Usability Issues in Differential Privacy Libraries: A Study from a Developers' Perspective

Ravi Mahankali
University of Bristol
Bristol, UK
ravi.mahankali@bristol.ac.uk

Jo Hallett
University of Bristol
Bristol, UK
joseph.hallett@bristol.ac.uk

***Abstract*—What usability issues do developers using Differential Privacy libraries face? We analyzed 2,021 GitHub issues from 5 major Differential Privacy libraries, identifying the usability problems like API confusion, poor error feedback, and documentation gaps. Unlike other privacy-preserving technologies, such as cryptographic libraries, that struggle with installation issues, Differential Privacy libraries face unique challenges. The main contributions of this work include: comprehensive taxonomy of 14 distinct usability issue categories identified through a systematic analysis of real-world developer experiences; empirical evidence that Differential Privacy libraries face different usability challenges compared to other privacy libraries, with API misuse dominating at 31.5% of all issues; and library-specific usability profiles revealing that specialized libraries (IBM DP and Google DP) show distinct patterns from general-purpose frameworks (PySyft), indicating the need for specialized library usability design approaches.**

I. Introduction

Privacy-preserving data analysis has become increasingly important as organizations seek to extract insights from sensitive data while protecting individual privacy [1], [2]. Differential Privacy is a formal framework that provides strong mathematical guarantees that an individual's data cannot be identified from the output of an analysis [3]. Several libraries have been developed for Differential Privacy, including Google Differential Privacy, IBM's Diffprivlib, OpenDP, PyDP, and PySyft [4]–[7]. Despite the availability of these libraries their adoption remains limited in practical applications outside of major technology companies [8]. This suggests a gap between the theoretical promises of Differential Privacy and its practical implementation, pointing to potential usability barriers preventing wider adoption [8], [9]. Research has focused on developing new Differential Privacy methods and the improvement of their theoretical properties [1], [10], but understanding the practical challenges developers face when attempting to implement these techniques is still underexplored. Usability issues in programming libraries can hinder adoption, particularly for complex concepts like Differential Privacy that already have a steep learning curve [9], [11].

We systematically analyzed GitHub issues—one of the major channels through which developers report problems and request features for open-source projects—and examining 2,021 GitHub issues between 2020 to 2025 to identify common usability challenges, categorize them thematically, and track their evolution over time. This provides insight into the real-world problems developers face.

This paper addresses the following research questions:

RQ1. What usability challenges are developers experiencing when using Differential Privacy libraries?

RQ2. Are the usability challenges developers face when using one Differential Privacy library the same as for all the others?

RQ3. Are the usability issues observed with Differential Privacy libraries any different to the issues Cryptography users face?

We find that confusion about API use represents the most significant usability barrier across all Differential Privacy libraries (RQ1), with 31.5% of all issues stemming from incorrect API usage, followed by requests for example code (16.7%) and usage guidance questions (14.1%). The distribution of usability challenges varies significantly across different libraries (RQ2), with specialized libraries like IBM DP and Google DP showing strong correlation in their issue patterns, while general-purpose frameworks like PySyft exhibit distinct usability profiles with weaker correlations to other libraries. We compared against an existing study in Cryptographic Libraries [9] and find differences in usability challenges between these two types of privacy technologies. Differential Privacy libraries face different challenges compared to cryptographic libraries (RQ3). This suggests that Differential Privacy libraries require specialized usability design approaches that address the unique conceptual complexity of privacy-utility tradeoffs rather than technical setup challenges.

II. Background

A. Differential Privacy

Differential Privacy, introduced by Dwork et al. [3], provides a formal privacy guarantee by ensuring that the ad-

Symposium on Usable Security and Privacy (USEC) 2026
27 February 2026, San Diego, CA, USA
ISBN 978-1-970672-07-7
https://dx.doi.org/10.14722/usec.2026.23033
www.ndss-symposium.org, https://www.usablesecurity.net/USEC/

dition or removal of a single record from a dataset does not significantly affect the output of an analysis. This is typically achieved by carefully adding calibrated noise to query responses.

The privacy guarantee is controlled by the privacy parameter ε, which quantifies the privacy loss: smaller values provide stronger privacy but typically at the cost of reduced utility [1]. In practice, a relaxed version known as (ε, δ)-Differential Privacy is often used, where δ represents the small probability of the guarantee not holding [1].

Differential Privacy is a powerful way for organizations to learn from sensitive data while protecting individual privacy. Unlike traditional anonymization techniques that can be vulnerable to re-identification attacks, Differential Privacy provides mathematically rigorous guarantees about the information leaked about any individual in the dataset. This makes it highly valuable for applications involving personal data, census information, healthcare records, and financial data where privacy protection is paramount.

The appeal of Differential Privacy lies in its composability—multiple differentially private queries can be combined while maintaining privacy guarantees—and its protection against arbitrary background knowledge held by attackers [1]. These theoretical properties have led to its adoption by major technology companies and government agencies for privacy-preserving data analysis [1], [2].

Several libraries have been developed to make Differential Privacy more accessible to practitioners. The differences in design philosophy, target audience, and implementation across these libraries create a rich environment for studying the usability challenges of Differential Privacy tools.

B. Usability of Privacy Tools

Prior research has identified usability as a critical factor in the adoption of privacy-enhancing technologies. The field of usable privacy and security has already been recognized that even technically strong privacy mechanisms can fail if they are too difficult for developers to implement correctly [9].

Grounded in the field of software usability, the work presented in this paper focuses mainly on the specific challenges of APIs and developer tools. The cognitive dimensions framework by Green and Petre [11] provides theoretical foundations for understanding usability challenges in programming environments. Their framework identifies key usability principles such as consistency, visibility, and error proneness that we apply to analyze Differential Privacy libraries.

Our research extends previous work on usable security tools by focusing specifically on the unique usability challenges posed by privacy-preserving technologies. We shift the focus from traditional end-user interfaces to the usability of developer tools and APIs. This domain poses a different class of challenges for users, including grasping complex concepts, correctly configuring parameters, and ensuring their implementation is error-free.

The importance of this research area is underscored by the growing recognition that privacy technologies must be not only theoretically sound but also practically deployable. Poor usability can lead to implementation errors that compromise privacy guarantees, making usability a critical component of overall privacy protection [12].

C. Differential Privacy versus Cryptographic Libraries

Differential Privacy presents fundamentally different usability challenges compared to traditional cryptographic libraries [7], [9], [13]. While cryptographic libraries focus on well-established algorithms with standardized implementations, Differential Privacy requires careful consideration of data-specific parameters and privacy-utility trade-offs [1], [14].

Cryptographic libraries generally have clear success and failure modes—encryption either works correctly or it fails—whereas Differential Privacy involves continuous trade-offs between privacy protection and data utility [2]. This difference creates new categories of usability challenges that do not exist in traditional cryptographic contexts [15].

While cryptographic libraries may require choosing key sizes or algorithms from a limited set of well-understood options, Differential Privacy requires setting epsilon and delta parameter values that depend on the specific use case, data characteristics, and desired privacy-utility balance [10], [16]. This creates a complex decision space for developers. Furthermore, the composition properties of Differential Privacy, while mathematically elegant, introduce additional complexity for practitioners who must track privacy budgets across multiple queries and understand how privacy guarantees degrade over time [1], [15]. These conceptual challenges don't exist in traditional cryptographic library usage.

We compare with the cryptographic usability work of Patnaik et al. [9] and reveal these differences and highlight the need for specialized usability research in the Differential Privacy domain (Section IV-B).

D. Related Empirical Studies

The foundational theoretical work by Dwork et al. [3] established the mathematical framework underlying modern Differential Privacy implementations, identifying key algorithmic challenges that continue to manifest as usability issues in contemporary libraries [1]. Building on this theoretical foundation, several researchers have examined how developers actually experience these challenges in practice.

Small-scale controlled studies have provided deep insights into individual developer experiences. Dankar et al. [17] examined Differential Privacy understanding in healthcare contexts, identifying parameter selection, sensitivity calculation, and composition as persistent pain points. These controlled studies reveal that even in supervised settings with direct support, developers struggle to translate Differential Privacy theory into working implementations. It is observed that privacy concepts like Differential Privacy are difficult to operationalize without concrete guidance.

Broader surveys have confirmed that these challenges extend beyond individual experiences to affect organizational adoption. Ngong et al. [13] conducted a comprehensive evaluation

TABLE I
ANALYZED DIFFERENTIAL PRIVACY LIBRARIES AND THEIR GITHUB REPOSITORIES BETWEEN 2020–2025

Library	Github Repository	Issues
PySyft	OpenMined/PySyft	1,387
OpenDP	opendp/opendp	469
PyDP	OpenMined/PyDP	99
Google DP	google/differential-privacy	35
IBM DP	IBM/differential-privacy-library	31
Total	5 libraries	2,021

of Differential Privacy tools with data practitioners, confirming the persistent usability challenges and the need for better tool design to bridge the gap between theoretical privacy guarantees and practical implementation.

What emerges from all this research is a pattern: Differential Privacy's mathematical complexity creates a translation problem that current tools fail to adequately address. While controlled studies reveal the moment-to-moment struggles developers face, surveys confirm these struggles translate into organizational resistance to adoption. The consistent identification of parameter configuration, composition understanding, and concrete guidance as pain points across diverse studies suggests these are fundamental challenges inherent to Differential Privacy rather than artifacts of specific implementations or study methodologies.

Our study complements this existing work by providing a large-scale analysis of real-world developer experiences with multiple Differential Privacy libraries. While previous research has typically examined individual libraries or involved limited participants (usually 10–30), our analysis of 2,021 GitHub issues provides a broader perspective on how these previously identified challenges manifest in practice. This scale allows us to quantify the relative importance of different usability challenges, providing empirical validation for the concerns raised in smaller-scale studies.

III. METHOD

A. Data Collection

1) Library Selection: We selected 5 Differential Privacy libraries that are the most popular differential privacy libraries on GitHub as they each had a reasonable number of issues and were actively being developed during the time period examined. These 5 libraries collectively represent the primary approaches to Differential Privacy implementation in practice, covering both Python-based tools (PySyft, PyDP, IBM DP) and multi-language frameworks (OpenDP, Google DP), as well as different abstraction levels from low-level noise mechanisms to high-level machine learning integrations (see Table I).

For each library, we extracted all issues created between January 2020 and March 2025, capturing a comprehensive view of developer interactions over time. This approach builds on the methodology of Bissyande et al. [18], who demonstrated that GitHub issue discussions provide valuable insights into developer challenges that may not be evident from code analysis alone. Our study adapts this methodology to the Differential Privacy domain by focusing on usability-specific challenges across multiple libraries. We developed a specialized coding scheme that categorizes issues according to usability principles and developer experience factors, enabling us to identify both library-specific patterns and universal challenges in Differential Privacy implementation. Additionally, analyzing issues in the timeframe (2020–2025) allows us to track how usability challenges evolve as libraries mature.

For each issue, we extracted the following information:

- Issue title and description text
- All comments and discussion threads
- Labels assigned by maintainers or contributors
- Creation and resolution timestamps
- Author information (anonymized for analysis)
- Whether the issue contained code examples or snippets
- Resolution status and method (closed, merged, rejected)
- Links to related issues or pull requests

The data collection process used the GitHub REST API to ensure consistency and completeness.

2) Inclusion and Exclusion Criteria: To determine whether a given issue qualified as a usability issue, we applied explicit inclusion and exclusion criteria. An issue was classified as a usability issue if it met at least one of the following criteria:

- The issue described problems developers encountered when attempting to use library APIs, functions, or features
- The issue requested help, examples, or guidance on how to correctly implement Differential Privacy functionality
- The issue reported confusion, errors, or unexpected behavior when using the library
- The issue identified gaps or problems in documentation that hindered effective library usage.

We excluded issues that did not relate to developer usability, such as:

- Administrative issues (such as repository management, CI/CD configuration, or project organization)
- Duplicate reports and spam
- Issues focused primarily on theoretical aspects of Differential Privacy rather than implementation challenges
- Issues that were purely feature requests without usability implications. For example, we excluded issues like: "Duplicate sphinx apidoc config" (OpenDP #1477) and "Resolve Duplicate without user-selected canonical" (OpenDP #1061) since these were administrative documentation concerns rather than usability challenges.

We also filtered out issues that focused primarily on academic understanding or theoretical concepts rather than practical implementation problems, such as feature requests for implementing theoretical frameworks or discussions about mathematical proofs that do not address concrete usability challenges.

This filtering ensured our analysis captured developer usability challenges rather than project management or theoretical concerns and discussions. To ensure we captured the

full developer experience, we included both open and closed issues, allowing us to analyze not only current challenges but also how issues evolved and were resolved over time. This temporal dimension enables us to track whether common usability problems persist or improve with library maturation.

B. Coding Scheme

To systematically analyze the issues, we developed a coding scheme based on a preliminary examination of the data and learning from prior literature on Differential Privacy implementation challenges. As a starting point for our codebook we took the usability issues from Patnaik et al.'s paper as an initial codebook [9] (which was itself derived from Green and Smith's earlier work [19]), but allowed new codes to emerge from our own dataset. Our approach was inspired by Humbatova et al. [20], who used a similar methodology to classify bugs in deep learning frameworks. Each issue was categorized according to the scheme shown in Table II.

The coding was performed using a semi-automated approach. First, we applied keyword-based filtering to exclude issues that lacked usability-related content. Then, we manually reviewed a subset of the remaining issues to validate our filtering criteria and ensure data quality. We created a code book with evolving codes as we went through the list of 2021 issues. Multiple codes could be assigned to a single issue if it addressed multiple aspects of Differential Privacy implementation. This approach aligns with the content analysis methodology used by Piorkowski et al. [21] in their study of developer information needs.

This coding scheme builds upon the challenges identified by Cummings et al. [2] for deploying Differential Privacy and follows the methodology of Patnaik et al. [9] for analyzing developer struggles with privacy libraries, with additional categories specific to library implementation concerns identified in our initial data exploration.

C. Validation

To ensure the reliability and validity of our coding scheme, two researchers with expertise in Differential Privacy and software engineering independently coded a random sample of 200 issues (approximately 10% of the dataset) using the preliminary coding scheme. Both researchers have prior experience with Differential Privacy libraries and usability analysis, ensuring domain knowledge for accurate categorization.

Key disagreements that required resolution included: (1) distinguishing between "API Misuse" and "How should I use this?" issues—we established that API Misuse applies when developers use incorrect parameters or method calls, while "How should I use this?" applies to general usage questions; (2) categorizing issues that mentioned both documentation problems and code examples—we created a rule that if the primary request was for working code, it was coded as "Example Code," otherwise as documentation-related; and (3) determining when privacy parameter configuration issues should be coded separately from general API misuse—we decided that only issues specifically discussing epsilon or delta values would be coded as "Privacy Parameter Configuration."

These refinements were documented in explicit decision rules, such as "Code as API Misuse only when specific method calls or parameters are incorrect; general questions about usage go to 'How should I use this?"' and "Privacy Parameter Configuration requires explicit mention of epsilon, delta, sensitivity, or privacy budget." After resolving disagreements, the refined coding scheme was applied to an additional validation set of 100 issues by the same two researchers to validate the consistency of the refined scheme.

The remaining 1,721 issues (approximately 85% of the dataset) were then recoded by the first author. To ensure consistency, a random sample of 100 issues from this remaining set was independently recoded by the second researcher, achieving an inter-coder agreement (Cohen's Kappa [22]) value of 0.87; suggesting reliable coding. Disagreements in this validation sample were resolved through discussion, and any resulting refinements to the coding rules were applied retroactively to the full dataset.

D. Analysis approach

We used a mixed-methods approach combining quantitative statistical analysis with qualitative thematic coding. We calculated descriptive statistics for issue categories across libraries, performed correlation analysis to identify patterns, and conducted chi-square tests to assess statistical significance of differences between libraries. The temporal analysis tracked issue patterns over time to identify trends and stability in usability challenges (see Table III). For qualitative analysis, we employed thematic coding to identify underlying usability principles violations and conducted comparative analysis with existing literature on cryptographic library usability to understand domain-specific challenges.

E. Threats to validity

We have identified the following threats to validity and implemented specific mitigation strategies:

1) Coding subjectivity: Our coding scheme is grounded in the first author's experience with differential privacy and security development, and may be biased by their experience. We mitigated by having a second coder (with a background in systems engineering and usable security) recode a subset, but our biases will still be implicit in the data as a whole [11].

2) GitHub issue sampling bias: Our study relies exclusively on GitHub issues, which represent a biased sample that is skewed toward certain types of problems and certain types of users. Library authors and maintainers often open issues themselves when they discover bugs during development, which may over-represent implementation issues relative to external user experiences. Second, GitHub's cultural norms favor reporting specific, actionable problems (e.g., bugs, error messages, missing documentation) over general questions or conceptual challenges. Questions about fundamental Differential Privacy concepts, such as parameter selection strategies or privacy-utility tradeoffs, are less likely to appear

TABLE II
CODE BOOK USED FOR CATEGORISING DIFFERENTIAL PRIVACY LIBRARY ISSUES (N=2021, 2020—2025)

Issue Category	Description	Example Issue	Count
API Misuse	Developers incorrectly using library features, wrong parameter values, or method calls	"When 'epsilon==0', 'PrivacyLossDistribution.from_privacy_parameters()' fails" (Google DP #110)	637
Example Code	Requests for code examples, tutorials, or working demonstrations of library functionality	"I can't found example about use Pysyft to train neural network" (PySyft #9185)	338
How should I use this?	Questions about correct usage, parameter selection, or implementation approaches	"How can we dynamically (based on data) determine maxContributions value for approximate bound algorithm?" (Google DP #258)	284
Compatibility Issues	Problems integrating with other libraries, version conflicts, or platform incompatibilities	"Facing error while Compiling pydp/carrot example" (PyDP #462)	154
Missing Documentation	Absence of documentation for specific features, functions, or use cases	"When developing the codebase, it is easy for notebooks to become outdated..." (PyDP #294)	143
Unsupported Feature	Requests for new functionality not currently available in the library	"this function would enable a separate privacy analysis using tools like AutoDP" (OpenDP #2179)	128
What's gone wrong here?	Debugging issues where code appears correct but fails or behaves unexpectedly	"LogisticRegression not working using example in logistic_regression.ipynb notebook" (IBM DP #97)	127
Clarity of Documentation	Existing documentation is unclear, confusing, or needs improvement	"Usability: Reference to `make_private_lazyframe` in error message when not used" (OpenDP #2100)	103
Should I use this?	Questions about choosing between different methods, algorithms, or approaches	"Usability: API Design Mirroring Popular Libraries" (OpenDP #1419)	54
Privacy Parameter Configuration	Specific issues with epsilon, delta, sensitivity, or other Differential Privacy parameter settings	"Could someone give me intuition on how to set bounds and epsilon?" (Google DP #15)	19
Lack of Knowledge	General confusion about Differential Privacy concepts or foundational understanding	"I've been trying to understand the paper... and it is difficult as the work is purely theoretical" (IBM DP #83)	16
Statistical Functions	Problems with statistical computations, aggregations, or mathematical operations	Add support for aggregating multiple values (Google DP #285)	8
Performance Issues	Reports of slow execution, memory problems, or efficiency concerns	"Out of memory with ZetaSQL" (Google DP #90)	6
Build Issues	Installation failures, compilation errors, or setup problems	"When I run the example, ModuleNotFoundError comes up every time" (PyDP #447)	4
Total			2021

TABLE III
SUMMARY OF ANALYZED DIFFERENTIAL PRIVACY LIBRARIES AND THEIR ISSUE DISTRIBUTIONS (2020—2025)

Library	Github Repository	Issues	Percentage	Primary Issues
PySyft	OpenMined/PySyft	1387	68.6%	API Misuse
OpenDP	opendp/opendp	469	23.2%	Example Code
PyDP	OpenMined/PyDP	99	4.9%	API Misuse
Google DP	google/differential-privacy	35	1.7%	API Misuse
IBM DP	IBM/differential-privacy-library	31	1.5%	How to use
Total	5 libraries	2021	100%	3 categories

in GitHub issues because they are not library-specific and are often addressed through other channels (e.g., academic forums, Stack Overflow, direct consultation). This bias may affect our conclusions, particularly regarding mathematical complexity barriers. Our finding that "the primary barriers surfaced in GitHub issues for developers are not the mathematical concepts of choosing privacy parameters themselves" (Section IV-A2) should be interpreted cautiously, as the sample may systematically under-represent these challenges due to the platform's cultural context. We acknowledge that this limitation means our study provides stronger evidence for some conclusions (e.g., documentation and API design challenges) than others (e.g., fundamental understanding of Differential Privacy concepts). Future work should triangulate

these findings with surveys, interviews, and other data sources to provide a more complete picture of developer challenges.

3) Platform bias: GitHub issues may not capture all developer struggles resolved through other channels (e.g., Stack Overflow, documentation, direct support). However, GitHub represents the primary platform for reporting technical issues in open-source libraries, and our focus on usability-specific issues helps ensure relevance to our research questions.

4) Library maturity effects: Libraries have different maturity levels affecting issue patterns. We addressed this by normalizing data by percentages and analyzing temporal trends.

5) Selection bias: Our focus on five major libraries may not capture the full spectrum of Differential Privacy implementations. However, these represent the most widely used implementations covering different approaches, design philosophies, and target audiences.

6) Temporal scope: Our 2020—2025 analysis period may not capture longer-term trends. However, this represents the most active development phase for Differential Privacy tools, providing a comprehensive view of usability challenges.

7) Comparison with prior qualitative studies: Our findings should be interpreted in the context of prior qualitative research on Differential Privacy usability. Sarathy et al. [23] and Garrido et al. [24] conducted interviews with library developers and users, identifying challenges including parameter selection, privacy-utility tradeoffs, and mathematical complexity. While our GitHub-based analysis finds relatively few explicit parameter selection issues (0.9%), this discrepancy may reflect the sampling bias discussed above rather than a true absence of these challenges. The interview studies provide evidence that mathematical and conceptual barriers do exist, suggesting that our GitHub-based sample may under-represent these fundamental challenges. This triangulation highlights the importance of considering multiple data sources when studying developer experiences with complex technologies like Differential Privacy.

F. Ethical Considerations

This study analyzes publicly available GitHub issues, which raises substantial ethical considerations regarding the use of public data for research purposes without explicit consent from the individuals who authored the comments. We have carefully considered these ethical implications and acknowledge the ongoing debate in the research community about best practices for public observation studies [25]. The GitHub issues analyzed in this study were created and posted publicly on GitHub, a platform where users explicitly make their content available under GitHub's Terms of Service. According to GitHub's Terms of Service (Section 5), users grant "a nonexclusive, worldwide license to use, display, and perform Your Content through the GitHub Service." The issues were created with no expectation of privacy or anonymity, as GitHub issues are inherently public-facing communication channels designed for community discussion and problem-solving. Following consultation with our institution's ethics board, this research did not require a formal ethics approval.

IV. Results

A. RQ1. What usability challenges are developers experiencing when using Differential Privacy libraries?

As shown in Table IV, developers face 14 distinct categories of usability issues when working with Differential Privacy libraries. The three most prevalent categories account for 62.3% of all issues: API Misuse (31.5%), Example Code requests (16.7%), and usage guidance questions (14.1%).

1) Technical Challenges: Our analysis shows that the most prevalent technical challenge for developers using Differential Privacy libraries is API misuse, accounting for 31.5% of all issues. This suggests that the abstractions provided by current libraries often do not align with developer expectations or mental models, leading to frequent errors in usage.

For example, developers struggle with incorrect parameter values, as evidenced by issues like "When ‘epsilon==0‘, ‘PrivacyLossDistribution.from_privacy_parameters()‘ fails" (Google DP #110) where developers incorrectly pass epsilon=0, causing the function to fail because "the second will overwrite the first (python should probably not silently do that)." Similarly, developers misuse bounded functions, as shown in "Bounded Functions fail when large dataset given" (Google DP #40) where "the int overflows and gives the output as the lowerbound." Other API misuse examples include missing required method calls like "If missing ‘fill_nan‘, ‘fill_null‘, can we do better than ‘unable to infer bounds‘?" (OpenDP #2312), incorrect attribute access like "AttributeError: 'BoundedSum' object has no attribute 'privacy_budget_left"' (PyDP #463).

Example code requests (16.7%) and questions about how to use the libraries (14.1%) are also common; suggesting a need for practical, implementation-focused guidance. Developers frequently request examples, as shown in issues like "I can't found example about use Pysyft to train neural network, I hope some helps" (PySyft #9185), "Error occurring during the execution of an example program" (PyDP #454), and "When I run the example, ModuleNotFoundError comes up every time" (PyDP #447). Usage questions are also prevalent, as demonstrated by "How can we dynamically (based on data) determine maxContributions value for approximate bound algorithm?" (Google DP #258), "What would be the expected way of implementing this?" (Google DP #212), "Fail early if user makes a margin for a column that doesn't exist" (OpenDP #2315), and "Facing error while Compiling pydp/carrot example" (PyDP #462).

While privacy-utility tradeoffs and privacy budget management are important technical aspects of Differential Privacy, our data indicates that these are less frequently reported as explicit issues. However, when they do occur, they reveal critical challenges. For example, developers struggle with privacy budget management, as shown in "C++ Proposal: Remove privacy_budget parameter" (Google DP #59) where the team notes that "anyone who wants to track overall expenditure of privacy loss budget will need to do extra work as soon

TABLE IV
DISTRIBUTION OF USABILITY ISSUE CATEGORIES ACROSS DIFFERENTIAL PRIVACY LIBRARIES (2020–2025). PERCENTAGES INDICATE THE PROPORTION OF EACH LIBRARY'S TOTAL ISSUES.

Issue Category	PySyft	OpenDP	PyDP	IBM DP	Google DP
API Misuse	537 (38.7%)	66 (14.1%)	16 (16.2%)	10 (32.3%)	8 (22.9%)
Example Code	222 (16.0%)	98 (20.9%)	15 (15.2%)	1 (3.2%)	2 (5.7%)
How should I use this?	164 (11.8%)	91 (19.4%)	15 (15.2%)	7 (22.6%)	7 (20.0%)
Compatibility Issues	116 (8.4%)	19 (4.1%)	14 (14.1%)	2 (6.5%)	3 (8.6%)
Missing Documentation	71 (5.1%)	57 (12.2%)	12 (12.1%)	1 (3.2%)	2 (5.7%)
Unsupported Feature	79 (5.7%)	33 (7.0%)	9 (9.1%)	1 (3.2%)	6 (17.1%)
What's gone wrong here?	91 (6.6%)	27 (5.8%)	5 (5.1%)	3 (9.7%)	1 (2.9%)
Clarity of Documentation	57 (4.1%)	38 (8.1%)	5 (5.1%)	1 (3.2%)	2 (5.7%)
Should I use this?	26 (1.9%)	21 (4.5%)	4 (4.0%)	1 (3.2%)	2 (5.7%)
Privacy Parameter Configuration	3 (0.2%)	9 (1.9%)	3 (3.0%)	4 (12.9%)	0 (0.0%)
Lack of Knowledge	12 (0.9%)	3 (0.6%)	1 (1.0%)	0 (0.0%)	0 (0.0%)
Statistical Functions	4 (0.3%)	4 (0.9%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
Performance Issues	3 (0.2%)	2 (0.4%)	0 (0.0%)	0 (0.0%)	1 (2.9%)
Build Issues	2 (0.1%)	1 (0.2%)	0 (0.0%)	0 (0.0%)	1 (2.9%)
Total	1387	469	99	31	35

as they're using more than one 'Algorithm'," highlighting the complexity of budget tracking across multiple operations.

Instead, developers are more likely to struggle with understanding how to correctly use library APIs, configure parameters, and adapt example code to their own use cases. The lack of standardization across libraries further compounds these challenges, as developers must often relearn concepts and interfaces when switching between tools [13].

2) Mathematical Complexity and Implementation Barriers: Although Differential Privacy is grounded in rigorous mathematical theory, our results indicate that among the barriers surfaced in GitHub issues, the most frequently reported challenges are related to the translation of Differential Privacy concepts into correct and effective code, rather than explicit questions about mathematical concepts themselves. The high frequency of *API misuse* and *requests for example code* suggests that developers need more concrete, actionable guidance rather than abstract theoretical explanations that don't translate to working code. However, we acknowledge that this finding may be influenced by the sampling bias inherent in GitHub issues, which tend to favor specific, actionable problems over general conceptual questions (see Section III-E) for discussion of limitations). Prior interview studies with library users have identified parameter selection and privacy-utility tradeoffs as significant challenges [23], [24], suggesting that our GitHub-based sample may under-represent these fundamental conceptual barriers.

Nevertheless, some issues do stem from the inherent complexity of Differential Privacy, such as difficulties with parameter configuration (0.9% of issues, but critical when they occur) and understanding the implications of different privacy settings. These challenges are exacerbated by the shortage of practitioners with both mathematical and software engineering expertise [26], and are reflected in persistent documentation-related problems across all libraries.

3) API Design and Documentation Challenges: The high rate of API misuse (31.5%) and the prevalence of example code requests (16.7%) highlight design and documentation challenges in making Differential Privace accessible to developers. Our findings align with recent usability studies that have identified similar patterns across multiple Differential Privacy libraries [2], [8]. Current Differential Privacy library APIs often fail to provide abstractions that match developer mental models, and documentation is frequently insufficient to bridge the gap between theory and practice.

Clarity and helpfulness of error messages is another challenge identified in usability studies of Differential Privacy libraries. This is directly related to the misuse of the APIs. Developers who raised issues in Github for these libraries reported difficulties in diagnosing and recovering from errors due to poorly designed messages. OpenDP users, primarily familiar with Python, encountered additional frustration as error messages were presented in Rust, compounding their difficulties in addressing errors effectively. This challenge was explicitly acknowledged by the OpenDP team, who created an issue to "Improve error messages throughout Rust codebase" (OpenDP #1992) and another addressing "Usability: Reference to $make\_private\_lazyframe$ in error message when not used" (OpenDP #2100), where users reported confusion when error messages referenced functions they weren't using.

Documentation navigation and quality represent another prevalent issue. Users have reported challenges in navigating the documentation, citing inconsistent content quality and a lack of up-to-date resources. Poor navigation within documentation has been noted as a barrier to effective implementation of Differential Privacy tools, with previous research in software engineering highlighting similar findings where inconsistent content and unclear navigation have been documented

as widespread usability problems for programming tools [27]. For example, one PyDP contributor noted: "When developing the codebase, it is easy for notebooks to become outdated. We should add test to our CI that warn us when notebooks stop working due to API changes. This is crucial, since notebooks are important pieces of documentation" (PyDP #294).

4) Mental Model Misalignments and Decision Implications: The alignment of user mental models with the operational models of Differential Privacy libraries represents another area of concern identified in recent usability research. Users often lack intuitive understanding of how these libraries implement computations and enforce privacy guarantees, leading to challenges in using them correctly. For example, an IBM DP user expressed confusion about the fundamental relationship between privacy parameters and model accuracy: "Up to a value of epsilon=10**7, there does not seem to be any relation between the value of epsilon and the accuracy of the model" (IBM DP #97), demonstrating a lack of understanding of how Differential Privacy mechanisms work internally.

Users with limited prior knowledge of Differential Privacy, often struggle to comprehend the implications of their decisions while using these libraries. This lack of understanding can lead to ineffective use of the tools and potential privacy violations, suggesting a need for better educational resources and user support tailored to varying levels of expertise. For instance, a Google DP user asked: "Could someone give me intuition on how to set bounds and epsilon?" (Google DP #15), highlighting the uncertainty about parameter implications and the need for better guidance on privacy-utility tradeoffs.

The community feedback mechanisms embedded within these libraries serve as a critical component for improvement, allowing developers to gather user experiences and adapt functionalities accordingly. Studies involving thousands of reviews have revealed vital insights into user satisfaction and challenges, emphasizing the need for better communication regarding decision implications and enhanced documentation quality [28]. Addressing these usability issues is essential for fostering trust and encouraging wider adoption of Differential Privacy frameworks among practitioners and researchers alike.

B. RQ2. Are the usability challenges developers face when using one Differential Privacy library the same as for all the others?

To answer this research question, we formulated the following hypothesis: **H0: The distribution of usability issue categories is uniform across all Differential Privacy libraries** (i.e., all libraries face the same types of challenges in the same proportions). The alternative hypothesis is **H1: The distribution of usability issue categories varies significantly across different Differential Privacy libraries** (i.e., different libraries face distinct usability challenges).

We tested this hypothesis by performing a comprehensive statistical analysis comparing issue distributions across the five analyzed libraries. If the null hypothesis is rejected, it would indicate that different libraries face unique usability challenges, requiring tailored approaches to usability improvement. Our study reveals significant differences in usability challenge patterns, driven by library design philosophies, target audiences, and implementation approaches.

TABLE V
CORRELATION COEFFICIENTS BETWEEN LIBRARIES' ISSUE CATEGORY DISTRIBUTIONS

Library	PySyft	OpenDP	PyDP	IBM DP	Google DP
PySyft	1.000	0.591	0.683	0.533	0.595
OpenDP	0.591	1.000	0.738	0.788	0.834
PyDP	0.683	0.738	1.000	0.702	0.770
IBM DP	0.533	0.788	0.702	1.000	0.889
Google DP	0.595	0.834	0.770	0.889	1.000

1) Library Design Philosophy Impact on Usability: Our analysis reveals that different Differential Privacy libraries' design approaches significantly impact their usability profiles, as evidenced by the distinct issue patterns we observed. The correlation analysis and chi-square tests demonstrate that libraries with similar design philosophies exhibit similar usability challenge patterns. For instance, IBM DP and Google DP, both specialized libraries focused specifically on Differential Privacy implementation, show the strongest correlation ($r = 0.889$) in their issue distributions. This finding suggests that design philosophy—whether a library is specialized for Differential Privacy or a general-purpose framework—influences the types of usability challenges developers encounter. The minimal API design of IBM DP enhances compatibility with data analytics libraries like Pandas [29], appears to result in a different usability profile compared to frameworks like PySyft, showing weaker correlations ($r = 0.533$–0.683) with other libraries.

2) Statistical Comparison of Issue Distributions: To quantify the similarity between libraries' issue patterns, we calculated Pearson correlation coefficients between the percentage distributions of issue categories.

We first normalized the data by calculating the percentage of issues in each category for each library, thus accounting for the differences in total issue counts (PySyft: 1,387; OpenDP: 469; PyDP: 99; IBM DP: 31; Google DP: 35). This allowed us to compare the relative importance of different issue types rather than raw counts.

Pearson correlation was chosen because it measures the linear relationship between two variables and is robust for comparing relative distributions. The correlation coefficient ranges from -1 to +1, where +1 indicates perfect positive correlation, 0 indicates no correlation, and -1 indicates perfect negative correlation. This approach allowed us to identify which libraries have similar usability challenge profiles, regardless of their absolute issue volumes.

The correlation analysis (Table V) reveals medium to high positive correlations between all libraries, indicating some consistency in the issue patterns. IBM DP and Google DP have a strong correlation ($r = 0.889$), as both are specialized libraries focused on the implementation of Differential Privacy.

TABLE VI
CHI-SQUARE TEST RESULTS COMPARING ISSUE CATEGORY DISTRIBUTIONS ACROSS LIBRARIES

Library Comparison	Chi-Square Statistic	df	p-value
PySyft vs. OpenDP	245.67	13	<0.001
PySyft vs. PyDP	89.34	13	<0.001
PySyft vs. IBM DP	156.78	13	<0.001
PySyft vs. Google DP	134.92	13	<0.001
OpenDP vs. PyDP	67.45	13	<0.001
OpenDP vs. IBM DP	98.23	13	<0.001
OpenDP vs. Google DP	112.56	13	<0.001
PyDP vs. IBM DP	45.67	13	<0.001
PyDP vs. Google DP	78.91	13	<0.001
IBM DP vs. Google DP	23.45	13	0.037
Overall Test	1,247.89	52	<0.001

In contrast, PySyft shows the weakest correlations with other libraries ($r = 0.533$–0.683), indicating that its issue profile differs substantially.

Table V confirms that IBM DP and Google DP have the most similar issue profiles (smallest distance), while PySyft has the most dissimilar profiles (largest distance).

To statistically validate these differences, we conducted chi-square tests of independence comparing the distribution of issue categories across libraries. The results, shown in Table VI, reveal highly significant differences ($p < 0.001$) between all library pairs, confirming that each library has a distinct usability challenge profile.

The chi-square test results provide strong statistical evidence that the usability challenges faced by developers vary significantly across different Differential Privacy libraries. The overall test statistic of 1,247.89 with 52 degrees of freedom ($p < 0.001$) indicates that the null hypothesis of uniform distribution across libraries can be confidently rejected. This supports our hypothesis (H1) of research that different libraries face distinct usability challenges based on their design philosophy, target audience, and implementation approach.

3) Key Differences in Issue Categories: Our analysis reveals that while some usability challenges—such as API misuse and the need for example code—are common across all Differential Privacy libraries, the relative importance of specific issue categories varies substantially between libraries. These findings suggest that each library has a unique usability profile shaped by its design philosophy, target audience, and technical focus. As a result, improving usability in the Differential Privacy ecosystem will require tailored solutions that address the specific pain points of each library, rather than a one-size-fits-all approach.

4) Top Issue Categories by Library: Each library exhibits a distinct pattern in its top three issue categories, as shown in Table IV. While API Misuse and Example Code requests are among the top issues for most libraries, the prominence of *How should I use this?* and *What's gone wrong here?* varies, reflecting differences in user needs and library design. These patterns underscore the importance of tailoring usability improvements to the specific issue profiles of each library.

C. RQ3. Are the usability issues observed with Differential Privacy libraries any different compared to Crypto libraries?

To answer this question, we compare our findings with the work of Patnaik et al. [9], who analyzed Stack Overflow questions about cryptographic libraries for usability issues. This comparison serves multiple purposes: it provides a baseline for usability research in privacy-enhancing technologies, leverages conceptual similarities between cryptographic and Differential Privacy libraries (both requiring complex mathematical understanding and parameter configuration), and enables methodological consistency through similar analysis frameworks.

Methodological Considerations: It is important to acknowledge the limitations of comparing results from two studies that used different (albeit similar) codebooks, as the perspectives of the coders will never be identical. The inherently subjective nature of qualitative coding means that categories may not map perfectly between studies, and differences in coding decisions could affect comparability. However, both studies followed similar qualitative coding methodologies with inter-rater validation, and we focus our comparison on high-level patterns and fundamental differences (e.g., installation vs. API misuse) rather than precise percentage comparisons. The substantial differences we observe (e.g., 22.1% vs. 0.2% for installation issues) are large enough to be meaningful despite potential codebook differences. We discuss these limitations explicitly to ensure readers interpret the comparison appropriately.

This comparison provides empirical evidence that Differential Privacy libraries face fundamentally different usability challenges than cryptographic libraries, supporting the need for specialized design approaches rather than applying general software usability principles.

1) Fundamental Differences in Problem Nature: Differential Privacy introduces conceptual challenges not present in cryptographic libraries. While cryptographic libraries typically deal with well-established algorithms with clear success/failure modes, Differential Privacy requires continuous decision-making about privacy-utility tradeoffs [16]. This fundamental difference is reflected in our data through significantly higher rates of usage uncertainty questions (16.8% vs. 12.3% for algorithm selection in crypto libraries).

The mathematical foundations of Differential Privacy create unique implementation barriers. Unlike cryptographic operations that often have standardized implementations, Differential Privacy requires understanding of noise mechanisms, sensitivity calculations, and composition properties that vary significantly based on data characteristics and use cases [13]. This complexity manifests in our observation that developers consistently struggle with parameter configuration, despite privacy parameter issues representing only 0.9% of total issues—their critical nature makes them disproportionately important for maintaining privacy guarantees.

Figure 1 shows these key differences in issue distribution patterns.

2) Key Differences: **API Usage Challenges:** Differential Privacy libraries show significantly higher rates of API misuse (31.5% vs. 14.2%). This substantial difference reflects the inherent complexity of Differential Privacy concepts, which create more intricate API interaction patterns than traditional cryptographic operations. Unlike cryptographic functions that typically have clear input-output relationships, Differential Privacy APIs must handle complex parameter spaces involving epsilon, delta, sensitivity, and noise mechanisms that interact in non-intuitive ways [2].

Example Dependency: Differential Privacy libraries show nearly double the rate of example code requests (16.7% vs. 8.7%), highlighting a critical difference in learning approaches. While cryptographic operations can often be understood through straightforward documentation, Differential Privacy requires concrete demonstrations to bridge the gap between mathematical theory and practical implementation [26]. This increased dependency on examples reflects the conceptual complexity of privacy-utility tradeoffs that cannot be easily conveyed through traditional API documentation.

Privacy Budget Complexity: A unique challenge in Differential Privacy libraries is privacy budget management, which has no analog in cryptographic libraries. Developers must track privacy loss across multiple computations and understand composition properties—challenges that are entirely absent from traditional cryptographic implementations [9]. This represents a fundamental shift from stateless cryptographic operations to stateful privacy accounting.

Mathematical Sophistication Requirements: Unlike cryptographic libraries where developers can often treat algorithms as black boxes, Differential Privacy requires deeper understanding of statistical concepts, noise distributions, and sensitivity analysis. This understanding of mathematical complexity is reflected in our finding that Differential Privacy libraries show different challenge patterns than general-purpose privacy frameworks. Developers struggle with understanding the relationship between epsilon values and model accuracy, as evidenced by the issue "Up to a value of epsilon=10**7, there does not seem to be any relation between the value of epsilon and the accuracy of the model" (IBM DP #97).

Similarly, developers face challenges with noise distribution complexity, as shown in "With the new APIs, it can be difficult to determine noise distribution, scale parameters and accuracy estimates" (OpenDP #1759). The need for sensitivity tracking and noise calibration is highlighted in issues like "OpenDP tracks the sensitivity and does the noise calibration, this function would enable a separate privacy analysis using tools like AutoDP" (OpenDP #2179), demonstrating the sophisticated mathematical concepts that developers must understand.

3) Temporal Analysis: Analysis of issue creation dates reveals that usability challenges in Differential Privacy libraries have remained relatively stable over time, unlike the declining installation issues observed in crypto libraries. The three most critical usability challenges—API misuse, example code requests, and documentation issues have maintained consistent prevalence rates over the 2020–2025 period. This temporal stability suggests that the fundamental usability challenges in Differential Privacy libraries are persistent and not easily resolved through incremental improvements.

For instance, API misuse issues persist across years, as evidenced by "Bounded Functions fail when large dataset given" (Google DP #40) from 2020, "Warn if dict is used for margins" (OpenDP #2297) from 2025, and similar issues throughout the analysis period. The consistent demand for example code is reflected in issues like "I can't found example about use Pysyft to train neural network" (PySyft #9185) and "Could someone give me intuition on how to set bounds and epsilon?" (Google DP #15), showing that these challenges are not diminishing over time despite ongoing library development.

V. Discussion

A. Usability Smells and Design Implications

Following Patnaik et al.'s methodology [9], we identified four usability smells that suggest when developers are facing usability challenges [11]:

1) **API Abstraction Mismatch (31.5%):** Current abstractions don't match developer mental models, violating the Consistency principle
2) **Example Dependency (16.7%):** Documentation fails to provide adequate affordances for learning
3) **Usage Uncertainty (16.8%):** Poor Closeness of Mapping between user goals and library features
4) **Parameter Complexity (0.9%):** Critical Error Proneness specific to Differential Privacy parameter selection

Static documentation alone is not enough; developers require concrete, executable examples and step-by-step guidance to successfully implement Differential Privacy. The persistent documentation-related issues (12.2% missing documentation, 5.1% clarity problems) across all libraries underscores the need for more practical, user-centered documentation and API design improvements. Based on our analysis, we recommend several solutions:

Interactive Documentation with Live Examples: Libraries should implement interactive documentation platforms that allow developers to run code examples directly in the browser, similar to Jupyter notebooks embedded in documentation. This addresses the high demand for example code (16.7% of issues) by providing immediate, executable demonstrations that developers can modify and test.

Privacy Parameter Visualization Tools: Given the complexity of privacy-utility tradeoffs, libraries should include interactive visualizations that show how epsilon and delta values affect both privacy guarantees and data utility. These tools could provide real-time feedback on parameter selection, helping developers understand the implications of their choices without requiring deep mathematical expertise [30].

Contextual Error Messages: Error messages should be enhanced with specific guidance for Differential Privacy contexts. For example, rather than showing a Rust trait bound

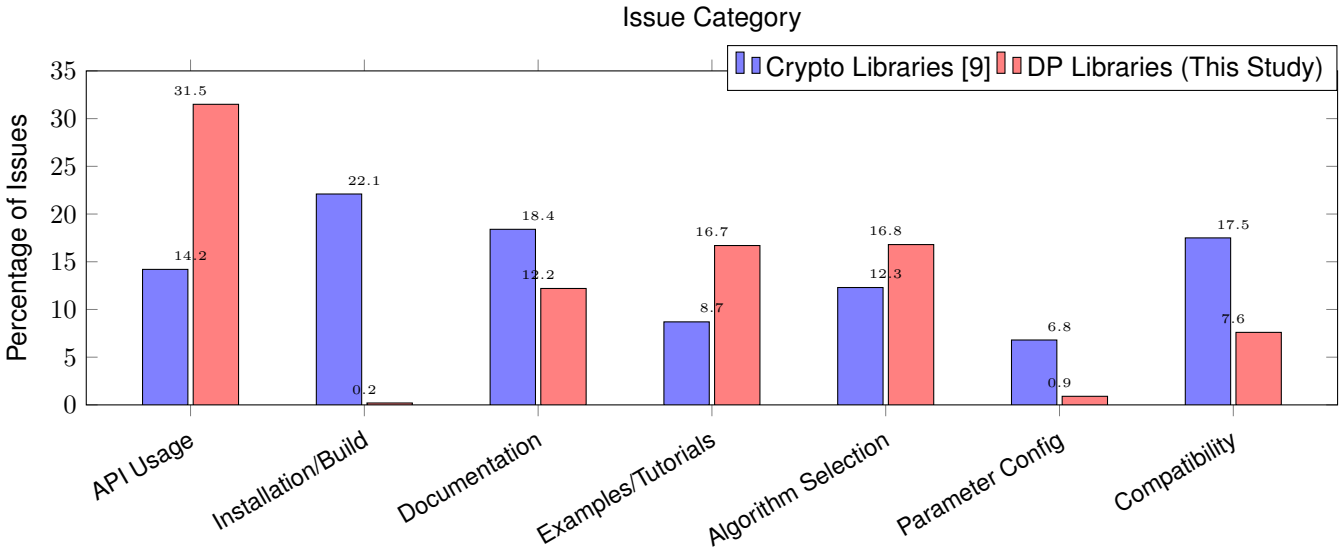

Fig. 1. Comparison of Issue Distributions: Cryptographic vs. Differential Privacy Libraries

error, the system should explain that the privacy budget has been exhausted and suggest alternative approaches.

Progressive Disclosure Documentation: Documentation should be structured with multiple levels of detail, allowing developers to start with simple examples and progressively access more complex implementations. This addresses the wide range of expertise levels among users, from beginners who need basic examples to advanced users requiring detailed parameter explanations.

Automated Documentation Testing: Implement continuous integration systems that automatically test documentation examples against the current API, ensuring that code snippets remain functional as libraries evolve. This prevents the common issue of outdated examples that no longer work with current versions. These improvements would directly address the most frequent usability challenges identified in our study: API misuse (31.5%), example code requests (16.7%), and documentation clarity issues (12.2%).

Design Recommendations:

- **Type-safe parameter interfaces** that prevent common epsilon/delta configuration errors, similar to how crypto libraries prevent key size mismatches
- **Fluent API designs** that guide developers through correct usage patterns, such as builder patterns that enforce parameter constraints
- **Validation layers** that catch parameter misconfigurations at development time, providing immediate feedback on privacy-utility trade-offs

1) Reducing Example Dependency: Evidence from Documentation Gaps: The 16.7% example code requests demonstrates the theory-practice gaps of documentation. Developers express frustration with abstract explanations:

Evidence: In issue (IBM DP #97), a developer expresses frustration: "When running the example notebook, the results are significantly different from what is in the committed example. Does the dp.LogisticRegression model still work? Has the interface changed significantly since this notebook was committed 3 years ago."

Design Recommendations:

- **Executable documentation** with interactive notebooks that allow developers to experiment with parameters and see immediate results
- **Use-case driven tutorials** rather than feature-driven documentation, focusing on common scenarios like "privacy-preserving analytics" rather than individual functions
- **Examples of Progressive complexity** from basic to advanced level scenarios, helping developers understand composition and privacy budget management.

2) Resolving Usage Uncertainty: Evidence from Decision Paralysis: The 16.8% usage uncertainty issues ("How should I use this?" + "Should I use this?") indicate decision-making paralysis that doesn't exist in crypto libraries:

Evidence: Issues like "Usability: API Design Mirroring Popular Libraries" (OpenDP #1419) show developers requesting better design guidance: "Consider leveraging the design of popular data science libraries in OpenDP API design for a learning scaffold."

Design Recommendations:

- **Algorithm recommendation systems** based on data characteristics, similar to how machine learning libraries suggest appropriate algorithms
- **Decision trees for parameter selection** that guide developers through epsilon/delta choices based on their

use case and privacy requirements
- **Comparative documentation** explaining when to use different approaches, with concrete examples of privacy-utility trade-offs

3) The Critical Challenge of Privacy Parameter Complexity: Although privacy parameter issues represent only 0.9% of the total issues, their criticality cannot be understated. Unlike the crypto-parameter mistakes that typically result in functionality failures, incorrect epsilon or delta values can completely undermine privacy guarantees.

Evidence: Issues like "LogisticRegression not working using example in logistic_regression.ipynb notebook" (IBM DP #97) show developers expressing confusion about parameter relationships: "Up to a value of epsilon=10**7, there does not seem to be any relation between the value of epsilon and the accuracy of the model."

Unique Design Challenge: This represents a fundamental difference from cryptographic libraries, where parameter validation is typically binary (correct/incorrect) rather than continuous (privacy/utility trade-off). Differential Privacy libraries need specialized validation that considers the context and consequences of parameter choices.

4) User-Centered Design and Continuous Improvement: The design of Differential Privacy tools should accommodate various levels of user expertise, ensuring accessibility for novices while providing advanced features for experienced users. By focusing on user-centered design principles, developers can foster broader adoption of Differential Privacy technologies across different domains. This includes providing educational materials that align with users' expected knowledge levels, which can enhance the usability of Differential Privacy tools.

Integrating usability criteria such as learnability, efficiency, and error prevention into the evaluation of Differential Privacy tools is crucial. Measuring error rates can help identify safety issues and design flaws, leading to improved user performance and satisfaction. Additionally, collecting feedback through post-task surveys and qualitative analyses can offer valuable insights into user experiences and the factors affecting Differential Privacy implementation.

To maintain the relevance and effectiveness of Differential Privacy tools, it is important to promote continuous learning and adaptation. This involves addressing challenges and constraints as they arise while staying current with the latest research and best practices in the field. Such an approach will help safeguard data privacy without sacrificing utility, thus improving user satisfaction and tool effectiveness.

Finally, collaboration with various stakeholders—including developers, downstream data users, and policymakers—should be prioritized and the Differential Privacy tools must be designed to facilitate a usable and adoptable privacy implementation.

VI. Conclusion

This study presents a comprehensive analysis of usability challenges in Differential Privacy libraries, analyzing 2,021 GitHub issues from five major Differential Privacy libraries over a five-year period (2020-2025). Our findings reveal that Differential Privacy libraries face fundamentally different usability challenges compared to other privacy-preserving tools like cryptographic libraries, with API misuse dominating at 31.5% of all issues. These findings align with recent usability studies that have identified similar patterns across multiple Differential Privacy libraries.

We identified four "usability smells" that characterize Differential Privacy library struggles: (1) API Abstraction Mismatch, (2) Example Dependency, (3) Usage Uncertainty, and (4) Parameter Complexity. These smells violate fundamental usability principles and create barriers to Differential Privacy adoption in practice.

Our mapping of issue categories to usability principles reveals that consistency violations (39.1% of issues) and affordance problems (23.1%) are the most critical challenges. Unlike cryptographic libraries where installation issues dominated, Differential Privacy libraries show a pattern focused on conceptual understanding and correct parameter usage.

The temporal analysis demonstrates that these usability challenges have remained stable over time, suggesting that current approaches to Differential Privacy library design are not adequately addressing fundamental usability barriers. The comparison with cryptographic libraries confirms that Differential Privacy introduces unique cognitive complexity that requires specialized usability considerations.

These findings have immediate implications for Differential Privacy library developers, who should prioritize type-safe parameter interfaces, executable documentation, and decision support systems. Our recommendations align with recent research that emphasizes the importance of user-centered design principles and continuous improvement in Differential Privacy tool development. For the broader privacy research community, our results highlight the critical need to consider usability as a first-class design constraint rather than an afterthought in library development.

Future work should investigate the effectiveness of our proposed design recommendations through controlled user studies and longitudinal analysis of library evolution. Additionally, extending this analysis to other privacy-enhancing technologies could reveal whether the patterns we identified are unique to Differential Privacy or are characteristics of privacy tools more broadly. Recent research has highlighted the importance of structured literature reviews and systematic approaches to understanding usability challenges in emerging technologies, which could provide valuable insights for future studies and research directions.

By providing empirical evidence of Differential Privacy library usability challenges at scale, this research contributes to the growing understanding that technical privacy solutions must be accompanied by thoughtful usability design to achieve meaningful adoption and impact in real-world applications.

Acknowledgment

This work was funded by EPSRC Grant EP/SO22465/1.

References

[1] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations Trends Theoretical Comput. Sci.*, vol. 9, no. 3–4, pp. 211–407, 2014.

[2] R. Cummings, D. Desfontaines, D. Evans, R. Geambasu, M. Jagielski, Y. Huang, P. Kairouz, G. Kamath, S. Oh, O. Ohrimenko *et al.*, "Challenges towards the next frontier in privacy," *arXiv preprint arXiv:2304.06929*, vol. 1, 2023.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *J. Privacy Confidentiality*, vol. 7, no. 3, pp. 17–51, 2016.

[4] R. J. Wilson, C. Y. Zhang, W. Lam, D. Desfontaines, D. Simmons-Marengo, and B. Gipson, "Differentially private SQL with bounded user contribution," in *Proc. Privacy Enhancing Technol.*, vol. 2020, 2020, pp. 230–250.

[5] N. Holohan, S. Braghin, P. Mac Aonghusa, and K. Levacher, "Diffprivlib: The IBM differential privacy library," *arXiv preprint arXiv:1907.02444*, 2019.

[6] OpenDP Core Team, "OpenDP: An open-source library for statistical analysis of sensitive data," 2019. [Online]. Available: https://github.com/opendp/opendp

[7] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach, "A generic framework for privacy preserving deep learning," *arXiv preprint arXiv:1811.04017*, 2018.

[8] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, "When the curious abandon honesty: Federated learning is not private," in *Proc. 2023 IEEE 8th Eur. Symp. Security Privacy (EuroS&P)*, 2023, pp. 175–199.

[9] N. Patnaik, J. Hallett, and A. Rashid, "Usability smells: An analysis of Developers' struggle with crypto libraries," in *Proc. 15th Symp. Usable Privacy Security (SOUPS)*, 2019, pp. 245–257.

[10] S. Vadhan, "The complexity of differential privacy," in *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. Springer, 2017, pp. 347–450.

[11] T. R. G. Green and M. Petre, "Usability analysis of visual programming environments: A 'cognitive dimensions' framework," *J. Visual Languages Comput.*, vol. 7, no. 2, pp. 131–174, 1996.

[12] A. Slavkovic and J. Reiter, "O privacy, where art thou?: Statistical disclosure limitation research and practice: Fascinating and growing areas of importance," *CHANCE*, vol. 25, no. 1, pp. 34–37, 2012.

[13] I. C. Ngong, B. Stenger, J. P. Near, and Y. Feng, "Evaluating the usability of differential privacy tools with data practitioners," 2024. [Online]. Available: https://arxiv.org/abs/2309.13506

[14] B. Z. H. Zhao, M. A. Kaafar, and N. Kourtellis, "Not one but many tradeoffs: Privacy vs. utility in differentially private machine learning," in *Proc. 2020 ACM SIGSAC Conf. Cloud Comput. Security Workshop*. New York, NY, USA: ACM, 2020, pp. 15–26.

[15] P. Song, J. Sarathy, M. Shoemate, and S. Vadhan, ""I inherently just trust that it works": Investigating mental models of open-source libraries for differential privacy," *Proc. ACM Human-Comput. Interact.*, vol. 8, no. CSCW2, pp. 1–39, 2024.

[16] R. Sarathy and K. Muralidhar, "Evaluating Laplace noise addition to satisfy differential privacy for numeric data," *Trans. Data Privacy*, vol. 4, no. 1, pp. 1–17, 2011.

[17] F. K. Dankar and K. El Emam, "Practicing differential privacy in health care: A review," *Trans. Data Privacy*, vol. 6, no. 1, pp. 35–67, 2013.

[18] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillere, J. Klein, and Y. Le Traon, "Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub," in *Proc. 2013 IEEE 24th Int. Symp. Software Rel. Eng. (ISSRE)*, 2013, pp. 188–197.

[19] M. Green and M. Smith, "Developers are not the enemy!: The need for usable security APIs," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 40–46, 2016.

[20] N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, "Taxonomy of real faults in deep learning systems," in *Proc. ACM/IEEE 42nd Int. Conf. Software Eng.*, 2020, pp. 1110–1121.

[21] D. Piorkowski, S. D. Fleming, I. Kwan, M. M. Burnett, C. Scaffidi, R. K. E. Bellamy, and J. Jordahl, "The whats and hows of programmers' foraging diets," in *Proc. 2016 CHI Conf. Human Factors Comput. Syst.*, 2016, pp. 3063–3074.

[22] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[23] J. Sarathy, S. Song, A. Haque, T. Schlatter, and S. Vadhan, "Don't look at the data! How differential privacy reconfigures the practices of data science," in *Proc. 2023 CHI Conf. Human Factors Comput. Syst.*, 2023, pp. 1–19.

[24] G. M. Garrido, X. Liu, F. Matthes, and D. Song, "Lessons learned: Surveying the practicality of differential privacy in the industry," 2022. [Online]. Available: https://arxiv.org/abs/2211.03898

[25] C. Fiesler, M. Zimmer, N. Proferes, S. Gilbert, and N. Jones, "Remember the human: A systematic review of ethical considerations in Reddit research," *Proc. ACM Human-Comput. Interact.*, vol. 8, no. GROUP, pp. 1–33, 2024.

[26] G. Barthe, M. Gaboardi, J. Hsu, and B. Pierce, "Programming language techniques for differential privacy," *ACM SIGLOG News*, vol. 3, no. 1, pp. 34–53, 2016.

[27] H. Tang and S. Nadi, "Evaluating software documentation quality," in *Proc. 2023 IEEE/ACM 20th Int. Conf. Mining Software Repositories (MSR)*, 2023, pp. 67–78.

[28] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in GitHub," in *Proc. 2015 10th Joint Meeting Foundations Software Eng.* ACM, 2015, pp. 805–816.

[29] R. McKenna, D. Sheldon, and G. Miklau, "Winning the NIST contest: A scalable and general approach to differentially private synthetic data," *J. Privacy Confidentiality*, vol. 11, no. 3, 2021.

[30] P. Nanayakkara, J. Bater, X. He, J. Hullman, and J. Rogers, "Visualizing privacy-utility trade-offs in differentially private data releases," in *Proc. Privacy Enhancing Technol.*, vol. 2022, no. 2, 2022, pp. 601–618.