DeCor: Decorrelating unsubscribed users' data from their identities

Anonymous Authors

Abstract

100-200 words that convince you to read this.

1 Introduction

1.1 Motivation

Web application companies face increasing legal requirements to protect users' data. These requirements pressure companies to properly delete and anonymize users' data when a user requests to *unsubscribe* from the service (i.e., revoke access to their personal data). For example, the GDPR requires that any user data remaining after a user unsubscribes is *decorrelated*, i.e., cannot be (directly or indirectly) used to identify the user [2].

In this paper, we propose DeCor, a new approach to managing user identities in web applications. DeCor meets the decorrelation requirements in the GDPR, and goes beyond: with DeCor, it is possible for users to switch between a privacy-preserving unsubscribed mode and an identity-revealing subscribed mode at any time. This facilitates important new web service paradigms, such as users granting a time-limited "lease" of data to a service instead of having a permanent service account.

1.2 Goals

DeCor's goal is to provide the following properties while preserving an application's semantics:

Decorrelation. Decorrelation ideally guarantees that it is impossible to distinguish between two records formerly associated with the same unsubscribed user and two records from different unsubscribed users.

Resubscription. Users should be able to easily switch between a privacy-preserving unsubscribed mode and an identity-revealing subscribed mode, without permanently losing their application data.

DeCor must implement these properties while ensuring (1) performance comparable to today's widely-used databases, and (2) easy adoption (decorrelation should be automated without needing to modify application schemas or semantics).

1.3 Decorrelation Guarantees

We address applications in which application data consists of *data records* and computations (such as aggregations) that may be performed over these data records. Data records are considered sensitive, private data records when they contain *user identifiers*; for example, a row in a table containing a column of user IDs would be a user's private data record. Data records containing multiple, potentially different, user IDs are considered shared data records private to the identified users. We refer to data records belonging to unsubscribed users as *remnants*.

Perfect decorrelation is achieved when queries to the application reveal no information allowing an observer to determine if two distinct user data records belong to the same user who has since unsubscribed (are *linked*), or belong to two different (real or unsubscribed) users. Observers gain no information that allows them to distinguish the two scenarios.

More formally, given user data records and a subset of R of these records that are remnants, we divide the R remnants among some number N of unsubscribed users. Perfect decorrelation guarantees that any division of the R remnants among N users is equally likely: each remnant is equally likely to be linked with any other remnant.

[lyt: I toyed with saying that $P(N = R) = P(N \leq R)$: the probability that every distinct remnant belongs to a distinct user equals the probability that any user owns multiple remnants, but this seemed unnecessarily complex when I believe that the above statement captures decorrelation.]

[lyt: Note: whether N is known or not also plays into the probability analysis]

[lyt: Note: this is the definition in which ghosts can be distinct from real users, so we're not trying to pretend that remnants cannot be identified]

In practical settings, however, perfect decorrelation is likely impossible: for example, a reposted screenshot may leak user IDs. Furthermore, an observer who can see application queries over time, or search web archives, can detect when a user unsubscribes and refer to prior snapshots in which user data records may have had the same user ID. Given these limitations, we seek to achieve the maximum decorrelation guarantees possible while assuming that the content of user data does not itself leak identifying information, and that observers cannot access past application database state. [lyt: Note: UIDs can also include things like email addrs, phone number, etc; so the notion of "identifiers" might be more broad than stated here].

Global Placeholder. A common strawman solution to decorrelation is to replace all unsubscribed user IDs with one global placeholder ID. This means that all remnants can be identified by one distinct UID, which we call *GP*.

A global placeholder can provide either complete privacy or no privacy at all. We first begin by assuming an observer has complete information about how many users are in the system (N).

- If $N = 1$ and only one user has unsubscribed, all remnants can be linked back to that user's identity: any data record with UID equal to *GP* belongs to that user.

- If all users unsubscribe, two remnants are equally as likely to belong to any two of the N individual users as they are to belong to one single user.

The amount of linkable information [lyt: (need to define this)], and the resulting deviance from perfect decorrelation, decreases as more users unsubscribe. Let us assume that an observer knows N and R, and the number of data records per user is uniformly distributed. Let $r = \frac{R}{N}$ be the number of remnants per unsubscribed user. Then an observer has probability p_{exact} of guessing the correct combination of records per users, where

$$p_{\text{exact}} = \frac{(r!)^N}{R!}$$

An observer has probability p_{user} of guessing the correct combination for one user where

$$p_{\text{user}} = \frac{r!(R-r)!}{R!}$$

[lyt: TODO—non-uniform distributions, not knowing N, not knowing R?]

Using a global placeholder, however, makes resubscription challenging: users can no identify which unsubscribed data records belong to them if all user-specific data has been erased from the system.

Data-Record Ghost IDs. To support [lyt: the same?]~~stronger~~ decorrelation guarantees while supporting resubscription, DeCor generates a unique ghost user for each data remnant. Unlike a global placeholder, DeCor allows users to reactivate their account and undo the decorrelation: user IDs can be linked back to a set of unique ghost IDs. This gives users the ability to freely unsubscribe to protect their privacy without worrying about losing their accounts. DeCor resubscribes users by transparently propagating updates to materialized views to expose real user identifiers in place of ghost identifiers.

DeCor relies on coarse-grained schema annotations to establish which associations to decorrelate, and builds a dataflow computation resulting in materialized views that answer application queries. Use of dataflow automatically propagates the correct updates to materialized views.

~~These ghosts, unlike a global placeholder, are indistinguishable from a real user in the system, ensuring that queries cannot correlate two ghosts with a single real (albeit unsubscribed) user. However, some linkable information is still leaked. For example, ghost users may be randomly generated in a pattern identifiable by an observer (e.g., if all ghosts have usernames which are random numbers, or arbitrary animals, but a real user may have more human-friendly usernames).~~

[lyt: Not sure how to incorporate DP or some notion of noninterference here (how do DeCor's responses to queries deviate from what an ideally decorrelated system would produce?) DP seems to deal with the change in probability of some event X, rather than the initial probability to begin with. Perhaps DP would be more applicable if we were looking at the database over time? And if we had some notion of noise. Without adding noise, the output would change dramatically when a user unsubscribes and UIDs are replaced by GIDs, which reveals complete linkability information.]

2 Background and Related Work

Companies have developed frameworks to avoid deletion bugs (e.g., DELF [1] at Facebook), but many applications decorrelate only coarsely by associating remnants of data with a global placeholder for all deleted users, or do not decorrelate at all (e.g., replacing usernames with a pseudonym).

[lyt: (cite Reddit, Lobsters, others?)]

3 Design

4 Implementation

5 Evaluation

6 Discussion

Acknowledgments

Availability

[lyt: USENIX program committees give extra points to submissions that are backed by artifacts that are publicly available. If you made your code or data available, it's worth mentioning this fact in a dedicated section.]

References

[1] Katriel Cohn-Gordon, Georgios Damaskinos, Divino Neto, Joshi Cordova, Benoît Reitz, Benjamin Strahs, Daniel Obenshain, Paul Pearce, and Ioannis Papagiannis. DELF: Safeguarding deletion correctness in online social networks. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, August 2020.

[2] E Parliament. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *Official Journal of the European Union*, L119(May 2016):1–88, 2016. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.