = PoK\{\tilde{r}_i, y_i : \tilde{c}_i = (Z^{b_2 \tilde{r}_i}, Z^{y_i} Z^{\tilde{r}_i})\} \; \forall \; i \in [1, n]$$

to its message to $\mathcal{MED}$.

10 Conclusions

This paper presented PIVOT that enables efficient and privacy-preserving consumer requests to data brokers. By combining proxy re-encryption with additively homomorphic encryption, PIVOT reduces consumer effort to a single submission that scales to all registered data brokers. PIVOT guarantees strong privacy: neither the intermediary nor data brokers acquire any new consumer PII. PIVOT enables consumers to efficiently and comprehensively exercise their statutorily granted privacy rights.

Acknowledgments

We acknowledge the use of AI (Claude, ChatGPT) for debugging and grammar checking. This work was supported in part by the NSF Award SATC 1956393, titled: "ProperData: Protecting Personal Data Flow on the Internet."

References

[1] Aydin Abadi, Sotirios Terzis, and Changyu Dong. 2020. Feather: Lightweight multi-party updatable delegated private set intersection. *Cryptology ePrint Archive* (2020).
[2] Aydin Abadi, Sotirios Terzis, Roberto Metere, and Changyu Dong. 2017. Efficient delegated private set intersection on outsourced private datasets. *IEEE Transactions on Dependable and Secure Computing* Volume 16, Number 4 (2017).
[3] Abine. [n. d.]. DeleteMe: Remove Your Personal Info from Search Results. https://joindeleteme.com/. Accessed: 11-13-2025.
[4] Supriya Adhatarao, Cédric Lauradoux, and Cristiana Santos. 2021. Why IP-based Subject Access Requests Are Denied? *arXiv preprint arXiv:2103.01019* (2021).
[5] Joseph A. Akinyele, Matthew D. Green, and Aviel D. Rubin. 2011. Charm: A Framework for Rapidly Prototyping Cryptosystems. Cryptology ePrint Archive, Paper 2011/617.
[6] Gary Anthes. 2014. Data brokers are watching you. *Commun. ACM* Volume 58, Number 1 (2014).
[7] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. 2005. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In *Proc. Network and Distributed System Security Symposium.*
[8] Aura. [n. d.]. Aura: All-In-One Digital Safety. https://buy.aura.com/. Accessed:11-13-2025.
[9] Coline Boniface, Imane Fouad, Nataliia Bielova, Cédric Lauradoux, and Cristiana Santos. 2019. Security Analysis of Subject Access Request Procedures - How to Authenticate Data Subjects Safely When They Request for Their Data. In *Proc. Privacy Technologies and Policy.*
[10] Luca Bufalieri, Massimo La Morgia, Alessandro Mei, and Julinda Stefa. 2020. GDPR: When the Right to Access Personal Data Becomes a Threat. In *Proc. IEEE International Conference on Web Services.*
[11] California Legislature. 2018. TITLE 1.81.5. California Consumer Privacy Act of 2018.
[12] California Privacy Protection Agency. [n. d.]. Data Broker Registry. https://cppa.ca.gov/data_broker_registry/. Accessed:11-13-2025.
[13] California Public Utilities Commission. 2025. Home Broadband Adoption Report. https://docs.cpuc.ca.gov/PublishedDocs/Efile/G000/M563/K203/563203680.PDF. Accessed: 11-13-2025.
[14] California State Legislature. 2023. SB-362: California Delete Act of 2023. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=202320240SB362. Accessed:11-13-2025.
[15] CalPrivacy. [n. d.]. https://consumer.drop.privacy.ca.gov/. Accessed:11-13-2025.
[16] CPPA. [n. d.]. Proposed Regulations on Accessible Deletion Mechanism. https://cppa.ca.gov/regulations/drop.html. Accessed:11-13-2025.
[17] European Parliament and Council. 2016. Regulation (EU) 2016/679, General Data Protection Regulation. https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng. Accessed:11-13-2025.
[18] McKenzie Funk. 2023. The Man Who Trapped Us in Databases. https://www.nytimes.com/2023/09/22/magazine/hank-asher-data.html. Accessed: 11-13-2025.
[19] Oded Goldreich. 2004. *Foundations of Cryptography, Volume 2.* Cambridge university press Cambridge.
[20] Governo do Brasil. 2023. LGPD - Lei Geral de Proteção de Dados. https://www.gov.br/esporte/pt-br/acesso-a-informacao/lgpd. Accessed: 2025-5-28.
[21] Jiahui He, Peter Snyder, Hamed Haddadi, Fabián E. Bustamante, and Gareth Tyson. 2025. Measuring the Accuracy and Effectiveness of PII Removal Services. *Proc. Privacy Enhancing Technologies Symposium* (2025).
[22] Dominik Herrmann and Jens Lindemann. 2016. Obtaining personal data and asking for erasure: do app vendors and website owners honour your privacy rights?. In *Sicherheit.*
[23] Jingwei Hu, Zhiqi Liu, and Cong Zuo. 2025. Delegated multi-party private set intersection from secret sharing. *Cryptology ePrint Archive* (2025).
[24] Seoyeon Hwang, Stanislaw Jarecki, Zane Karl, Elina van Kempen, and Gene Tsudik. 2024. PIVA: Privacy-Preserving Identity Verification Methods for Accountless Users via Private List Intersection and Variants. In *Proc. European Symposium on Research in Computer Security.*
[25] Scott Jordan, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, and Gene Tsudik. 2023. VICEROY: GDPR-/CCPA-compliant Enforcement of Verifiable Accountless Consumer Requests. In *Proc. Network and Distributed System Security Symposium.*
[26] Florian Kerschbaum. 2012. Collusion-resistant outsourcing of private set intersection. In *Proc. Annual ACM Symposium on Applied Computing.*
[27] Jacob Leon Kröger, Jens Lindemann, and Dominik Herrmann. 2020. How do app vendors respond to subject access requests?: a longitudinal privacy study on iOS and Android Apps. In *Proc. Conference on Availability, Reliability and Security.*
[28] California Legislature. [n. d.]. California Code of Regulations. https://govt.westlaw.com/calregs/Index?transitionType=Default&contextData=%28sc.Default%29. Accessed: 11-13-2025.
[29] Fang Liu, Wee Keong Ng, Wei Zhang, Do Hoang Giang, and Shuguo Han. 2014. Encrypted set intersection protocol for outsourced datasets. In *Proc. IEEE International Conference on Cloud Engineering.*
[30] Login.gov. [n. d.]. https://login.gov/. Accessed: 2026-02-04.
[31] Mariano Di Martino, Isaac Meers, Peter Quax, Ken Andries, and Wim Lamotte. 2022. Revisiting Identification Issues in GDPR 'Right Of Access' Policies: A Technical and Longitudinal Analysis. *Proc. Privacy Enhancing Technologies Symposium* (2022).
[32] Matthias Marx, Ephraim Zimmer, Tobias Mueller, Maximilian Blochberger, and Hannes Federrath. 2018. Hashing of personally identifiable information is not sufficient. In *Sicherheit.*
[33] Nikita Samarin, Shayna Kothari, Zaina Siyed, Oscar Bjorkman, Reena Yuan, Primal Wijesekera, Noura Alomar, Jordan Fischer, Chris Jay Hoofnagle, and Serge Egelman. 2023. Lessons in VCR Repair: Compliance of Android App Developers with the California Consumer Privacy Act (CCPA). *Proc. Privacy Enhancing Technologies Symposium* (2023).
[34] Surfshark. [n. d.]. Incogni: Automatic Personal Data Removal. https://incogni.com/. Accessed: 11-13-2025.
[35] Kejsi Take, Jordyn Young, Rasika Bhalerao, Kevin Gallagher, Andrea Forte, Damon McCoy, and Rachel Greenstadt. 2024. What to Expect When You're Accessing: An Exploration of User Privacy Rights in People Search Websites. *Proc. Privacy Enhancing Technologies Symposium* (2024).
[36] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. 2019. A Study on Subject Data Access in Online Advertising After the GDPR. In *Proc. Data Privacy Management, Cryptocurrencies and Blockchain Technology.*
[37] U.S. Census Bureau. 2024. United States Population by Age and Sex. https://www.census.gov/popclock/data_tables.php?component=pyramid. Accessed: 11-13-2025.
[38] Elina van Kempen, Isita Bagayatkar, Pavel Frolikov, Chloe Georgiou, and Gene Tsudik. 2026. Consumer Beware! Exploring Data Brokers' CCPA Compliance. In *IEEE Symposium on Security and Privacy (S&P), 2026.*
[39] Elina van Kempen, Pavel Frolikov, Chloe Georgiou, and Gene Tsudik. [n. d.]. Source code for PIVOT. https://github.com/y76/IV-Drip/.
[40] Janis Wong and Tristan Henderson. 2019. The right to data portability in practice: exploring the implications of the technologically neutral GDPR. *International Data Privacy Law* Volume 9, Issue 3 (2019).
[41] Hao Zhao, Wenqi Zhang, Xiong Li, Shuai Shang, Ke Huang, and Xiaosong Zhang. 2024. Towards efficient delegated private set intersection cardinality protocol. In *Proc. International Conference on Computer Supported Cooperative Work in Design.*
[42] Qingji Zheng and Shouhuai Xu. 2015. Verifiable delegated set intersection operations on outsourced encrypted data. In *Proc. IEEE International Conference on Cloud Engineering.*

A Security Proofs Outlines

A.1 Malicious $\mathcal{MED}$ and $\mathcal{BR}_j$ in PIVOT

Privacy against malicious $\mathcal{MED}$. For a malicious $\mathcal{MED}$, a simulator $\mathsf{SIM}_{\mathcal{MED}}$ verifies π_{a_i}, extracts w_i for all $i \in [n]$, and computes $D = (a_i^{(\beta)})^{1/w_i}$. $\mathsf{SIM}_{\mathcal{MED}}$ computes $Z^{-x_i} Z^{r_i} \cdot Z^{y_i} Z^{\tilde{r}_i}$ from values received from C and $\mathcal{BR}_j$ and checks whether it equals D. If yes, $\mathsf{SIM}_{\mathcal{MED}}$ sets $A_i = a_i$, and otherwise $A_i \leftarrow \$$, and sends all values to $\mathcal{BR}_j$. Thus the simulated and real views are indistinguishable.

Privacy against malicious $\mathcal{BR}_j$. For a malicious $\mathcal{BR}_j$, a simulator $\mathsf{SIM}_{\mathcal{BR}_j}$ extracts $\tilde{r}_i, y_i$. The proof then follows the same reasoning as for a malicious C.

A.2 Additively homomorphic proxy re-encryption

The security of the additively homomorphic proxy re-encryption scheme follows the proof of the (non-additively homomorphic) proxy re-encryption scheme by [7], modifying it to put the message m in the exponent of Z.