DOI:10.1145/3807515

With this look at implementing *Right to Be Forgotten* from a computing and data-management perspective, the authors attempt to bridge the disconnect between legal and computing domains.

BY CHEN SUN, NIKOLAS GUGGENBERGER, AND SUPREETH SHASTRI

Faults and Pitfalls in Implementing the Right to Be Forgotten

For every complex problem, there is an answer that is clear, simple, and wrong.

—H.L. MENCKEN

ANCIENT WISDOM SAYS that everything that has a beginning has an ending. However, when it comes to the life cycle of personal data, the ending is nowhere in sight. In fact, for much of its existence, the computing community has evolved without treating deletion as a first-class operation. This practice had to change when the European Union introduced the Right to Be Forgotten (RTBF)—first in 2014, as a standalone right applicable to online search engines and then in 2018, as a universal right applicable to all data controllers through the General Data Protection Regulation (GDPR).[a]

"Isn't it just deletion?" has been the computing community's standard reaction to RTBF's requirements. While the end goal of RTBF is indeed the deletion of data, casting RTBF as just deletion is akin to saying that eating is just for nutrition. It is not surprising that over the first five years of GDPR, an RTBF penalty is issued once every nine days[25]—a clear sign that the computing and data-management communities have continued to oversimplify, misunderstand, and poorly implement RTBF. Our work is an attempt to remedy this disconnect.

This article demonstrates how RTBF exposes computing systems to uncertainties and the challenges at all stages of design and operation, and how RTBF has invalidated principles and practices of data management with decades of precedent. To address these challenges, we propose a principled approach for introducing RTBF capability in computing systems. Our solution is rooted in two key insights:

1. The need to bridge an intrinsic dichotomy existing between computing and law; that is, computing systems are created to be precise and static, but laws are written to be abstract and interpretable

2. Modeling compliance as a *via negativa* problem; that is, instead of trying to build a perfectly compliant system, it is much easier to weed out known violations from the system.

We ground our work firmly in both

a 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. *Official Journal of the European Union 59*, 1–88 (2016).

IMAGE BY KOTOIMAGES

law and computing—the former, by basing our analysis on GDPR, the most prominent data regulation, and the latter, by implementing our findings in Elasticsearch, the most widely used open source search engine. The interdisciplinary nature of this work allows us to integrate techniques from both law (e.g., tracking legal precedents and applying the Issue-Rule-Application-Conclusion or IRAC method) and computing (e.g., layered design and anti-patterns), and thus be able to make evidence-based recommendations that previous domain-specific investigations have lacked. In particular, we make three key contributions:

1. Designing End-to-End RTBF Capability. We propose a novel two-phase approach—*law-driven design* followed by *enforcement-driven refinement*—that helps us make applications RTBF capable. We evaluate its efficacy via a longitudinal study on GDPR enforcement.

2. RTBF-Capable Elasticsearch. We analyze the RTBF capabilities of Elasticsearch and identify the functionalities that are lacking. Then, we implement timed and thorough delete operations in Elasticsearch and make it available on Github.[b]

3. RTBF Anti-Patterns. We identify six RTBF anti-patterns, which are longstanding principles and practices of computing and data-management systems that serve their original purpose well but make it difficult to support RTBF.

Background and Motivation

Right to Be Forgotten, alternatively referred to as Right to Erasure, is the right of a person to request an organization to delete their personal data. RTBF gives people the ability to prevent others from seeing their personal information that they deem inappropriate, irrelevant, or simply no longer consent to its being used. This right is distinct from right to privacy,[3] which prevents governments and other entities from forcing a person to reveal their personal data; instead, RTBF is to be used for information that is already disclosed (either to an organization or to the public), but the individual no longer wants to remain so. The need for RTBF arose in the early 21st century, when organizations began collecting personal information at scale and search engines made these accessible globally. As Viktor Mayer-Schonberger has chronicled,[18] this was in stark contrast to past human history, where forgetting was the norm and remembering was the exception.

The interdisciplinary nature of this work allows us to integrate techniques from both law and computing, and thus be able to make evidence-based recommendations that previous domain-specific investigations have lacked.

RTBF became a legal right for the first time in 2014. In a case against Google[c] the European Court of Justice ruled that people can request search engines to remove certain links from their search results, if their privacy concerns outweigh the public interest in those links. Later, when GDPR went into effect in 2018, RTBF was expanded to cover all data controllers, not just search engines. Even GDPR did not make RTBF an absolute right; that is, for an RTBF request to be honored, it has to meet one of the six conditions and not fall under one of the five exemptions (we show these in Figure 1, where we produce GDPR article 17 verbatim). To keep our discussion concrete, we focus exclusively on GDPR's version of RTBF and prefix its articles with *G*.

Challenges in complying with RTBF. When new laws are enacted, policymakers tend to limit their expositions to core legal principles that are broadly interpretable and will stand the test of time (e.g., think how a country's constitution remains relevant even hundreds of years after drafting it). While legally prudent, this strategy leaves computing systems with several challenges: Lack of precise specifications, undefined trade-offs in cost, and uncertainties in managing new technologies, among others. We now illustrate how these can manifest at different stages of system design and operation.

Examples at design and implementation level. Systems designed with little or no prior consideration for RTBF find it hard to add that capability later. For example, an organiza-

b https://github.com/lawfulcomputing/RTBF

c Google Spain SL and Google Inc. v. Agencia Española de Protección de Datos (AEPD) and Mario Costeja González. The Court of Justice of the European Union—Reports of Cases; EUR-Lex (2014); https://tinyurl.com/22xozuav

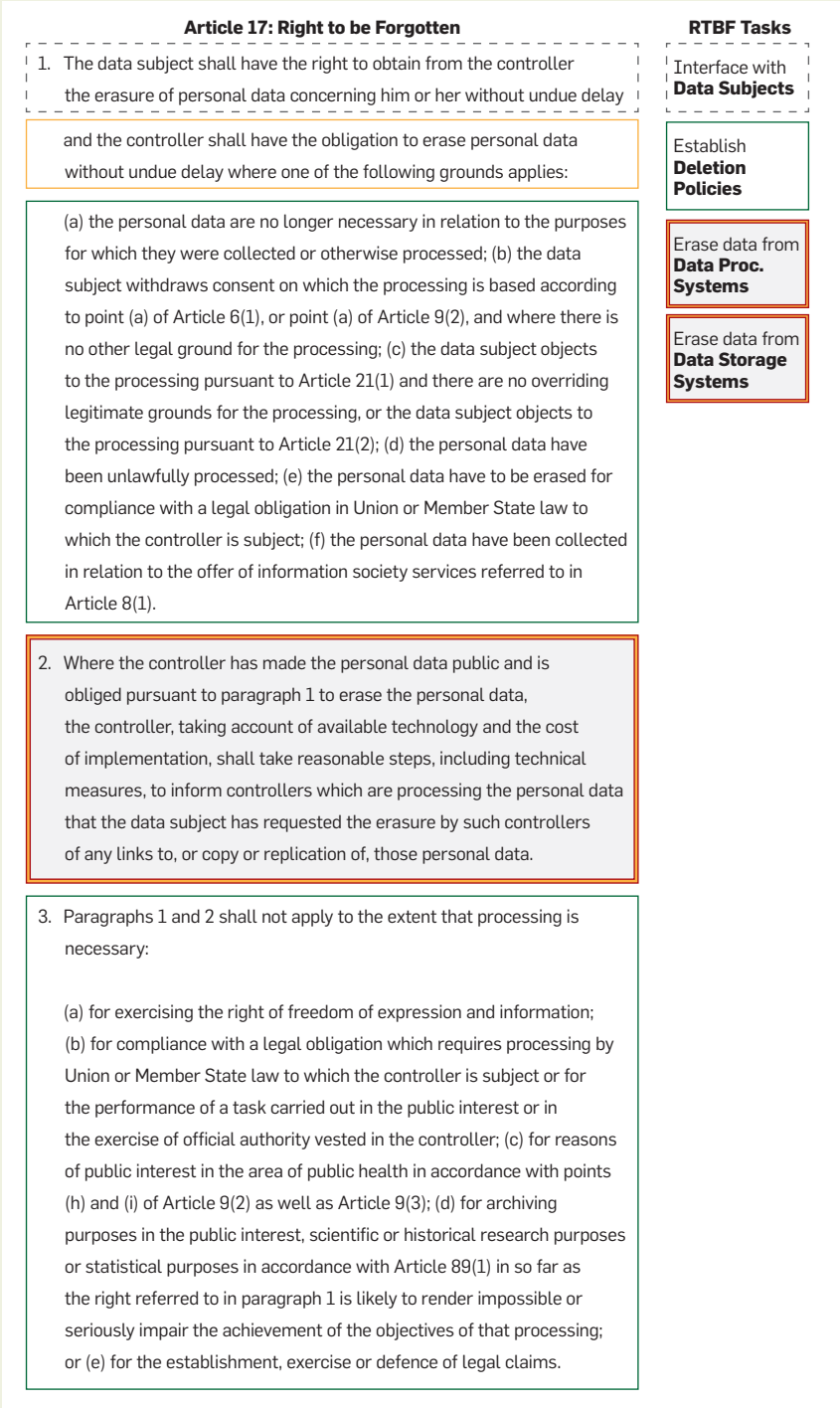
tion that uses personal data as a primary key in its databases would find it tricky to delete that item. Problems could also stem from unwise choices in organizing the data. For example, Clearview AI built its facial-recognition system by training on billions of images from the Internet. However, when people approached them with RTBF requests, the firm realized it could not identify all the photos that belonged to a given person (since it had not tagged the images at the time of collecting or processing). Clearview AI was fined in 2022 by multiple regulators[14,29,d] for this limitation.

Examples at operation level. RTBF is not an absolute right; that is, just because an RTBF request is made on valid personal data by a verified data subject does not mean it should be honored. GDPR requires all controllers to balance the rights of individuals with the interests and obligations of society. This is challenging for organizations, since it turns RTBF from a *generic operation* to a *highly individualized process*, thereby making it difficult to fully automate. The gravity of this challenge is evident when you consider that RTBF is operated largely as a manual process at Google and Microsoft,[1,e] and that they take about six days, on average, to arrive at an RTBF decision.

Related work. *RTBF in computing systems.* Several organizations including Google,[1] Microsoft,[e] and Wikipedia[f] have shared details about how their applications support RTBF. These reports primarily focus on aggregate-level characterization of the received RTBF requests and their responses but do not offer much detail on the computing aspects of their internal RTBF implementation. Orthogonal to this perspective, researchers have explored how people perceive RTBF support on social media websites[19] and the challenges they face while exercising RTBF.[13,26,30] This body of work focuses on human-computer interfacing for RTBF and treats the target computing systems and services as black boxes. In contrast, we focus on end-to-end implementation of RTBF.

Deletion in computing systems. RTBF and the onset of other data rights have brought renewed relevance for the deletion operation. Two practical examples of this trend are Google Cloud publishing its deletion pipeline and guaranteeing to erase all copies of the data within 180 days of a request,[8] and Facebook designing a system that can assure correctness of deletion.[16] Other key advances

Figure 1. Mapping RTBF's legal requirements into the computing domain. We identify four key tasks of RTBF-capable systems (on the right), and the text of the regulation that requires these (on the left).



d CNIL. Facial recognition: 20 million euros penalty against CLEARVIEW AI. Restricted Committee Deliberation No. SAN-2022-019. (Oct. 20, 2022).

e Microsoft. "Right to be forgotten" requests (2023); https://tinyurl.com/27zm4jqj

f Wikimedia Foundation. Transparency Reports (2023); https://tinyurl.com/2y65vsrh

in deletion research include Lethe,[23] a key-value database that lets users control deletion latency, and a cryptographic framework for data deletion by Garg et al.[10] The notion of machine unlearning, that is, making ML systems forget all they learned from given data, has gained traction in the AI/ML community.[5] These works primarily focus on making the deletion efficient for certain models[2,17,31] or specific applications.[6,11]

A key trait of all these works is they use RTBF as a motivation to implement deletion in a target computing system (say, databases, cloud computing, AI/ML systems, and so on), but they largely abstract out the legal and policy aspects of RTBF. A central thesis of this article is that RTBF is not just deletion, and by ignoring the interplay between law and computing, these prior works fail to recognize important system design issues and trade-offs.

Designing for RTBF

We propose a novel two-phase approach to designing RTBF capability. This is motivated by a core dichotomy between computing and law—namely, while computing applications are created to be precise and specific, laws are written to be abstract and interpretable—and our attempt to bridge this divide. In phase-1, we analyze the *language of the law* (which changes rarely) and map it to a set of high-level computing tasks. In phase-2, we examine the *enforcement of the law* (which evolves frequently) to identify available design choices and trade-offs, and to weed out noncompliant ones.

Phase-1: Law-driven design. *Mapping legal intentions to computing tasks.* By analyzing RTBF from a computing perspective, we identify four key tasks that cover all of its legal requirements. Figure 1 highlights this process by showing the legal text, the computing tasks, and the mapping between the two. The first task, *interfacing with data subjects*, encompasses all IO operations of the RTBF machinery. The second task *establishes deletion policies* the organization will follow in determining whether an RTBF request meets the legal standard to be honored or be rejected. The third task is responsible for *erasing data from all data-processing systems,* including application software, libraries, OS and system software, hardware processors, cloud, and other external processing systems. The last task involves *erasing data from storage systems* that offer long-term, persistent storage, thereby completing the RTBF process. It is important to keep the RTBF tasks high-level in phase-1, so we do not lose the generality and interpretability afforded by the law, while at the same time, we set up a foundation to methodically explore low-level design and implementation choices in phase-2.

Phase-2: Enforcement-driven refinement. Enforcement is the process of ensuring people and organizations are complying with the law. This is carried out by Data Protection Authorities (DPAs) and EU courts that investigate any reported or discovered violations of GDPR, interpret what the law requires of that situation, and then adjudicate any punishment, correction, or fine as appropriate. Such enforcement decisions collectively identify a set of behaviors that is noncompliant with the current interpretation of the law. In the legal domain, this is commonly referred to as the *precedent,* or *state of the art*.

In phase-2, we use this collective knowledge from RTBF enforcements to refine the design and operation of computing systems. To do so, we use the enforcement corpus from GDPRxiv,[25] an open source project that automatically crawls, collects, and annotates decisions by DPAs and courts. While there were 4,206 enforcement decisions in the first five years of GDPR (between May 25, 2018, and May 25, 2023), we are interested in 205 of them that cite *G*17. Our analysis and findings in the rest of the section are based on these RTBF-related decisions.

Understanding RTBF enforcements in the field. We undertook a qualitative analysis on the RTBF corpora to identify how RTBF enforcements influence the design and trade-offs in computing systems. This analysis was performed by the lead author, a Ph.D. candidate in computer science with prior research experience in GDPR. First, by employing an open coding methodology, she extracted the main reason(s) that led to the RTBF violation. While a vast majority of them stemmed from a single reason, 18% of them pointed to two reasons, and 1.5% of them had three reasons for failure. Next, based on the failure reason, each of these cases was mapped to one or more of the foundational blocks of RTBF. To validate the correctness and accuracy of the process, the other two authors of this article (professors in law and computing) randomly picked 10% of the documents to carry out coding and mapping independently. We did not find any substantive discrepancies between the two analyses. Though this *catalog of failures* is a useful reference for RTBF designers, the annotations are too lengthy to be included verbatim, so we make them available on the project website. However, the failures exhibit several recurring themes and patterns, which allow them to be studied in logical groups according to the RTBF properties they violate. Presented next are eight of the most commonly occurring themes.

1. Exercising RTBF. Regulators have issued several sanctions against controllers who made it more difficult for people to exercise RTBF. In the first category (which we call *UI-TROUBLE*), we group eight cases where controllers did not specify how to submit RTBF requests, pointed to RTBF webpages that errored out, or provided RTBF emails to which mail was undeliverable, among others. In one case, a Spanish regulator fined Google for making people fill out a pre-questionnaire before allowing them to submit RTBF requests. The second broad category (*VERIFY-MORE*) includes eight cases where controllers employed stringent verification before allowing RTBF. For instance, an Austrian online service required a physical signature; Groupon and Twitter required a national ID. These practices were deemed disproportionate since such verifications were not required when creating accounts or using their services.

2. Response time. It measures the time between making an RTBF request and receiving the first response from the controller. An upper bound comes from *G*12(3), which states all

data right requests should be responded to within one month of receipt (while allowing exemptions for cases requiring complex processing). In 54 cases, the controller never responded (we refer to this category as *NO-RESPONSE*) and in 25 cases, the controller took more than 30 days to respond (category *LATE-RESPONSE*). These enforcements have clarified the minimum expectation: Controllers must acknowledge RTBF requests within a month of receipt, though RTBF decision making and actual data deletion may take longer.

3. Explainability. This represents the ability of a data subject to understand how their RTBF request was handled. There are eight cases in this category (we denote them by *EXPLAIN*), all of which made it clear that explanations are important when rejecting RTBF requests. For instance, in two separate cases against Google and Spotify, regulators agreed the companies were correct in rejecting the RTBF requests but fined them for lack of a proper explanation in their responses. So, the prudent option when rejecting RTBF is to include a precise and individualized explanation.

4. Exemption handling. This is the process of vetting an RTBF request to ensure it meets all the legal criteria. Thus far, 41 cases say exemptions from within GDPR were misinterpreted (we denote this as *EXMPT-GDPR*), 24 cases say the interplay of RTBF with non-GDPR laws was disregarded (category *EXMPT-OTH*), and 13 cases say both were incorrectly handled (category *EXMPT-ALL*). The complexity of this process is reflected in Google search's approach of using skilled human arbiters who take up to six days on average to arrive at an RTBF decision.[1] Despite those guard rails, Google has been fined four times for incorrect exemption handling. Interestingly, we have not seen a case where a controller was penalized for honoring an RTBF request that should have been rejected. So, it may be prudent to weigh more in favor of honoring the RTBF request unless the applicability of exemptions is glaringly obvious.

5. Deletion APIs. This category (denoted by *NO-API*) comprises 15 cases where data-processing systems did not support deletion well. The issue manifested in several forms: organizations relying on employees to manually enter delete requests to multiple disjoint systems, which led to errors; software bugs that resulted in delete requests being silently dropped before processing; setting of incorrect values for the expiry time of the data; companies switching to new software systems that could not make deletions on older data; organizations using third-party processors that did not offer deletion APIs; an AI company unable to detect all the data items in its training set that matched the RTBF criterion; and a government organization that did not have administrative privileges to delete data from their Web system.

We propose a novel two-phase approach to designing RTBF capability ... motivated by a core dichotomy between computing and law—namely, while computing applications are created to be precise and specific, laws are written to be abstract and interpretable.

6. Deletion propagation. This is the process of conveying the delete request to all software and hardware systems that have copies of the RTBF data and ensuring the data is indeed deleted. In this category (*PROPAGATE*), five enforcements have highlighted shortcomings in both internal and external propagations. For example, the Swedish regulator reprimanded Rebtel Networks for continuing to send marketing emails, even after an RTBF on email addresses was granted—a failure to propagate deletion across all internal data-processing/storage systems. On the external propagation side, the Danish regulator issued a criticism against Høje-Taastrup municipality for using a third-party processor that did not offer APIs for deletion on demand.

7. Service degradation. Enforcements in this category represent cases where the controller refused RTBF because honoring it would result in service degradation or no service to the customer. There are five examples in this category (denoted by *BAD-SERV*), including a company in Austria that could not offer loyalty program benefits to people who request RTBF via email address and phone number; Taxa in Denmark would erase all prior transactions of the customer if they exercised RTBF on phone number, in turn degrading their future service. However, Spotify was allowed to reject RTBF requests on credit card information for those customers who used free trials to prevent future

abuse of the program.

8. Database constraints. This captures issues stemming from database and storage systems that prevent (or make it difficult in) implementing RTBF. There are three enforcements in this category (which we denote by *DBMS*), including Carrefour in France could not delete email IDs and Taxa in Denmark could not remove phone numbers due to primary- and foreign-key constraints in their databases. In all these cases, regulators deemed it an insufficient reason for rejecting RTBF requests and ordered them to re-architect their data-management systems.

Other frequent decisions. We observe a number of enforcement decisions that likely stem from incompetence or ignorance of the controller. In 17 cases, the controllers falsely responded that RTBF was completed when in reality they took no such action internally (category *OMIT*); in three cases, the controllers tried to avoid RTBF responsibility by redirecting people to another controller (category *DEFLECT*); lastly, in two cases, the controllers could not authenticate the requester, as the data was obtained by unknown processes (category *NO-SRC*).

Visualizing RTBF enforcements. We plot a treemap in Figure 2a, where each rectangular box represents a named category and its area is proportional to the number of cases within that category. Also, we color the boxes to show which of the four RTBF tasks they failed. This helps us recognize two trends: First, we see that 49% of enforcements cite failures in interfacing with data subjects and 37% cite failures in policy resolution. While this distribution is skewed, it is not entirely surprising since user interface (UI) and policy failures are easier to catch and they impact people immediately. Next, we see that more than half the failures stem from three categories (*NO-RESPONSE, LATE-RESPONSE, EXMPT-GDPR*). Understanding these macro trends could help organizations reevaluate their RTBF activities and focus on areas of high enforcement.

Scope and limitations. First, we developed our two-phase approach for the specific context of GDPR and its enforcement. Though the two-phase approach is generic and could be extended to other privacy regulations, such as California's CCPA and China's PIPL, our findings may not generalize as is. Second, compliance evolves with time; that is, to stay compliant, an organization must continuously evaluate its design choices and operating practices to ensure consistency with new precedents. Thus, phase-2 is never complete but instead treated as a periodic health-check activity for systems and processes. Lastly, our approach cannot provide clarity on those aspects of RTBF where there are no precedents. For instance, there are no RTBF rulings yet that directly address generative AI technologies.

The notion of machine unlearning, that is, making ML systems forget all they learned from given data, has gained traction in the AI/ML community.

Applying Our Technique in Practice

We assess the usefulness of our technique via two questions: [EV.Q1] Could it help reduce RTBF violations in the future, and [EV.Q2] Could it strengthen the RTBF capabilities of real-world systems?

Longitudinal evaluation. To answer EV.Q1, we undertook a longitudinal study that explored: Could knowledge from the first five years of RTBF help reduce violations that occurred in year six of GDPR?

Experiment setup. Because it had been more than a year since we started the project, enforcements from year six were already available on GDPRxiv. So, we gathered all enforcements from year six (i.e., those between May 25, 2023, and May 25, 2024) that cite *G*17. First, we annotated these 40 enforcements following the same methodology as outlined in the section titled Phase-2: Enforcement-Driven Refinement. Next, we compared each of these annotations to the annotated corpora from the first five years. To do that, we employed the IRAC method.[4] The key advantage of this method is it splits the analysis into four distinct parts:

1. The *Issue* identifies the legal question or dispute that needs to be resolved.
2. The *Rule* lists all the applicable rules for the case.
3. The *Application* consists of applying the identified rules to the given situation.

4. The *Conclusion* outlines the final decision.

The first and the fourth parts of all our cases were similar, since they captured enforced failures in RTBF. However, the Rule (the failure category, e.g., *NO-API*) and Application (the specific way in which the failure manifested, e.g., using a third-party processor that did not offer deletion APIs) helped us define a mapping function, as noted in Table 1.

Then, we applied this mapping function to all year-six cases, comparing them with all the named categories and their constituent cases. A no_match was easy to determine, since we only had 15 named categories. For cases that did match a category, we iterated until a strong_match was found or until the iterations terminated, having run out of enforcement cases in all its matched categories, at which point, we noted it down as a weak_match. A complete mapping of all year-six cases, along with our justification for the mapping choices, is available on the project website. However, Figure 2b represents this pictorially, with each case being represented by a rectangular point and mappings indicated by lines (thicker lines for strong mapping and dotted lines for weak mapping).

First, we see that none of the 40 enforcements of year six created any new categories. Thirty-two cases exhibited strong_match, while the remaining eight were weak_matches. This suggests that learning about prior RTBF enforcements and then refining one's design and operation could be effective in avoiding future RTBF failures. Of course, this assumes the controllers are willing and able to rework their systems to avoid past RTBF failures once they come to know of them. Next, the figure also shows how year-six enforcements are distributed differently compared to prior enforcements. For example, we see a reduction in exemption handling, explainability, and verification categories; whereas, we note an increase in the categories of *NO-API* and *UI-TROUBLE*. This is an indication that future enforcement priorities may vary over time and evolve with the field. Though it did not happen in year six, it is quite possible that new

Figure 2. Treemap of RTBF enforcements. In (a), each box represents a named category. The area of the box is proportional to the number of cases in that category, while the color of the box indicates which of the four RTBF tasks it failed. In (b), we show strong and weak mapping of year-six enforcements.

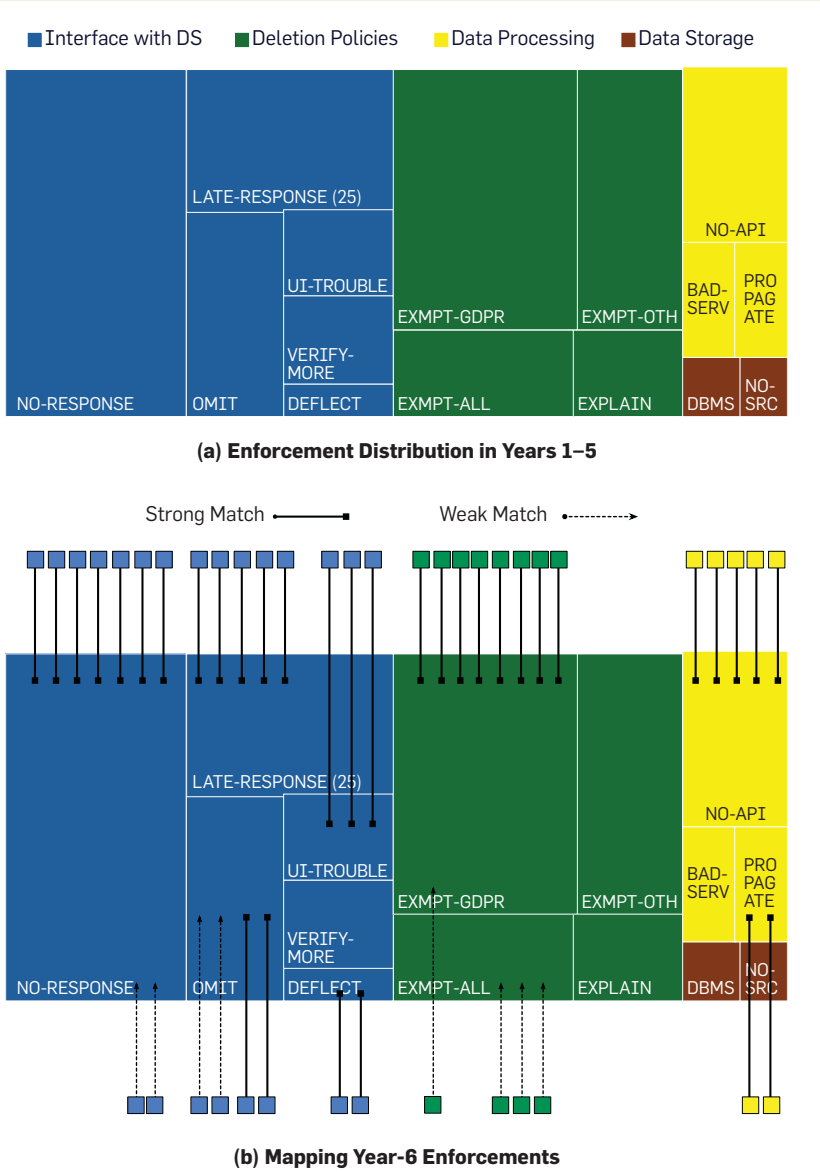


Table 1. Mapping function.

strong_match(e, c): An enforcement **e** has a strong match to category **c** if its annotation points to **c** as a failure category (*Rule* match) and there exists a prior enforcement **e'** in **c** that has the same application of that specific rule (*Application* match). This mapping indicates that if the controller had known about **e'** and fixed the specific RTBF condition that led to a failure in **e'**, they could have avoided **e**.
weak_match(e, c): An enforcement **e** has a weak match to category **c** if its annotation points to **c** as a failure category (*Rule* match) but there is no enforcement **e'** in **c** that has the same application of the specific rule (no *Application* match). This mapping indicates that if the controller had known about category **c** and its constituent cases, there would have been a reasonable chance to avoid **e**.
no_match(e, c): An enforcement **e** has no match to category **c** if its annotation does not indicate **c** as a failure category (no *Rule* match) and by extension, no *Application* match. This mapping indicates that even if the controller had known about category **c** and its constituent cases, they would have gained no knowledge to help them avoid **e**.

Table 2. Exploring the component-level data storage in Elasticsearch.

	Data Structures	Elastic Caches	Translog	Snapshots	Event Log	Replica Shard
Purpose	Performance	Performance	Fault Tolerance	Fault Tolerance	Monitoring	Availability
Storage location	Memory, disk	Memory, cache	Disk	Disk	Disk	External server
Retention beyond deletion	Until merge	Depends on system load	Until flush	No limit	No limit	Same as primary
API to delete select data	Yes, but timing is not guaranteed	No	No	No	No	No

Table 3. RTBF anti-patterns in computing and data-management systems.

RTBF Anti-Patterns	Rationale and Impact on RTBF	Affecting Categories and Examples
1. Using personal data as primary keys	Database constraints may cause practical difficulties in deleting data.	[DBMS] A Copenhagen company unable to delete phone numbers without redesigning their database[27]
2. Anonymizing data instead of deleting	Robust anonymization is hard;[21,22] Also, GDPR does not allow data to be anonymized w/o a legitimate purpose.	[EXMPT-GDPR] AXA anonymizing personal data to preserve its knowledge beyond the expiry date[h]
3. Keeping personal data untagged	Tagging data at ingestion is costly,[24] but not all types of data can be easily identified during runtime.	[NO-API] Clearview AI requiring users to provide exact copies of their photos to be deleted[14,29]
4. Mismanaging the depth of deletion	Many deletion workflows are not RTBF compatible; the depth of deletion must be guided by the operational context.	[PROPAGATE] A company simply deactivates user handles without removing their data;[i] Google fined for propagating URL delist req. to websites that published the original URL[28]
5. Logging without checks and bounds	Logs, even when used for demonstrating compliance, must not contain RTBF data.	[EXMPT-GDPR] A Belgian company storing excessive info about RTBF in their logs[j]
6. Employing excessive user verification	Though RTBF is a one-time, irreversible operation, its user-verification mechanism must not impose undue burdens.	[VERIFY-MORE] Twitter requiring photo ID for RTBF but not for account creation[k]

failure categories are recognized, especially with the emergence of new technologies and as controllers avoid making commonly identified mistakes.

RTBF-capable Elasticsearch. To answer EV.Q2, we investigate the RTBF capabilities of Elasticsearch, a popular open source search engine.[12] We explore this along two dimensions: *What RTBF tasks can Elasticsearch perform?* and *How well does it perform them*? Since Elasticsearch is primarily a data-processing and data-storage system, our analysis naturally focuses on those tasks:

▸ *Deletion APIs.* Elasticsearch offers two direct options: the RESTful method DELETE and the API _delete_by_query. The former can be used to delete a document (equivalent of tuples in DBMS) or an index (equivalent of an entire database), while the latter is useful for deleting a set of documents that match a query. Elasticsearch also provides a mechanism to remove contents within a document (equivalent of setting a field to null in DBMS) through the _update_by_query API. Finally, Elasticsearch used to support a time-to-live field for each document, but this has been deprecated since version 5 (circa June 2018).

▸ *Deletion latency.* Elasticsearch takes a lazy approach to deletion, that is, all deletion APIs simply *mark the data as deleted* without actually erasing them from data structures and files; the actual deletion is carried out at a later time depending on the runtime state of the system and user-defined configuration parameters. While this approach helps Elasticsearch be highly performant, it makes it hard to determine the latency of deletions.

▸ *Deletion propagation.* Elasticsearch's deletion can be best categorized as *shallow*. Deletion APIs make the deleted data unavailable for the application and eventually remove them from internal data structures. However, the deletion request is not propagated to all the underlying subsystems (e.g., query cache, translog, snapshots), which may contain the RTBF data.

In summary, Elasticsearch has basic RTBF capability but falls short on two aspects: It does not completely erase the RTBF data from all underlying subsystems, and even for those subsystems that it does, it is hard to estimate when the actual erasure happens. This behavior, while not unique to Elasticsearch,[9,23] conflicts with four enforcements in *NO-API* and *PROPAGATE* categories. Next, we propose a resolution to this.

Timed and thorough deletion in Elasticsearch. We describe a deletion operation as *timed* if it has a notion of time associated with it—for example, TTL-based deletion, where data would be deleted at a predetermined time; or real-time deletion, where data would be deleted synchronously in real time. Similarly, we characterize a deletion to be *thorough* if all copies of the given data are erased from all parts of the system. This is distinct from sanitizing deletes,[15] where data has to be irrevocably erased from the physical hardware; instead, a thorough delete simply goes through all the subsystems that may contain the given data and issues a delete request on them. The notion of timed and thorough deletion is not only rooted in RTBF precedents but also implemented in practice. For instance, Google Cloud's deletion guarantee states[8] all copies of the data would be deleted from all underlying subsystems within 180 days of requesting it. In contrast, Elasticsearch's deletion is neither timed nor thorough.

To fix this problem, we chart out

Elasticsearch's data footprint. Table 2 lists all the components of Elasticsearch in which data is stored during create, read, and update operations. For each component, it shows how data is stored, if it is retained beyond deletion, and if there are built-in APIs to trigger deletion. On our project website,[g] we share a deeper system analysis and our implementation of the timed-and-thorough deletion. However, it is important to mention that we were able to introduce timed and through deletion capability in ~450 lines of Python code. Then, we evaluated it using the official Rally benchmark in the StackOverflow track, which contains 36 million documents derived from the public Stack Overflow content.[20] It demonstrates that deletions in the modified Elasticsearch are guaranteed to complete in less than 30 minutes while incurring a transient drop in throughput lasting a few seconds. Thus, based on the ease of implementation and minimal impact on performance, we establish that timed and thorough deletion can be integrated into Elasticsearch, especially if the RTBF workflows could be batch scheduled at times of lower system load, say nightly or weekly.

RTBF Anti-Patterns

Finally, we apply learnings from the two-phase technique to reevaluate how certain longstanding practices of computing and data-management systems create pitfalls for RTBF compliance. This exploration is inspired by Andrew Koenig's notion of anti-patterns, that is, principles and practices of software design that seem intuitive and useful in a narrow context but prove ineffective in a broader system and over the long term[16]—for example, premature optimization of individual functions in a large software system. In a similar vein, we define *RTBF anti-patterns* as longstanding principles and practices of computing systems that serve their original purpose well but make it harder to support RTBF. To identify anti-patterns, we iterated through all 205 RTBF failures, both individually and collectively, probing if they stem from well-known and well-adapted principles and practices of computing systems. This process has yielded six anti-patterns, listed in Table 3.

Scope and limitations. The anti-patterns presented here should not be considered *exhaustive* (i.e., it may be possible to identify additional anti-patterns by analyzing the corpora with a different perspective) or *prescriptive* (i.e., there is no guarantee that having these in your systems will incur a penalty nor that avoiding them will be enough to avoid a violation). Instead, these anti-patterns should be treated as starting points toward making systems adapt better to RTBF.

Conclusion

Ever since its introduction, RTBF has triggered vigorous debates in our society, from being hailed as a counterbalance to the aggressive data practices of the 21st century to being criticized as a means to rewrite history, weaken the freedom of expression, and enable censorship. This work is an attempt to bring clarity to implementing RTBF from a computing and data-management perspective. We believe our principled approach to understanding the law and enforcement of RTBF, and translating that understanding to implementable actions in computing systems, will bridge the disconnect between legal and computing domains. C

g https://tinyurl.com/23ccsyl6

References

1. Bertram, T., et al. Five years of the right to be forgotten. In *Proceedings of the ACM SIGSAC Conf. on Computer and Communications Security* (2019), 959–972.
2. Bourtoule, L. et al. Machine unlearning. In *Proceedings of the IEEE Symp. on Security and Privacy* (2021), 141–159.
3. Charter of Fundamental Rights of the European Union. *Official J. of the European Union* 55 (2012), 391–407.
4. Calleros, C.R. *Legal Method and Writing*. Wolters Kluwer, New York (2014).
5. Cao, Y. and Yang, J. Towards making systems forget with machine unlearning. In *Proceedings of the IEEE Symp. on Security and Privacy* (2015).
6. Chen, M. et al. Graph unlearning. In *Proceedings of the ACM SIGSAC Conf. on Computer and Communications Security* (Nov. 2022), 499–513.
7. Cohn-Gordon, K. et al. DELF: Safeguarding deletion correctness in online social networks. In *Proceedings of the 29th USENIX Security Symp.* (2020).
8. Data deletion on Google Cloud platform. Google Cloud; https://cloud.google.com/security/deletion/.
9. Dong, S. et al. Optimizing space amplification in RocksDB. In *Proceedings of the Conf. Innovative Data Systems Research* (2017).
10. Garg, S., Goldwasser, S., and Vasudevan, P.N. Formalizing data deletion in the context of the right to be forgotten. In *Proceedings of EUROCRYPT 2020: 39th Annual Intern. Conf. on the Theory and Applications of Cryptographic Techniques* (May 2020), 373–402.
11. Ginart, A., Guan, M., Valiant, V., and Zou, J.Y. Making AI forget you: Data deletion in machine learning. In *Proceedings of the 33rd Intern. Conf. on Neural Information Processing Systems* 32 (2019).
12. *GDPR Compliance and The Elastic Stack*. Elastic (2024).
13. Habib, H. et al. "It's a scavenger hunt": Usability of websites' opt-out and data deletion choices. In *Proceedings of the 2020 CHI Conf. on Human Factors in Computing Systems* (2020), 1–12.
14. Hellenic DPA fines Clearview AI 20 million euros. *European Data Protection Board News* (Jul. 20, 2022).
15. Hughes, G.F., Coughlin, T., and Commins, D.M. Disposal of disk and tape data by secure sanitization. *IEEE Security & Privacy* 7, 4 (2009), 29–34.
16. Koenig, A. Patterns and antipatterns. *J. of Object-Oriented Programming* 8, 1 (1995).
17. Liu, Y. et al. The right to be forgotten in federated learning: An efficient realization with rapid retraining. In *Proceedings of IEEE INFOCOM* 2022, 1749–1758.
18. Mayer-Schönberger, V. *Delete: The Virtue of Forgetting in the Digital Age*. Princeton University Press (2011).
19. Minaei, M., Mondal, M., and Kate, A. Empirical understanding of deletion privacy: Experiences, expectations, and measures. In *Proceedings of the 31st USENIX Security* (2022).
20. Mitterdorfer, D. Announcing Rally: Our benchmarking tool for Elasticsearch. *Elastic Blog* (Apr. 19, 2016); https://tinyurl.com/2brdavf8
21. Narayanan, A. and Shmatikov, V. Robust de-anonymization of large sparse datasets. In *Proceedings of the IEEE Symp. on Security and Privacy* (2008), 111–125.
22. Rocher, L., Hendrickx, J.M., and Yves-Alexandre, D. M. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature Communications* 10, 1 (2019).
23. Sarkar, S., Papon, T.I., Staratzis, D., and Athanassoulis, M. Lethe: A tunable delete-aware LSM engine. In *Proceedings of ACM SIGMOD Intern. Conf. on Management of Data* (2020), 893–908.
24. Shastri, S. et al. Understanding and benchmarking the impact of GDPR on database systems. In *Proceedings of the VLDB Endowment* 13, 7 (2020).
25. Sun, C. et al. GDPRxiv: Establishing the state of the art in GDPR enforcement. In *Proceedings on Privacy Enhancing Technologies* 4 (2023).
26. Take, K. et al. It feels like whack-a-mole: User experiences of data removal from people search websites. In *Proceedings on Privacy Enhancing Technologies* 3 (2022), 159–178.
27. The Danish Data Protection Agency proposes a DKK 1.2 million fine for Danish taxi company. European Data Protection Board. (Mar. 25, 2019); https://tinyurl.com/y4mbtq8c.
28. The Swedish Data Protection Authority imposes administrative fine on Google. European Data Protection Board (2020); https://tinyurl.com/ycosjn55.
29. UK Information Commissioner's Office. ICO fines facial recognition database company Clearview AI Inc. more than 7.5M GBP and orders U.K. data to be deleted. *Monetary Penalty Notices* (May 22, 2022).
30. Xue, M. et al. The right to be forgotten in the media: A data-driven study. In *Proceedings on Privacy Enhancing Technologies* (2016).
31. Zhang, D. et al. Right to be forgotten in the era of large language models: Implications, challenges, and solutions. *AI and Ethics* 5, Springer (2024), 2445–2454.

Chen Sun is a computer science graduate student at the University of Iowa, Iowa City, IA, USA.

Nikolas Guggenberger is an assistant professor of law at the University of Houston Law Center, Houston, TX, USA.

Supreeth Shastri is an assistant professor of computer science at the University of North Texas, Denton, TX, USA.