Edna: Data Disguises for Web Applications

Anonymous Authors

1 Introduction

Web applications must protect users' data and rights to data privacy as codified in laws such as the GDPR. Current web applications take an all-or-nothing approach: they either make minimal effort to provide data privacy (e.g., Reddit and Lobsters), or permanently delete users' data upon request. Either option gives users little fine-grained control over how web applications retain and use their private data.

Furthermore, even these efforts are nontrivial, and require careful understanding and manipulation of the web application's database. Companies with resources such as FB have implemented internal systems that handle e.g., user account deletion (cite DELF); however, solutions like these fail to generalize and require time and money that smaller-scale companies may lack.

We propose a new system architecture for web applications to enable support for *data disguising*, namely transforming users' data stored by the appliaction in a number of privacy-preserving ways. Data disguising enables data decay (gradual anonymization over time); revealing of disguises (restoration of deleted data by the user, while providing strong privacy guarantees when deleted); and decorrelation and temporary recorrelation of users with anonymized data. [lyt: <– Fix this.] Furthermore, data disguising generalizes across web applications, handling the technical challenges of securing private data, while being aware of application database schemas and semantics.

1.1 Utility Goals

Our design considers typical database-backed applications that link a library implementing our disguising tool (Edna). Edna provides an API that the application uses to specify disguises, apply them, and (in some cases), reveal disguised data with proper authorization (Table 2). Edna executes the database transformations required as part of a disguise, and stores disguised data in a secure way, subject to the threat model we describe in §1.3.

Example: HotCRP. We imagine three useful disguises that aid privacy in HotCRP:

1. Per-user GDPR Account Deletion, which removes all of a particular user's data such as reviews and papers.
2. Universal Comment Removal, which removes all comments on papers.
3. Universal Decorrelation, which decorrelates users from their authored reviews and papers. Decorrelation of a paper or review (1) generates an anonymous user in the user table, and (2) rewrites the foreign key from the paper or review to the users table to point to the newly created anonymous user.

Users who invoke GDPR deletion are given the option to return by revealing their disguise; however, between disguising and revealing their account, no other entity learns the identity or existence of the user (ensuring that the disguise complies with the GDPR).

After several years, the admin can decay identifiers in the conference data by decorrelating all users from their authored reviews and papers. This helps protect reviewers and paper authors from being unblinded in the case of a data breach.

Even after universal decorrelation, it would be nice if HotCRP allowed users to apply GDPR account deletion to delete (even decorrelated) papers and reviews that they wrote: a disguise that applies to a user's data may, in some circumstances, need to apply to data decorrelated from the user.

Application of universal decorrelation also should not prevent the application of universal comment removal: the admin may decide, after applying universal decorrelation, that arbitrary text in paper comments also identifies their authors (which are now anonymous users), and invoke universal comment removal. This demonstrates how the developer can simply apply a new disguise to fix mistakes in a previously applied disguise that left some identifying information around, even if the new disguise now applies to data of anonymous users.

Finally, HotCRP would retain much of its usefulness if users could be permitted to view reviews on their papers, or

edit their reviews, even if these papers and reviews have been decorrelated and belong to anonymous users. However, this should be possible *without* revealing to other users of the system which actual user authored these anonymized papers and reviews.

Usability. These disguising properties should be achieved with minimal burden on users.

When Edna disguises a user's data for universal decorrelation or comment removal, Edna emails a unique URL (a *capability*) to the user corresponding to that a paper or review only known to the user. When the user wants to ask for permission to view (or edit, if permitted) the undisguised paper or review, the user simply has to click the URL link and log into the application. Thus, application of universal disguises should not require a user to be online in order to gain the capability to apply further disguises or view their disguised data.

Similarly, when Edna disguises a user's data for GDPR deletion, the invoking user receives a URL that allows the user to reveal the disguise, restoring their account.

General Use Cases. From the HotCRP example, we generalize a handful of the most beneficial and practical use cases that Edna should support:

- **1st-Party GDPR-Compliant Disguising and Revealing.** Edna should support user-invoked disguise(s) to modify, decorrelate and/or delete the user's data and account that enable applications to meet the requirements of the GDPR's right to be forgotten. These disguises can optionally be revealable, which enables users to restore the original state of disguised data to e.g., permanently restore their accounts and/or data ownership.
- **3rd-Party Disguising.** Edna should support disguise applied by users (such as an admin) that modify, decorrelate, and/or delete data of other users in the system. For example, universal conference anonymization or comment removal alters data not associated with the invoking admin.
- **Temporary Recorrelation w/out Database Changes.**
  - *For Disguising Decorrelated Data*: A disguise targeting a user's data can, if authorized, also operate on data decorrelated from the user by prior disguises without modifying the database's persistent links between data and owning users. For example, GDPR account deletion removes user u's reviews even if Edna has decorrelated these reviews from u.
  - *For Per-User Revealed Views and Permissions*: Users should be able to operate with ownership permissions on data previously correlated with their account, but decorrelated by a disguise. Users do so by interacting with a personalized, temporary database view when they visit their decorrelated data. This should require no modifications to the actual database contents.

    For example, a user whose authored papers have been decorrelated can log in and view their papers and reviews on their papers.
- **Disguising Anonymized Users.** Edna can disguise data of anonymized users. For example, universal comment removal may removes the data of anonymized users generated by universal decorrelation. This allows disguises to be developed over time, and remove identifying data potentially mistakenly missed by prior disguises, but which now belongs to anonymous users.

1.2 Edna Abstractions

Edna supports these use cases by exposing three abstractions to applications:

- Application *principals*, which correspond to users of the application, and which are uniquely identifiable (e.g., via a user ID). We denote an application principal as $p \in P$, where P is the set of all application principals.
- Application *pseudoprincipals*, which correspond to anonymous users not tied to any natural person, and which are uniquely identifiable (e.g., via a user ID). Pseudoprincipals are part of the set of application principals $p \in P$. A pseudoprincipal may or may not be indistinguishable from a real principal: for example, it may not have an associated password.
- A disguise $d \in D$ invoked by some principal p, which transforms the application database.

Edna operates on two categories of data:

1. ***Database Contents***: The application reads and writes database contents, which include visible, undisguised and currently-disguised data. Edna modifies the data database contents when it applies and reveals disguises.
2. ***Disguise Diffs***: Edna records the set of disguise diffs—which contain the original database contents and the modifications performed to them—and uses diffs to compose disguises on top of one another, or to reveal the updates done by a disguise. Every disguise diff has an associated principal (namely the one associated with the modified data).

Data Access Control. The application's normal permissions logic controls data access to database contents. The application invokes Edna to access and needs to parse diff contents directly: Edna exposes an API (§2) allowing the application to learn specific semantic information contained in diffs (e.g.,

whether a particular principal used to own a now-decorrelated piece of data) if provided appropriate authorization. This authorization also allows the application to ask Edna to use diff information to reveal or apply disguises.

Two types of *capabilities* control access to diffs:

1. ***Data Capability*** $\mathcal{C}_p^{\texttt{data}}$: Grants read access to disguise diffs associated with principal p.
2. ***Locating Capability*** $\mathcal{C}_{pd}^{\texttt{loc}}$: Allows locating the disguise diffs associated with principal p produced from applying d, but grants no access to the diffs' data.

Edna needs both the data and locating capabilities $(\mathcal{C}_p^{\texttt{data}}, \mathcal{C}_{pd}^{\texttt{loc}})$ to access disguise diffs of disguise d. Each capability alone is insufficient: an attacker who can spoof $\mathcal{C}_{pd}^{\texttt{loc}}$ will not be able to access disguise diffs without $\mathcal{C}_p^{\texttt{data}}$, although they can learn that d disguised some of p's data; and an attacker holding $\mathcal{C}_p^{\texttt{data}}$ cannot locate the disguise diffs to access without $\mathcal{C}_{pd}^{\texttt{loc}}$.

1.3 Threat Model

Data disguises protect user information in a web application against external observation and service compromise. An external observer is a user of the web application (authenticated or unauthenticated) who observes information exposed through using the application. A service compromise occurs when an attacker compromises the web application and gains full access to the server. The attacker therefore can access any data stored, perform any actions the application can perform, and access any information available to Edna.

A data disguise guarantees that the disguised data is hidden from any future attackers unless explicitly revealed by an authorized principal. In particular, an attacker who compromises Edna at time t learns *nothing but*:

1. the (plaintext) contents of the application database at or after time t;
2. the disguises invoked, and the identity of the principals invoking them, after time t; and
3. all of a principal's disguise diffs acquired prior to time t, should the principal authorize access to disguise diffs for purposes of revealing or performing some application action after time t.

We make standard assumptions about the security of cryptographic primitives: attackers cannot break encryption and keys stored with non-colluding clients are safe.

Edna operates in an honest-but-curious setting: even if compromised, Edna faithfully executes its protocols, but exposes all data accessed to the attacker.

1.4 Security Goals

With this threat model, Edna seeks to meet four security goals:

(1) Authorized Disguises. Only a client properly authenticated as a principal p who is authorized to invoke d can apply (and later reveal) d.

(2) Secure Disguise Diffs. Only an authenticated principal who provides both locating capability $\mathcal{C}_{pd}^{\texttt{loc}}$ and data capability $\mathcal{C}_p^{\texttt{data}}$ can authorize access to the disguise diffs, as well as any information (e.g., ownership permissions) derived from diffs.

(3) Privacy of Disguise History. An attacker cannot learn the set of disguises that have disguised a principal's data. An attacker only learns a disguise d exists if an authenticated principal p provides locating capability $\mathcal{C}_{pd}^{\texttt{loc}}$ to Edna.

Non-Goals. Under Edna's threat model, the following properties are out of scope:

- Security of disguised data when an attacker has prior snapshots of the system.
- Security of metadata such as the number of disguises applied to, or number of disguise diffs associated with, *some* principal (where the principal ID is unknown).
- Security of metadata when an attacker has $\mathcal{C}_{pd}^{\texttt{loc}}$: the attacker can learn that d applied to a particular principal p, and the number of disguise diffs for that p and d.
- Privacy guarantees about undisguised data: if identifying data about p is not covered by disguise specification d, it remains visible to attackers after corresponding instance d applies. For example, a disguise that leaves contents of posts unmodified does not hide identifying references to users in that content.
- Hiding whether data is disguised or not. For example, if a post's author is anonymized as "anonFox", Edna leaks the fact that a post's author has been disguised.

2 Edna API.

Edna's API in Table 2 enables the application to support disguise application and revealing. The application also uses Edna to access and interpret diffs authorized by the provided capabilities. Applications integrating with Edna may modify the client API so that the client can pass capabilities to the application along with the action to perform. For example, the application can send users URLs to click that invoke application actions with capability arguments.

We next describe example usages of the API.

Principal Registration. Every client who creates an application account for principal p registers an email and public key with Edna. Edna remembers each public key $\texttt{k}_p^{\texttt{pub}}$ along with p's ID.

Disguising and Revealing. When a client speaking for principal p wants to apply a disguise d, the application invokes `ApplyDisguise` with the client-provided data capability and a set of locating capabilities. Each locating capability $\mathcal{C}_{pd'}^{\texttt{loc}}$ allows Edna to access p's disguise diffs from disguise d', and to compose the new disguise on top of these disguises

Symbol	Description
p	application principals, corresponding to a user ID in the application
d	disguise invoked by authorized principals that transforms the application database in privacy-preserving ways
$\texttt{op}_d$	a disguise operation (either a removal, modification, or decorrelation)
Δ_{pd}	database diff associated with p produced by disguise d.
$\mathcal{C}_p^{\texttt{data}}$	data capability that grants access to all disguises' diffs associated with principal p.
$\mathcal{C}_{pd}^{\texttt{loc}}$	locating capability that allows the holder to find the database diffs from d associated with a principal p.
$\texttt{k}_p^{\texttt{pub}}$	public key of p
$\texttt{k}_p^{\texttt{pri}}$	private key of p

Table 1: Notation used to describe Edna's design.

API Call	Description
`RegisterPrincipal(`p`, Email email, PubKey` $\texttt{k}_p^{\texttt{pub}}$`)` $\rightarrow ()$	Registers p as a principal with public key $\texttt{k}_p^{\texttt{pub}}$.
`ApplyDisguise(`p`, Disguise` d`, DataCap` $\mathcal{C}_p^{\texttt{data}}$`, Vec<LocCap> lcaps)` $\rightarrow$`Vec<LocCap>`	Applies disguise d and returns the corresponding disguise d and locating capabilities $\mathcal{C}_p^{\texttt{loc}}$d. The argument capabilities allow Edna to locate and access disguise diffs, which Edna uses to compose disguise d on top of prior disguises.
`RevealDisguise(`p`, Disguise` d`, DataCap` $\mathcal{C}_p^{\texttt{data}}$`, LocCap` $\mathcal{C}_{pd}^{\texttt{loc}}$`)` $\rightarrow ()$	Reveals disguise d. The argument capabilities grant Edna access to database diffs from applying d, which allows Edna to reveal undisguised data.
`CapEstablishesOwnership(`p`, DatabaseObj` x`, DataCap` $\mathcal{C}_p^{\texttt{data}}$`, LocCap` $\mathcal{C}_{pd}^{\texttt{loc}}$`)` $\rightarrow$ `bool`	Returns whether principal p has ownership rights to the provided database object. The capabilities allow Edna to access diffs that can prove that p had been correlated with the object.

Table 2: The Edna Library API

during application. Applying a disguise returns the locating capability $\mathcal{C}_{pd}^{\texttt{loc}}$ that the application sends to the client (via e.g., emailing the client a link).

In order to reveal data disguised by d for principal p, the application invokes `RevealDisguise` on behalf of a client speaking for p. The client provides its data capability $\mathcal{C}_p^{\texttt{data}}$ and locating capability $\mathcal{C}_{pd}^{\texttt{loc}}$. The application passes these to Edna to restore the original, undisguised data with access to p's diffs corresponding to d.

Performing Permitted Application Actions. Imagine that HotCRP disguise d anonymizes all reviewers and authors of a conference by decorrelating them from their papers and reviews respectively. As described above, HotCRP emails the client a link when that their data has been decorrelated, which contains the locating capability in the URL. Assume that HotCRP sends a unique link for every piece of data (e.g., every review or paper); HotCRP could also send a single link for all decorrelated data.

The application normally ensures that only a client speaking for p should have read-only access to reviews for p's authored papers, and read-write access to reviews p wrote.

If the client speaking for p wants to edit a review as it would normally, the application (post-d) will not grant the client any read/write permissions to perform actions on any reviews of the conference: any previously correlated paper or review is now associated with pseudoprincipal q.

Instead, for the client speaking for p to regain authorship or reviewership permissions of a particular review or paper:

- The client clicks on the URL emailed to the client for that piece of data when it was decorrelated, which opens a HotCRP page.
- Javascript on the webpage retrieves the client's data capability stored by the client's browser, and extracts the locating capability and the data object the client wants to access from the URL.
- The client then must log into HotCRP as principal p.

- HotCRP invokes Edna's `CapEstablishesOwnership` for the data object in question, using the capabilities passed in by the client.
- If Edna returns `true`, the application permits the client to view the object with ownership rights (e.g., ability to see reviews on authored papers, or edit their review); otherwise, the application does not allow the client to perform privileged actions.

3 Design

3.1 Disguise Operations.

Because disguises are inherently application-specific, the application developer specifies a disguise as a set of predicated disguise operations $\texttt{op}_d$ to perform.

Operations $\texttt{op}_d$ of disguise d take data objects as input and execute updates to application data. Edna automatically generates database change records when applying $\texttt{op}_d$.

Developers describe which principal(s) an operation's generated database change record corresponds to. For example, a database change record generated by removing comment may correspond to the principal whose ID is referenced by the author column.

Developers also specify an application-aware pseudoprincipal generation policy, namely how to generate new user accounts in a manner that the application can handle (e.g., pseudoprincipals may not have email addresses).

Finally, developers specify which principals are authorized to apply the disguise: enforcing access control for disguising is left to the application.

Operations come in three forms:

1. Modify: change an attribute of the data object.
2. Remove: delete the data object.
3. Decorrelate: generate a *pseudoprincipal* q, and rewrite the foreign key to original principal p from the data object to instead point to q.

For each $\texttt{op}_d$, the application developer specifies:

- An associated predicate over the application database that selects $\texttt{op}_d$'s input objects in a SQL-like fashion.
- The type of operation and its arguments (e.g., which attributes of the data object to modify).
- The corresponding principal(s) that should have the capability to access $\texttt{op}_d$'s generated disguise change or correlation record.

3.2 Database Diffs.

Every $\texttt{op}_d$ produces a *database diff* Δ_{pd} associated with the disguise δ and a principal p. Each Δ_{pd} contains the ID of the disguise, the associated principal p's ID, and a random nonce. In addition to this data, a decorrelation Δ_{pd} contains the created pseudoprincipal's ID; a removal Δ_{pd} contains the removed object's value; and a modification Δ_{pd} contains the old and new value of the modified object.

3.3 Diff Access Control.

Securing Access to Database Diffs. Edna's design uses principal p's private key $\texttt{k}_p^{\texttt{pri}}$ as the data capability $\mathcal{C}_p^{\texttt{data}}$. Edna secures p's diffs Δ_{pd} by encrypting them with $\texttt{k}_p^{\texttt{pub}}$. The diff's nonce ensures safety against known-plaintext attacks. Only a client who knows $\texttt{k}_p^{\texttt{pri}}$ can access diffs.

Securing Disguise History. Edna's design may leak information that d disguised principal p because an adversary may learn that ciphertexts for Δ_{pd} exist for a particular p and d. This is a problem: for example, Edna should not store information that a principal p has invoked GDPR deletion.

To avoid this problem, Edna stores an array of Δ_{pd} ciphertexts at random location pointed to by locating capability $\mathcal{C}_{pd}^{\texttt{loc}}$. This dissociates one disguise applied to a principal p from all other disguises, so an adversary only ever learns about a single disguise if Edna gains permission to reveal or compose upon that disguise.

Note that an adversary without access to any $\mathcal{C}_{pd}^{\texttt{loc}}$ can learn that n diffs exist for *some* p and d, but cannot directly identify which p or d. If an adversary has access to $\mathcal{C}_{pd}^{\texttt{loc}}$, the adversary can learn that d applied to p, and the number of Δ_{pd} diffs for that p and d. This metadata is out of scope of our threat model. [lyt: An alternative design might remove the encrypted data completely (and email it to the client), so that an adversary doesn't even know of its existence. However, since we're allowing disguised data to be distinguished from undisguised data, this seems potentially unnecessary.]

Storage of Capabilities. Edna should not store $\mathcal{C}_p^{\texttt{data}}$ or $\mathcal{C}_{pd}^{\texttt{loc}}$: an adversary could then learn that d has applied to principal p. Thus, these capabilities must be stored externally, even when no client speaking for p is currently online.

Clients already hold $\mathcal{C}_p^{\texttt{data}}$: a client generates a keypair for a new principal p and stores $\texttt{k}_p^{\texttt{pri}}$, while registering $\texttt{k}_p^{\texttt{pub}}$ with Edna. Edna only stores $\texttt{k}_p^{\texttt{pub}}$.

Because Edna generates $\mathcal{C}_{pd}^{\texttt{loc}}$ during disguising, Edna must communicate $\mathcal{C}_{pd}^{\texttt{loc}}$ to clients. Edna emails $\mathcal{C}_{pd}^{\texttt{loc}}$ to the corresponding email address associated with p. This allows Edna to disguise data of principals even when no client authenticated as p has an active session open.

3.4 Current Design: Discussion

As the current design stands, Edna supports 1st-person and 3rd-person disguising, as well as 1st-person revealing. Application developers can write GDPR-compliant and/or universal disguises with the primitives exposed by Edna to write disguise specifications. Edna uses diffs produced from disguising to reveal data when authorized to do so, and when revealing

does not revert updates made to the data since the time of disguise application.

Edna also supports "Temporary Recorrelation without Database Changes" because Edna can determine the original owner of data as long as Edna has access to decorrelation diffs (which the client or the application provides via the appropriate capabilities). Edna can use information from decorrelation diffs to disguise decorrelated data as if it were owned by the original user; and Edna can support the API discussed in §2, which allows applications to query Edna to check ownership properties and grant authorized users personalized views and permissions to access data objects.

Furthermore, Edna does this while meeting all security goals: Edna supports authorized disguises and ensures the security of ownership claims, disguise diffs, and disguise history.

Our current design falls short, however, by failing to support "Disguising Anonymized Users" which we explain next.

3.5 Disguising Anonymized Users

To apply one disguise on top of another in a way that supports, e.g., comment removal after universal decorrelation, Edna must support *pseudoprincipal diffs*, namely diffs associated with pseudoprincipals.

These diffs can occur after at least one disguise has been applied: subsequent disguises may associate diffs with pseudoprincipals. For example, after universal decorrelation, every paper or review correlates with a pseudoprincipal user. When Edna applies universal decorrelation, decorrelation of all papers and reviews generates correlation diffs for these pseudoprincipals.

However, we now have a problem. Diffs are encrypted with $\mathcal{C}_p^{\texttt{data}}$ (p's private key) and stored at $\mathcal{C}_{pd}^{\texttt{loc}}$, and Edna emails $\mathcal{C}_{pd}^{\texttt{loc}}$ to (potentially offline) clients and does not retain $\mathcal{C}_{pd}^{\texttt{loc}}$. However, pseudoprincipals have no corresponding real user (and, for unlinkability, Edna cannot store which pseudoprincipal-user correspondences). Thus, Edna has no way to communicate $\mathcal{C}_{pd}^{\texttt{loc}}$ to a user who speaks for pseudoprincipal p!

We have three potential approaches to solve this problem. Two we see as strawmen because they fail to meet our secury and use case goals respectively:

1. *Weak Security:* Edna can store $\mathcal{C}_{pd}^{\texttt{loc}}$ for pseudoprincipal p. This means an adversary will be able to learn p's undisguised data, even if it had been disguised by d.

2. *Permanent Disguises:* Edna throws away $\mathcal{C}_{pd}^{\texttt{loc}}$ for pseudoprincipal p. This means that the disguise modifications are permanent, and no links (to determine ownership) between p and other pseudoprincipals can be made. This affects temporary recorrelation, because decorrelation links between principals cannot be derived from discarded diffs.

We next describe a third option, which meets our security and use case goals while trading off efficiency and usability.

Pseudoprincipal Private Keys. When a pseudoprincipal q is generated to decorrelate data from p:

1. Edna generates keypair $(\texttt{k}_q^{\texttt{pub}}, \texttt{k}_q^{\texttt{pri}})$ for q
2. encrypts the private key of q with $\texttt{k}_p^{\texttt{pub}}$ to produce $\mathsf{Enc}(\texttt{k}_p^{\texttt{pub}}, \texttt{k}_q^{\texttt{pri}})$
3. stores $\mathsf{Enc}(\texttt{k}_p^{\texttt{pub}}, \texttt{k}_q^{\texttt{pri}})$ at location $\mathcal{C}_{pd}^{\texttt{loc}}$ along with p's encrypted diffs from disguise d

This ensures that a client with $\mathcal{C}_{pd}^{\texttt{loc}}$ and $\mathcal{C}_p^{\texttt{data}}$ ($\texttt{k}_p^{\texttt{pri}}$) can access $\texttt{k}_q^{\texttt{pri}}$, the data capability of pseudoprincipal q. Furthermore, only a client who holds $\mathcal{C}_{pd}^{\texttt{loc}}$ can learn that p has decorrelated data because associated pseudoprincipal private keys are not stored publically with p's ID.

When Edna applies disguise d' to further disguise pseudoprincipal q's data, Edna:

1. stores disguise diffs $\Delta_{qd'}$ at location $\mathcal{C}_{qd}^{\texttt{loc}}$
2. encrypts $\mathcal{C}_{qd}^{\texttt{loc}}$ with $\texttt{k}_p^{\texttt{pub}}$ to produce $\mathsf{Enc}(\texttt{k}_p^{\texttt{pub}}, \mathcal{C}_{qd}^{\texttt{loc}})$
3. stores $\mathsf{Enc}(\texttt{k}_p^{\texttt{pub}}, \mathcal{C}_{qd}^{\texttt{loc}})$ mapped to by principal q's ID

Note that Edna stores $\mathcal{C}_{qd}^{\texttt{loc}}$ ciphertexts associated with q's ID, whereas for a real principal p, Edna does not store any similar metadata indicating how many disguises have applied to p. This means that an adversary can learn that *some* disguise has applied to and decorrelated data from q. However, this falls out of scope in our threat model: q is a pseudoprincipal created from a decorrelation operation, and therefore any disguise metadata regarding q has already been dissociated from the original principal p's identity.

Extending capabilities to authorize pseudoprincipal diffs access. With this extended design, a client speaking for p that provides data and locating capabilities $(\mathcal{C}_p^{\texttt{data}}, \mathcal{C}_{pd}^{\texttt{loc}})$ authorizes access to *both* p's diffs for disguise d, *and* diffs of a pseudoprincipal q created by decorrelating p's data. [lyt: I don't think we're necessarily allowing clients to speak for q, but rather to perform actions using q's diffs.]

Given a data capability $\texttt{k}_p^{\texttt{pri}}$, Edna can can access $\texttt{k}_q^{\texttt{pri}}$ by decrypting $\mathsf{Enc}(\texttt{k}_p^{\texttt{pub}}, \texttt{k}_q^{\texttt{pri}})$. With $\texttt{k}_q^{\texttt{pri}}$, Edna can decrypt all $\mathsf{Enc}(\texttt{k}_q^{\texttt{pub}}, \mathcal{C}_{qd}^{\texttt{loc}})$ so that Edna has both the data and location capabilities for all of q's disguise diffs. This allows the application to reveal, disguise, or perform application actions with access to diffs from all disguises applying to q.

While clearly more expensive for Edna to execute, this protocol allows Edna to recursively disguise pseudoprincipals, yet still require only the top-level original principal capabilities to support the desired use cases. Furthermore, this design still meets our security goals: data capabilities secure all disguise diffs, and an adversary learns no disguise metadata for real users.

4 Disguising and Revealing Semantics

This section describes the semantics of disguising provided by Edna. In particular, we describe the end state of a data object given some history of disguise applications and reveals.

Some Notation. We describe disguise histories as a list of $\texttt{A}(d_i)$ and $\texttt{R}(d_i)$ actions, where $\texttt{A}(d_i)$ corresponds to the application of disguise d_i, and $\texttt{R}(d_i)$ corresponds to the reveal of d_i's diffs. Time moves to the right in the list.

For every data object x in the system, $x_{\texttt{start}}$ describes its initial state, and $x_{[\texttt{A}(d_1),\ldots]}$ describes its state after Edna has applied the history $[\texttt{A}(d_1), \ldots]$.

Composing Multiple Disguise Applications. Let d_1 and d_2 be two disguises, where d_1 is applied after d_2. to produce disguise history $[\texttt{A}(d_1), \texttt{A}(d_2)\ ]$. Let x be some application data object, where $x_{\texttt{start}}$ is its initial state, and $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$ is the final state after Edna applies d_1 and d_2. If both d_1 and d_2 update x, what is $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$? We consider two possible end states:

($\mathcal{S}_1$) $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$ reflects the application of *both* d_1 *and* d_2 to $x_{\texttt{start}}$.

In other words, if an $\texttt{op}_{d_2}$'s predicate `pred` matches $x_{\texttt{start}}$, then $\texttt{op}_{d_2}$ is applied to $x_{[\ \texttt{A}(d_1)\ ]}$, even if `pred` does not match $x_{[\texttt{A}(d_1)]}$.

$\texttt{op}_{d_2}$ updates are applied sequentially after d_1's updates if the two modify the same object attribute.

($\mathcal{S}_2$) $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$ reflects the application of d_1 to $x_{\texttt{start}}$, followed by the application of d_2 to $x_{[\texttt{A}(d_1)]}$. When deciding whether to apply to x, Edna matches $\texttt{op}_{d_2}$'s predicate only against $x_{[\texttt{A}(d_1)]}$, and not against the original state $x_{\texttt{start}}$.

For example, let $\texttt{op}_{d_1}$ decorrelate all posts from authors predicated on `author = Bea`, and let $\texttt{op}_{d_2}$ remove all posts predicated on `author = Bea`.

($\mathcal{S}_1$) Both $\texttt{op}_{d_1}$ and $\texttt{op}_{d_2}$ update posts originally having author "Bea", resulting in the removal of all posts originally with author Bea.

($\mathcal{S}_2$) Applying $\texttt{op}_{d_1}$ results in all posts with author "Bea" having pseudoprinicipal authors such as "anonFox." $\texttt{op}_{d_2}$ only knows the state of posts after $\texttt{op}_{d_1}$ occurs, and does not remove any posts because no post has author "Bea".

The choice of whether $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$ should result in $\mathcal{S}_1$ or $\mathcal{S}_2$ depends on the specific application and is left to the developer to specify.

In certain scenarios, however, Edna can only achieve end state $\mathcal{S}_2$. In order to achieve $\mathcal{S}_1$, d_2 must evalute predicates against $x_{\texttt{start}}$, which requires knowing the value of $x_{\texttt{start}}$. To learn $x_{\texttt{start}}$, Edna must know how d_1 modified x (e.g., knowing that a post with author "anonFox" originally was a post with author "Bea"); this information, however, is stored in in the diffs produced by d_1.

Edna may not have access to these diffs when applying d_2: a client who invokes d_2 while authenticated as principal $q \neq p$, and who does not provide the relevant capability pairs $(\mathcal{C}^{\texttt{data}}_{pd_1}, \mathcal{C}^{\texttt{loc}}_{pd_1},)$ cannot access p's diffs for d_1.

In this case, d_2 can only match against $x_{[\texttt{A}(d_1)]}$ and achieve end state $\mathcal{S}_2$. This is shown in Table 3.

	Access d_1's diffs while $\texttt{A}(d_2)$?	
	Yes	**No**
$x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$	$\mathcal{S}_1$ or $\mathcal{S}_2$	$\mathcal{S}_2$

Table 3: Possibilities for $x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$ depending on whether Edna has access to diffs from d_1 regarding x.

Semantics of Disguise Reversals. We now consider composition of disguises in the presence of disguise reversals.

	Access to d_i's diffs while $\texttt{A}(d_i)$?	
	Yes	**No**
$x_{[\texttt{A}(d_i),\texttt{R}(d_i)]}$	$x_{\texttt{start}}$	$x_{[\texttt{A}(d_i)]}$

Table 4: Possibilities for $x_{[\texttt{A}(d_i),\texttt{R}(d_i)]}$ depending on whether Edna has access to diffs from d_i regarding x.

We first consider $x_{[\texttt{A}(d_i),\texttt{R}(d_i)]}$; Table 4 illustrates how this state depends the authorization a client invoking $\texttt{R}(d_i)$ provides to Edna to access d_i's diffs for x. As expected, the updates applied by d_i to x cannot be revealed when the relevant diffs are inaccessible.

d_1 and d_2's Updates	**Access to d_1's diffs while $\texttt{R}(d_1)$?**	
	Yes	**No**
Independent	$x_{[\texttt{A}(d_2)]}$	$x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$
Conflict	$x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$	$x_{[\texttt{A}(d_1),\texttt{A}(d_2)]}$

Table 5: $x_{[\texttt{A}(d_1),\texttt{A}(d_2),\texttt{R}(d_1)]}$ depending on whether Edna has access to d_1's diffs when revealing d_1, and whether updates from d_1 and d_2 to x modify the same data.

We next consider $x_{[\texttt{A}(d_1),\texttt{A}(d_2),\texttt{R}(d_1)]}$: the end state of x when a disguise d_1 is revealed after a subsequent disguise d_2 has been applied. Table 5 illustrates how the result depends on Edna's access to d_1 diffs, and whether d_1 and d_2 performed conflicting modifications to x's attributes.

When the client invoking $\texttt{R}(d_1)$ provides no authorization to access to d_1's diffs, then the updates applied by d_1 to x cannot be revealed. However, even if the client grants access to d_1's diffs, Edna will fail to reveal d_1's updates to x if d_2 and d_1 both modify x in a conflicting manner: $\texttt{R}(d_1)$ must not restore data that was also disguised by d_2, and until d_2 is revealed, invocations of $\texttt{R}(d_1)$ will not reveal $x_{\texttt{start}}$.

4.1 Composition Implementation Techniques

Edna's algorithm for disguising and disguise reversal in most scenarios is straightforward.

To apply disguise d_i, Edna applies each $\texttt{op}_{d_i}$ to all data objects satisfying $\texttt{op}_{d_i}$'s predicate, while also taking into account the information in accessible diffs(to e.g., predicate against undisguised versions of objects). Each $\texttt{op}_{d_i}$ produces one or more diffs, which Edna stores either globally or encrypted by a data capability at $\mathcal{C}^{\texttt{loc}}_{pi_i}$.

To reveal disguise d_i, Edna reveals the undisguised version of data stored in accessible diffs corresponding to d_i by updating the relevant objects x in the database.

However, Edna must be careful of two potential problems: (1) Edna cannot accidentally reveal data that must be disguised; and (2) Edna cannot let global diffs recording updates to data x leak information if x is subsequently (privately) disguised.

Preventing Accidental Data Revelation. Edna records the disguised state of data x in each Δ_{pd} recording a update performed by d_i to x. If the state of x does not match the disguised state in d_i's diff for x, then Edna knows the diff records a stale and overwritten update to x, and refuses to reveal the undisguised state of x stored in the diff.

Preventing Global Diff Information Leakage. A disguise may optionally store diffs in global storage, accessible to Edna without any client authorization. Such a global disguise provides no privacy guarantees against an adversary who compromises the application server, but does transform the application database so external users see disguised data.

If d_1 is a global disguise and d_2 a normal, privacy-preserving disguise, $\texttt{op}_{d_2}$ may update a data object x previously updated by $\texttt{op}_{d_1}$.

$\texttt{op}_{d_1}$ produces a global diff Δ_{pd_1}; $\texttt{op}_{d_2}$ produces a normal, encrypted and secured diff Δ_{pd_2}.

If Edna simply performs $\texttt{op}_{d_2}$'s update to x, an adversary can still read undisguised data from the Δ_{pd_1} diff because the diff is global and records a now-stale state of x! To prevent global diffs like Δ_{pd_1} from leaking data that should be disguised, Edna updates the data in Δ_{pd_1} with $\texttt{op}_{d_2}$'s update. An update to a diff such as Δ_{pd_1} itself generates a (private) diff Δ_{pd_2}, just as updating a data object generates a diff.

Revealing d_2 uses Δ_{pd_2} to reveal the modification to Δ_{pd_1}. If d_1 has already been revealed, and the database state of x updated to reflect the contents of Δ_{pd_1} (which were updated by the application of d_2), then Edna simply uses Δ_{pd_2} to reveal the current state of x in the database.

If a future d_3 has further updated Δ_{pd_1} in a way that conflicts with d_2's update, then upon reversal of d_2, Edna will notice that Δ_{pd_2} records an overwritten update to Δ_{pd_1}, and will not revert the state of Δ_{pd_1} until the future disguise d_3 has been revealed .

[lyt: Note that removal is just an interesting case of this, in which Δ_{pd_1} is removed, by a diff Δ_{pd_2} stored in $\mathcal{C}^{\texttt{loc}}_{pd_2}$ that saves the removal of Δ_{pd_1} for reversal.]