Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

Proposal for Thesis Research in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

Title: Helping Web Developers Give Users Control Over Their Data

Submitted by: Lillian Tsai
Stata Center
Cambridge, MA 02139

Signature of author: _____

Date of Submission: February 14, 2023

Expected Date of Completion: May 25, 2024

Laboratory: CSAIL

Brief Statement of the Problem:

Users today have little to no control over the data they give to web applications. They must either avoid using an application, or leave their data susceptible to leaks, e.g., via SQL injections or compromise of other users' accounts. Incentivized by growing public demand and laws such as the GDPR and CCPA [4, 11], developers increasingly need to implement mechanisms to give users control over their data's exposure.

Supervision Agreement:

The program outlined in this proposal is adequate for a PhD thesis. The supplies and facilities required are available, and I am willing to supervise the research and evaluate the thesis report.

Frans Kaashoek, Charles Piper Professor of EECS
Malte Schwarzkopf, Professor of Computer Science, Brown University

Helping Web Developers Give Users Control Over Their Data

Lillian Tsai
Massachusetts Institute of Technology
tsilyai@mit.edu

February 14, 2023

1 Introduction

The question of who controls personal data, and how it is used, has increasingly surfaced in the web ecosystem. Recent legal developments such as the GDPR and CCPA, as well as growing public demand and news coverage of e.g., data breaches, have increased the need for better user data handling in web applications. This thesis identifies key challenges that web developers face today in implementing better user data controls, and contributes practical systems that can help to give control of user data in web services back to the user. In particular, this thesis introduces a novel system design that allows applications to implement a variety of user data controls while still remaining functional.

2 Motivation

Users today have little to no control over the data they give to web applications. They must either avoid using an application, or leave their data susceptible to leaks, e.g., via SQL injections or compromise of other users' accounts. Incentivized by growing public demand and laws such as the GDPR and CCPA [4, 11], developers increasingly need to implement mechanisms to give users control over their data's exposure.

Central to this conversation are two needs: users want to protect their data, and applications require data to be available in order to function. Problems arise when application functionality requires the same data users want to protect: for example, deleting a post may orphan other users' comments on it, crashing the application when it expects comments to have an associated post. Handling this correctly is difficult for developers—naively removing user data can break the application or reduce its functionality.

Removal is also usually permanent, hurting both users and applications: once a user deletes their data, there is no coming back.

Large companies have built systems to help developers get data deletion right [9], but the problem goes beyond data deletion. Users may want to take a break from using an application, but rest easy that their data is protected while they're inactive; or users may want more fine-grained control—i.e., have their data exist in a state where it is both visible and protected. For example, a user might want to disassociate content with a specific hashtag from a social media account. Because applications today lack support for this, users resort to ad-hoc alternatives like throwaway accounts and "finstas" [22, 31] to post content that is visible but disconnected from the "real" account. This burdens users with managing multiple accounts and requires planning ahead, since moving posts into a throwaway account or reclaiming them is impossible.

These problems call for research into systems that let web application developers provide users with means to manage their data in the gray space where users' need for data protection and application functionality requirements overlap.

3 Background and Related Work

Existing systems for data protection aim to support data deletion, prevent unauthorized data access, or protect against database server compromise. However, none of these systems are explicitly designed to flexibly handle user data (via fine-grained removals or redactions), and to do so without breaking the application. Instead, many of these systems provide orthogonal or partial solutions.

Laws and regulations such as the EU's General Data Protection Regulation (GDPR) [11], the California Consumer Privacy Act (CCPA) [4], and others [28, 29] give users the right to request deletion of their data from web applications. **Data deletion tools** developed at large organizations, such as DELF at Meta [9], help to correctly delete data in response to such requests. DELF helps developers specify correct deletion policies via annotations on social graph edges and object types, and ensures correct cascading data deletion. However, data deletion systems support only simple deletion, and do not address the entire range of possible user data controls (e.g., temporarily removing old data of inactive users).

Policy enforcement systems such as Qapla [18] aim to prevent unauthorized access to data and protect against leakage via compromised, unauthorized accounts or SQL injections. They enforce developer-specified visibility and access control policies via information flow control [8, 12, 14, 26, 32], authorized views [2] or per-user views [17], or blocking or rewriting database queries [18, 21, 33].

Encrypted storage systems such as CryptDB [23] and Mylar [24] aim to protect against database server compromise (with some limitations [13]), as well as SQL injection attacks. These systems encrypt data in the database, and ensure that only users with legitimate reasons to view the data (i.e., the owner or an admin) can decrypt the data. Applications handle keys, and send queries either through trusted proxies that decrypt data [23], or move application functionality client-side [24].

Policy-enforcing and encrypted storage systems do not help users redact data, nor maintain application functionality in the face of partially or fully redacted data. Their goal is orthogonal: to protect unauthorized data access via access control mechanisms (encryption or policy checks). These systems also keep data available in the database, thus leaving user data vulnerable to attacks by compromised accounts with authorized access (e.g., an admin).

3.1 Other Related Work.

Some systems modify the data layout [27] or use fine-grained information flow tracking [15] to track the ownership and provenance of database rows. This tracks how information propagates and provides information to support wholesale user data deletion. However, these solve orthogonal problems than the one of practical user data controls, and do not support more nuanced redactions such as partial anonymization of shared data.

Sypse [10] pseudonymizes user data and partitions personally-identifying information (PII) from other data. However, Sypse does not securely store (e.g., with encryption) PII, nor modify the actual data stored by the application

Decentralized platforms such as Solid [25], BSTORE [7], Databox [19], and others [1, 5, 6, 16, 20] put data fully and directly under user control, since users store their own data. But this burdens users with maintaining infrastructure, and decentralized platforms lack the capacity for server-side compute and break today's ad-based business model.

Some platforms can prove that server-side processing respects user-defined data policies via cryptographic means [3] or systems security mechanisms [30]. This may restrict feasible application functionality (e.g., to additively homomorphic functions), or restrict combining data with different policies.

4 Approach

Edna is a library that integrates with web applications and provides abstractions that let developers specify not just wholesale removal of data, but also flexible redaction or decorrelation of data. Edna then changes the database contents in well-defined ways that avoid breaking the application. Developers and users both benefit from Edna: applications can expose and advertise new privacy-protection features and reduce their exposure in the event of a data breach, while users gain more control over their sealed data, including the convenience of returning to the application as if their data had been there all along.

Edna introduces *sealing*, which removes or redacts some or all of a user's data, and *revealing*, which restores the sealed data at a user's request. Sealed data remains on the server, but is encrypted and inaccessible to the web application. Sealing changes the database contents and replaces the data to seal with placeholder values where necessary (e.g., comments require an associated post).

Edna supports user data control in web applications using application-specific *sealing transformations*. To integrate an application with Edna, the developer writes seal specifications and adds hooks to seal or reveal data using Edna's API. Applications (1) register users (human principals) with Edna, (2) invoke Edna to apply transformations that seal user data, and (3) invoke Edna to reveal that data when requested.

(1) Applications register users with a public–private keypair that either the application or the user's client generates; Edna stores the public key in its database, while the user remembers the private key for use in future reveal operations.

(2) When the application wants to seal some data, it invokes Edna with the corresponding developer-provided seal specification and any necessary parameters (such as a user ID). Seal specifications can remove data, modify data (replacing some or all of its contents with placeholder values), or decorrelate data, replacing links to users with links to pseudoprincipals (fake users). Edna takes the data it removed or replaced and the connection between the user and any pseudoprincipals it created, encrypts that data with the user's public key, and stores the resulting ciphertext—the *sealed data*—such that it cannot be linked to the user without the user's private key.

(3) When a user wishes to reveal their sealed data, they pass credentials to the application, which calls into Edna to reveal the data. Credentials are application-specific: users may either provide their private key or other credentials sufficient for Edna to re-derive the private key. Edna gets the sealed data and decrypts it, undoing the changes to the application database that sealing introduced.

Edna provides the developer with sensible default sealing and revealing semantics (e.g., revealing makes sure not to overwrite changes made since sealing), which the developer can customize via a low-level API provided by Edna.

Threat Model. Edna assumes well-intentioned developers who write seal specifications that capture the desired application semantics. Edna seals data according to these specifications, but because some data must be retained in the database to keep application semantics intact, Edna cannot protect against inference attacks. Nevertheless, Edna protects a user's *sealed* data against:

1. arbitrary application database accesses (e.g., SQL injection attacks);
2. public exposure or other users' ability to view the data through the application (e.g., compromised accounts, including privileged ones); and
3. remote code execution with the application's privileges (e.g., PHP's `eval()`).

Edna guarantees that a user's sealed data is protected unless an attacker obtains their credentials. While Edna hides the contents of sealed data and relationships between sealed data and users, it does not hide the existence of sealed data. (An attacker can see whether a user has sealed some data, but cannot see which sealed data corresponds to this user.) Some transformations, especially decorrelation, cannot protect against statistical inference attacks over information still visible in the database. Edna also cannot handle copies of data embedded in other data (e.g., quoted text) or snapshots saved when the data was still in the database. We make standard assumptions about the security of cryptographic

primitives: attackers cannot break encryption, and keys stored with clients are safe.

5 Contributions

Edna provides the following contributions:

1. A new paradigm for developers to provide flexible user control over data in web applications, with cryptographic protection and while maintaining application functionality and invariants.
2. Abstractions for sealing and revealing, and a small set of data-anonymizing primitives (remove, modify, decorrelate) that cover a wide range of application needs and compose cleanly.
3. The design and implementation of Edna, our prototype library that implements user data control via sealing and revealing, as well as Ednacrypt, which additionally encrypts unsealed data.
4. Case studies of integrating Edna with three real-world web applications, an evaluation of Edna's effectiveness, performance, and security, and a comparison to Qapla.

5.1 Design

5.2 Evaluation

Edna is complementary to existing data protection mechanisms. For example, we combine Edna with an encrypted database to achieve stronger guarantees. Ednacrypt simultaneously protects against database server compromises even for unsealed data and adds Edna's protections for sealed data to encrypted databases, which have no built-in support for removing sensitive data without breaking the application.

To investigate the need for Edna as a new system, we tried to realize Edna's functionality atop Qapla [18], a framework that rewrites SQL queries to conform to access control policies. We found that Qapla requires invasive application changes, its abstractions are awkward for sealing and revealing, and Qapla's query rewriting slows down common queries.

5.3 Limitations

Edna has some limitations. Its usability goals, and specifically the fact that placeholder data continues to exist in the database, prevent Edna from protecting against inference attacks (e.g., statistical correlation attacks). Edna also assumes bug-free seal specifications, and that applications use Edna correctly. Finally, Edna provides limited protection for sealed data against attackers who compromise the database server, as auxiliary database information (e.g., logs) may contain old plaintext data; Edna-CryptDB strengthens these protections.

5.4 Extensions: Hydra

Hydra narrows down on a specific aspect of Edna, namely how users control their identities and links between their identities and data in web applications. Users today will manually implement ad-hoc alternatives like throwaway accounts and "finstas" [22, 31] to post content that is visible but disconnected from the "real" account.

Hydra provides a library that enables web applications to provide their users inbuilt for splintering and merging their web identities within the application. With Hydra, applications can let users decorrelate data to context-specific identities (e.g., for work or for family), easily move their data between identities, spin off new identities when necessary, and merge identities if desired. Importantly, only the user can determine which web application identities are linked to themselves. Hydra explores the result of taking application-supported data decorrelation to the extreme, while preserving application functionality and protecting correlations between data and users. Applications both retain (decorrelated) data for e.g., analytical or advertising purposes, while users gain security and control over what data is linked to which of their web application identities.

6 Timeline

- **Spring 2023**:
  - Focused work on Funhouse to bring it to paper-ready status.
  - Prepare a conference-ready paper and talk on Edna.
  - Complete the RQE using an Edna talk.
  - Complete the thesis proposal (this document).
- **Summer 2023**:
  - Focused work on Funhouse, or potential internship.
  - Potential conference presentation of Edna.
  - Potential workshop presentation of Funhouse HotOS.
- **Fall 2023**:
  - Funhouse conference submission if ready.
  - Prepare job talk materials (Funhouse or Edna).
  - TA (OS or systems security)
- **Spring 2024**:

- Finalize last Edna-related thesis research (Hydra).
- Finalize Funhouse contributions for thesis, if any.
- Write and defend thesis.

References

[1] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J. Freedman. Blockstack: A global naming and storage system secured by blockchains. In *Proceedings of the 2016 USENIX Annual Technical Conference (ATC)*, pages 181–194, Denver, Colorado, USA, June 2016. ISBN 978-1-931971-30-0. URL https://www.usenix.org/conference/atc16/technical-sessions/presentation/ali.

[2] Kristy Browder and Mary Ann Davidson. The virtual private database in oracle9ir2. *Oracle Technical White Paper, Oracle Corporation*, 500(280), 2002.

[3] Lukas Burkhalter, Nicolas Küchler, Alexander Viand, Hossein Shafagh, and Anwar Hithnawi. Zeph: Cryptographic enforcement of end-to-end data privacy. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 387–404, July 2021. ISBN 978-1-939133-22-9. URL https://www.usenix.org/conference/osdi21/presentation/burkhalter.

[4] California Legislature. The California Consumer Privacy Act of 2018, June 2018. URL https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375.

[5] Tej Chajed, Jon Gjengset, Jelle van den Hooff, M. Frans Kaashoek, James Mickens, Robert Morris, and Nickolai Zeldovich. Amber: Decoupling user data from web applications. In *15th Workshop on Hot Topics in Operating Systems (HotOS)*, Kartause Ittingen, Switzerland, May 2015. USENIX Association. URL https://www.usenix.org/conference/hotos15/workshop-program/presentation/chajed.

[6] Tej Chajed, Jon Gjengset, M. Frans Kaashoek, James Mickens, Robert Morris, and Nickolai Zeldovich. Oort: User-centric cloud storage with global queries. Technical Report MIT-CSAIL-TR-2016-015, MIT CSAIL, 2016.

[7] Ramesh Chandra, Priya Gupta, and Nickolai Zeldovich. Separating web applications from user data storage with bstore. In *Proceedings of the 2010 USENIX Conference on Web Application Development*, WebApps'10, page 1, USA, 2010.

[8] Adam Chlipala. Static checking of dynamically-varying security policies in database-backed applications. In *Proceedings of the 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 105–118, 01 2010.

[9] Katriel Cohn-Gordon, Georgios Damaskinos, Divino Neto, Joshi Cordova, Benoít Reitz, Benjamin Strahs, Daniel Obenshain, Paul Pearce, and Ioannis Papagiannis. DELF: Safeguarding deletion correctness in online social networks. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*, August 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/cohn-gordon.

[10] Amol Deshpande. Sypse: Privacy-first data management through pseudonymization and partitioning. In *The Conference on Innovative Data Systems Research (CIDR)*, Chaminade, CA, January 2021.

[11] European Union. 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*, L119:1–88, May 2016. URL http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.

[12] Daniel B. Giffin, Amit Levy, Deian Stefan, David Terei, David Maziéres, John C. Mitchell, and Alejandro Russo. Hails: Protecting data privacy in untrusted web applications. In *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 47–60, Hollywood, California, USA, October 2012. ISBN 978-1-931971-96-6. URL https://www.usenix.org/conference/osdi12/technical-sessions/presentation/giffin.

[13] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. Why your encrypted database is not secure. pages 162–168, 05 2017. doi: 10.1145/3102980.3103007.

[14] Katia Hayati and Martín Abadi. Language-based enforcement of privacy policies. In *International Workshop on Privacy Enhancing Technologies*, pages 302–313. Springer, 2004.

[15] Tim Kraska, Michael Stonebraker, Michael Brodie, Sacha Servan-Schreiber, and Daniel Weitzner. Schengendb: A data protection database proposal. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, pages 24–38. Springer, 2019. ISBN 978-3-030-33752-0.

[16] Maxwell Krohn, Alex Yip, Micah Brodsky, Robert Morris, and Michael Walfish. A World Wide Web Without Walls. In *Proceedings of the 6th ACM Workshop on Hot Topics in Networking (HotNets)*, Atlanta, Georgia, USA, November 2007.

[17] Alana Marzoev, Lara Timbó Araújo, Malte Schwarzkopf, Samyukta Yagati, Eddie Kohler, Robert Morris, M. Frans Kaashoek, and Sam Madden. Towards multiverse databases. In *Proceedings of the Workshop on Hot Topics in Operating Systems (HotOS)*, pages 88–95, 2019.

[18] Aastha Mehta, Eslam Elnikety, Katura Harvey, Deepak Garg, and Peter Druschel. Qapla: Policy compliance for database-backed systems. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security)*, pages 1463–1479, Vancouver, British Columbia, August 2017. ISBN 978-1-931971-40-9. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/mehta.

[19] Richard Mortier, Jianxin Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James Colley, Tom Lodge, et al. Personal data management with the databox: What's inside the box? In *Proceedings of the 2016 ACM Workshop on Cloud-Assisted Networking*, pages 49–54, 2016.

[20] Shoumik Palkar and Matei Zaharia. Diy hosting for online privacy. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks (HotNets)*, pages 1–7, 2017.

[21] Primal Pappachan, Roberto Yus, Sharad Mehrotra, and Johann-Christoph Freytag. Sieve: A middleware approach to scalable access control for database management systems. *arXiv preprint arXiv:2004.07498*, 2020.

[22] Pranay Parab. How to make a burner account on reddit, even though they don't want you to anymore. Lifehacker, January 2022. URL https://lifehacker.com/how-to-make-a-burner-account-on-reddit-even-though-the-1848336857.

[23] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proceedings of the 23th ACM Symposium on Operating Systems Principles (SOSP)*, page 85–100, 2011. ISBN 9781450309776. doi: 10.1145/2043556.2043566. URL https://doi.org/10.1145/2043556.2043566.

[24] Raluca Ada Popa, Emily Stark, Jonas Helfer, Steven Valdez, Nickolai Zeldovich, M. Frans Kaashoek, and Hari Balakrishnan. Building web applications on top of encrypted data using mylar. In *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, pages 157–172, 2014. ISBN 978-1-931971-09-6. URL http://dl.acm.org/citation.cfm?id=2616448.2616464.

[25] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: a platform for decentralized social applications based on linked data. Technical report, MIT CSAIL & Qatar Computing Research Institute, 2016.

[26] David Schultz and Barbara Liskov. Ifdb: Decentralized information flow control for databases. In *Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys)*, pages 43–56, 2013.

[27] Malte Schwarzkopf, Eddie Kohler, M. Frans Kaashoek, and Robert Morris. Position: Gdpr compliance by construction. In *Heterogeneous Data Management, Polystores, and Analytics for Healthcare*, pages 39–53, Cham, 2019. Springer International Publishing. ISBN 978-3-030-33752-0.

[28] Alexander H. Southwell, Ryan T. Bergsieker, Cassandra L Gaedt-Sheckter, Frances A. Waldmann, and Lisa V. Zivkovic. Virginia passes comprehensive privacy law". Gibson Dunn, March 2021. URL https://www.gibsondunn.com/virginia-passes-comprehensive-privacy-law/.

[29] Griffin Thorne. Gdpr meets its match ... in china. China Law Blog, July 2019. URL https://www.chinalawblog.com/2019/07/gdpr-meets-its-match-in-china.html.

[30] Frank Wang, Ronny Ko, and James Mickens. Riverbed: Enforcing user-defined privacy constraints in distributed web services. In *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 615–630, Boston, Massachusetts, USA, February 2019. ISBN 978-1-931971-49-2. URL https://www.usenix.org/conference/nsdi19/presentation/wang-frank.

[31] Caity Weaver and Danya Issawi. 'finsta,' explained. The New York Times, September 2021. URL https://www.nytimes.com/2021/09/30/style/finsta-instagram-accounts-senate.html.

[32] Jean Yang, Kuat Yessenov, and Armando Solar-Lezama. A language for automatically enforcing privacy policies. *ACM SIGPLAN Notices*, 47(1):85–96, 2012.

[33] Wen Zhang, Eric Sheng, Michael Chang, Aurojit Panda, Mooly Sagiv, and Scott Shenker. Blockaid: Data access policy enforcement for web applications. In *16th USENIX Symposium on Operating*

Systems Design and Implementation (OSDI 22), pages 701–718, Carlsbad, CA, July 2022. USENIX Association. ISBN 978-1-939133-28-1. URL https://www.usenix.org/conference/osdi22/presentation/zhang.