Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Between Accuracy and Privacy: Rethinking Developer-Identity De-Aliasing in Git Repositories

Thomas Bock · Shyam Agarwal · Curtis Atkisson · Audris Mockus · Bogdan Vasilescu

Received: 4 March 2026 / Accepted: 6 November 2026

Abstract Developer-identity de-aliasing (i.e., linking multiple aliases to the same person) is essential for accurate attribution of contributions in software repositories—a common need in many Mining Software Repositories (MSR) studies that rely on correct developer identities. While developing de-aliasing techniques has been a long-standing focus in the MSR community, linking developers' aliases can pose ethical problems as some contributors may not want their aliases linked. As privacy implications of de-aliasing have not been sufficiently explored, we investigate the tension between the promises of de-aliasing (better attribution, improved data quality) and its perils (potential privacy violations). We use GitHub's privacy-preserving email addresses as an example of a privacy signal and systematically analyze how respecting such signals affects de-aliasing accuracy. To accomplish that, we 1) develop GitAuthority, a configurable reimplementation of established de-aliasing techniques, as existing tools are not designed for this purpose, and 2) devise a sampling strategy to construct a large, recent dataset containing privacy-preserving aliases at scale, as existing datasets do not contain sufficient examples of this privacy indicator. We find that 44% of all GitHub projects (excluding forks) and 22% of the commits contain a privacy-preserving email address. Furthermore, we observe a substantial drop in de-aliasing accuracy when privacy signals are respected (recall drops from 0.77 to 0.47), underscoring the tension between privacy and accuracy. We provide practical guidance to help researchers navigate this complex landscape, acknowledging that the scientific value of de-aliasing must be balanced with ethical considerations around privacy protection.

Thomas Bock
Carnegie Mellon University, Pittsburgh, PA, USA. Email: bockthom@cmu.edu
✉ Corresponding author.

Shyam Agarwal
Carnegie Mellon University, Pittsburgh, PA, USA. Email: shyamaga@andrew.cmu.edu

Curtis Atkisson
University of Washington, Seattle, WA, USA. Email: catkiss@uw.edu

Audris Mockus
University of Tennessee, Knoxville, TN, USA. Email: audris@utk.edu

Bogdan Vasilescu
Carnegie Mellon University, Pittsburgh, PA, USA. Email: vasilescu@cmu.edu

Keywords Developer De-Aliasing · Version-Control Systems · Mining Software Repositories · Privacy Concerns

1 Introduction

In software development, each commit in version-control systems such as Git is associated with a developer's name and email address (if properly configured in their client). However, developers often use multiple email addresses, name variations, or aliases (Fry et al., 2020), making it challenging to accurately attribute contributions, especially between different projects (Amreen et al., 2020). For decades, the Mining Software Repositories (MSR) community has developed increasingly sophisticated techniques to *de-alias* these identities—linking different aliases to the same developer to enable accurate attribution for empirical studies on collaboration patterns, productivity, and software evolution (Hemmati et al., 2013; Bettenburg et al., 2009).

While using different aliases is often unintentional (e.g., inconsistent Git configurations across environments), developers may also *deliberately* use different identities to protect their privacy (Knothe and Pietschmann, 2019). For example, the social coding platform GitHub provides privacy-preserving email addresses (e.g., `username@users.noreply.github.com`), which obscure a developer's actual email address while still enabling attribution to their GitHub account. Despite the growing prevalence of such privacy signals (Knothe and Pietschmann, 2019; Alfieri et al., 2024; Sharma et al., 2025; Feitelson, 2023), existing de-aliasing approaches typically pursue maximum accuracy by linking as many aliases as possible, often disregarding indicators that suggest the developer's will to keep certain identities separate.

This privacy-agnostic stance creates a fundamental tension. On the one hand, comprehensive de-aliasing enables better attribution and improved data quality essential for mining software repositories research. It allows researchers to understand developers' contributions across projects, track developer evolution, and measure productivity and collaboration patterns reliably.

On the other hand, aggressive de-aliasing can violate developer privacy by linking aliases that developers deliberately keep separate, potentially revealing (1) connections between corporate work and voluntary personal projects, (2) employment history through institutional affiliations, or (3) real identities behind pseudonymous contributions. When developers actively choose privacy-preserving aliases, linking them to other identities may directly contravene their expressed preferences.

However, some developers may use privacy-preserving aliases simply to avoid spam or to keep their personal email address private (Baltes and Diehl, 2016), while others may use them to intentionally separate different aspects of their online presence or to maintain anonymity in certain contexts. Unfortunately, contacting developers to clarify their intentions behind using privacy-preserving aliases would itself violate their privacy and is therefore not a viable option for determining whether their aliases should be linked. Instead, we assume that a developer using a privacy-preserving alias prefers that such aliases should not be linked to other identities. While this assumption cannot capture every individual motivation for using such an alias, GitHub's privacy-preserving email addresses require a developer to explicitly opt in via a dedicated, clearly labeled setting that is disabled by default (see Section 5.1), which makes their usage an observable, deliberate action rather than an

incidental byproduct of platform defaults; we discuss the limitations of this proxy in more detail in Section 6.

Surprisingly, this privacy dimension has received little attention in the MSR literature (Vidoni and Díaz Ferreyra, 2022), despite its ethical importance (Gold and Krinke, 2020; Feitelson, 2023; Berry, 2004) and potential impact on research validity. Most existing approaches optimize for recall—merging as many aliases as possible—without considering whether some aliases *should not* be merged. This raises a critical question: *What is the cost of respecting developer privacy preferences in identity resolution?* More specifically:

> **RQ:** How does privacy-aware de-aliasing perform compared to privacy-agnostic approaches in terms of accuracy?

To investigate this question, we need to systematically compare privacy-aware and privacy-agnostic de-aliasing strategies. However, existing de-aliasing tools are not designed for this purpose—they lack the configurability to selectively honor privacy signals. Furthermore, existing ground-truth datasets predate the widespread adoption of GitHub's privacy-preserving email addresses and do not contain sufficient examples of this privacy indicator.

To address these challenges, we develop GitAuthority, a configurable reimplementation of established heuristic-based de-aliasing techniques, enhanced with refinements to address corner cases encountered during our analysis of privacy-sensitive patterns. The primary purpose of GitAuthority is to serve as a methodological tool that enables experimentation with different developer-identity de-aliasing strategies. By making key parameters of the de-aliasing techniques configurable (e.g., whether to respect privacy-preserving `username@users.noreply.github.com` email addresses), GitAuthority allows us to compare different privacy policies while holding other factors constant. In our study, we validate that GitAuthority with all privacy-preserving configuration options turned off achieves comparable accuracy to state-of-the-art approaches on four existing ground-truth datasets, ensuring that any observed differences in our privacy experiments stem from policy choices rather than algorithmic deficiencies.

We also address the data gap by developing a sampling strategy to construct a large, recent dataset containing privacy-preserving aliases at scale. Using this dataset, we systematically analyze how respecting privacy signals affects de-aliasing accuracy and provide insights into the trade-offs involved.

Our results reveal that about 30% of commits made in 2024 use GitHub's privacy-preserving email addresses, underscoring the prevalence of privacy signals in modern software repositories. Moreover, the use of GitHub's privacy-preserving email addresses has been growing steadily over the last decade (see Figure 3), indicating that they are becoming an increasingly common feature of developer identities rather than a marginal edge case. Honoring privacy signals leads to a measurable decrease in de-aliasing accuracy—recall drops from 0.77 to 0.47 on our large-scale GitHub dataset, and the F1-score drops from 0.80 to 0.70 on a manually validated sample—with varying impacts depending on the specific empirical analysis and its need for accurate de-aliasing. These findings have important implications for mining software repositories research. While comprehensive de-aliasing improves data completeness, it may come at the cost of violating developer privacy and potentially introducing

bias (e.g., if privacy-conscious developers are systematically misrepresented). Conversely, overly conservative privacy-aware approaches may fragment developer identities and distort aggregate statistics. Our work quantifies these trade-offs, providing guidance for researchers to make informed decisions based on their specific research contexts and ethical commitments.

These considerations become even more salient in the era of large language models and generative AI. As AI tools increasingly assist with code generation and commit message writing, behavioral signals traditionally used for identity matching (e.g., coding style, message patterns) may become less distinctive (Zhang and Saber, 2025; Terčon and Dobrovoljc, 2025; Kazemitabaar et al., 2023). Simultaneously, automated configuration of aliases may inadvertently affect identity representation, making deliberate privacy choices more important to detect and respect. Regardless of how code is produced, developers remain accountable for their contributions, underscoring the need for attribution systems that balance accuracy with ethical responsibility.

In summary, we make the following contributions:

- GitAuthority, a configurable de-aliasing tool, validated against four existing ground-truth datasets (Section 4). In addition to its full-accuracy configuration, GitAuthority enables controlled experimentation with privacy-aware strategies that honor GitHub's noreply addresses or first-name-only aliases as privacy signals. It also supports additional configurations of matching heuristics and blocklists that go beyond those evaluated in this article.
- An empirical characterization of GitHub's privacy-preserving noreply email addresses as a widespread and growing privacy signal (Section 5.1).
- A sampling strategy for constructing a large-scale ground-truth dataset containing this privacy signal at scale, addressing the lack of suitable existing data for this purpose (Section 5.2).
- Empirical results that quantify the resulting accuracy–privacy trade-off in developer-identity de-aliasing (Section 5.3).
- A discussion of the implications of this trade-off for common empirical analyses, along with practical guidance on how researchers can navigate this trade-off when applying developer-identity de-aliasing (Section 7).

While we cannot publicly release our datasets due to privacy concerns, we provide the implementation of GitAuthority and our evaluation scripts in a supplementary repository: `https://github.com/bockthom/gitauthority/`
In addition, it is also available at `https://doi.org/10.5281/zenodo.23790141`.

By explicitly examining both the promises and perils of developer-identity de-aliasing, we aim to foster more responsible and ethically informed practices in empirical software engineering research.

2 Definitions

As the literature uses key terms related to developer-identity de-aliasing differently, let us define them to ensure consistent usage:

Alias An *alias* (also called a *Git handle* or *Git identity*) is a combination of a name and an email address in the format `Name <prefix@domain.com>` that appears in the author or committer fields of Git commits. In Git, developers can configure their name and email address arbitrarily for each commit, for each repository, or for each

machine they use, leading to multiple aliases per person. For example, a developer might use `John Doe <john.doe@company.com>`, `J. Doe <j.doe@company.com>`, and `johndoe <johndoe@users.noreply.github.com>` as three different aliases.[1]

Identity, Developer, and Author An *identity* refers to a unique real-world person who contributes to software projects. Each identity may have multiple aliases. De-aliasing aims to map aliases to identities, recognizing that one person (identity) may use many different aliases across or within projects.[2]

We use the term *developer* or *developer identity* to refer to any person who contributes source code to a software project, which is also referred to as an *author* of a commit. In this article, we use "developer," "author," and "identity" interchangeably when referring to unique real-world persons.

User A *user* (or *GitHub user*, *GitHub handle*, or *GitHub account*) refers to an account on the social coding platform GitHub, identified by a unique username (the handle). Such an account can host multiple repositories. While there is often a one-to-one correspondence between users and developers, developers may maintain multiple GitHub accounts (e.g., for work and personal projects), and conversely, a single GitHub account might be shared by multiple people (e.g., bot accounts or organizational accounts).

De-Aliasing *De-aliasing* is the process of establishing a many-to-one mapping from multiple aliases to a single identity. This involves identifying aliases that belong to the same real-world person and grouping them accordingly. For example, de-aliasing would recognize that the aliases `Jane Smith <jane@uni.edu>`, `J. Smith <jane.smith@company.com>`, and `jsmith <jsmith@users.noreply.github.com>` all refer to the same person.

Disambiguation *Disambiguation* is the complementary process of determining whether a given alias corresponds to one or multiple identities. This is particularly relevant when an alias is ambiguous and could potentially refer to different people in different contexts (e.g., `Admin <admin@company.com>` might be a shared account used by multiple administrators). While de-aliasing focuses on merging aliases, disambiguation focuses on separating them when appropriate. The former may be conceptualized as grouping synonyms, while the latter as contextualizing homonyms.

Email Prefix & Email Domain The *email prefix*, also called *email username* (Amreen et al., 2020) or *email base* (Gote and Zingg, 2021), is the part of an email address before the @ sign (e.g., `john.doe` in `john.doe@company.com`). The *email domain* is the part after the @ sign (e.g., `company.com` in `john.doe@company.com`). For de-aliasing, these two components of an email address are often used separately.

Privacy-Preserving Email Address We refer to an email address that is designed to protect a developer's privacy by not exposing personal email addresses, such as GitHub's `username@users.noreply.github.com` (where the email prefix

[1] Here, we use "John Doe" as a placeholder for an actual name. However, if "John Doe" appears in actual repository data, such an instance could serve as a privacy signal.

[2] Note that an identity could also represent a bot (i.e., a non-human tool automating some task). However, we do not investigate identification of bot identities in the context of this article and, thus, we treat bots as if they were human developers.

usually is a numeric user ID followed by "+" and the user's username[3]), as a *privacy-preserving email address*. For brevity, we henceforth refer to such an email address as *noreply address*.

3 Background & Related Work

Before we analyze the trade-offs between accuracy and privacy in developer-identity de-aliasing, we provide background information and related work that addresses the two main topics of this work: (1) privacy and ethical considerations when dealing with developer identities in repository-mining studies, and (2) established approaches to disambiguate and de-alias developer identities in software repositories.

3.1 Privacy & Ethical Considerations

The trade-off between privacy and accuracy is a long-lasting and ongoing debate in data mining and data science (Li and Li, 2009; Palen and Dourish, 2003), but it has not been sufficiently explored in the context of mining software repositories.

3.1.1 Privacy in Socio-Technical Systems

Nissenbaum (2009) investigated privacy in the context of socio-technical systems and argued that privacy is not about strictly restricting access to personal information, but rather about ensuring appropriate flow of personal information in a particular context. To protect their privacy and obscure their real identities in online communities, people use technical approaches (e.g., using VPNs, using the Tor browser, using throwaway accounts, or having multiple accounts) as well as behavioral and operational approaches (e.g., using fictitious names, uploading faked profile pictures, hiding biographical information, or having multiple accounts, etc.) (Kang et al., 2013; Forte et al., 2017; Andalibi et al., 2016; Yamak et al., 2016). While these approaches aim to achieve privacy through anonymity, past research acknowledges that full anonymity is not achievable and that there is a trade-off between anonymity and identifiability (Forte et al., 2017; Brennan et al., 2012). For example, some participants of online communities hide their email addresses primarily to avoid receiving spam, but this may also hinder "maintaining their credibility" (Kang et al., 2013). Marx (1999) reflected on the rationale behind anonymity and identifiability from a sociological perspective, and suggested that there is no one-size-fits-all solution, but that the decision to favor anonymity or identifiability should be made on a case-by-case basis. Reasons for anonymity include protection from economic interests, the avoidance of persecution, or providing unidentifiable data for research, while reasons for identifiability include accountability and reputation reasons, legal obligations, and the support of longitudinal research, as "research may benefit from links to other types of personal data" or "may require tracking unique individuals" (Marx, 1999). In a study on Tor users and Wikipedia users, Forte et al. (2017) identified risks of surveillance, harassment, or reputation loss as reasons for anonymity and privacy. As

[3] https://docs.github.com/en/account-and-profile/reference/email-addresses-reference#your-noreply-email-address (accessed: 2025-10-20)

a consequence, they suggested that systems could offer participants the possibility to create new technical identities that are not linked to their past identities when taking on new roles. Although people are also often concerned about organizational practices (Smith et al., 1996) or identity theft, participation in online communities often requires disclosing some personal information, and people often have to make trade-offs between privacy and utility (Palen and Dourish, 2003). Nevertheless, in the age of large language models, automated de-anonymization utilizing agentic web searches across online platforms becomes easier than ever (Lermen et al., 2026), which puts online privacy in an entirely new light and raises even more questions with respect to ethical considerations and privacy concerns.

3.1.2 Ethical Considerations in Mining Software Repositories Research

Mining software repositories research is not exempt from these privacy concerns, as developers may have legitimate reasons to use different aliases to protect their privacy, such as separating work and personal projects, avoiding harassment, or maintaining anonymity in open-source contributions. Gold and Krinke (2020) discussed the ethical implications of mining software repositories research and highlighted the need for "balancing risks and benefits" when mining version-control data. After conducting a survey with 224 open-source software developers, Feitelson (2023) concluded that using version-control data for research is generally acceptable to developers, but that researchers should be transparent about their data collection and handling practices. In a similar vein, Vidoni and Díaz Ferreyra (2022) raised the question of whether mining software repositories research should be more privacy-aware, and they also provided a set of ethical guidelines for responsible data handling in this context. In particular, they suggested (1) to develop privacy-aware methodologies to prevent de-anonymization through data linking, (2) to assess the nuances of developers' involvement, and (3) to balance between privacy and utility and discuss the trade-offs. In this article, we take a step in this direction by investigating the tension between privacy and accuracy of de-aliasing approaches, and we also provide guidance for researchers on how to handle this tension in a responsible way.

3.1.3 Privacy Concerns in the Context of Software Repositories

In recent years, researchers have started to investigate privacy concerns in the context of software repositories on GitHub. Hennig et al. (2026) as well as Levi-Eshkol and Ribak (2025) analyzed developers' discussions about privacy and found that these discussions are primarily about public policies and commercial regulations. In contrast to the focus of our study, these discussions target the privacy of users of the software that is developed but not the privacy of the developers themselves. Aghili et al. (2025) examined privacy in software logs and identified that these often contain sensitive attributes such as IP addresses, file paths, usernames, or email addresses. While industry practitioners perceived technical attributes such as IP addresses or file paths as more sensitive than personal attributes such as usernames and email addresses, they also found it challenging to define a balance between privacy and utility. Whereas their analysis solely considered privacy-sensitive data in avoidable repository contents, in our study, we investigate this tension in the context of developer names and email addresses in version-control logs, which are an inherent part of the metadata in software repositories.

Closer to our work, Alfieri et al. (2024) paid attention to users' privacy settings on GitHub and identified that even if users adjust their setting for email addresses to be private in the user profile, this does not necessarily mean that their email addresses are not publicly available. For example, they found sensitive privacy information in pull-request comments and in the history of repositories, which can be discovered even if the user uses privacy settings. They therefore suggest that a more sophisticated approach for privacy protection is needed. Díaz Ferreyra et al. (2023) asked developers about their privacy concerns, and most developers reported that they feel safe in publishing information, while they also acknowledge the concern that their publicly available information could be collected by others. While this paradox is also present in similar research on privacy in social media (Barth et al., 2019; Kim and Kim, 2020), Díaz Ferreyra et al. (2023) call for more research on that in the software-engineering domain, requesting more transparency on how personal information is stored in repositories and how privacy-enhancing mechanisms work. In our study, we add to this line of research by shedding light on developer-identity de-aliasing and investigating the tension between privacy and accuracy in this context, which has not been investigated before.

3.2 De-Aliasing and Disambiguation Approaches

Identity matching (i.e., mapping aliases to unique persons) is a general problem that has been studied across various domains. Christen (2006) noted that persons use different name variations (e.g., with or without middle names or initials, different order of first and last names, abbreviating parts of names, etc.) and nicknames that are entirely different from their actual names. Therefore, Christen (2006) suggested different matching techniques, such as phonetic-based approaches (e.g., Soundex, Metaphone, etc.), string-similarity-based approaches (e.g., Levenshtein distance, Jaro-Winkler distance, etc.), and combinations thereof. Approaches from social network analysis try to match user identities across different platforms using additional information such as username, avatar, location, or occupation (Iofciu et al., 2011; Golbeck and Rothstein, 2008; Qian et al., 2011; Malhotra et al., 2012; Zhang and Yu, 2015), writing style (Liu et al., 2013, 2014), or user behavior (Liu et al., 2014).

While these approaches work in general-purpose settings such as social networks, Mo et al. (2015) argued that software communities are different in the way they use tools, provide names, and use a variety of different email addresses, for example, and therefore need different person disambiguation approaches that are dedicated to the software-engineering domain. Although social coding platforms such as GitHub might provide the intuition that they make identity matching obsolete as they provide unique user handles, previous research has shown that this is not the case: Approximately 22% of commits in GitHub repositories are not assigned to a registered user (i.e., the email addresses used by the commit authors are not associated with a GitHub account) (Kalliamvakou et al., 2016). Moreover, Kamola (2019) reported that user data cleaning is necessary to get rid of "ubiquitous, artificial 'developers' " that have wrongly been merged on GitHub due to using a common (fake) email address. Consequently, developer-identity de-aliasing and disambiguation on version-control data is still a relevant problem.

3.2.1 Heuristic-Based Approaches in the Software-Engineering Domain

Research in the software-engineering domain has developed a variety of different approaches to disambiguate developer identities based on names and email addresses that are used in version-control systems, as these are the most commonly available attributes in version-control systems. In addition, researchers have also enriched these attributes with other information. In what follows, we provide an overview of different heuristic-based approaches that have been proposed in the literature to disambiguate developer identities in software repositories. In our study, we make use of these heuristics (as long as they are simply based on substring matching and do not involve the computation of string similarity distances) to implement a configurable de-aliasing approach that allows us to investigate the trade-off between privacy and accuracy in developer-identity de-aliasing (see Section 4).

More than 20 years ago, Robles and Gonzalez-Barahona (2005) demonstrated the relevance of developer-identity de-aliasing when analyzing software projects and their various interaction channels (e.g., version-control systems, mailing lists, issue trackers). They proposed a set of heuristics to match aliases based on names and email prefixes, as well as GPG keys to map different email addresses belonging to the same developer. Notably, they pointed out that manual verification is necessary to ensure high precision, as there will always be false positives and false negatives when applying automated heuristics to large datasets of developer identities (Robles and Gonzalez-Barahona, 2005).

Similarly, Bird et al. (2006) figured out that any developer-identity de-aliasing strategy will be "inexact" and should be "supplemented by subsequent manual analyses" because people can choose any arbitrary alias that might not be related to their name—up to 11 aliases per person were not uncommon. Focusing on mailing-list data as a primary communication channel of developers, they applied string similarity measures (such as Levenshtein distance) between first names, last names, and email addresses to build clusters of aliases that might belong to the same developer. However, due to too many false positives, manual inspection was necessary to split up the resulting clusters. As these false positives primarily originated from the use of string similarity measures, Canfora et al. (2011) refined this approach and avoided using string similarity measures: They first matched identities based on various name parts (e.g., first name, last name, initials), then by email addresses (e.g., full email address, or email prefix only), and finally by matching email prefixes with name parts. For mapping names, they employed a number of different heuristics (e.g., ignoring middle names, using initials of first names if they are unique in the dataset, and composing name parts in various ways).

Poncin et al. (2011) searched for compositions of any initials with any name parts in nicknames and email prefixes. In addition, they assigned weights to alias mappings based on the number of matching heuristics (e.g., full email match and email prefix match), but still manual corrections remained necessary.

Kouters et al. (2012) applied latent semantic analysis using various normalization and transliteration steps to names and email addresses (e.g., removing accents/diacritics, digits-only words, short names like jr., dr., etc.), computed Levenshtein distances to account for misspellings, and weighted names based on their frequency of occurrence to avoid merging persons just because of having common first names. Gharehyazie et al. (2015) noticed that some common email prefixes or generic names

(e.g., “jira”, “admin”) need to be ignored, as they are often used by different developers and therefore lead to false positives if they are used for matching.

Oliva et al. (2012) matched developers only by names. Goeminne and Mens (2013) normalized names and email addresses (e.g., removing accents, trimming whitespace, and converting to lowercase), and matched them if they were at least 3 characters long. Kouters et al. (2012) showed that this simple approach performed better than the more complex approach of Bird et al. (2006) that used string similarity measures, while the more advanced approaches performed better on noisy data.

Vasilescu et al. (2015) used simple heuristics from previous approaches, but recommended being “as conservative as possible” to avoid wrong merges. Therefore, they merged identities primarily by email addresses (excluding “fictitious email addresses”), and considered other pieces of information (e.g., GitHub usernames, names, email prefixes, email domains, and locations) if email addresses did not match. In addition, Vasilescu et al. (2014) identified project-specific name formats (e.g., developer names in GNOME were prefixed by a timestamp, requiring timestamp removal) and noted that similarity measures often fail if first name and last name are swapped or middle names are used. To allow transitive merges of aliases, they built a graph with identities as nodes and matches between them as edges, and performed de-aliasing via identifying connected components. This way, they found up to 168 aliases belonging to a single person (who used commit messages instead of their name) in their GNOME dataset, with an average of 1.36 aliases per person (Vasilescu et al., 2014). Lamba et al. (2020) extended the graph-based algorithm of Vasilescu et al. (2014) by introducing two passes: In the first pass, matching heuristics are applied only to identities within projects; in the second pass, matching heuristics are applied across projects using only a subset of the heuristics, to minimize false positives arising from name collisions (as name collisions of different developers are more likely across projects than within projects).

Gote and Zingg (2021) proposed the tool GAMBIT that applies a set of previously used developer-identity matching heuristics by applying similarity measures on names and email addresses. Similar to our goal, their approach does not need training or external information from outside the version-control history and therefore scales well to large datasets. Using a semi-automatically created ground-truth dataset of 1 896 aliases from project GNOME GTK, their approach achieved a higher precision with similar recall when compared to the approaches of Goeminne and Mens (2013) and Bird et al. (2006), which also rely on similarity measures.

3.2.2 Comparison of Different Approaches

To compare different identity merge algorithms, Goeminne and Mens (2013) created a dataset of developer identities based on information from AUTHORS and MAINTAINERS files in 3 projects, manually validated by 3 persons that were knowledgeable about the investigated projects without being involved in them.[4] They found that merging aliases reduces the number of identities by 9–12%. Strict naming conventions in projects ease de-aliasing of identities and make the use of similarity measures unnecessary, as best precision and recall were achieved for direct matches. Restricting names to at least 3 characters achieves best precision, as shorter names

[4] Privacy reasons hindered ground-truth validation by the projects themselves.

are often ambiguous and requiring more than 3 characters leads to many false negatives. In addition, the more people are engaged in a project, the more difficult it is to match identities correctly, as people tend to have a higher preference for "fancy" names, nicknames, or anonymity in larger projects (Goeminne and Mens, 2013).

Wiese et al. (2016) compared 6 identity merge algorithms (Robles and Gonzalez-Barahona, 2005; Bird et al., 2006; Canfora et al., 2011; Oliva et al., 2012; Kouters et al., 2012; Goeminne and Mens, 2013) using data from 150 developer mailing lists from the Apache Software Foundation, which contained more than 315 000 different email addresses. To create a ground truth from that, they collected name and email data from a public key repository, previous literature, and the projects' JIRA issue tracker, ending up with 4 563 email addresses that belonged to 1 639 different persons (with an average of 2 email addresses per person and a maximum of 15 email addresses per person). All 6 algorithms performed similarly well on this dataset, while simple heuristics (such as Oliva et al., 2012) achieved higher precision than the more complex ones that used similarity measures or latent semantic analysis. Obviously, the larger the projects and the longer a project's history, the worse was the performance of the algorithms, as larger datasets contain more email addresses and therefore more possibilities for false or missing merges.

3.2.3 Machine-Learning-Based Approaches in the Software-Engineering Domain

In recent years, machine-learning-based de-aliasing approaches emerged, aiming for higher accuracy and less manual intervention than heuristic-based approaches, despite the fact that they require training data and are more complex to implement.

Mo et al. (2015) used decision trees with greedy matching to match developer identities from GitHub and StackOverflow (which does not expose email addresses), applying similarity measures to usernames, discussed topics, and tags that describe the developers' skills. They figured out that using such tags could be problematic because they are often self-assigned and thus might not match if people assign different tags to themselves on different platforms. For evaluation, they created a small ground-truth dataset of 6 000 persons by solely relying on the 10% of identities that linked to the same personal website in their GitHub and StackOverflow profiles.

Similarly, Xiong et al. (2017) used supervised machine learning with three groups of features: a variety of different string similarity measures on usernames, behavioral features (e.g., times being active, used programming languages, etc.), and the writing style of developers (e.g., length of comments, vocabulary, etc.). Using a similar dataset as Mo et al. (2015) of 8 000 persons, combining all these features proved to be useful, but the combination only performed slightly better than relying on usernames alone. We deem writing style and behavioral features less useful in times of generative AI, as AI models tend to produce text in a more uniform style (Zhang and Saber, 2025; Terčon and Dobrovoljc, 2025), and developers might use AI tools to generate code snippets in programming languages they are not familiar with (Kazemitabaar et al., 2023; Xu et al., 2022; Wong et al., 2023).

Fry et al. (2020) used random forests to match developer identities based on names and email addresses in the Git version-control archive *World of Code (WoC)*, as well as mapping data from a small dataset of GitHub usernames. Their approach reduces the entire dataset of authors by creating blocks of authors that are considered to be disconnected using basic heuristics (such as email matching or name matching), and performs pairwise comparisons of author identities only within these blocks, to

reduce the number of comparisons. However, they acknowledge inaccuracy of their approach for missing separation of first and last names, non-English names, blank names, email addresses, shared email addresses, and people with the same name. While they applied their approach on 14.9M authors in WoC (being reduced to 5.4M after merging identities, with an average developer having 2 aliases), they evaluated their approach only on a small ground-truth dataset of 44 authors that has been manually created by actual developers, accounting for 208 different aliases.

Amreen et al. (2020) used active learning with name frequency and so-called behavioral fingerprints (i.e., similarity of modified files, timezones, and commit messages) to match developer identities. First, they identified common error patterns for synonyms (different aliases belonging to the same person, for example due to spelling mistakes or name variations) and homonyms (different people using the same name, for example due to common names such as "John Smith" or generic names such as "admin"). Then, they trained a random-forest classifier on sampled datasets that contain these error patterns. To evaluate their approach, they created a manually validated ground-truth dataset of 1 060 author–ID pairs from OpenStack, as well as a dataset of 10 344 authors from a commercial company and a GitHub dataset with 23 493 projects. Behavioral fingerprints (such as commit messages or timezones) may become less reliable as generative AI produces code of a more uniform style (Zhang and Saber, 2025; Terčon and Dobrovoljc, 2025) and work on remote servers or VPNs can hide actual timezones (Liu, 2023). Crucially, using inverse name frequency is essential for avoiding false positives in large cross-project collections (i.e., preventing totally unrelated identities from being merged into huge aliased networks), but it dramatically reduces the accuracy of fully connecting the aliases of developers with common names. To mitigate the risk of people with common names being merged incorrectly, in our study, we ignore the frequency of names entirely. To avoid creating massive networks of unrelated aliases based on false positive matches, we suggest restricting across-project merges to email addresses only. The same (proper) email address tends to be used by multiple developers much less frequently than the same names are shared by multiple developers. This, however, prevents linking authors who exclusively use work email for work projects and home email for personal projects.

3.2.4 Interactive Tools and Visualization Tools

Moreno et al. (2019) proposed the tool SortingHat, which enables users to add author instances via files, applies matching heuristics (based on names, usernames, and email addresses), and allows users to interactively add authors and manually merge identities or split wrongly merged identities. Furthermore, their tool allows adding a blocklist for excluding organization names or generic email addresses that should not be considered for merging identities.

Campanella and Lanza (2024) developed a tool to visualize developer identities and their contributions to a project over time, applying various similarity measures to match developer identities based on names, email addresses, or fingerprints (e.g., similarity of modified files). This way, they allow visualizing per-alias contributions of a developer and clusters of aliases that were active around the same time.

3.2.5 Source-Code-Authorship Attribution Approaches

Besides applying de-aliasing strategies to developers' metadata (such as name or email address), researchers have also developed multiple approaches to attribute source-code authorship to developers by trying to identify authors of files using different kinds of models (Gong and Zhong, 2025; Caliskan-Islam et al., 2015; Yang et al., 2017; Bogomolov et al., 2021; Li et al., 2022; Gong and Zhong, 2022; Li et al., 2024). While such approaches can be promising in some scenarios, they have their own caveats and are only applicable in certain contexts. For example, they only work well for authors that are part of the training data, and for files mostly authored by a single developer (Gong and Zhong, 2025), but do not work well when applied on real-world data (Bogomolov et al., 2021). In times of large-scale usage of generative AI, individual code styles might become less distinguishable, as AI models tend to produce code in a more uniform style (Zhang and Saber, 2025; Terčon and Dobrovoljc, 2025), but might still be committed by different human developers who are accountable for the committed changes. Thus, such developer identification approaches based on code style might become even less reliable in the future, but there still is a need to attribute developers' contributions correctly. In addition, approaches based on code style usually require a substantial amount of code to be effective, which is not always available (e.g., for developers with only few commits or small contributions). Also, approaches that rely on code style usually cannot be applied to other types of contributions, such as communication on mailing lists or issue trackers, which are also important parts of developers' contributions to software projects. Consequently, our approach focuses on de-aliasing developers based on their metadata, as this is more generally applicable (also to other interaction channels outside of the software-engineering domain which developers might use, such as chat applications like Slack or Zulip) and does not have the limitations of source-code-related approaches.

4 Configurable Developer-Identity De-Aliasing

To analyze the trade-off between accuracy and privacy in developer-identity de-aliasing, we need a de-aliasing approach that is accurate and that can be configured to accommodate different privacy needs in order to enable us to study this trade-off systematically. Previous approaches from the literature either do not provide such configurability out of the box, or they are not accurate enough to be used as a basis for our study. To that aim, we reimplement existing heuristic-based approaches from the literature and add refinements to these heuristics to address specific corner cases that we have identified through working with existing approaches. In line with the contributions of Gote and Zingg (2021), we aim for an approach that does not require training data or external information, as this makes the approach more scalable to large datasets and easier to apply in practice. Consequently, we decided to reimplement a heuristic-based approach, as these approaches are more transparent and easier to configure than machine-learning-based approaches, which usually require retraining when new data is added or when the configuration needs to be changed.

4.1 Implementation of GitAuthority

Our de-aliasing approach GitAuthority uses a variety of existing heuristics from the literature (see Section 3.2.1), in particular the ones of the most recently developed approach by Gote and Zingg (2021), which has been shown to be accurate and scalable, but also draws on heuristics from other approaches. In what follows, we describe our reimplementation of these heuristics in detail (see the respective references to the literature for each of the steps) as well as the refinements that we have added to it (i.e., special treatment of German umlauts, leetspeak normalization, nickname resolution, GitHub username matching, dedicated matching rules with regard to parts of email domains, etc.).

GitAuthority is implemented in Python and therefore easy to set up, and it comes with an external configuration file in which each of the following preprocessing and matching heuristics can be enabled or disabled individually to accommodate different privacy needs and other requirements. Note that the default configuration enables all heuristics and preprocessing steps to maximize accuracy. In particular, our approach consists of four components: (1) normalization of aliases, (2) blocklisting of generic and other special aliases that might lead to merging identities incorrectly, (3) heuristic-based matching of developer aliases via matching rules, and (4) merging of aliases into identities and name selection to assign a unique identifier to each developer identity. Figure 1 provides an overview of this pipeline, illustrating how multiple raw aliases of the same developer (e.g., differing in name format and email address, or using a privacy-preserving email address) are normalized, filtered using blocklists, compared pairwise to collect clues, and finally merged into a single developer identity with a selected representative name.

4.1.1 Normalization

Before applying matching heuristics, we normalize developer aliases to standardize their representation and reduce variations. Each of the following normalization steps can be enabled or disabled via the configuration file. Normalization steps that are annotated with a reference to the literature have been used in previous approaches, while the steps without such a reference are refinements that we added.

Lowercasing Converts an alias to lowercase at the beginning of the normalization process to enable case-insensitive matching (Canfora et al., 2011; Kouters et al., 2012; Goeminne and Mens, 2013; Gote and Zingg, 2021).

Unicode Normalization Normalizes an alias to Unicode NFC (composed form) to ensure consistent character representation across encodings (e.g., representation of "ü" as a single character vs. as "u" plus diacritic[5]) (Kouters et al., 2012).

Metadata Removal Removes artifacts prepended or appended to aliases, such as timestamps ("13:37:28") and timezone abbreviations ("UTC") (Vasilescu et al., 2014; Gote and Zingg, 2021).

[5] Note that this is a purely encoding-level change with no effect on appearance: both the decomposed form ("u" + combining diacritic) and the composed NFC form render identically as "ü".

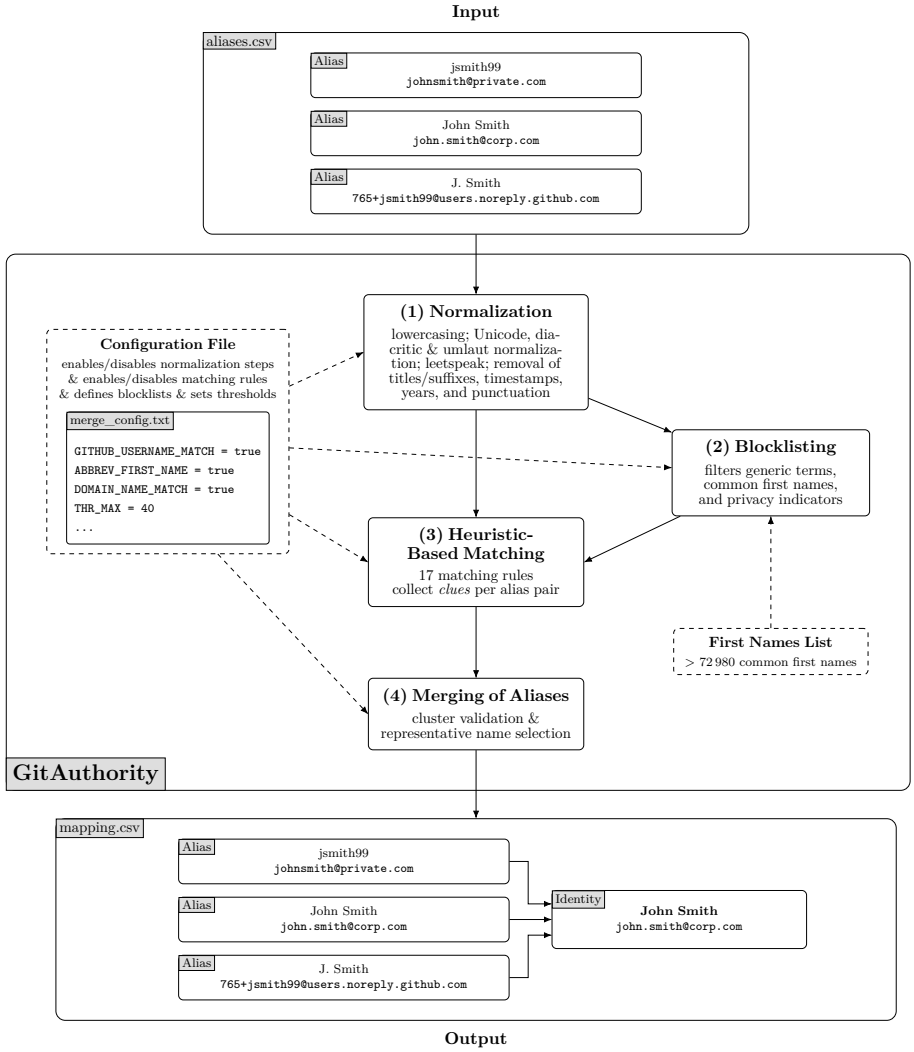


Fig. 1 Overview of the GitAuthority de-aliasing pipeline, from raw developer aliases to a merged developer identity.

Character Transliteration Context-aware character mapping handles multiple writing systems including Latin-based scripts (Romance, Germanic, Slavic), Cyrillic, Greek, and special characters from Turkish, Hungarian, and Baltic languages (Kouters et al., 2012).

Diacritics Removal Diacritics (accents) are removed from characters to standardize names across different languages and typing conventions (e.g., “é” to “e”, “ñ” to “n”) (Kouters et al., 2012; Goeminne and Mens, 2013).

German-Specific Normalization of Umlauts While diacritics are generally replaced with base characters (e.g., “ú” with “u”), German umlauts need to be represented by adding an “e” after the base character (e.g., “Müller” is transcribed

to “Mueller” when umlaut characters are not available). We replace “ä” with “ae”, “ö” with “oe”, “ü” with “ue”, and “ß” with “ss” for German and Nordic names. Since we cannot determine if a name is German, we apply both general diacritic removal and German-specific normalization, to use both string variants for matching.

Leetspeak Normalization Common leetspeak[6] substitutions are reversed to their standard alphabetic equivalents (e.g., “4” to “a”, “3” to “e”, “1” to “l”, “0” to “o”). As we cannot determine if a name is leetspeak or just contains digits for other reasons, we apply both the original and the leet-normalized versions for matching.

Encoding Correction Corrects encoding issues such as double-encoded characters (e.g., “Ã¡” to “á”), and replaces or removes Unicode replacement characters (“”) (Kouters, 2014).

Prefix and Suffix Removal Removes academic titles (Dr., Prof., Ph.D.), honorifics (Mr., Mrs., Ms.), and generational suffixes (Jr., Sr., II–X) to focus on core name components (Kouters et al., 2012; Goeminne and Mens, 2013; Gharehyazie et al., 2015; Gote and Zingg, 2021).

Punctuation Removal Removes all punctuation marks, including periods, commas, hyphens, underscores, quotes, parentheses, brackets, slashes, and special characters (.,:;!?’"()[]{}/@#$%^&*+=|<>~`-_) (Bird et al., 2006; Canfora et al., 2011; Gote and Zingg, 2021).

Digit Handling Preserves numbers if a text string starts with a digit; otherwise, digits are stripped to allow matches of actual names while accommodating identifiers beginning with numbers (Kouters et al., 2012).

Middle Name Removal For three-part names, middle names are removed if they are single-character initials or common first names, enabling matches between “Firstname M. Lastname” and “Firstname Lastname” (Canfora et al., 2011).

4.1.2 Blocklists

To prevent false-positive matches, we use blocklists applied to substrings used for matching (e.g., full names, name parts, email prefixes, domains). Generic names (e.g., “admin”, “user”, “support”) or organizational names (e.g., “team”) are blocklisted (Bird et al., 2006; Goeminne and Mens, 2013; Gharehyazie et al., 2015; Gote and Zingg, 2021; Moreno et al., 2019). We use predefined lists from previous literature supplemented with names from our preliminary investigations and suggestions from GitHub Copilot after manual review. Our predefined generic name blocklist contains:

- Git hosting providers (github, gitlab, bitbucket, etc.)
- Email service providers (gmail, yahoo, hotmail, etc.)
- Generic organizational terms or users (admin, support, team, developer, bot, etc.)
- Machine-generated names (localhost, ubuntu.local, macbook-pro.local, etc.)

[6] https://christianespinosa.com/blog/leetspeak-the-history-of-hacking-subcultures-native-tongue/ (accessed: 2025-10-20)

All entries can be configured via the external configuration file, and users can add their own entries. The complete blocklist is available in our supplementary materials.

Furthermore, we also maintain predefined blocklists for first names, as merging based solely on common first names is too risky (i.e., if people use only their first name as their name in a commit, merging two of such aliases would likely lead to many false positives, as many people share this first name if it is a common first name). We use common first names from the Wiktionary Names Appendix[7], which contains more than 72 980 first names from different languages.

In addition to the above blocklists that are used to prevent false-positive matches, we allow configuring a blocklist of privacy indicators such as privacy-preserving email patterns (e.g., GitHub's email domain "users.noreply.github.com"). Optionally, aliases whose entire name consists of nothing but a common first name (e.g., an alias simply named "Josh") can also be excluded from merging entirely via a dedicated configuration flag.

4.1.3 Heuristic-Based Matching

After normalization, we apply a set of heuristics to match developer aliases based on rules that consider different aspects of names and email addresses, as long as the respective components that are to be compared in a specific rule are not part of the blocklist. A matching rule defines whether two aliases match based on a specific criterion (e.g., exact email match), but does not directly decide whether two aliases belong to the same developer. All the matching rules that match for each pair of aliases are collected as *clues* to decide later (see Section 4.1.4) whether the two aliases belong to the same developer (Vasilescu et al., 2014; Lamba et al., 2020). This way, when deciding later, we can require a certain combination of matching rules to increase the confidence of matching aliases.

Each of the following matching rules can be individually configured to be used or not, and all of the applied rules are performed independently of each other on the normalized aliases. We apply a matching rule only if the components that it affects are not part of the configured blocklist and are at least 3 characters long (configurable via THR_MIN_LENGTH), following previous research (Goeminne and Mens, 2013) to avoid incorrectly merging initials. Rules that are annotated with a reference to the literature have been used in previous approaches, while the rules without such a reference are refinements that we added.

Email-Based Rules

1. EMAIL:
   Exact email match (Robles and Gonzalez-Barahona, 2005; Bird et al., 2006; Vasilescu et al., 2015, etc.)
2. SIMPLE_EMAIL_PREFIX:
   Matching email prefixes (Canfora et al., 2011; Vasilescu et al., 2015)
3. COMP_EMAIL_PREFIX:
   Matching compound email prefixes that contain a dot, hyphen, or underscore, including reversed component orders (e.g., "john.smith" matches "smith.john")
4. DOMAIN:
   Exact email domain match (Vasilescu et al., 2015)

[7] https://github.com/solvenium/names-dataset/ (accessed: 2025-09-14)

5. DOMAIN_MAIN_PART:
   Matching domain main part (second-level domain) with identical email prefixes (e.g., "john.smith@institute1.uni-location.edu" matches "john.smith@department2.uni-location.edu")

Cross-Field Rules

6. PREFIX_LOGIN:
   Email prefix matches login/username (Poncin et al., 2011; Vasilescu et al., 2015)
7. PREFIX_NAME:
   Email prefix matches full name (normalized, and spaces, hyphens, and underscores removed) (Canfora et al., 2011; Vasilescu et al., 2015)
8. LOGIN_NAME:
   Login/username matches full name (Poncin et al., 2011; Vasilescu et al., 2015)
9. DOMAIN_NAME_MATCH:
   Domain part matches normalized name components, including patterns like "firstname@lastname.com" matching "Firstname Lastname"
10. GITHUB_USERNAME_MATCH:
    GitHub's privacy-preserving noreply email pattern (e.g., "12345678+username@users.noreply.github.com") matches the full name (in this example, "username")

Name-Based Rules

11. FULL_NAME:
    Exact full name match (after normalization) (Oliva et al., 2012; Goeminne and Mens, 2013; Vasilescu et al., 2015, etc.)
12. SWITCHED_NAME:
    Names with first and last name switched (e.g., "John Smith" and "Smith John") (Canfora et al., 2011)
13. COMMA_SUFFIX_MATCH:
    Names identical except for comma-separated suffixes (e.g., "John Smith" and "John Smith, Ph.D.")
14. ABBREV_FIRST_NAME:
    Abbreviated first name matches full first name (e.g., "J. Smith" matches "John Smith", or "jsmith" email prefix matches "John Smith") (Canfora et al., 2011; Poncin et al., 2011)
15. ABBREV_LAST_NAME:
    Abbreviated last name matches full last name (e.g., "John S." matches "John Smith", or "johns" email prefix matches "John Smith") (Canfora et al., 2011; Poncin et al., 2011)
16. ABBREV_MIDDLE_NAME:
    Abbreviated middle name matches full middle name (e.g., "John M. Smith" matches "John Michael Smith", or "jmsmith" email prefix matches "John Michael Smith") (Canfora et al., 2011; Poncin et al., 2011)
17. NICKNAME_MATCH:
    First name is a known nickname of the other alias's first name, with an identical last name (e.g., "Jim Smith" matches "James Smith"), using an existing

curated nickname dictionary[8]; ambiguous nicknames that could resolve to several full names (e.g., "Bobby" could be short for "Robert" or "Bob") are excluded by default to limit false positives, but can be enabled or manually resolved to a specific full name via the configuration file

Abbreviation matching includes sophisticated pattern recognition for email prefixes, handling formats such as "jsmith" (first initial + last name), "smithj" (last name + first initial), "johns" (first name + last initial), "jmsmith" (first initial + middle initial + last name), and "smithjm" (last name + first initial + middle initial); see also Canfora et al. (2011) and Poncin et al. (2011). Single-word names are parsed to detect potential abbreviation patterns.

4.1.4 Merging of Aliases

The final step in our de-aliasing approach is to merge aliases into a single identity when an appropriate set of clues has been collected for a pair of aliases through the application of sufficiently many matching rules (to avoid low-confidence matches), and we finally select a representative name for each cluster of merged aliases.

Clue Collection For each alias pair, GitAuthority collects all matching rules from Section 4.1.3 that apply to this pair, creating a set of clues, as Vasilescu et al. (2015) did. Aliases are merged based on the following priority of clues (i.e., they are merged if one of the following three conditions is met):

1. EMAIL match (highest confidence)
2. Two or more distinct clues (high confidence through corroboration) (DOMAIN_NAME_MATCH does not count as independent from DOMAIN or DOMAIN_MAIN_PART here, since all three derive from the same underlying email-domain signal)
3. Single strong clue: FULL_NAME, SWITCHED_NAME, COMP_EMAIL_PREFIX, PREFIX_NAME, DOMAIN_NAME_MATCH, GITHUB_USERNAME_MATCH, COMMA_SUFFIX_MATCH, NICKNAME_MATCH, or any of the three abbreviation rules ABBREV_*

Then, the formed clusters of merged aliases are validated against multiple conditions (as done by Vasilescu et al., 2015) to prevent spurious merges. In particular, our approach uses configurable thresholds THR_MIN (default: 1) and THR_MAX (default: 40) to control minimum and maximum cluster sizes, preventing spurious merges of overly common identifiers.

Representative Name Selection For each valid cluster, a representative name is selected from all aliases of this cluster. While previous work has employed named entity recognition models to select the most informative name (Campanella and Lanza, 2024), we use a simpler heuristic-based approach that prioritizes more complete and informative names based on the following priority of three criteria:

1. Two-part name with both components having at least 2 characters, avoiding email prefixes containing "+" symbols
2. Longest/most complete name with proper title case capitalization
3. Earliest name in the cluster (which refers to the alias that first occurred in the dataset, assuming aliases are processed in chronological order)

[8] Using the Python package `nicknames` (https://pypi.org/project/nicknames/), which is based on an existing hand-curated dictionary of names and nicknames available at https://github.com/carltonnorthern/nicknames/ (accessed: 2026-08-13).

4.2 Evaluation of GitAuthority

To demonstrate that our approach GitAuthority meets the accuracy of previous state-of-the-art approaches, we evaluate it using four ground-truth datasets from the literature and compare it against two approaches (a heuristic-based one and a machine-learning-based one).

4.2.1 Datasets and Approaches

We use four existing ground-truth datasets that vary in size and origin (see Table 1): The smallest dataset (in terms of the number of aliases it contains) has been created by Amreen et al. (2019) and contains 147 aliases from 31 developers who manually selected their aliases from Git commits in *World of Code (WoC)* using their Developer Reputation Estimator (DRE) tool. Later on, Amreen et al. (2020) created a dataset of 1 060 author–ID pairs from 1 482 aliases of 1 030 OpenStack users via manual annotation. Wiese et al. (2016) created a dataset from mailing lists of the Apache Software Foundation (ASF) with 315 000 email addresses from public key repositories, project websites, JIRA issue trackers, and Git commits. Their final dataset contains only 1 572 aliases of 1 298 persons. Most recently, Gote and Zingg (2021) semi-automatically created a dataset of 1 896 unique aliases from Gnome GTK commits. Notice that the datasets from Amreen et al. (2019) and Wiese et al. (2016) had been created across multiple projects, whereas the datasets from Gote and Zingg (2021) and Amreen et al. (2020) had been created from a single project. While Wiese et al. (2016) restricted their project selection to a particular ecosystem and relied on its specific technical infrastructure, such restrictions do not apply to the dataset from Amreen et al. (2019).

Table 1 Overview of existing ground-truth datasets for developer-identity de-aliasing

Dataset	Source	#Aliases	#Persons	# Evaluation Pairs of Aliases
Amreen et al. (2019)	WoC + DRE	147	31	10 731
Amreen et al. (2020)	OpenStack	1 482	1 030	1 060
Wiese et al. (2016)	ASF	1 572	1 298	1 234 806
Gote and Zingg (2021)	Gnome GTK	1 896	1 156	1 796 460

To evaluate our approach GitAuthority on these ground-truth datasets, we compare it against two state-of-the-art approaches: a heuristic-based one and a machine-learning-based one. For the heuristic-based baseline approach, we choose gambit (Gote and Zingg, 2021) as it is the most recently developed tool, can be easily applied out of the box, and shows the highest accuracy based on recent literature. For the machine-learning-based baseline approach, we choose ALFAA (Amreen et al., 2020) for pragmatic reasons as it is implemented and practically used in the large-scale version-control repository archive *World of Code (WoC)*, and thus we are able to use ALFAA's results without the need to train its parameters on our datasets on our own (which would otherwise be necessary and time-intensive). Consequently, whereas we run GitAuthority and gambit on each of the four datasets ourselves,

we instead rely on ALFAA's de-aliasing results from WoC (version V3), which has already applied ALFAA to the aliases of these datasets.

We use standard evaluation metrics from the literature (Amreen et al., 2020; Gote and Zingg, 2021), where TP denotes correctly merged alias pairs, TN correctly unmerged pairs, FP incorrectly merged pairs, and FN incorrectly unmerged pairs:

Precision	= TP / (TP + FP)
Recall	= TP / (TP + FN)
F1-Score	= 2 × (Precision × Recall) / (Precision + Recall)
Accuracy	= (TP + TN) / (TP + TN + FP + FN)
Balanced Accuracy	= (Recall + (TN / (TN + FP))) / 2
Lumping	= FP / (TP + FN)
Splitting	= FN / (TP + FN)

Lumping and splitting (Smalheiser and Torvik, 2009) are more easily interpretable measures for de-aliasing than precision or recall. Following the definitions of Ventura et al. (2015) and Amreen et al. (2020), lumping represents the rate of incorrectly merged pairs (i.e., different persons merged into one identity), while splitting represents the rate of incorrectly unmerged pairs (i.e., aliases of the same person split into multiple identities).

4.2.2 Evaluation Results

Table 2 shows that GitAuthority (in its privacy-agnostic full-accuracy mode) outperforms ALFAA and matches gambit's performance across all four datasets, demonstrating that our approach meets and exceeds state-of-the-art accuracy. Notably, GitAuthority achieves very low splitting rates on all datasets, preventing contributions from being split across multiple identities, with slightly higher recall than gambit on three of the four datasets while maintaining similar precision. The higher recall of GitAuthority is due to its more comprehensive set of matching rules that were not used in gambit, such as nickname matching and GitHub username matching, as well as more advanced normalization techniques (e.g., leetspeak normalization, normalization of umlauts, and middle name removal).

To obtain an understanding of the limitations of GitAuthority, we manually inspected its false positives (FPs) and false negatives (FNs) across all datasets:

Gnome GTK:	10 FPs,	12 FNs
WoC + DRE:	0 FPs,	37 FNs
OpenStack:	5 FPs,	1 FN
ASF:	0 FPs,	8 FNs

For the vast majority of FPs, we are highly confident that GitAuthority made the correct decision to merge aliases and believe that the ground-truth dataset is wrong in these cases, acknowledging that also human labelers who had created these ground-truth datasets could have made mistakes or overlooked common strings (e.g., people's nicknames or pseudonyms appearing in email prefixes or domains). The FNs

Table 2 Evaluation results of developer-identity de-aliasing approaches on existing ground-truth datasets

Dataset	Approach	Precision	Recall	F1-Score	Accuracy	Balanced Accuracy	Lumping	Splitting
WoC + DRE	ALFAA (Amreen et al., 2020)	**1.0000**	0.5399	0.7012	0.9796	0.7700	**0.0000**	0.4601
	GAMBIT (Gote and Zingg, 2021)	**1.0000**	0.8361	0.9108	0.9927	0.9181	**0.0000**	0.1639
	GitAuthority (full-accuracy mode)	**1.0000**	**0.9223**	**0.9596**	**0.9966**	**0.9611**	**0.0000**	**0.0777**
	GitAuthority (privacy-preserving mode)	1.0000	0.7794	0.8760	0.9902	0.8897	0.0000	0.2206
OpenStack	ALFAA (Amreen et al., 2020)	0.9868	0.9230	0.9538	0.9453	0.9517	0.0123	0.0770
	GAMBIT (Gote and Zingg, 2021)	**0.9923**	**0.9985**	**0.9954**	**0.9943**	**0.9931**	**0.0077**	**0.0015**
	GitAuthority (full-accuracy mode)	**0.9923**	**0.9985**	**0.9954**	**0.9943**	**0.9931**	**0.0077**	**0.0015**
	GitAuthority (privacy-preserving mode)	0.9923	0.9923	0.9923	0.9906	0.9901	0.0077	0.0077
ASF	ALFAA (Amreen et al., 2020)	**1.0000**	0.1911	0.3209	0.9998	0.5956	**0.0000**	0.8089
	GAMBIT (Gote and Zingg, 2021)	0.9965	0.9693	0.9827	**1.0000**	0.9846	0.0034	0.0307
	GitAuthority (full-accuracy mode)	**1.0000**	**0.9727**	**0.9862**	**1.0000**	**0.9863**	**0.0000**	**0.0273**
	GitAuthority (privacy-preserving mode)	1.0000	0.9727	0.9862	1.0000	0.9863	0.0000	0.0273
Gnome GTK	ALFAA (Amreen et al., 2020)	0.9922	0.2369	0.3825	0.9991	0.6185	0.0019	0.7631
	GAMBIT (Gote and Zingg, 2021)	**0.9986**	0.9699	0.9840	**1.0000**	0.9850	**0.0014**	0.0301
	GitAuthority (full-accuracy mode)	0.9954	**0.9940**	**0.9947**	**1.0000**	**0.9970**	0.0046	**0.0060**
	GitAuthority (privacy-preserving mode)	0.9954	0.9935	0.9944	1.0000	0.9968	0.0046	0.0065

Bold: Best value per dataset and metric (not considering GitAuthority in privacy-preserving mode). Higher values are better for precision, recall, F1-score, accuracy, and balanced accuracy; lower values are better for lumping and splitting.

result from significant differences in aliases that cannot be linked without additional external information (e.g., completely different names and email addresses).

These results demonstrate that GitAuthority meets and exceeds state-of-the-art accuracy levels, providing a suitable basis for studying the trade-off between accuracy and privacy in developer-identity de-aliasing. As we implemented GitAuthority in a configurable way to accommodate different privacy needs, we can now explore how its accuracy changes when enabling or disabling certain heuristics to preserve privacy, which we will focus on in the remainder of this article.

5 The Impact of Privacy Preferences on Applications of De-Aliasing

To investigate the trade-off between accuracy and privacy in developer-identity de-aliasing, we conduct an empirical study. This is a difficult endeavor, as it is not publicly observable whether a developer cares about privacy or not, and if so, to what extent. While it might sound plausible to conduct surveys among developers to ask them about their privacy preferences, this is a sensitive topic. Asking developers about their privacy preferences by directly contacting them via their email addresses provided in their GitHub accounts feels like an intrusion into their privacy itself, and it is questionable with respect to ethical research guidelines (Baltes and Diehl, 2016). There are ongoing discussions about how to conduct research in a privacy-respecting and ethical manner (Gold and Krinke, 2020; Feitelson, 2023), and we do not want to contribute to unethical research practices. Following the suggestions of Robillard et al. (2024), we communicate the trade-offs of our study design openly to make our process transparent and to allow others to judge the ethical implications of our study design themselves. Instead of contacting developers directly, we therefore decided to focus on specific key indicators of privacy preferences that are publicly observable via trace data from software repositories without collecting any additional private information about developers.

5.1 Indicators of Privacy Preferences

The key innovation of our study design is to focus on specific privacy-preserving artifacts that developers voluntarily use to hide their actual identity, potentially indicating privacy preferences. GitHub allows its users to use privacy-preserving pseudo email addresses for their commits to hide their actual email address from public view. Each GitHub user can enable this feature in their settings (see Figure 2 for an example screenshot). Notably, this feature is presented as an explicit, dedicated setting labeled “Keep my email addresses private”, which is disabled by default. That is, using a noreply address requires an active, deliberate choice by the developer rather than being nudged through a pre-selected default or being a side effect of enabling some other, unrelated setting. When this feature is enabled, GitHub automatically generates a pseudo email address of the following form:[9]

```
<user-id>+<github-username>@users.noreply.github.com
```

[9] https://docs.github.com/en/account-and-profile/reference/email-addresses-reference#your-noreply-email-address (accessed: 2025-10-20)

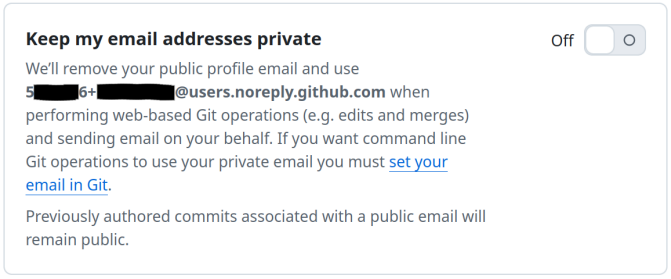


Fig. 2 Screenshot of a GitHub user's settings page showing the option to use a privacy-preserving `users.noreply.github.com` email address for commits. (We have blacked out the actual username and user ID in this screenshot for privacy reasons.)

(or `<github-username>@users.noreply.github.com` in earlier versions). This pseudo email address, which we refer to as *noreply address*, can be used for commits instead of the user's actual email address. As can be seen from the screenshot, GitHub clearly indicates that this email address is used only for web-based Git operations, but not for Git operations of a developer's local Git client, unless the developer explicitly configures their local Git client to use this email address as suggested by GitHub. Also, GitHub clearly indicates that using this privacy-preserving email address does not affect any previously authored commits for which the developer might have used their actual email address publicly. This demonstrates that privacy is not even fully enforced by GitHub itself, but it is up to the developer to decide whether they want to use this privacy-preserving email address for future activities and for their local Git operations or not. The reasons for using such privacy-preserving email addresses are not documented, nor does GitHub provide any suggestions why a user might want to use this feature. Thus, keeping their actual email address private might be a matter of personal preference, and it is not clear whether developers use this feature primarily to protect their privacy or for other practical reasons (e.g., to avoid spam emails). As it is not fully understood whether and how people use such privacy-preserving email addresses on GitHub, we decided to focus on this specific artifact as a proxy for privacy preferences in our study and investigate how prevalent the usage of such email addresses is on GitHub.

To obtain an understanding of how prevalent the usage of noreply addresses is on GitHub, we collect basic descriptive statistics. For this purpose, we use *World of Code (WoC)* (Ma et al., 2019), which is a large-scale archive of Git version-control data that can be freely used for research purposes. In its most recent version at the time of conducting this empirical study, it contained 5.7B commits from 269M public repositories on GitHub and other code-hosting platforms such as GitLab or Bitbucket, spanning the entire history until the end of 2024. This dataset contains all metadata stored in the Git commit objects, including author names and email addresses of the authors of the commits. There are 104M different author strings (i.e., developer aliases) in WoC version V2412, which we use for this study.

In particular, we extract all author email addresses from the WoC dataset and count how many of these email addresses belong to the `users.noreply.github.com` domain, how many projects are affected by such noreply addresses, and how many commits have been made using such noreply addresses. In Table 3, we provide an overview of our basic descriptive statistics. We can see that about 30% of all author

Table 3 Descriptive statistics about the usage of `users.noreply.github.com` email addresses in commits on GitHub projects and beyond, based on the *World of Code (WoC)* dataset

Descriptive Statistic	Value
Distinct author aliases in WoC	103 560 700
Distinct author aliases in GitHub repositories	97 632 968
Distinct author aliases with `users.noreply.github.com` email	31 208 737
Percentage of `users.noreply.github.com` aliases (% of all)	30.14%
Percentage of `users.noreply.github.com` aliases (% of GitHub)	31.97%
Distinct `users.noreply.github.com` emails	27 333 308
GitHub handles using `users.noreply.github.com` emails	27 066 905
Projects in WoC (excluding forks)	189 785 613
GitHub projects (excluding forks)	189 567 577
Projects in WoC with $\geq$1 `users.noreply.github.com` (excluding forks)	84 342 114
GitHub projects with $\geq$1 `users.noreply.github.com` (excluding forks)	84 339 049
Percentage of projects with $\geq$1 `users.noreply.github.com` (excl. forks)	44.44%
Percent. of GitHub proj. with $\geq$1 `users.noreply.github.com` (excl. forks)	44.49%
Commits in WoC	5 708 204 922
Commits to GitHub repositories	5 477 051 067
Commits with `users.noreply.github.com` email	1 204 884 532
Commits to GitHub repositories with `users.noreply.github.com`	1 199 641 382
Percentage of commits with `users.noreply.github.com` (% of all)	21.11%
Percentage of commits with `users.noreply.github.com` (% of GitHub)	21.90%

aliases in WoC and about 32% of all author aliases in GitHub repositories use noreply addresses, indicating that a substantial number of developers care about their privacy and want to hide their actual email address from public view, at least on some occasions. In total, 44% of GitHub projects (not considering forks) have at least one commit with a noreply address, which shows that the usage of such privacy-preserving email addresses is widespread across almost half of the projects. Notably, more than 3 000 projects from other platforms (such as GitLab or Bitbucket) also have at least one commit with a `users.noreply.github.com` email address, which indicates that some developers also use such privacy-preserving email addresses in projects that are not hosted on GitHub. All in all, about 21% of all commits in WoC and about 22% of all commits to GitHub repositories have been made using such noreply email addresses.

In Figure 3, we present how the number of commits made using a noreply address has evolved over time. Noreply addresses have been used since around 2016, and the number of commits made using noreply email addresses has steadily increased over time, alongside a general increase in the overall number of commits. The steady growth in adoption suggests that this privacy-conscious behavior is becoming increasingly common over time.

5.2 Sampling Strategy to Obtain Privacy-Relevant Ground-Truth Data

To study the trade-off between accuracy and privacy in large-scale developer-identity de-aliasing, we need ground-truth data containing developers who use noreply addresses (e.g., `users.noreply.github.com`). The use of these email addresses is a fairly recent phenomenon that has gained traction in the past few years, likely driven by rising privacy concerns. Thus, existing ground-truth datasets, created before this

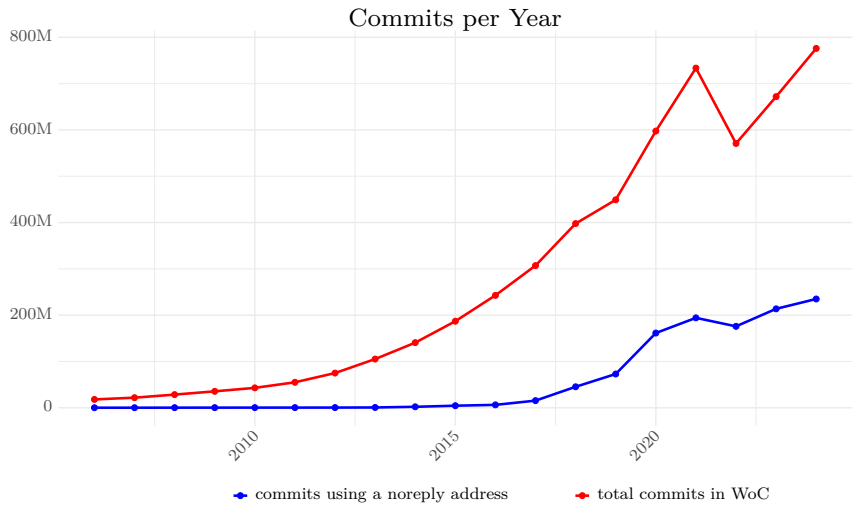


Fig. 3 Overview of the number of commits made with `users.noreply.github.com` email addresses over time (per year), based on the WoC dataset.

practice gained traction, are mostly unsuitable for our work as they lack sufficient privacy-preserving email samples. In particular, looking at the datasets presented in Table 1, the OpenStack dataset contains 5 such samples, the WoC + DRE dataset contains 2 samples, the Gnome GTK dataset contains 7 samples, and the ASF dataset contains 0 samples.

To overcome this limitation of existing datasets, we develop a sampling strategy to create a large-scale ground-truth dataset containing many noreply addresses. Our goal is not to construct a perfect dataset, but rather to build one sufficiently large to enable studying the interplay between accuracy and privacy in developer-identity de-aliasing. In what follows, we first describe our sampling strategy, then we explain how we apply this strategy to create a large-scale dataset, and finally we apply a threshold-based filtering to reduce the risk of false positives in this dataset.

5.2.1 Sampling Strategy

Our strategy leverages the fact that GitHub users can create multiple repositories under their account, each with its own commit history. Since repositories created by the same GitHub user are very likely owned and maintained by the same person, we can use this signal to construct ground-truth identity clusters. We focus specifically on *single-author repositories* (i.e., repositories where all commits in the repository have *the same alias* in the commit author field), as these indicate that only one person has contributed—most likely the repository owner. Of course, exceptions exist: Repository owners may grant write access to another person who ends up being the sole commit author in the repository, or organizational accounts may be managed by multiple developers. Similarly, educators using GitHub Education[10] may create repositories where a student is the sole commit author. Despite

[10] https://github.com/education/teachers/ (accessed: 2025-10-20)

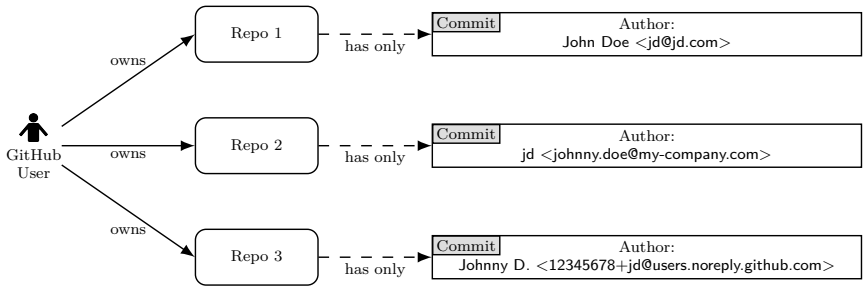


Fig. 4 Illustration of our sampling strategy: A GitHub user with 3 single-author repositories uses different aliases across repositories, enabling ground-truth identity linkage.

these edge cases, single-author repositories of the same GitHub user provide a reasonably reliable signal for ground-truth identity linkage at scale: When a GitHub user has multiple single-author repositories, we assume all their commits originate from the same person—the repository owner (as long as the number of different aliases is reasonably low, which we control for by implementing a threshold that limits the number of aliases). This assumption enables us to link different identities used by the same developer. For instance, the commits in one repository might be made through GitHub's web interface while the commits in another repository might have been created via the command-line Git client with a different name or email address. Similarly, commits made to repositories at different times may reflect outdated versus current email addresses, work versus personal email addresses, or actual versus privacy-preserving addresses. Figure 4 further illustrates this strategy, where a GitHub user with three single-author repositories uses different aliases across repositories—ranging from full names to abbreviated forms and privacy-preserving noreply addresses—all of which we link to the same identity.

5.2.2 Dataset Creation

We use the *World of Code (WoC)* dataset (Ma et al., 2019) in its version V2412 to identify all repositories that contain exactly one distinct author alias across all their commits, which we define as single-author repositories. We assume that all author aliases within such repositories belong to the same developer, that is, the repository owner. For this study, we focus exclusively on public GitHub repositories, excluding repositories from other platforms (e.g., GitLab, Bitbucket) that are also present in WoC. We additionally remove 6% of the aliases from these repositories that were ill-encoded and could not be reliably processed due to unknown character encoding.

We group the identified single-author repositories by their GitHub owner and collect all aliases of their commit authors (name and email address combinations) used across them. To minimize false positives, we remove aliases that appear with multiple GitHub users. Such cases suggest either shared organizational accounts or developers with multiple GitHub accounts using the same email configuration—neither representing the typical single-developer scenario we seek to study. By excluding all author aliases that are shared among multiple GitHub users, we obtain a cleaner dataset where identity clusters more reliably represent individual developers.

In Table 4, we present descriptive statistics about our created dataset. We initially identified 11 180 447 GitHub users with at least one single-author repository,

Table 4 Descriptive statistics about our GitHub dataset of single-author repositories

Descriptive Statistic	Value
Entire set of GitHub users with at least 1 single-author repository	11 180 447
Non-user accounts contained in the above set (organizations, bots, unknown)	1 050 582
Remaining GitHub users after removing non-user accounts	10 129 865
Remaining GitHub users after removing aliases that occur with multiple users	9 568 385
Total number of distinct author aliases in the remaining dataset	21 477 370
Distinct aliases in the remaining dataset that contain a noreply address	6 023 589
Distinct noreply addresses occurring in the remaining dataset	5 145 195
Author aliases per remaining GitHub user	*median:* 2, *p90:* 3, *p99:* 6, *max:* 6 606
Author aliases per organization user	*median:* 3, *p90:* 9, *p99:* 49, *max:* 12 635

yielding 41 985 057 total author aliases. Since some GitHub accounts represent organizations or bots rather than individual developers, we queried each account's type via the GitHub GraphQL API.[11] We removed 1 050 582 accounts that were either organizations or had indeterminate types, retaining 10 129 865 individual developer accounts with 36 257 066 total author aliases. In addition, we removed 14 779 696 aliases that were shared among multiple GitHub users, suggesting that having multiple GitHub accounts or sharing a single account among multiple users (thus, using personal accounts the way organization accounts are used) may be fairly common. We discuss these and similar potential limitations of our dataset in Section 6. After these removals, our final dataset has 9 568 385 GitHub users with 21 477 370 distinct author aliases. Among these 9 568 385 GitHub accounts, 51.3% have an alias containing a noreply address in one of their single-author repositories, which is a surprisingly high percentage. Moreover, among those with at least one alias containing a noreply address in their single-author repositories, 51.8% also have at least one alias with another email address, indicating that many developers use both privacy-preserving and actual email addresses across their repositories. However, some of these GitHub accounts still exhibit organizational usage patterns—such as shared accounts for multiple developers or instructor accounts creating repositories for students—which cannot be automatically detected. Manual inspection revealed that users with more than 10 single-author repositories often fall into these categories. In Figure 5, we present the distribution of the number of aliases per user in our dataset, which shows that most users have only a few aliases, while a small number of users have many aliases. In particular, the distribution among single-author repositories shows a median of 2 aliases per user (90% quantile: 3 aliases; 99% quantile: 6 aliases) with a maximum of 6 606 aliases. Examination of extreme cases confirmed that users with thousands of single-author repositories with different aliases typically represent organizational accounts or contain thousands of automatically generated repositories with a different numeric alias used in each of them, inconsistent with individual developer behavior.

The single-author repositories may not be representative of typical open-source software developers at all, so the number of aliases detected should not be extrapolated to the overall developer population, but they represent a way to automatically collect a large alias-resolution dataset.

[11] https://docs.github.com/en/graphql/ (accessed: 2025-10-20)

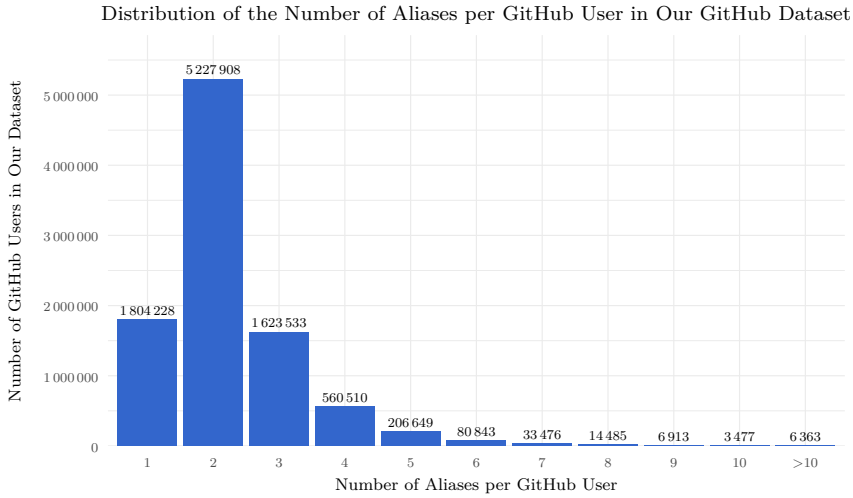


Fig. 5 Distribution of the number of aliases per GitHub user in our GitHub dataset of single-author repositories (9 568 385 GitHub users in total).

5.2.3 Threshold-Based Filtering

To focus on individual developers rather than users that behave rather like an organizational account, we apply threshold-based filtering on our dataset. In our main analysis, we use a threshold of 6 for the maximum number of author aliases per user, which corresponds to the 99% quantile of aliases per user in our dataset. In addition, we conduct a sensitivity analysis using thresholds of 2, 3, and 10 for the maximum number of author aliases per user (used in the single-author repositories). The threshold of 2 represents the median of aliases, the threshold of 3 the 90% quantile, and the threshold of 10 the 99.9% quantile, retaining the majority of users while excluding extreme outliers. These thresholds align with prior empirical evidence showing developers typically maintain 2–3 aliases on average, with maximums going up to 10–15 in most studies (Bird et al., 2006; Vasilescu et al., 2015; Wiese et al., 2016; Fry et al., 2020). While this filtering may exclude some legitimate developers who genuinely use more than 10 aliases, such cases are rare according to the literature, which is also supported by our own analysis of the distribution of aliases per user in our dataset (see Figure 5). We accept this trade-off to ensure dataset quality and confidence in our ground truth.

5.3 Evaluation of De-Aliasing with Respect to Privacy Preferences

Using our newly created dataset, we can now evaluate the performance of different de-aliasing approaches with respect to privacy preferences. To do so, we run GitAuthority in two different configurations: (1) a full-accuracy configuration that uses all matching rules, including those relying on privacy-sensitive information, and (2) a privacy-preserving configuration that disables matching rules relying on privacy-sensitive information. In particular, for our privacy-preserving configuration,

Table 5 Evaluation results of de-aliasing approaches on the GitHub dataset

Approach	Fully Merged	Excluding Single	Recall (with 95% CI)
Threshold: max. 6 aliases in single-author repositories per GitHub user (99.3% of dataset)			
ALFAA (Amreen et al., 2020)	41.3%	27.6%	0.451 [0.4510, 0.4516]
GAMBIT (Gote and Zingg, 2021)	71.2%	64.5%	0.747 [0.7466, 0.7471]
GitAuthority (full-accuracy mode)	73.8%	67.7%	0.771 [0.7709, 0.7714]
GitAuthority (privacy-preserving mode)	42.5%	29.0%	0.472 [0.4715, 0.4721]

Fully Merged: Percentage of GitHub users whose aliases are fully merged by the respective approach
Excluding Single: Same as Fully Merged, excluding GitHub users with only 1 alias
95% CI: Estimated 95% confidence intervals using bootstrap resampling with 10 000 iterations

we disable the GITHUB_USERNAME_MATCH rule, which relies on the GitHub username in the noreply address as a signal for matching aliases (see Section 4.1.3), and we also blocklist the email domain "users.noreply.github.com" (see Section 4.1.2). We deliberately scope our notion of privacy protection to this specific mechanism: a privacy-preserving configuration aims to prevent an alias for which a developer has used GitHub's explicit privacy signal (a noreply address) from being merged with any of that developer's other aliases, so that the activity performed under the privacy-protected alias cannot be attributed to the same person as their other, potentially identifying activity. This configuration does not attempt to protect aliases that carry no such signal, nor does it address privacy risks that are unrelated to alias merging, such as inferring a developer's identity from coding style or activity patterns (Section 3.2.5); we discuss this scope and its justification in more detail in Section 6. Table 2 shows that the privacy-preserving configuration of GitAuthority performs similarly well as the full-accuracy configuration on the four existing ground-truth datasets, which contain only very few samples of privacy-preserving email addresses (as we have discussed in Section 5.2). In particular, on the ASF dataset, which contains no noreply addresses, both configurations of GitAuthority achieve exactly the same performance, as expected.

To evaluate the effect of privacy-preserving de-aliasing, we use our newly created GitHub dataset (see Section 5.2), which contains a substantial number of more than 5M distinct privacy-preserving email addresses (see Table 4). For this evaluation, we applied a threshold of at most 6 aliases per user to filter users with many aliases used across their single-author repositories (see Section 5.2.3) and conduct a sensitivity analysis of the thresholds later in Section 5.3.1. We run both configurations of GitAuthority (the privacy-agnostic one and the privacy-preserving one) on our dataset and report the corresponding results in Table 5.[12] When disabling the matching rules that rely on information extracted from GitHub's noreply addresses, the percentage of GitHub users whose aliases are fully merged drops from 73.8% to 42.5% in our GitHub dataset. In general, the recall of GitAuthority drops from 0.77 to 0.47 (with each having very narrow 95% confidence intervals using bootstrap

[12] For the sake of completeness, in Table 5, we also report the performance of the existing baseline approaches ALFAA (Amreen et al., 2020) and GAMBIT (Gote and Zingg, 2021), although these approaches cannot be used to answer our research question as they do not have a privacy-preserving configuration. Note that it is not our primary goal to compare the performance of GitAuthority with the performance of existing approaches here, as we already did this comparison in Section 4.2 on existing datasets.

resampling with 10 000 iterations, see Table 5) when switching from the full-accuracy configuration to the privacy-preserving configuration.

Note that we only report the recall here, as the way we sampled our dataset does not allow us to reliably determine false positives (which would be necessary to report reliable values for precision or F1-score). In our dataset, all aliases that we link to the same user are considered as positives, and thus all aliases that are merged by the de-aliasing approach are considered as true positives, while the aliases that are not merged by the de-aliasing approach although they should have been merged according to our dataset are considered as false negatives. If the de-aliasing approach merges two aliases that should not have been merged according to our dataset, we cannot reliably determine whether this is a false positive or not, as a person could use different GitHub accounts with different aliases. Even so, when assuming that no such cases would exist in our dataset, the precision would drop from about 0.83 to about 0.56 when switching from the full-accuracy configuration to the privacy-preserving configuration. Similarly, the F1-score would drop from about 0.79 to about 0.49. To obtain a better understanding of how reliable the differences in precision and F1-score are in our dataset, we provide a manually validated precision and accuracy estimate from a sample of 360 users in Section 5.3.4. Furthermore, we discuss likely causes for the substantially lower recall of ALFAA (and, to a lesser extent, GAMBIT) compared to GitAuthority on our GitHub dataset, and to what extent these causes can be attributed to our dataset construction as opposed to the approaches themselves, in Section 6.

5.3.1 Sensitivity Analysis of the Filtering Thresholds

As the results presented above might be sensitive to the specific choice of threshold for the maximum number of aliases per user, we conduct a sensitivity analysis by evaluating the performance of the de-aliasing approaches on our GitHub dataset using different thresholds. In Table 6, we report the evaluation results of the de-aliasing approaches for four different thresholds for the maximum number of aliases per GitHub user (2, 3, 6, and 10), confirming that our findings are not sensitive to the specific choice of threshold. As described in Section 5.2.3, the threshold of 2 represents the median number of aliases per user in our dataset, the threshold of 3 the 90% quantile, the threshold of 6 the 99% quantile, and the threshold of 10 the 99.9% quantile. Across all four thresholds, the relative ordering of the approaches remains stable, and the performance drop (in terms of fully merged user aliases and recall) between the full-accuracy and privacy-preserving configurations of GitAuthority remains consistently large. This suggests that our findings are robust and not overly sensitive to the specific choice of threshold for the maximum number of aliases per user. Notably, all approaches show a slight increase in performance when the threshold is lowered, which is expected as the lower thresholds exclude users with many aliases, which are more likely to be organizational accounts or otherwise atypical users whose aliases should not be merged although they are linked to the same GitHub user. We take a closer look at the number of aliases per user in our dataset when we manually investigate the validity of our assumption that all aliases of a GitHub user in single-author repositories belong to the same person in Section 5.3.4.

Table 6 Threshold-sensitivity analysis of the evaluation results of de-aliasing approaches on the GitHub dataset

Approach	Fully Merged	Excluding Single	Recall (with 95% CI)
Threshold: max. 10 aliases in single-author repositories per GitHub user (99.9% of dataset)			
ALFAA (Amreen et al., 2020)	41.1%	27.4%	0.450 [0.4495, 0.4501]
GAMBIT (Gote and Zingg, 2021)	71.0%	64.3%	0.746 [0.7461, 0.7466]
GITAUTHORITY (full-accuracy mode)	73.6%	67.4%	0.770 [0.7702, 0.7707]
GITAUTHORITY (privacy-preserving mode)	42.4%	29.0%	0.472 [0.4717, 0.4723]
Threshold: max. 6 aliases in single-author repositories per GitHub user (99.3% of dataset)			
ALFAA (Amreen et al., 2020)	41.3%	27.6%	0.451 [0.4510, 0.4516]
GAMBIT (Gote and Zingg, 2021)	71.2%	64.5%	0.747 [0.7466, 0.7471]
GITAUTHORITY (full-accuracy mode)	73.8%	67.7%	0.771 [0.7709, 0.7714]
GITAUTHORITY (privacy-preserving mode)	42.5%	29.0%	0.472 [0.4715, 0.4721]
Threshold: max. 3 aliases in single-author repositories per GitHub user (90.5% of dataset)			
ALFAA (Amreen et al., 2020)	44.4%	29.7%	0.467 [0.4665, 0.4671]
GAMBIT (Gote and Zingg, 2021)	73.0%	65.9%	0.750 [0.7497, 0.7502]
GITAUTHORITY (full-accuracy mode)	75.8%	69.4%	0.776 [0.7760, 0.7765]
GITAUTHORITY (privacy-preserving mode)	44.5%	29.9%	0.472 [0.4717, 0.4723]
Threshold: max. 2 aliases in single-author repositories per GitHub user (73.5% of dataset)			
ALFAA (Amreen et al., 2020)	50.0%	32.7%	0.500 [0.4993, 0.5001]
GAMBIT (Gote and Zingg, 2021)	75.9%	67.6%	0.759 [0.7591, 0.7597]
GITAUTHORITY (full-accuracy mode)	78.7%	71.4%	0.787 [0.7869, 0.7875]
GITAUTHORITY (privacy-preserving mode)	48.9%	31.3%	0.489 [0.4887, 0.4894]

Fully Merged: Percentage of GitHub users whose aliases are fully merged by the respective approach
Excluding Single: Same as Fully Merged, excluding GitHub users with only 1 alias
95% CI: Estimated 95% confidence intervals using bootstrap resampling with 10 000 iterations

5.3.2 Ablation Study of the Impact of the Blocklist

As GITAUTHORITY makes use of a manually curated blocklist of frequent placeholder identities and other aliases that do not correspond to a real person (see Section 4.1.2), we conduct an ablation study to evaluate the impact of this blocklist on the performance of GITAUTHORITY. To do so, we run both the full-accuracy configuration and the privacy-preserving configuration of GITAUTHORITY on our GitHub dataset with and without enabling the blocklist. In particular, to disable the blocklist, we simply remove all blocklist entries in the configuration file of GITAUTHORITY (except for the list of common first names, which is not manually curated). While this does not allow us to assess the impact of a specific blocklist entry, it allows us to assess the overall impact of the blocklist on the recall of GITAUTHORITY. We report the corresponding results in Table 7. The results indicate that removing the blocklist has only a marginal effect on the performance of GITAUTHORITY in either configuration. Although the recall is slightly higher when the blocklist is disabled, which might create the intuition that the blocklist is detrimental to the performance of GITAUTHORITY, we note that the blocklist is designed to prevent false positives rather than to increase recall. We show in Section 5.3.4 on a manually validated sample that the blocklist is indeed effective in preventing false positives, which is crucial for the performance of GITAUTHORITY.

Table 7 Analysis of the impact of blocklists on the evaluation results of de-aliasing approaches on the GitHub dataset

Configuration of GitAuthority	Fully Merged	Excluding Single	Recall (with 95% CI)
Threshold: max. 6 aliases in single-author repositories per GitHub user (99.3% of dataset)			
full-accuracy mode	73.8%	67.7%	0.771 [0.7709, 0.7714]
full-accuracy mode without blocklist	74.0%	67.9%	0.773 [0.7729, 0.7734]
privacy-preserving mode	42.5%	29.0%	0.472 [0.4715, 0.4721]
privacy-preserving mode without blocklist	42.6%	29.2%	0.473 [0.4728, 0.4734]

Fully Merged: Percentage of GitHub users whose aliases are fully merged by the respective approach
Excluding Single: Same as Fully Merged, excluding GitHub users with only 1 alias
95% CI: Estimated 95% confidence intervals using bootstrap resampling with 10 000 iterations

5.3.3 Comparative Study on the Chosen Privacy Signal

The privacy-preserving configuration evaluated so far operationalizes a developer's privacy preference in one specific way, namely by treating the usage of a noreply address as a privacy signal and thus withholding aliases that contain a noreply address from being merged with other aliases. However, this is only one of several plausible ways for a developer to indicate that they likely care more about privacy than others. To assess whether our findings depend on this specific choice of privacy signal, we therefore conduct a comparative study using an alternative, less obvious privacy signal. Such a comparative study helps determine whether the observed performance drop under privacy-preserving de-aliasing is an inherent consequence of treating noreply addresses as a privacy signal or whether an alternative choice of privacy signal can lead to a similar performance drop. This way, we gain additional insights into the accuracy–privacy trade-off.

For our comparative study, we consider aliases whose name part (i.e., ignoring the email address) consists of only a common first name (e.g., "John") without having any last name or other information as an alternative privacy signal. People might choose to use only their first name as it is less personally identifiable than using both first and last names. While there might also be other possible reasons for using only a first name (such as laziness or a preference for brevity), we consider it a reasonable assumption that using only a first name is a weaker privacy signal than using a noreply address, but still could indicate some level of privacy concern. Consequently, we compare the performance of GitAuthority when using the first name as a privacy signal (i.e., withholding aliases from merging when their name consists of only a first name) with the performance of GitAuthority when using the noreply address as a privacy signal (which we refer to as privacy-preserving mode) and with the performance of the full-accuracy mode of GitAuthority. In Table 8, we report the corresponding results of this comparative study. Using the first name as a privacy signal yields intermediate performance (recall 0.71), falling between the full-accuracy (recall 0.77) and the privacy-preserving configuration (recall 0.47) of GitAuthority, which is consistent with our expectation that using only a first name is a weaker privacy signal than using a noreply address and less commonly used by developers than a noreply address.

Table 8 Evaluation results of de-aliasing approaches on the GitHub dataset using an alternative privacy signal (first name) instead of a noreply address

Configuration of GitAuthority	Fully Merged	Excluding Single	Recall (with 95% CI)
Threshold: max. 6 aliases in single-author repositories per GitHub user (99.3% of dataset)			
full-accuracy mode	73.8%	67.7%	0.771 [0.7709, 0.7714]
first-name privacy signal	66.9%	59.1%	0.707 [0.7070, 0.7075]
privacy-preserving mode	42.5%	29.0%	0.472 [0.4715, 0.4721]

Fully Merged: Percentage of GitHub users whose aliases are fully merged by the respective approach
Excluding Single: Same as Fully Merged, excluding GitHub users with only 1 alias
95% CI: Estimated 95% confidence intervals using bootstrap resampling with 10 000 iterations

5.3.4 Manual Validation of a Sample of the Dataset

As our ground truth is derived automatically from single-author repositories (see Section 5.2), we additionally manually validate a sample of our dataset to assess the reliability of the automatically derived ground truth and to obtain a precision estimate of the evaluation results reported above. Figure 5 shows that the distribution of the number of aliases per GitHub user in our dataset is highly skewed, with the vast majority of users having only 2 or 3 aliases and comparatively few users having more than 6 aliases. A simple random sample drawn from this population would therefore be dominated by users with few aliases and would contain very few (if any) users with a larger number of aliases, making it impossible to reliably assess the performance of de-aliasing approaches for such users. We therefore draw a *stratified sample* (Cochran, 1977), using the number of aliases per GitHub user as the stratification variable and applying equal allocation across strata: for each stratum from 2 to 10 aliases per user, we randomly sample 40 GitHub users in our dataset, yielding a total of 360 sampled GitHub users. We exclude the stratum of users with only a single alias, as such users cannot form alias pairs and are therefore not informative for evaluating the de-aliasing approaches. For a 95% confidence level and a 5% margin of error, common guidelines for determining an adequate sample size when estimating a population proportion (Cochran, 1977) recommend a sample size of approximately 385. Our sample of 360 GitHub users is close to this commonly recommended threshold, and, since it comprises 2 160 aliases in total, it exceeds this threshold at the level of individual aliases, providing a reasonable basis for estimating the performance of the de-aliasing approaches. Equal allocation across strata ensures that each stratum (including the strata of users with comparatively many aliases, who are underrepresented in the overall population but particularly relevant for assessing whether de-aliasing approaches can correctly merge a larger number of aliases) is represented with the same number of observations, thereby avoiding that rare but important cases are completely missed, as would happen under simple random sampling given the observed skew in the alias-count distribution. At the same time, stratifying by the number of aliases per user reduces the variance of our estimates relative to simple random sampling, since aliases within the same stratum are more homogeneous with respect to the number of aliases to be merged than aliases from different strata (Cochran, 1977).

Manually inspecting the 2 160 aliases of the 360 sampled GitHub users revealed that for 60 out of the 360 sampled users (16.7%), our assumption that all aliases of a GitHub user in single-author repositories belong to the same person does not hold.

Table 9 Manual validation of the assumption that all aliases of a GitHub user in single-author repositories belong to the same person on our stratified sample of 360 GitHub users

Aliases per user (stratum)	2	3	4	5	6	7	8	9	10
Users in stratum	40	40	40	40	40	40	40	40	40
Users with assumption violation	6	2	5	4	3	5	6	10	19

In most of these cases, either a bot was the single commit author in one of the user's repositories, or another human developer contributed to one of the user's repositories as a single commit author. As we show in Table 9, the number of users for whom our assumption does not hold is exceptionally high for users with more than 8 aliases, which is consistent with the intuition that users with many aliases are more likely to be organizational accounts or otherwise atypical users whose aliases should not be merged although they are linked to the same GitHub user. A manual investigation of these cases revealed that the vast majority of these single commit authors were students who were assigned a repository for an individual course project set up by an educator who had created repositories for each student under the educator's GitHub account. While our assumption is violated in 16.7% of the sampled users, we deem this to be a reasonable trade-off that does not substantially affect the validity of our evaluation results, especially as violations are more likely to occur for users with many aliases and our sensitivity analysis in Section 5.3.1 shows that our findings are robust to the choice of threshold for the maximum number of aliases per user.

After identifying these assumption violations, we manually corrected the corresponding annotation errors in our sample of the 2 160 aliases that belong to the 360 sampled GitHub users. That is, we marked aliases that obviously belong to different identities (e.g., completely different names) as not belonging to the same identity although they are linked to the same GitHub user in our dataset. Note that we only corrected obvious cases, but did not correct cases where it was unclear whether two aliases belong to the same identity or not (e.g., when name and email of the alias are general terms that are not uniquely related to a particular identity, such as "admin"). In total, the sample comprises 2 331 720 alias pairs. Under the automatically derived ground truth, 6 600 of these are positive pairs (i.e., both aliases belong to the same identity) and 2 325 120 are negative pairs (i.e., both aliases belong to different identities). The manual correction described above reclassifies 1 064 of these positive pairs as negative, so the corrected version of the sample contains 5 536 positive pairs and 2 326 184 negative pairs. We applied the different de-aliasing approaches to this manually validated sample, both to the original version without manual corrections and to the manually corrected version, and compared the results for each approach.

In Table 10, we report the numbers of false positives (FPs), false negatives (FNs), and the resulting F1-scores of the different de-aliasing approaches on our manually validated sample of the dataset. Comparing the original and corrected versions shows that the annotation errors we identified predominantly concern recall rather than precision: the number of FNs decreases substantially for every approach after correction (e.g., from 5 451 to 4 389 for ALFAA, and from 2 889 to 1 824 for GitAuthority in full-accuracy mode), whereas the number of FPs remains almost unchanged, differing by at most two alias pairs for every approach. This indicates that our evaluation using the automatically derived ground truth tends to underestimate the performance of the approaches, and correcting it consequently yields a moderately higher

Table 10 Evaluation results of de-aliasing approaches on the GitHub dataset based on a manually validated sample of the dataset

Approach	Original Sample			Corrected Sample		
	FPs	FNs	F1	FPs	FNs	F1
ALFAA (Amreen et al., 2020)	0	5 451	0.30	2	4 389	0.34
GAMBIT (Gote and Zingg, 2021)	3 500	2 642	0.56	3 500	1 578	0.61
GitAuthority (full-accuracy mode)	29	2 889	0.72	28	1 824	0.80
GitAuthority (first-name privacy signal)	29	3 468	0.64	28	2 403	0.72
GitAuthority (privacy-preserving mode)	6	3 597	0.63	5	2 532	0.70
GitAuthority (full-accuracy, without blocklist)	584	3 036	0.66	583	1 971	0.74
GitAuthority (privacy-pres., without blocklist)	487	3 723	0.58	486	2 658	0.65

Total sample: 2 160 aliases, 2 331 720 alias pairs. Positive pairs: 6 600 (original), 5 536 (corrected). Negative pairs: 2 325 120 (original), 2 326 184 (corrected).

F1-score for every approach (e.g., from 0.56 to 0.61 for GAMBIT, and from 0.72 to 0.80 for GitAuthority in full-accuracy mode). We therefore focus on the corrected sample, which we consider the more reliable basis for comparing the approaches.

The results confirm the trends observed on the full dataset, with GitAuthority in full-accuracy mode achieving the highest F1-score (0.80), followed by the first-name privacy signal (0.72) and the privacy-preserving mode (0.70). The 28 false positives of GitAuthority in full-accuracy mode are caused by only two pairs of GitHub users: one pair shares the same first name (which is not among the more than 72 980 common first names in our blocklist) and gets merged because both users use a first-name-only alias, and the other pair shares exactly the same first and last name and the same GitHub username when stripping the numbers from the username. That is, the 28 false positives caused by these two pairs of GitHub users are unavoidable given the information available in our dataset, which is negligible compared to the total number of 2 331 720 alias pairs in our manually validated sample. Among the previously existing approaches, ALFAA achieves the highest precision of all approaches (1.00, with only 2 FPs), matched only by GitAuthority in privacy-preserving mode, but by far the lowest recall (0.21, with 4 389 FNs), resulting in the lowest F1-score overall (0.34); GAMBIT shows the opposite profile, combining a comparatively high recall (0.72) with the lowest precision of all approaches (0.53, with 3 500 FPs), yielding an F1-score of 0.61 that trails all five GitAuthority configurations.

Comparing GitAuthority with and without the blocklist further confirms the role of the blocklist in preventing false positives (see also Section 5.3.2): disabling the blocklist increases the number of false positives by more than an order of magnitude, both in full-accuracy mode (from 28 to 583) and in privacy-preserving mode (from 5 to 486), lowering precision from 0.99 to 0.86 and from 1.00 to 0.86, respectively. For example, the sample contains multiple users that use "Your Name" as one of their aliases, which are merged to a single identity when the blocklist is disabled, but correctly kept separate otherwise. Unlike on the full, automatically derived dataset, where disabling the blocklist slightly increases recall (see Section 5.3.2), on this manually validated sample disabling the blocklist instead slightly *decreases* recall in both configurations (from 0.67 to 0.64 in full-accuracy mode, and from 0.54 to 0.52 in privacy-preserving mode). Consequently, on this manually validated sample, enabling the blocklist improves precision, recall, and F1-score simultaneously (F1 increases from 0.74 to 0.80 in full-accuracy mode, and from 0.65 to 0.70 in privacy-

preserving mode), providing further evidence that the blocklist is beneficial for the overall performance of GitAuthority, beyond merely preventing false positives.

All in all, the results on this manually validated sample of the dataset confirm the trends observed on the full dataset, while providing a more reliable basis and a precision estimate for comparing the different approaches and configurations.

Answer to our RQ: When deactivating specific matching rules in GitAuthority to enhance privacy, we observe a trade-off between accuracy and privacy. Privacy-aware de-aliasing incurs a substantial accuracy penalty compared to privacy-agnostic approaches. While GitAuthority in full-accuracy mode achieves the highest performance with a recall of 0.77 and fully merges the aliases of around 74% of GitHub users, switching to privacy-preserving mode reduces the recall to 0.47 and fully merges the aliases of only around 43% of GitHub users. This represents a reduction in recall by approximately 39%, demonstrating that privacy-sensitive signals are critical for effective de-aliasing and that respecting privacy preferences comes at a substantial cost to performance on the available evaluation metrics.

6 Threats to Validity

Our empirical study has several limitations that may affect the validity of our findings. We discuss these limitations in what follows and group them into four established categories (Wohlin et al., 2012).

6.1 Internal Validity

Repository data and ground-truth datasets may contain errors or inconsistencies in names, emails, or identity mappings (Goeminne and Mens, 2013), potentially affecting accuracy. Nevertheless, GitAuthority's stable performance across all four ground-truth datasets from the literature as well as on our own ground-truth dataset indicates that our approach generalizes well across different contexts, implements previous heuristics effectively (even with a slight improvement over previous approaches), and is robust to dataset-specific variations.

Our sampling strategy for a privacy-relevant ground-truth dataset relies on the assumption that commits in single-author repositories on GitHub are authored by the repository owner. While this appears to be a reasonable heuristic, it may not hold for (1) organization accounts where different developers create repositories, (2) GitHub Classroom or educational contexts where instructors create repositories for students, (3) repositories where the owner granted write access to another developer who authored the commits in the repository, and (4) shared or bot accounts. To account for that, we excluded aliases that occurred with multiple GitHub accounts from our dataset during dataset creation. In addition, we applied a threshold that limited the maximum number of aliases and conducted a sensitivity analysis, which shows that our results are robust to this threshold (Section 5.3.1). Our manual investigation of a sample of our dataset (Section 5.3.4) indicates that, while our assumption that all aliases of a GitHub user in single-author repositories belong to the same person

is still violated in some cases, tool evaluation results are largely consistent when manually correcting these violations.

Regarding the substantially lower recall of ALFAA (and, to a lesser extent, GAMBIT) compared to GITAUTHORITY on our GitHub dataset (e.g., see Table 5), several observations suggest that this gap is unlikely to be primarily an artifact of our dataset construction. First, 6 023 589 of the aliases in our dataset (28.1%, affecting 51.3% of GitHub users) are noreply addresses, yet neither ALFAA nor GAMBIT implements any mechanism to link a GitHub noreply alias to a developer's other aliases via the username embedded in it, unlike GITAUTHORITY's dedicated GITHUB_USERNAME_MATCH rule (Section 4.1.3); both previously existing approaches instead rely exclusively on name and email similarity (Sections 3.2.1 and 3.2.3), so a large share of these aliases is essentially unmatchable for them by design, independent of any property of our sampling strategy. Second, ALFAA explicitly weights matches by inverse name frequency to avoid merging developers with common names in large cross-project collections (Amreen et al., 2020); while effective at keeping precision high (as we also confirm on our manually validated sample, Section 5.3.4), this design choice becomes increasingly costly for recall as the scale and heterogeneity of the underlying population grows—exactly the setting of our GitHub dataset, which spans millions of developers across the entire GitHub ecosystem, unlike the substantially smaller, single-project or single-ecosystem datasets ALFAA and GAMBIT were originally developed and evaluated on (OpenStack and Gnome GTK, respectively; see Table 1). Third, our manual validation (Section 5.3.4) indicates that ground-truth noise is not the dominant driver of the accuracy differences either: correcting the identified annotation errors raises the F1-score of every approach, including the baselines, by a similar margin without closing the relative gap between them (ALFAA's F1-score on the corrected sample remains less than half of GITAUTHORITY's), while leaving the number of false positives almost unchanged. Our sensitivity analysis (Section 5.3.1) further shows that the ordering and magnitude of this gap are stable across four different alias-count thresholds, indicating that it is not an artifact of a particular filtering choice either.

GITAUTHORITY's performance depends on the configuration of matching rules and thresholds. Different combinations of configured matching rules and normalization techniques can yield substantially different results. Our reported results reflect five specific configurations optimized for accuracy or privacy, complemented with an ablation study and a comparative study on an alternative privacy signal with largely consistent results, but other configurations may produce different trade-offs.

6.2 External Validity

Our empirical study on privacy aspects in de-aliasing focuses exclusively on GitHub and its privacy-preserving `users.noreply.github.com` addresses. Therefore, our findings may not generalize to other platforms (e.g., GitLab, Bitbucket, or self-hosted Git instances), which may have different privacy features, user populations, and usage patterns. However, GitHub is by far the largest and most popular code-hosting platform—according to WoC, more than 99% of open-source repositories are hosted on GitHub (see Table 3)—and thus our findings are likely to apply to the vast majority of repositories and developers in the open-source ecosystem. Notably, we also found that GitHub's noreply addresses are used in more than 3 000 repositories

that are not hosted on GitHub, which suggests that our findings may also be relevant for other platforms.

GitHub's privacy-preserving email feature is relatively recent, with steadily increasing usage over time. Consequently, historical datasets (such as the four datasets we used as ground truth for tool evaluation) contain few or no privacy-preserving email addresses, limiting our ability to evaluate privacy-aware de-aliasing on older datasets. Nonetheless, we investigate the trade-off between accuracy and privacy on a new large-scale ground-truth dataset that contains more than 21M aliases (containing more than 5M distinct noreply addresses) until the end of 2024. Given the large scale and recency of our dataset, our findings are most likely also applicable to contemporary and future repository data.

Our study is limited to publicly observable trace data in WoC, thus private repositories and proprietary enterprise settings are not represented there. Also, our focus on single-author repositories represents only a specific subset of repository activity. Still, given the large scale and diversity of repositories in WoC (see Table 3), our findings are likely relevant for a wide range of repositories and contexts. Also, it is noteworthy that our ground-truth dataset is built from a large set of different repositories, whereas most previous ground-truth datasets were built from a single repository or a single software ecosystem, which may enhance the generalizability of our findings compared to previous work.

6.3 Construct Validity

We use GitHub's noreply addresses as a proxy for privacy preferences, which is imperfect because (1) developers may be unaware of this feature but still care about privacy, (2) they may use other privacy-preserving mechanisms like pseudonyms, generic names (e.g., "Your Name" or "John Doe"), or non-existent email addresses, and (3) the usage of noreply addresses may be influenced by factors other than personal privacy concerns (e.g., organizational policies, convenience, etc.), such as wanting to avoid spam without necessarily objecting to identity linkage itself. Without directly asking developers about their motivation for using a noreply address—which would itself constitute a privacy intrusion (Section 3.1.2)—we cannot determine with certainty which of these motivations applies in a given case, or what fraction of noreply-address users are primarily concerned with identity linkage specifically. However, as discussed in Section 5.1, GitHub's noreply-address feature is disabled by default and requires a developer to actively navigate to their account settings and enable a dedicated, clearly labeled "Keep my email addresses private" option (Figure 2). We therefore expect the share of developers who care about privacy to be higher among those who made this deliberate choice than among GitHub users in general, even though we cannot determine the exact proportion or precisely which conception of privacy applies to each of them. Given the lack of any more direct, non-intrusive measure of privacy preferences and the widespread use of GitHub's noreply addresses, we consider this proxy to be a reasonable and deliberately conservative starting point for investigating privacy-related aspects in de-aliasing. Importantly, GitAuthority is not conceptually limited to noreply addresses: it is designed to be configurable with respect to which matching rules are treated as privacy-sensitive, so it can be readily extended to honor other, currently unavailable privacy signals

(e.g., a dedicated privacy-preference indicator that a platform might expose in the future) without changes to the overall approach.

The privacy protection we operationalize in this article targets one specific, narrowly defined risk: that de-aliasing could reveal that an alias for which a developer has used an explicit privacy signal belongs to the same person as their other, potentially identifying aliases. We chose this scope because de-aliasing is, by construction, precisely the step that creates this particular risk—only linking (i.e., merging) aliases can turn an otherwise privacy-protected alias into a non-private one by tying it to identifying information already present in a developer's other aliases. We consider this scope appropriate for our study because it isolates the privacy risk that a de-aliasing approach can directly control, and is therefore where a configurable de-aliasing approach like GitAuthority can make a concrete difference. Privacy risks that are unrelated to alias merging—for example, inferring a developer's identity from coding style, commit timing, or other behavioral patterns (Section 3.2.5), or risks that already exist in the raw, publicly available repository data irrespective of any de-aliasing—are outside the scope of what a de-aliasing approach could reasonably be expected to address. Likewise, linking two aliases that both lack any explicit privacy signal (e.g., two email addresses that both contain a developer's real name) is not treated as a privacy violation in our study, since neither alias was intentionally kept separate by the developer as far as we can observe, although these developers still might care about their privacy or even might not want their aliases merged.

Our approach treats privacy as a binary concept (i.e., using noreply addresses vs. not using them), when in reality privacy preferences exist on a spectrum. Other traces of privacy preferences, such as intentional names or email addresses (e.g., "John Doe <hidemyname@me.com>") that are difficult to identify also exist and shall be subjects of future research. Although we demonstrate the effects of ignoring or respecting privacy in a limited way, our study sheds light on the substantial trade-offs between accuracy and privacy in de-aliasing, and highlights the need for more nuanced approaches that can better capture the complexity of privacy preferences in repository data. Considering the merging of aliases with privacy-preserving emails as a violation of privacy may be overly conservative (developers may use privacy-preserving emails for some projects while being comfortable with identity linkage for others). Conversely, it may be insufficiently protective for developers who use pseudonyms without GitHub's privacy feature. Detecting and appropriately respecting such nuanced privacy preferences remains an open challenge for future research.

6.4 Conclusion Validity

While our large-scale analysis provides substantial statistical power, edge cases and unusual patterns may be underrepresented, and filtering may inadvertently exclude legitimate accounts with atypical patterns. Results may vary with different filtering approaches, rule applications, and threshold selections; yet, our sensitivity analysis (Section 5.3.1), ablation study (Section 5.3.2), and comparative study (Section 5.3.3) indicate that our findings are robust to such variations. Although our findings are derived from a specific set of configurations in the configuration space and therefore may not generalize to all possible configurations, our results present a minimal illustrative example of how a particular privacy signal affects the accuracy–privacy trade-off.

The 17 matching rules used in GitAuthority were derived from prior literature and refined through evaluation on existing datasets, but the relative contribution of each rule to accuracy and privacy may vary across developer populations. What works well for repositories of the Apache Software Foundation or OpenStack may not generalize to populations that use different email patterns, contribution behaviors, or privacy norms. Nevertheless, we rely on a comprehensive set of matching rules that have been shown to be effective in prior work, and our evaluation on a variety of datasets suggests that our approach is well applicable to different populations.

7 Discussion

Optimizing developer-identity de-aliasing for accuracy while respecting privacy is a balancing act. Our study demonstrates the critical role privacy plays in large-scale de-aliasing: Ignoring privacy-preserving email addresses substantially reduces accuracy, threatening validity and introducing matching biases. However, entirely ignoring developer privacy expectations raises ethical concerns. In the following, we first discuss the insights we gained into the accuracy–privacy trade-off in developer-identity de-aliasing, then we discuss the implications of our findings for research relying on de-aliasing, and finally we provide perspectives and practical guidance for the research community on how to handle this trade-off in future research.

7.1 Insights into the Accuracy–Privacy Trade-Off in Developer-Identity De-Aliasing

Developers using GitHub's privacy-preserving email feature may not fully understand its implications or may have varying expectations—some seeking anonymity, others merely avoiding spam. Even so, using a noreply address does not guarantee that it completely hides their real email address, as it does not apply to commits created by means other than the web interface (e.g., in a local development environment), unless a developer consistently configures a noreply address across all their devices and repositories, but even then past commits with real email addresses would still be visible. Indeed, for more than half of the GitHub users in our dataset that use a noreply address, we also found at least one alias with another email address. It is unclear how many of these users are aware that their real email address is still visible in some of their commits, and how many of them deliberately use both privacy-preserving and actual email addresses across their repositories.

Furthermore, there exists a fundamental tension between privacy and accountability: While developers have legitimate privacy concerns, attribution is important for recognizing contributions, establishing reputation, and enabling empirical research. Sometimes, developers use pseudonymous identities to hide some of their characteristics such as gender, as previous studies suggest women sometimes hide their real names to avoid disadvantages in open-source projects (Trinkenreich et al., 2022; Frluckaj et al., 2022). Also, some developers decide to use pseudonyms as a consequence of toxic behavior (Bharadwaj et al., 2025). All of these different aspects need to be carefully considered when de-aliasing developer identities, and there is no one-size-fits-all solution that respects all the different goals and preferences of all developers. This is further illustrated by developers who have explicitly requested the removal of their open-source contributions from datasets such as WoC, indicating

that privacy preferences in repository data go well beyond what noreply addresses alone can capture. Nevertheless, the accuracy penalty that comes with respecting even this one privacy signal might already have substantial consequences for research relying on de-aliasing, as we discuss in what follows.

7.2 Implications for Research Relying on De-Aliasing

In our GitHub dataset, switching to privacy-preserving de-aliasing (i.e., disabling matching rules that rely on noreply addresses as a privacy signal) reduces the recall from 0.77 to 0.47 and the percentage of GitHub users whose aliases are fully merged from approximately 74% to 43%. This drop in accuracy suggests that using privacy-preserving de-aliasing may have potential downstream consequences for any research that builds on developer-identity de-aliasing. Notably, previous research has shown that even simple metrics such as the number of commits attributed to a developer can be substantially affected by different tooling and analysis pipelines (Hoess et al., 2025), and our findings suggest that the choice of de-aliasing configuration can be a significant source of such variability. De-aliasing underpins a wide range of empirical software engineering research, from productivity analyses to collaboration network studies and career trajectory research (Hemmati et al., 2013; Bettenburg et al., 2009). Below, we highlight potential downstream effects for several common use cases.

Under privacy-preserving de-aliasing, studies that measure individual developer productivity (e.g., number of commits, lines of code, or files changed; Oliveira et al., 2020; Mockus, 2009) may substantially undercount the output of developers who use noreply addresses alongside regular aliases, since their aliases cannot be fully merged when respecting privacy preferences. When the two aliases of a developer (the median number of aliases in our dataset) cannot be linked, privacy-preserving de-aliasing would treat them as two different developers. This way, only a fraction of the contributions of this developer would be attributed to the developer. In projects where a core contributor happens to use both a noreply address and a regular email address with equal frequency, they may no longer be identified as a core contributor, while another developer with a single alias but fewer contributions may be incorrectly classified as a core contributor. Such misattributions could be particularly consequential for research on code ownership (e.g., Bird et al., 2011; Linares-Vásquez et al., 2012), developer coreness (Hechtl et al., 2026), and core-developer classification (e.g., Joblin et al., 2017; Bock et al., 2023; Çetin and Tüzün, 2020), where incorrect attribution of commits can lead to wrong conclusions.

To illustrate this concretely, we applied the full-accuracy and privacy-preserving configurations of GitAuthority to the 25 projects previously investigated by Bock et al. (2023) to validate core-developer classifications and tracked individual developers across configurations. Among 1 529 core developers (using the traditional 80/20 core-developer classification established in the literature; Joblin et al., 2017; Yamashita et al., 2015; Bock et al., 2023) with at least 20 commits under the full-accuracy configuration, 269 (17.6%) have their commit history fragmented across multiple aliases under the privacy-preserving configuration; for these developers, the largest surviving identity retains a mean of 87.0% (median 93.9%) of their original commit count, and 22 of them (8.2% of all fragmented developers) fall out of the core-developer bracket entirely. While this might sound negligible at first glance, the effect on individual rankings by commit count can be substantial: in project Keras,

a developer ranked 4th under the full-accuracy configuration (230 commits) is split into three identities under the privacy-preserving configuration, the largest retaining only 86 commits (37%), dropping the developer to rank 15. Similar effects occur in other projects: in React, the 2nd-ranked contributor (1 950 commits) fragments into five identities and drops to rank 7 (1 305 commits, 67%); in VS Code, a top-4 developer with 9 017 commits drops to rank 9; and in Next.js, a top-10 developer's 857 commits are split into two identities under the privacy-preserving configuration, the larger of which (595 commits) drops to rank 18, while the smaller fragment (262 commits) falls to rank 31. These concrete cases show that privacy-preserving de-aliasing can concretely change which developers are classified as core contributors and how they rank relative to one another.

This can also affect the identification of one-time contributors (e.g., Lee and Carver, 2017), the analysis of developers' roles within the project's organizational structure (e.g., Joblin et al., 2023), and studies on developer networks (e.g., Gharehyazie et al., 2015; Wang and Nagappan, 2021; Wang et al., 2022; Bock et al., 2022b). In particular, Amreen et al. (2020) found that even small errors in developer-identity de-aliasing can have a high impact on the resulting collaboration networks, and therefore also on metrics such as network centrality, clustering coefficients, or community detection. Also, studies that investigate synchronous development patterns (e.g., Bock et al., 2022a; Xuan and Filkov, 2014), socio-technical congruence (e.g., Mauerer et al., 2022), and coordination requirements (e.g., Cataldo and Herbsleb, 2013; Hunsen et al., 2020) are likely to be affected by such misattributions, as they rely on accurate mappings of who contributed what and when. In general, research on collaboration patterns in open-source ecosystems (e.g., Chen et al., 2022; Sun et al., 2023) or commit traceability to specific developers (e.g., Jiang et al., 2014), such as analyses of cross-project bug fixing (e.g., Ma et al., 2017), is susceptible to such fragmentation. Beyond mere collaboration-related research, research that tracks developers' career trajectories (e.g., van Wesel et al., 2017; Joblin et al., 2023; Zhou and Mockus, 2012), the geographic location or gender of developers (e.g., Gonzalez-Barahona et al., 2008; Hechtl et al., 2025; Vasilescu et al., 2015; Qiu et al., 2023), developer turnover and disengagement (e.g., Foucault et al., 2015; Lin et al., 2017; Rigby et al., 2016; Miller et al., 2019), or the participation of commercial contributors (e.g., Zhang et al., 2021) may be affected by such misattributions, as they rely on correctly linking a developer's contributions over time.

While we demonstrate that privacy-preserving de-aliasing incurs a substantial accuracy penalty, the concrete downstream effects on specific research findings remain an open question. At the level of an individual repository, misattributions may be small or even cancel out; however, at the large scale at which much of today's mining software repositories research operates—spanning millions of developers and hundreds of thousands of projects—small per-developer errors can compound into significant biases in aggregate results. Systematically quantifying these effects across the breadth of MSR research requires dedicated replication studies for each downstream use case, each demanding substantial methodological effort in its own right (Bock et al., 2026). We therefore call for future work that empirically investigates how privacy-aware de-aliasing affects the conclusions of downstream studies to better understand when and to what extent the accuracy–privacy trade-off materially changes research findings.

7.3 Perspectives and Practical Guidance for the Research Community

Despite de-aliasing being a recurring issue in MSR research for more than 20 years (see Section 3.2), research practice has fundamentally not changed. While privacy concerns around developer aliases have been discussed in MSR literature (Gold and Krinke, 2020; Robles and Gonzalez-Barahona, 2005), none of the existing de-aliasing algorithms explicitly handles privacy prior to merging developer aliases. The community, us included, keeps improving the algorithms without much regard for privacy, even if people's perceptions of privacy online have evolved substantially compared to 20 years ago. The lack of emphasis on privacy was arguably necessary, as (1) the research insights generated were valuable, "the end justifying the means"; and (2) there is no easy solution to preserving privacy: For example, about 30% of developers' Git aliases in the WoC dataset (containing 104M unique author aliases at the end of 2024) are noreply addresses, and ignoring these aliases while sampling for research would threaten external validity. Ignoring all the projects these developers touched would require dropping 44% of repositories as these repositories contain at least one commit from a noreply address; alternatively, contacting developers for consent may be perceived as spam or privacy violation, and would likely have a low response rate (Baltes and Diehl, 2016).

Our intent is to generate some momentum on this important issue. On a technical level, we quantify the extent of "the problem" and demonstrate how MSR researchers can begin making more deliberate decisions regarding de-aliasing privacy. While our approach does not aim for completeness (i.e., it does not cover all sorts of privacy-sensitive signals), our modular, configurable design of GitAuthority enables the research community to easily contribute additional heuristics and signals that can be used to enhance privacy in de-aliasing. Still, one might argue that solely focusing on noreply addresses as a single sort of privacy signal is not enough, but we believe that being respectful of many people's privacy sooner (we can already capture a high-frequency pattern) is better than being respectful of potentially more people's privacy later. We call for community-wide efforts to develop best practices and ethical guidelines for developer-identity de-aliasing, which require input from empirical software engineering researchers, data privacy experts, and ethicists. Dedicated workshops at software-engineering conferences or Dagstuhl seminars could be organized to collaboratively establish guidelines that balance scientific value with privacy protection.

While broadly applicable solutions will likely take some time as community discussions are necessary to figure out what an acceptable middle ground is (which requires input from multiple stakeholders, including GitHub), in the short term researchers could start by mining privacy-aware datasets (which do not contain authors or projects that use privacy signals). To that aim, we suggest adding such privacy-aware views of authors and projects to WoC, to make sampling easy. Regardless of how data are sampled, researchers could report conclusion differences between mining privacy-aware and privacy-agnostic datasets, which our configurable tool enables.

Configurable de-aliasing approaches like GitAuthority enable researchers to deliberately choose matching heuristics and respect privacy preferences based on the specific context and goals of their research. This could be used to increase accuracy within projects but favor privacy across projects. For example, using a two-step approach (Lamba et al., 2020)—first de-aliasing within projects using full-accuracy mode, then linking across projects using privacy-preserving mode—can balance pri-

vacy and accuracy, especially in cases where developers decided to stay anonymous in one project but not in another project.

A common practice for protecting developers' privacy is post-hoc pseudonymization, that is, de-aliasing in full-accuracy mode and pseudonymizing the resulting identities afterwards. While post-hoc pseudonymization hides who a developer is, it still links aliases that a developer might have chosen to keep separate, and such linked activity can enable re-identification. Post-hoc pseudonymization is, at the very least, necessary at the end of the data analysis pipeline when results are published, to honor privacy and prevent re-identification of individuals. Privacy-aware de-aliasing, in contrast, tackles this problem at the beginning of the data analysis pipeline: It prevents links between aliases in the first place, at the cost of lower recall, which makes it preferable when linking developer activity across sensitive sources carries re-identification risk, or when the analysis crosses contexts that developers may wish to keep separate. Both approaches are complementary: Pseudonymizing the output of privacy-aware de-aliasing respects developers' privacy signals before data analysis and protects against re-identification at publication stage. We therefore encourage researchers to reflect on this trade-off before conducting their analysis and to choose their strategy based on the requirements of the analysis and its implications for the individuals affected.

Finally, de-aliasing is also relevant to disciplines other than software engineering. The idea of making de-aliasing algorithms configurable to different needs is applicable in general, and thus GitAuthority can also be used in other contexts that use names and email addresses. While noreply addresses similar to GitHub's may not be used on other platforms, other traces of privacy sensitivity like "John Doe" instead of an actual name are likely to exist also in other contexts. The awareness of privacy in de-aliasing is also relevant in many other fields that use personal data, and our work might inspire researchers of other domains to create better approaches for their own scenarios.

8 Conclusion

In this article, we examined the tension between accuracy and privacy in developer-identity de-aliasing for Git repositories. Through our configurable approach GitAuthority, we reimplemented and refined existing heuristic-based de-aliasing techniques (i.e., handling German umlauts, leetspeak, GitHub usernames, nickname resolution, etc.), achieving accuracy comparable to or even slightly better than state-of-the-art methods across four ground-truth datasets from the literature. This enabled us to systematically investigate the implications of respecting privacy signals on de-aliasing accuracy. Our empirical study focused on GitHub's privacy-preserving `users.noreply.github.com` email addresses as observable indicators of potential privacy preferences. Using a novel sampling strategy based on single-author repositories, we analyzed the prevalence and implications of these privacy-sensitive aliases at scale. In particular, we showed that privacy-preserving de-aliasing can substantially reduce recall compared to privacy-agnostic de-aliasing, affecting about one third of the developers in our dataset (whose aliases are no longer fully merged under privacy-preserving de-aliasing). Entirely excluding projects that contain at least one privacy-preserving email address from repository mining studies would lead to excluding 44% of repositories, which would severely threaten the validity

of any empirical study. Our findings reveal a growing trend of developers using privacy-preserving email addresses, presenting a fundamental challenge for the MSR community: Respecting these privacy signals reduces de-aliasing performance on the available evaluation metrics, yet ignoring these signals raises ethical concerns and may violate developers' explicit privacy choices. However, no one-size-fits-all solution exists. We demonstrated that a configurable de-aliasing approach such as ours can be adapted to respect privacy preferences, and we provided practical guidance for researchers on how to navigate the accuracy–privacy trade-off in their specific contexts. Ultimately, we call for community-wide discussions to establish best practices and ethical guidelines for developer-identity de-aliasing that balance the scientific value of accurate attribution with the imperative to respect developer privacy in an era where privacy expectations are increasingly prevalent.

Acknowledgements We thank Christoph Gote and Igor Steinmacher for providing us with their ground-truth datasets of developer identities from their previous work. The work presented in this article began at the WoC Hackathon at the University of Tennessee, Knoxville, in May 2025.

Statements and Declarations

Data Availability While we cannot publicly release our datasets due to privacy concerns, we provide the implementation of GitAuthority as well as all our evaluation scripts and the scripts to construct our ground-truth dataset in a publicly available supplementary repository: https://github.com/bockthom/gitauthority/. In addition, it is also persistently available at https://doi.org/10.5281/zenodo.23790141 . Researchers interested in using our datasets for further research can contact the corresponding author to discuss potential options for data sharing for research purposes that respect privacy concerns.

Funding This work was partially funded by the National Science Foundation (NSF) under grant numbers 2120323 & 2120429, by the National Institutes of Health (NIH) under project number 1R01GM164731-01, by the Alfred P. Sloan Foundation under grant number g-2025-25171, and by Schmidt Sciences.

Conflicts of Interests/Competing Interests The authors have no competing interests to declare that are relevant to the content of this article.

Author Contributions ***Thomas Bock:*** Conceptualization, Methodology, Investigation, Data curation, Formal analysis, Software, Validation, Writing – original draft, Writing – review & editing, Visualization, Funding acquisition. ***Shyam Agarwal:*** Methodology, Investigation, Data curation, Validation, Writing – review & editing. ***Curtis Atkisson:*** Conceptualization, Methodology, Writing – review & editing, Funding acquisition. ***Audris Mockus:*** Conceptualization, Methodology, Investigation, Data curation, Writing – review & editing, Funding acquisition, Resources, Supervision. ***Bogdan Vasilescu:*** Conceptualization, Methodology, Writing – review & editing, Funding acquisition, Supervision.

Ethical Approval This article does not contain any studies with human participants.

References

Aghili R, Li H, Khomh F (2025) Protecting Privacy in Software Logs: What Should Be Anonymized? Proc. ACM on Software Engineering (PACMSE) 2(FSE):60

Alfieri C, Di Rocco J, Inverardi P, Nguyen PT (2024) Exploring User Privacy Awareness on GitHub: An Empirical Study. Empirical Software Engineering (EMSE) 29(6):156

Amreen S, Karnauch A, Mockus A (2019) Developer Reputation Estimator (DRE). In: Proc. Int. Conf. Automated Software Engineering (ASE), IEEE, pp 1082–1085
Amreen S, Mockus A, Zaretzki R, Bogart C, Zhang Y (2020) ALFAA: Active Learning Fingerprint Based Anti-Aliasing for Correcting Developer Identity Errors in Version Control Systems. Empirical Software Engineering (EMSE) 25(2):1136–1167
Andalibi N, Haimson OL, De Choudhury M, Forte A (2016) Understanding Social Media Disclosures of Sexual Abuse through the Lenses of Support Seeking and Anonymity. In: Proc. Conf. Human Factors in Computing Systems (CHI), ACM, pp 3906–3918
Baltes S, Diehl S (2016) Worse Than Spam: Issues in Sampling Software Developers. In: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), ACM, art. 52
Barth S, De Jong MDT, Junger M, Hartel PH, Roppelt JC (2019) Putting the Privacy Paradox to the Test: Online Privacy and Security Behaviors Among Users with Technical Knowledge, Privacy Awareness, and Financial Resources. Telematics and Informatics 41:55–69
Berry DM (2004) Internet Research: Privacy, Ethics and Alienation: An Open Source Approach. Internet Research 14(4):323–332
Bettenburg N, Shihab E, Hassan AE (2009) An Empirical Study on the Risks of Using Off-the-Shelf Techniques for Processing Mailing List Data. In: Proc. Int. Conf. Software Maintenance (ICSM), IEEE, pp 539–542
Bharadwaj S, Santos F, Trinkenreich B (2025) The Shifting Sands of Toxicity: The Evolving Nature of Interpersonal Challenges in Open Source. In: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), IEEE, pp 242–252
Bird C, Gourley A, Devanbu PT, Gertz M, Swaminathan A (2006) Mining Email Social Networks. In: Proc. Int. Conf. Mining Software Repositories (MSR), ACM, pp 137–143
Bird C, Nagappan N, Murphy B, Gall HC, Devanbu PT (2011) Don't Touch My Code! Examining the Effects of Ownership on Software Quality. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 4–14
Bock T, Hunsen C, Joblin M, Apel S (2022a) Synchronous Development in Open-Source Projects: A Higher-Level Perspective. Automated Software Engineering 29(1):3
Bock T, Schmid A, Apel S (2022b) Measuring and Modeling Group Dynamics in Open-Source Software Development: A Tensor Decomposition Approach. ACM Transactions on Software Engineering and Methodology (TOSEM) 31(2):19
Bock T, Alznauer N, Joblin M, Apel S (2023) Automatic Core-Developer Identification on GitHub: A Validation Study. ACM Transactions on Software Engineering and Methodology (TOSEM) 32(6):138
Bock T, Mockus A, Vasilescu B (2026) The Ground Is Shifting: A Reflection on the Foundations of Software Measurement. In: Proc. Int. Conf. Automated Software Engineering (ASE), ACM, pp 4595–4600
Bogomolov E, Kovalenko V, Rebryk Y, Bacchelli A, Bryksin T (2021) Authorship Attribution of Source Code: A Language-Agnostic Approach and Applicability in Software Engineering. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 932–944
Brennan M, Afroz S, Greenstadt R (2012) Adversarial Stylometry: Circumventing Authorship Recognition to Preserve Privacy and Anonymity. ACM Transactions on Information and System Security (TISSEC) 15(3):12
Caliskan-Islam A, Harang R, Liu A, Narayanan A, Voss C, Yamaguchi F, Greenstadt R (2015) De-Anonymizing Programmers via Code Stylometry. In: USENIX Security Symposium (USENIX Security), USENIX Association, pp 255–270
Campanella S, Lanza M (2024) Hidden in the Code: Visualizing True Developer Identities. In: Working Conference on Software Visualization (VISSOFT), IEEE, pp 24–35
Canfora G, Cerulo L, Cimitile M, Di Penta M (2011) Social Interactions Around Cross-System Bug Fixings: The Case of FreeBSD and OpenBSD. In: Proc. Int. Conf. Mining Software Repositories (MSR), ACM, pp 143–152
Cataldo M, Herbsleb JD (2013) Coordination Breakdowns and Their Impact on Development Productivity and Software Failures. IEEE Transactions on Software Engineering (TSE) 39(3):343–360
Çetin HA, Tüzün E (2020) Identifying Key Developers Using Artifact Traceability Graphs. In: Proc. Int. Conf. Predictive Models in Software Engineering (PROMISE), ACM, pp 51–60
Chen Z, Ma W, Chen L, Song W (2022) Collaboration in Software Ecosystems: A Study of Work Groups in Open Environment. Information and Software Technology (IST) 145:106849

Christen P (2006) A Comparison of Personal Name Matching: Techniques and Practical Issues. In: Proc. Int. Conf. Data Mining (ICDM), IEEE, pp 290–294
Cochran WG (1977) Sampling Techniques, 3rd edn. John Wiley & Sons
Díaz Ferreyra NE, Imine A, Vidoni M, Scandariato R (2023) Developers Need Protection, Too: Perspectives and Research Challenges for Privacy in Social Coding Platforms. In: Proc. Int. Conf. Cooperative and Human Aspects of Software Engineering (CHASE), IEEE, pp 105–110
Feitelson DG (2023) "We Do Not Appreciate Being Experimented On": Developer and Researcher Views on the Ethics of Experiments on Open-Source Projects. Journal of Systems and Software (JSS) 204:111774
Forte A, Andalibi N, Greenstadt R (2017) Privacy, Anonymity, and Perceived Risk in Open Collaboration: A Study of Tor Users and Wikipedians. In: Proc. Int. Conf. Computer-Supported Cooperative Work (CSCW), ACM, pp 1800–1811
Foucault M, Palyart M, Blanc X, Murphy GC, Falleri JR (2015) Impact of Developer Turnover on Quality in Open-Source Software. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 829–841
Frluckaj H, Dabbish LA, Widder DG, Qiu HS, Herbsleb JD (2022) Gender and Participation in Open Source Software Development. Proceedings of the ACM on Human-Computer Interaction (HCI) 6(CSCW2):299
Fry T, Dey T, Karnauch A, Mockus A (2020) A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits. In: Proc. Int. Conf. Mining Software Repositories (MSR), ACM, pp 518–522
Gharehyazie M, Posnett D, Vasilescu B, Filkov V (2015) Developer Initiation and Social Interactions in OSS: A Case Study of the Apache Software Foundation. Empirical Software Engineering (EMSE) 20(5):1318–1353
Goeminne M, Mens T (2013) A Comparison of Identity Merge Algorithms for Software Repositories. Science of Computer Programming (SCP) 78(8):971–986
Golbeck J, Rothstein M (2008) Linking Social Networks on the Web with FOAF: A Semantic Web Case Study. In: Proc. Conf. Artificial Intelligence (AAAI), AAAI, pp 1138–1143
Gold NE, Krinke J (2020) Ethics in Mining: A Case Study on MSR Mining Challenges. In: Proc. Int. Conf. Mining Software Repositories (MSR), ACM, pp 265–276
Gong S, Zhong H (2022) A Study on Identifying Code Author from Real Development. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 1627–1631
Gong S, Zhong H (2025) Incremental Learning of Code Authors Over Time. Journal of Systems and Software (JSS) 230:112527
Gonzalez-Barahona JM, Robles G, Andradas-Izquierdo R, Ghosh RA (2008) Geographic Origin of Libre Software Developers. Information Economics and Policy 20(4):356–363
Gote C, Zingg C (2021) gambit – An Open Source Name Disambiguation Tool for Version Control Systems. In: Proc. Int. Conf. Mining Software Repositories (MSR), IEEE, pp 80–84
Hechtl C, Joblin M, Apel S (2025) Is Perceived Gender Related to Contributions and Standing in Open-Source Software Projects? Empirical Software Engineering (EMSE) 30(5):123
Hechtl C, Bock T, Ramsauer R, Mauerer W, Apel S (2026) How Developer Coreness Influences the Patch-Review Process: A Mixed-Method Study. Automated Software Engineering 33(4):108
Hemmati H, Nadi S, Baysal O, Kononenko O, Wang W, Holmes R, Godfrey MW (2013) The MSR Cookbook: Mining a Decade of Research. In: Proc. Int. Conf. Mining Software Repositories (MSR), IEEE, pp 343–352
Hennig A, Schulte L, Herbold S, Kulyk O, Mayer P (2026) The Whos, Whats, and Whys of Issues Related to Personal Data and Data Protection in Open-Source Projects on GitHub. Empirical Software Engineering (EMSE) 31(1):9
Hoess N, Paradis C, Kazman R, Mauerer W (2025) Does the Tool Matter? Exploring Some Causes of Threats to Validity in Mining Software Repositories. In: Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), IEEE, pp 645–656
Hunsen C, Siegmund J, Apel S (2020) On the Fulfillment of Coordination Requirements in Open-Source Software Projects: An Exploratory Study. Empirical Software Engineering (EMSE) 25(6):4379–4426
Iofciu T, Fankhauser P, Abel F, Bischoff K (2011) Identifying Users Across Social Tagging Systems. Proc. Int. AAAI Conf. Web and Social Media (ICWSM) 5(1):522–525

Jiang Y, Adams B, Khomh F, Germán DM (2014) Tracing Back the History of Commits in Low-Tech Reviewing Environments: A Case Study of the Linux Kernel. In: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), ACM, art. 51

Joblin M, Apel S, Hunsen C, Mauerer W (2017) Classifying Developers into Core and Peripheral: An Empirical Study on Count and Network Metrics. In: Proc. Int. Conf. Software Engineering (ICSE), IEEE, pp 164–174

Joblin M, Eckl-Ganser B, Bock T, Schmid A, Siegmund J, Apel S (2023) Hierarchical and Hybrid Organizational Structures in Open-Source Software Projects: A Longitudinal Study. ACM Transactions on Software Engineering and Methodology (TOSEM) 32(4):86

Kalliamvakou E, Gousios G, Blincoe K, Singer L, Germán DM, Damian D (2016) An In-Depth Study of the Promises and Perils of Mining GitHub. Empirical Software Engineering (EMSE) 21(5):2035–2071

Kamola M (2019) How to Verify Conway's Law for Open Source Projects. IEEE Access 7:38469–38480

Kang R, Brown S, Kiesler S (2013) Why Do People Seek Anonymity on the Internet? Informing Policy and Design. In: Proc. Conf. Human Factors in Computing Systems (CHI), ACM, pp 2657–2666

Kazemitabaar M, Chow J, Ma CKT, Ericson BJ, Weintrop D, Grossman T (2023) Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In: Proc. Conf. Human Factors in Computing Systems (CHI), ACM, art. 455

Kim B, Kim D (2020) Understanding the Key Antecedents of Users' Disclosing Behaviors on Social Networking Sites: The Privacy Paradox. Sustainability 12(12):5163

Knothe D, Pietschmann F (2019) Large-Scale-Exploit of GitHub Repository Metadata and Preventive Measures. arXiv preprint arXiv:1908.05354

Kouters E (2014) Identity Matching and Geographical Movement of Open-Source Software Mailing List Participants. Master's thesis, Eindhoven University of Technology

Kouters E, Vasilescu B, Serebrenik A, van den Brand MGJ (2012) Who's Who in Gnome: Using LSA to Merge Software Repository Identities. In: Proc. Int. Conf. Software Maintenance (ICSM), IEEE, pp 592–595

Lamba H, Trockman A, Armanios D, Kästner C, Miller H, Vasilescu B (2020) Heard It Through the Gitvine: An Empirical Study of Tool Diffusion across the *npm* Ecosystem. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 505–517

Lee A, Carver JC (2017) Are One-Time Contributors Different? A Comparison to Core and Periphery Developers in FLOSS Repositories. In: Proc. Int. Sympos. Empirical Software Engineering and Measurement (ESEM), IEEE, pp 1–10

Lermen S, Paleka D, Swanson J, Aerni M, Carlini N, Tramèr F (2026) Large-Scale Online Deanonymization with LLMs. In: USENIX Security Symposium (USENIX Security), USENIX Association, pp 1947–1966

Levi-Eshkol K, Ribak R (2025) "Track Every Move": Analyzing Developers' Privacy Discourse in GitHub README Files. new media & society 27(12):6534–6550

Li T, Li N (2009) On the Tradeoff between Privacy and Utility in Data Publishing. In: Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD), ACM, pp 517–526

Li Z, Chen GQ, Chen C, Zou Y, Xu S (2022) RoPGen: Towards Robust Code Authorship Attribution via Automatic Coding Style Transformation. In: Proc. Int. Conf. Software Engineering (ICSE), ACM, pp 1906–1918

Li Z, Zhao S, Chen C, Chen Q (2024) Reducing the Impact of Time Evolution on Source Code Authorship Attribution via Domain Adaptation. ACM Transactions on Software Engineering and Methodology (TOSEM) 33(6):159

Lin B, Robles G, Serebrenik A (2017) Developer Turnover in Global, Industrial Open Source Projects: Insights from Applying Survival Analysis. In: Proc. Int. Conf. Global Software Engineering (ICGSE), IEEE, pp 66–75

Linares-Vásquez M, Hossen K, Dang H, Kagdi H, Gethers M, Poshyvanyk D (2012) Triaging Incoming Change Requests: Bug or Commit History, or Code Authorship? In: Proc. Int. Conf. Software Maintenance (ICSM), IEEE, pp 451–460

Liu J, Zhang F, Song X, Song YI, Lin CY, Hon HW (2013) What's in a Name? An Unsupervised Approach to Link Users Across Communities. In: Proc. Int. Conf. Web Search and Data Mining (WSDM), ACM, pp 495–504

Liu S, Wang S, Zhu F, Zhang J, Krishnan R (2014) HYDRA: Large-Scale Social Identity Linkage via Heterogeneous Behavior Modeling. In: Proc. Int. Conf. Management of Data

(SIGMOD), ACM, pp 51–62
Liu Z (2023) Application and Security Analysis of Virtual Private Network (VPN) in Network Communication. Academic Journal of Computing & Information Science 6(11):52–59
Ma W, Chen L, Zhang X, Zhou Y, Xu B (2017) How Do Developers Fix Cross-Project Correlated Bugs? A Case Study on the GitHub Scientific Python Ecosystem. In: Proc. Int. Conf. Software Engineering (ICSE), IEEE, pp 381–392
Ma Y, Bogart C, Amreen S, Zaretzki R, Mockus A (2019) World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data. In: Proc. Int. Conf. Mining Software Repositories (MSR), IEEE, pp 143–154
Malhotra A, Totti L, Meira Jr W, Kumaraguru P, Almeida V (2012) Studying User Footprints in Different Online Social Networks. In: Proc. Int. Conf. Advances in Social Networks Analysis and Mining (ASONAM), IEEE, pp 1065–1070
Marx GT (1999) What's in a Name? Some Reflections on the Sociology of Anonymity. The Information Society 15(2):99–112
Mauerer W, Joblin M, Tamburri DA, Paradis C, Kazman R, Apel S (2022) In Search of Socio-Technical Congruence: A Large-Scale Longitudinal Study. IEEE Transactions on Software Engineering (TSE) 48(8):3159–3184
Miller C, Widder DG, Kästner C, Vasilescu B (2019) Why Do People Give Up FLOSSing? A Study of Contributor Disengagement in Open Source. In: Proc. IFIP Int. Conf. Open Source Systems, Springer, pp 116–129
Mo W, Shen B, Chen Y, Zhu J (2015) TBIL: A Tagging-Based Approach to Identity Linkage Across Software Communities. In: Proc. Asia-Pacific Software Engineering Conf. (APSEC), IEEE, pp 56–63
Mockus A (2009) Succession: Measuring Transfer of Code and Developer Productivity. In: Proc. Int. Conf. Software Engineering (ICSE), IEEE, pp 67–77
Moreno D, Dueñas S, Cosentino V, Fernandez MA, Zerouali A, Robles G, Gonzalez-Barahona JM (2019) SortingHat: Wizardry on Software Project Members. In: Companion Volume ICSE, IEEE, pp 51–54
Nissenbaum H (2009) Privacy in Context: Technology, Policy, and the Integrity of Social Life. Stanford University Press
Oliva GA, Santana FW, de Oliveira KCM, de Souza CRB, Gerosa MA (2012) Characterizing Key Developers: A Case Study With Apache Ant. In: Proc. Int. Conf. Collaboration and Technology (CRIWG), Springer, pp 97–112
Oliveira E, Fernandes E, Steinmacher I, Cristo M, Conte T, Garcia A (2020) Code and Commit Metrics of Developer Productivity: A Study on Team Leaders Perceptions. Empirical Software Engineering (EMSE) 25(4):2519–2549
Palen L, Dourish P (2003) Unpacking "Privacy" for a Networked World. In: Proc. Conf. Human Factors in Computing Systems (CHI), ACM, pp 129–136
Poncin W, Serebrenik A, van den Brand MGJ (2011) Process Mining Software Repositories. In: Proc. Europ. Conf. Software Maintenance and Reengineering (CSMR), IEEE, pp 5–14
Qian Y, Hu Y, Cui J, Zheng Q, Nie Z (2011) Combining Machine Learning and Human Judgment in Author Disambiguation. In: Proc. Int. Conf. Information and Knowledge Management (CIKM), ACM, pp 1241–1246
Qiu HS, Zhao ZH, Yu TK, Wang J, Ma A, Fang H, Dabbish LA, Vasilescu B (2023) Gender Representation Among Contributors to Open-Source Infrastructure: An Analysis of 20 Package Manager Ecosystems. In: Proc. Int. Conf. Software Engineering: Software Engineering in Society (ICSE-SEIS), IEEE, pp 180–187
Rigby PC, Zhu YC, Donadelli SM, Mockus A (2016) Quantifying and Mitigating Turnover-Induced Knowledge Loss: Case Studies of Chrome and a Project at Avaya. In: Proc. Int. Conf. Software Engineering (ICSE), IEEE, pp 1006–1016
Robillard MP, Arya DM, Ernst NA, Guo JLC, Lamothe M, Nassif M, Novielli N, Serebrenik A, Steinmacher I, Stol KJ (2024) Communicating Study Design Trade-offs in Software Engineering. ACM Transactions on Software Engineering and Methodology (TOSEM) 33(5):112
Robles G, Gonzalez-Barahona JM (2005) Developer Identification Methods for Integrated Data from Various Sources. ACM SIGSOFT Software Engineering Notes 30(4):1–5
Sharma A, Karmakar S, Kancherla GP, Bichhawat A (2025) On the Prevalence and Usage of Commit Signing on GitHub: A Longitudinal and Cross-Domain Study. In: Evaluation and Assessment in Software Engineering (EASE), ACM, pp 360–370
Smalheiser NR, Torvik VI (2009) Author Name Disambiguation. Annual Review of Information Science and Technology 43(1):6

Smith HJ, Milberg SJ, Burke SJ (1996) Information Privacy: Measuring Individuals' Concerns About Organizational Practices. MIS Quarterly 20(2):167–196

Sun W, Iwuchukwu S, Bangash AA, Hindle A (2023) An Empirical Study to Investigate Collaboration Among Developers in Open Source Software (OSS). In: Proc. Int. Conf. Mining Software Repositories (MSR), IEEE, pp 352–356

Terčon L, Dobrovoljc K (2025) Linguistic Characteristics of AI-Generated Text: A Survey. arXiv preprint arXiv:2510.05136

Trinkenreich B, Wiese I, Sarma A, Gerosa MA, Steinmacher I (2022) Women's Participation in Open Source Software: A Survey of the Literature. ACM Transactions on Software Engineering and Methodology (TOSEM) 31(4):81

Vasilescu B, Serebrenik A, Goeminne M, Mens T (2014) On the Variation and Specialisation of Workload—A Case Study of the Gnome Ecosystem Community. Empirical Software Engineering (EMSE) 19(4):955–1008

Vasilescu B, Serebrenik A, Filkov V (2015) A Dataset for Social Diversity Studies of GitHub Teams. In: Proc. Int. Conf. Mining Software Repositories (MSR), IEEE, pp 514–517

Ventura SL, Nugent R, Fuchs ERH (2015) Seeing the Non-Stars: (Some) Sources of Bias in Past Disambiguation Approaches and a New Public Tool Leveraging Labeled Records. Research Policy 44(9):1672–1701

Vidoni M, Díaz Ferreyra NE (2022) Should I Get Involved? On the Privacy Perils of Mining Software Repositories for Research Participants. In: Proc. Int. Workshop on Recruiting Participants for Empirical Software Engineering (RoPES), ACM, art. 3

Wang L, Li Y, Zhang J, Tao X (2022) Quantifying Community Evolution in Developer Social Networks. In: Proc. Europ. Software Engineering Conf. and Int. Sympos. Foundations of Software Engineering (ESEC/FSE), ACM, pp 157–169

Wang S, Nagappan N (2021) Characterizing and Understanding Software Developer Networks in Security Development. In: Proc. Int. Sympos. Software Reliability Engineering (ISSRE), IEEE, pp 534–545

van Wesel P, Lin B, Robles G, Serebrenik A (2017) Reviewing Career Paths of the OpenStack Developers. In: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE, pp 544–548

Wiese IS, da Silva JT, Steinmacher I, Treude C, Gerosa MA (2016) Who is Who in the Mailing List? Comparing Six Disambiguation Heuristics to Identify Multiple Addresses of a Participant. In: Proc. Int. Conf. Software Maintenance and Evolution (ICSME), IEEE, pp 345–355

Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in Software Engineering, 2nd edn. Springer

Wong MF, Guo S, Hang CN, Ho SW, Tan CW (2023) Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review. Entropy 25(6):888

Xiong Y, Meng Z, Shen B, Yin W (2017) Developer Identity Linkage and Behavior Mining Across GitHub and Stack Overflow. International Journal of Software Engineering and Knowledge Engineering (IJSEKE) 27(09&10):1409–1425

Xu FF, Vasilescu B, Neubig G (2022) In-IDE Code Generation from Natural Language: Promise and Challenges. ACM Transactions on Software Engineering and Methodology (TOSEM) 31(2):29

Xuan Q, Filkov V (2014) Building It Together: Synchronous Development in OSS. In: Proc. Int. Conf. Software Engineering (ICSE), ACM, pp 222–233

Yamak Z, Saunier J, Vercouter L (2016) Detection of Multiple Identity Manipulation in Collaborative Projects. In: Proc. Int. Conf. World Wide Web (WWW), ACM, pp 955–960

Yamashita K, McIntosh S, Kamei Y, Hassan AE, Ubayashi N (2015) Revisiting the Applicability of the Pareto Principle to Core Development Teams in Open Source Software Projects. In: Proc. Int. Workshop on Principles of Software Evolution (IWPSE), ACM, pp 46–55

Yang X, Xu G, Li Q, Guo Y, Zhang M (2017) Authorship Attribution of Source Code by Using Back Propagation Neural Network Based on Particle Swarm Optimization. PLOS ONE 12(11):e0187204

Zhang J, Yu PS (2015) Multiple Anonymized Social Networks Alignment. In: Proc. Int. Conf. Data Mining (ICDM), IEEE, pp 599–608

Zhang Y, Zhou M, Mockus A, Jin Z (2021) Companies' Participation in OSS Development–An Empirical Study of OpenStack. IEEE Transactions on Software Engineering (TSE) 47(10):2242–2259

Zhang Z, Saber T (2025) Exploring the Boundaries Between LLM Code Clone Detection and Code Similarity Assessment on Human and AI-Generated Code. Big Data and Cognitive Computing 9(2):41
Zhou M, Mockus A (2012) What Make Long Term Contributors: Willingness and Opportunity in OSS Community. In: Proc. Int. Conf. Software Engineering (ICSE), IEEE, pp 518–528



Thomas Bock is currently a post-doctoral researcher in the Software and Societal Systems Department in the School of Computer Science at Carnegie Mellon University. Previously, he was a Ph.D. student at the Chair of Software Engineering at Saarland University, Saarland Informatics Campus, Germany. He received his Ph.D. in Computer Science from Saarland University in 2024. His research interests are focused on empirical software engineering, the social aspects of software engineering, and scientific software. In his research, he investigates coordination and communication among software developers, developers' roles, and organizational patterns in evolving software projects.



Shyam Agarwal is a Ph.D. student in the Software and Societal Systems Department in the School of Computer Science at Carnegie Mellon University. As AI becomes part of everyday software development, his work aims to understand, evaluate, and build intelligent systems that reshape how people create software. He works at the intersection of AI, software engineering, and human–computer interaction, where he studies how coding agents change developer workflows, evaluates human–AI programming interaction, and builds socially aware coding agents that collaborate with developers and augment their capabilities. His goal is to create AI tools that make software engineering more effective, sustainable, and aligned with human needs.



Curtis Atkisson is an academic data scientist and researcher at the University of Washington's eScience Institute. Trained as an anthropologist, his research examines how social structures, institutions, and patterns of interaction shape collective behavior and outcomes. His work combines computational, statistical, and qualitative methods across domains including scientific collaboration, research software engineering, and open-source software development. He received his Ph.D. in anthropology from the University of California, Davis, and completed a postdoctoral fellowship in public policy at the University of Massachusetts Amherst.



Audris Mockus is a Research Scientist at Meta and the Erickson Harlan Mills Chair Professor in the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville (UTK). He received his Bachelor of Science and his Master of Science in Applied Mathematics from the Moscow Institute of Physics and Technology, and his Master of Science and Ph.D. in Statistics from Carnegie Mellon University. Prior to UTK and Meta, he was with Avaya Labs Research and Bell Labs. His research develops data-driven methods for understanding complex software systems and the communities that produce them, including the World of Code infrastructure for mining the global open-source software universe. He received the Most Influential Paper Award at ICSE 2010 and the Foundational Contribution Award at MSR 2023.



Bogdan Vasilescu is an Associate Professor in the Software and Societal Systems Department in the School of Computer Science at Carnegie Mellon University (CMU). Prior to CMU, he was a post-doctoral researcher in the Davis Eclectic Computational Analytics Lab (DECAL) at the University of California, Davis. He obtained his Ph.D. in Computer Science at Eindhoven University of Technology, the Netherlands, in 2014, and holds a master's degree in Computer Science and Engineering from Eindhoven University of Technology and an engineer's degree in Systems and Computer Science from the University of Ploieşti, Romania. He is engaged in interdisciplinary research to offer data-driven answers to questions in empirical software engineering, social computing, and computer-supported cooperative work. He received the Ric Holt Early Career Achievement Award at MSR 2021 and his work received multiple ACM SIGSOFT Distinguished Paper Awards at ICSE and FSE.