UNIVERSITÄT
BERN

Profiling Cryptography Developers

Bachelor Thesis

Said Ali

from

Rüfenacht BE, Switzerland

Faculty of Science
University of Bern

August 2020

Prof. Dr. Oscar Nierstrasz

Mohammadreza Hazhirpasand

Software Composition Group
Institute of Computer Science
University of Bern, Switzerland

Abstract

Profiling developer expertise on the internet can provide valuable information for a multitude of applications such as recruiting. Studies have shown that it is feasible to track and profile developer activity on various platforms, (*e.g.*, *Stack Overflow* and *GitHub*). Furthermore, tracking developer expertise can shed some light on whether developer activity on one platform is in line with the same developer's activity on another platform. Recently, studies have shown that developers often rely on vulnerable cryptography code snippets, which are commonly found on *Stack Overflow* or *GitHub* repositories. Therefore, we are interested to investigate to what extent cryptography experts on *Stack Overflow* employ cryptography on their open-source projects on *GitHub*.

To achieve our goal, we build a five-stage pipeline. (1) We extract 40 crypto-related tags from *Stack Overflow*. (2) We identify 1,000 users who have accepted answers (crypto accepted answers) in discussions where the selected crypto tags were used. (3) We automatically and manually scrape the selected users' profiles on *Stack Overflow* and find 522 *GitHub* links (*i.e.*, users). (4) The 522 users contribute to 23,633 repositories, in which 3.4% are crypto-related. (5) Finally, we extract the contributors (*i.e.*, crypto contributors) of crypto files in the crypto-related repositories.
We use statistical and visual analyses to observe whether different groups of developers differ in terms of crypto activities (crypto score, reputation, and number of crypto accepted answers) on *Stack Overflow* and the number of crypto file contributions on *GitHub*. Our findings reveal that crypto activities between crypto contributors (189) and users without crypto contributions (332) do not differ significantly. Moreover, crypto contributors with a high number of crypto activities on *Stack Overflow* do not have a higher number of crypto contributions on *GitHub*. Overall we are unable to find any correlation between crypto developer activity on *Stack Overflow* and crypto developer contribution on *GitHub*.

Contents

1 Introduction

Developer expertise is an essential factor in software development [6]. Profiling developer expertise is used to estimate developer expertise qualitatively and quantitatively. Hence, the results of such a profiling approach indicates the skills that a developer has obtained and the levels that a developer performs on those skills. Profiling developer expertise has been shown to help improve effective task allocation and developer recruitment [5]. For instance, recruiters often rely on reputation and badges from *Stack Overflow* to measure developer expertise [28]. However, profiling developer expertise is a great challenge as their activities are often spread across multiple online communities [5].

Some studies have focused on profiling developer expertise on a single platform while other have conducted such studies cross-platform [11, 14, 4,22]. Since developer activity is often spread across multiple online communities, conducting such studies cross-platform can provide more valuable information. However, identity linkage is considered a challenge as developers may use different aliases on different platforms [22]. It is of great interest among researchers to link developer identity between *Stack Overflow* and *GitHub* in order to observe how developers perform on both platforms [2,3,4,22]. Recently Zhang *et al.* identified cross-platform profiles by matching the hash (*i.e.*, MD5) of email addresses from public active users from *Stack Overflow* and *GitHub* [4].

We should not lose sight of the fact that such platforms host vulnerable code snippets, and this issue often adversely affects the developer performance [34]. A recent study showed that developers blindly rely on

Stack Overflow discussions to resolve their programming challenges [29]. Yang *et al.* concluded that 1.9M out of 290M function definitions on *GitHub* contain snippets captured in *Stack Overflow* [29]. A study has revealed that developers rely on unvalidated code from online sources (*e.g.*, *Stack Overflow* and *GitHub*) where security vulnerabilities are common [27]. Unfortunately, the connection between experts on *Stack Overflow* and their developer productivity is not well-understood [5]. It is of great interest to investigate developer' expertise cross-platform (*e.g.*, *Stack Overflow* and *GitHub*).

The above security challenges have become critical for the cryptography domain in recent years. To date, we were unable to find any research paper working on profiling developer expertise in the domain of cryptography. A series of recent studies have indicated that lack of developer knowledge in the domain of cryptography has led to many software vulnerabilities [27,30]. In particular, developers commonly resolve their crypto challenge on online sources such as *Stack Overflow* or *GitHub*, which are often not secure [27]. An empirical study of cryptographic misuse in android applications shows that 88% of 11,748 applications that use cryptographic APIs make at least one mistake [30]. As a result, developers do not use cryptographic APIs in a way that maximizes overall security.

In this work, we investigate the correlation of crypto developer activity on *Stack Overflow* and crypto developer contribution on *GitHub*. To achieve our goal, we have built a five-stage pipeline.

(1) In the tag analysis, we choose "cryptography" as our base tag, which is used in 11,130 discussions on *Stack Overflow*. The discussions contain 2,184 tags. Of the total tags we collect, we extract 40 crypto-related tags. (2) We fetch 1,000 users who have accepted answers (crypto accepted answers) in discussions where the selected crypto tags are used. We store a unique identifier and for each user crypto activities (crypto score, reputation, and number of crypto accepted answers). (3) We automatically and manually scrape the selected users' profiles on *Stack Overflow* and find 522 *GitHub* links (*i.e.*, users). (4) The 522 users contribute to 23,633 repositories, in which 812 repositories (*i.e.*, 3.4%) contain cryptographic APIs. (5) Finally, we collect contributors (*i.e.*, crypto contributors) of the crypto files in the repositories (*i.e.*, 812) and check whether the *Stack Overflow* developers (*i.e.*, 522) are among the crypto contributors.

In our analysis, we use seven approaches to look for significant differences in the data. We compare the crypto activities of the 189 crypto contributors with the 332 users who did not contribute to crypto files. The result of the Mann-Whitney U-test shows that there is no correlation in crypto activities between the two groups. In other words, a randomly selected value of crypto activities (crypto score, reputation, and number of crypto accepted answers) from the first group is considered to be equal to a randomly selected value of the second group. Finally, we were unable to find any significant differences among the 189 crypto contributors based on their number of *Stack Overflow* crypto activities and their *GitHub* crypto contribution. Therefore, we can conclude that there is no correlation between crypto developer activity on *Stack Overflow* and crypto developer contribution on *GitHub*.

The main contributions of this work are to share the analyzed dataset, to identify the extent to which crypto experts use cryptography on their open-source projects, and the pipeline. The data analyses demonstrate that crypto developer activity on *Stack Overflow* is not in line with their crypto contribution on *GitHub*. However, this matter needs further investigation and may not reflect the developers' real expertise. To do so, the pipeline can be extended by adding additional programming languages or crypto libraries. Furthermore, researchers can employ the pipeline to profile developer activity in other fields (*e.g.*, machine learning). Eventually, the dataset contains the developers' social media links (*e.g.*, Twitter, LinkedIn, and personal websites), which can be used to investigate their shared content on social media.

The remainder of this thesis is organized as follows. After discussing the related work in chapter 2, we discuss the methodology and the five-stage pipeline in chapter 3. In chapter 4, we present the results of our investigation and afterwards discuss our findings. In chapter 5, we follow up with threats to validity. We conclude in chapter 6.

2

Related Work

Profiling developer expertise has recently gained considerable attention in research. Some researchers have focused on investigating developer expertise in open-source software communities. For instance, Saxena *et al.* presented a method to create a detailed technology skill profile of developers based on their contributions to *GitHub* repositories [7]. Zhao *et al.* proposed a ranking metric network learning framework for finding experts [10]. They focused on users' quality relative to given questions and their social relations. Furthermore, they developed a random walk based learning method with recurrent neural networks to match the similarities between a user's question and historical questions posed by other users. Guo *et al.* recommended an answer provider model, where a question is given as a query, and a ranked list of users is returned according to the likelihood of answering the question [1]. Hauff *et al.* proposed a pipeline that automatically suggests matching job advertisements to developers, based on extracted signals from developers' activities on *GitHub* [9].

Several methods are reported in the literature to investigate developer expertise on community question answering (CQA) sites. For instance Zhang *et al.* defined a model to identify developer expertise in a CQA site [13]. Zhao *et al.* considered the problem of expert finding from the viewpoint of missing value estimation [15]. To improve the performance of expert finding in CQA systems, users' social networks are considered in the user model. They analyzed the missing value of the rating matrix between questions and users with a graph-regularized matrix completion algorithm. Zhou *et al.* proposed a topic-sensitive probabilistic model that finds experts from CQA sites by considering the topical similarity among users and link structure [16]. Yung *et al.* investigated the challenge of expert finding with the Topic Expertise Model (TEM)[17]. The probabilistic generative TEM jointly modeled topics and expertise by integrating textual content model and link structure analysis. Bouguessa *et al.* approached a method to automatically identify authoritative actors in CQA sites [14]. They evaluated developer expertise based on the provided number of best answers and multiple algorithms for estimating ranking scores. Liu *et al.* investigated the relative expertise score of users CQA sites [12]. The study focused on the implicit pairwise comparison between two users that participated in the best answer selection. Zhou *et al.* investigated how developers become experts in software projects [18]. They concluded that developer productivity in terms of the number of tasks per month increases with project time.

Unlike previous studies, some research studied developer expertise cross-platform. The challenging part of profiling developer expertise cross-platform is to link developer identity. Mo *et al.* described a tagging-based approach to identity linkage across software communities [2]. The essential idea of the approach is to use skills (measured by tags), usernames, concerned topics of developers as hints, and a decision tree-based algorithm to link user identity. Liu *et al.* proposed HYDRA, a solution framework, that models heterogeneous user behavior for cross-platform identity linkage [3]. Recently, Zhang *et al.* identified profiles by matching the hash (*i.e.*, MD5) of email addresses from public active users of *Stack Overflow* and *GitHub* [4]. Kouters developed an identity matching algorithm that matches identities and email addresses that belong to the same individual [22]. Yan *et al.* proposed an approach to profile developer expertise across software communities by the heterogeneous information network (HIN) analysis [11]. The HIN is first built by analyzing developer activity in various communities, in order to estimate the proximity of developer and skills with their relation. Vasilescu *et al.* investigated the interaction between *Stack Overflow's* activities and development process, reflected by code changes committed to *GitHub* [5]. Huang *et al.* proposed CPDScorer, which models and scores the programming expertise of developers through mining heterogeneous information from both CQA sites and Open-Source Software (OSS) communities [6]. CPDScorer analyzes the answers posted in CQA sites and evaluates the projects submitted in OSS communities to assign expertise scores to developers, considering both the quantitative and qualitative factors. Venkataramani *et al.* found the most frequent terms on *GitHub* and mapped them to question tags found on *Stack Overflow* [8]. Xiong *et al.* proposed an approach to mine developer behavior across *GitHub* and *Stack Overflow* [20]. The identity linkage is made through a CART decision tree, leveraging the features from usernames, user behavior, and writing styles. Sajedi *et al.* investigated the features overlap of *GitHub* and *Stack Overflow*. They analyze the members' core contributions, editorial activities, and influence in the two networks [26].

Further studies investigated task recommendation or code analyses on *GitHub*. Fu *et al.* proposed a novel recommendation approach for task routing in competitive crowdsourced software development [21]. Melnik *et al.* presented a matching algorithm based on a fixpoint computation that is usable across different scenarios [23]. The algorithm takes two graphs as input and produces a mapping between corresponding nodes of the graphs. Shi *et al.* studied the relevance search problem in heterogeneous networks, where the task is to measure the relativity of heterogeneous objects [24]. Ying *et al.* proposed a reviewer recommendation approach that simultaneously considers developer expertise and authority on pull-requests in *GitHub* [19]. Dabbish *et al.* contributed to the body of knowledge on social coding by investigating the network structure of social coding in *GitHub* [25].

A series of recent studies have indicated that lack of developer knowledge in the domain of cryptography has led to many software vulnerabilities [27, 31,32]. For instance, Hazhirpasand *et al.* conducted a study on how developers perform in using cryptographic APIs. On average, 2.5 out of 3.9 crypto uses in each project are not secure, and developers have considerable difficulties using more than half of the APIs [27]. Nadi *et al.* surveyed 11 developers who asked crypto-related questions on *Stack Overflow*, as well as 37 developers who used Java cryptography APIs. They concluded that developers are confident in selecting the right cryptography concepts, but they have difficulties in correctly using certain cryptographic algorithms. They found out that crypto APIs are generally too low-level, and developers prefer more task-based solutions [31]. Shuai *et al.* created a prototype system (*i.e.*, Crypto Misuse Analyzer), which can efficiently identify the crypto misuse vulnerabilities [32]. They concluded that more than half of the analyzed Android applications have cryptographic misuse vulnerabilities. It is of great interest to find a relation between crypto developer expertise and crypto developer contribution to online sources.

To conclude, there exists research that profiles developer expertise cross-platform. To the best of our knowledge, no prior study has examined developer expertise across platforms in the domain of cryptography.

3 Methodology

3.1 Steps

The objective of this study is to investigate to what extent cryptography experts on *Stack Overflow* employ cryptography on their open-source projects on *GitHub*. In order to meet this objective, we defined the following pipeline:

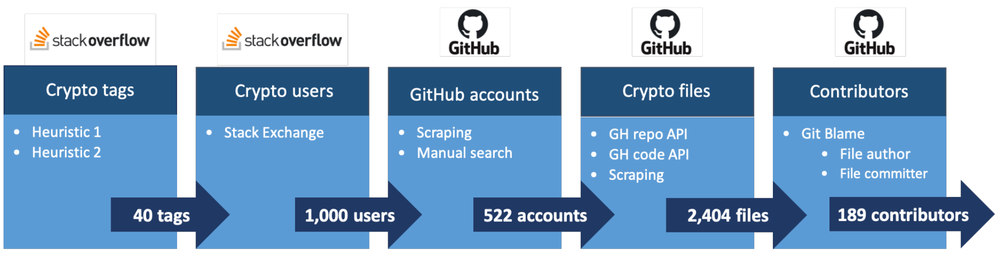


Figure 3.1: A pipeline to identify crypto experts on *Stack Overflow*, and check their crypto contributions in open-source projects on *GitHub*

In each phase of the pipeline, we experienced various challenges. In the following, the challenges, as well as the pipeline, are explained in detail.

3.1.1 Crypto Tag

We assumed that crypto experts participate in crypto discussions on *Stack Overflow*. To identify crypto-related discussions, we focused on the tags that are attached to a question. Meier conducted a study on finding frequent topics on *Stack Overflow*. The author used an approach to identify crypto-related tags on *Stack Overflow*, which we explain in the following. All discussions containing the base tag, *i.e.*, "cryptography" were extracted with the help of the Data Explorer platform (*Stack Exchange*).

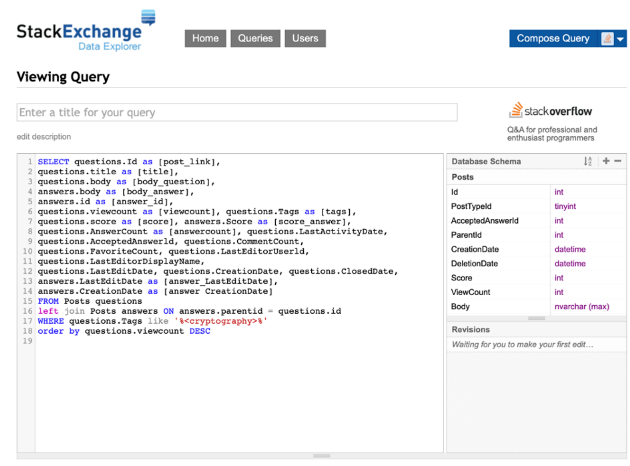


Figure 3.2: Query crypto-related tags from *Stack Overflow*

The query returned 11,130 discussions from *Stack Overflow*, which contained 2,184 tags (candidate tags) that appeared beside "cryptography". Nevertheless, not all candidate tags are crypto-related (*e.g.*, language tags). To identify crypto-related tags, two heuristics H1 and H2 were employed. The first heuristic (H1) investigates to what extent a candidate tag is exclusively associated with the base tag, *i.e.*, "cryptography". Therefore, the number of posts containing a candidate tag and "cryptography" are divided by the number of posts containing a candidate tag. H1 returns a value between zero and one; the nearer the value is to one the more related it is with "cryptography". The first heuristic causes a problem when a candidate tag is used only once in the whole *Stack Overflow* dataset. Although H1 equals one, it is not significant and a second heuristic (H2) is needed. In H2, the posts containing a candidate tag and "cryptography" are divided by the posts containing "cryptography". For example, if H2 returns a value of 0.01, only 1% of discussions use the candidate tag with the base tag, *i.e.*, "cryptography". After a number of observations, the researcher chose the following values for the two heuristics, H1 (*i.e.*, 0.025) and H2 (*i.e.*, 0.005). The candidate tags that their two heuristic values are above the specified threshold are considered as crypto tags. In total, the tag analysis returns 40 crypto-related tags (see Table 3.1).

Table 3.1: Selected tags and their frequencies on *Stack Overflow*

Tag	Freq.	Tag	Freq.
3des	384	keystore	3256
aes	6586	md5	6682
bouncycastle	2639	openssl	16093
3des	384	keystore	3256
aes	6586	md5	6682
bouncycastle	2639	openssl	16093
cng	141	pbkdf2	339
crypto++	707	pkcs 7	414
cryptoapi	474	pkcs11	678
cryptographic-hash-function	74	private-key	1565
cryptography	11130	public-key	1375
cryptojs	954	public-key-encryption	1797
des	680	pycrypto	988
diffie-hellman	315	rijndael	482
digital-signature	3167	rsa	5905
ecdsa	361	salt	1879
elliptic-curve	371	sha	1511
encryption	41971	sha1	2700
encryption-asymmetric	594	sha256	1813
encryption-symmetric	757	smartcard	2167
hash	39886	x509	2064
hmac	1329	x509certificate	3287
jce	554	xor	2514

3.1.2 Crypto User

In this study, we are interested in the developers who provided crypto accepted answers on *Stack Overflow*. To do so, we wrote a query to fetch developers who had at least ten accepted answers in discussions where the crypto-related tags (*i.e.*, 40 tags) were used.

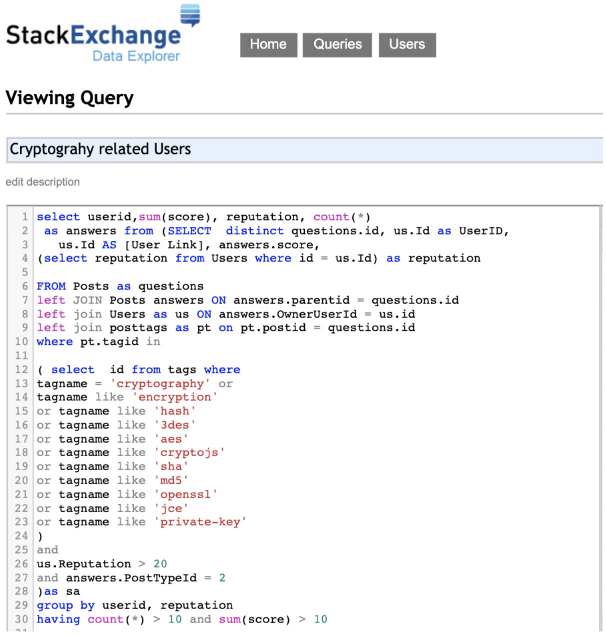


Figure 3.3: Query crypto developers from *Stack Overflow*

Figure 4.7 shows the query to fetch developers who contributed to crypto discussions on *Stack Overflow*. The query returned 1,000 developers who contributed to more than 10 crypto accepted answers. Additionally, in the query, we filtered users with crypto score above 10 (*e.g.*, one upvote on a crypto answer) and a reputation above 20. The query returned the following information: users' unique identifier and their crypto activities (crypto score, reputation, and number of crypto accepted answers). The data is stored in a MySQL relational database. The reputation is calculated by the sum of upvotes from activities such as questions and answers. A crypto accepted answer is accepted by the asker in the area of cryptography (40 crypto-tags). The crypto score is the total number of upvotes of a developer's crypto accepted answers.

Due to the limitations of the resources of Stack Exchange, we had to split the query into multiple queries with fewer criteria. Finally, this step provides us with 1,000 users (crypto users) who met our criteria.

3.1.3 GitHub Account

The users of *Stack Overflow* can share their social media addresses (*e.g.*, Twitter, *GitHub*, and personal websites) on their profile. however, the users' profile information was not available on the Data Explorer platform (*Stack Exchange*). In this phase, we automatically and manually scraped profiles of the 1,000 *Stack Overflow* crypto users, who contributed to at least 10 accepted crypto answers in order to check if their *GitHub* page is available.

```
for Id in UsersID:
 # scrap Profile links
 page = requests.get('https://stackOverflow.com/users/' + Id)
```

```
soup = BeautifulSoup(page.content, 'html.parser')
profileClass = soup.find(class_="list-reset grid gs8 gsy fd-column fc-medium")
profileLink = profileClass.find_all(class_="url")
aboutMeLinks = []
aboutMeClass = soup.find(class_="grid--cell mt16 fs-body2 profile-user--bio")
aboutMeLink = aboutMeClass.find_all("a")
profileLinks = [Id]
for link in profileLink:
profileLinks.append(link.get('href'))
for link in aboutMeLink:
aboutMeLinks.append(link.get('href'))
writer.writerow({"UserId": profileLinks[0], "twitter": profileLinks[1],
     "\GH": profileLinks[2],"webPage": profileLinks[3],
     "AboutMe": aboutMeLinks})
```

Listing 1: Snippet scraping profile links

We employed the code snippet from Listing 1 to extract *GitHub* links from the developer profile on *Stack Overflow*. To parse HTML pages and search for specific elements, we used the BeautifulSoup library. We used the developer tools to identify div classes that display such links on user profile on the *Stack Overflow* profiles. We wrote a Python script to extract information from the identified div classes. In case that the *GitHub* page was not linked on the profile, we manually searched for *GitHub* page. Our manual investigation consist of two phases. First, we examined the other links provided by users (*i.e.*, Twitter and their personal websites). For *Stack Overflow* users with profile pictures, we manually looked for their *GitHub* accounts with google search (*i.e.*,'*Stack Overflow*-full name' + *GitHub* "). If the *GitHub* profile pictures were identical to *Stack Overflow*, we considered that as a match.
Finally, we stored the results of our manual and automatic scraping in a MySQL database, (*e.g.*, Twitter, *GitHub*, personal websites, and about me).

```
  select * from cryptouserslinks where twitter like "\%github.com/\%" or
     gitHub like "\%github.com/\%" or webPage like "\%github.com/\%" or
     AboutMe like "\%github.com/\%"
```

Listing 2: Query GitHub links

Listing 2 returned a total of 522 *Stack Overflow* developers whose *GitHub* links were found.

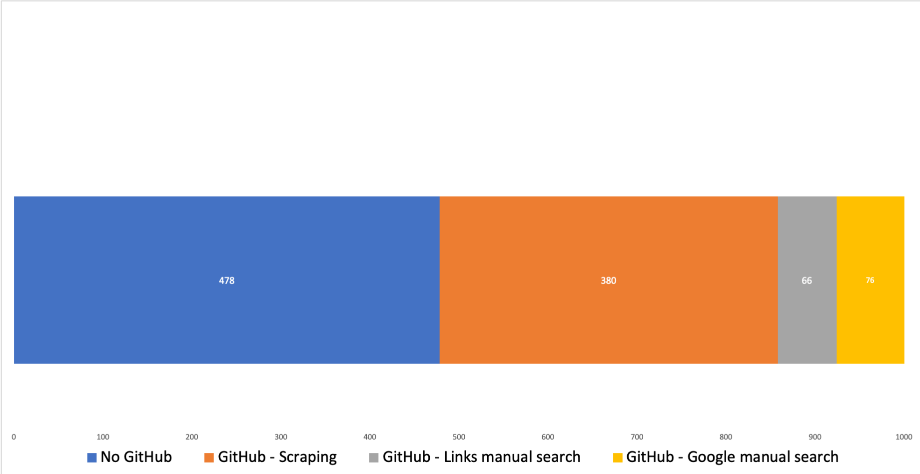


Figure 3.4: Rate of *GitHub* accounts of *Stack Overflow* users

Figure 3.4 shows the status of users' *GitHub* links found by different approaches. Most of the *GitHub* accounts (380) were directly extracted by scraping *Stack Overflow* profiles. Another 142 developers were found by manual search. In the manual search step, 66 users linked their *GitHub* accounts in their social media or personal websites, whereas 76 were found with a google search.

On the whole, our manual and automatic scraping techniques provide us with 522 *GitHub* links, which is 52.2% of the total number of users.

3.1.4 Crypto File

We collected repositories of the 522 *Stack Overflow* crypto users using the GitHub API. In particular, we used PyGithub, which is a Python library that eases the usage of the GitHub APIs for the most common operations, such as repository, issue, and branch requests.

```
for Name in githubName:

   try:
      user = g1.get\_user(Name)
      repositories = user.get\_repos()
      for repositorie in repositories:
         \# Get languages used in a repo
         languages = repositorie.get\_languages()
         languages\_list = []
         for language in languages:
            languages\_list.append(language)
            writer.writerow({"FullName": user.name, "Mail": user.email,
            "UserId": row['UserId'], "Repositorie": repositorie.name,
            "Language": languages\_list})
```

Listing 3: Snippet *GitHub* repositories

We used the Listing 3 code snippet to collect user repository with the *GitHub* Repository API. We fetched *GitHub* username, repository name, and the programming languages used in a repository. We extracted a total of 23,633 public repositories from the 522 *Stack Overflow* developers. In the following, we search for repositories (crypto repositories) that use crypto APIs. We selected seven languages and searched for their crypto libraries (see Listing 3).

```
queryArray = codeSearchArray(cryptolibraries, row['UserName'],
                  row['RepositoryName'], language)
   for query in queryArray:
      for file in g1.search_code(query):
         fileURL.add(file.html_url)
         codeSearchFiles.add(file.path)
      for codeSearchFile, file in zip(codeSearchFiles, fileURL):
         writer.writerow({"UserName": row['UserName'], "RepositoryName":
         row['RepositoryName'], "UserId": row['UserId'],
         "Language": row['Language'],"CodeSearchFiles": codeSearchFile,
         "FileURL": file})
```

Listing 4: Snippet crypto repositories

In Listing 4, we extract crypto repositories using the *GitHub* Code Search API. The method codeSearchArray builds a search query for a given user repository, a language, and their crypto libraries (see Listing 3).

For every search query, we sent a request to the Code Search API. Due to the API limitations, every query searches a single crypto library. The Code Search API returns 812 crypto repositories (*i.e.*, containing cryptographic APIs) out of the 23,633 repositories. A challenging part was that the Code Search API currently does not support exact matches. As a result, we retrieved some repositories that do not use crypto APIs. Therefore, we wrote a regex script to find the exact matches of crypto usages. Finally, the regex script returned a total of 2,404 crypto files.

We experienced some challenges when using the *GitHub* APIs. Due to the *GitHub* API rate limit, user repositories collection took 142 hours for the 23,633 repositories. The Repository API rate limit for authenticated users is 5,000 requests per hour. However, the crypto file collection was much slower as the Search API only allows 30 requests per minute for authenticated requests.

3.1.4.1 Crypto Libraries

Python	Ruby	C	C++	Java	C#	Javascript	Rust
from passlib.	require 'rbnacl'	include "tomcrypt_	include <botan/	Java.security	using Org.BouncyCastle	require("crypto")	use octavo
import passlib	require 'digest'	include <tomcrypt.h>	include "cryptlib.h"	Javax.crypto	using Sodium	aes.js	use recrypt
import pbkdf2_sha256	require 'openssl'	include "paillier.h"	include <cryptlib>		using System.Security.Cryptography	rsa.js	use ring
import nacl	require 'bcrypt'	include "rsa.h"	include "aes.h"		using PCLCrypto	hash.js	use crypto
from nacl		include "x509.h"	using CryptoPP::			nacl_facto	use openssl
import hashlib		include "crypto_"	include <des.h>			ry.js	use rustls
from hashlib		include <openssl/	include "blowfish.h"			sjcl.js	use md5
from crypto		include <themis/ include <wolfssl/	include "secblock.h"			hashes.js	use blake2
import crypto		include "xxhash.h"	include "eccrypto.h"			require("js-nacl")	use digest
from pyelliptic		include "aes.h"	include <helib/			require('crypto')')	use themis
import bcrypt		include "md5.h"	include "cryptopp/			require('hashes')	
from bcrypt		include "sha1.h"	include <cryptopp/			require('crypto-js')	
		include "sha256.h"	include <openssl/			goog.module('goog.crypt	
		include "blowfish.h"				JSEncrypt()	
		include "des.h				require('jsthemis	

Figure 3.5: Crypto libraries

To find common crypto libraries in each language, we consulted with two crypto experts. Then, we checked the crypto libraries' GitHub page to observe how popular (*i.e.*, star and fork) they are. Finally, we compiled a list of common crypto libraries in each language. Finding each library's APIs required a considerable amount of work and time. Moreover, this approach could produce false positives in our results as developers may use similar class names in their repositories. Therefore, we studied what namespaces one must import to use them. Figure 3.5 shows the list of the namespaces that we consider to look for in users' repositories.

3.1.5 Crypto Contributor

In the last step of the pipeline, we used git blame to get the crypto contributors of the 2,404 crypto files.

```
if currentRepository != row['RepositoryName']:
   dest = path + "/" + currentRepository
   if currentRepository != "":
      shutil.rmtree(dest, ignore_errors=True)
      currentRepository = row['RepositoryName']
      clone = "git clone https://github.com/" + row['UserName'] + "/"
      + row['RepositoryName'] + ".git"
      os.system(clone) # Cloning
   os.chdir(row['RepositoryName'])
   blame = "git blame \"" + row['CodeSearchFiles'] + "\" --porcelain | egrep
   \"author |committer \" | sort | uniq"
   fileAuthor = os.popen(blame).read()
   os.chdir('..')
   writer.writerow({"UserId": row['UserId'], "UserName": row['UserName'],
      "RepositoryName": row['RepositoryName'],
      "FilePath": row['CodeSearchFiles'],"Author": fileAuthor})
```

Listing 5: Snippet crypto file

In Listing 5 we cloned the 812 crypto repositories. We extracted crypto files' authors and committers with the help of git blame. Then we checked whether the *Stack Overflow* developers contributed to the crypto files. To this end, we used *GitHub* User API to collect three key elements: email, username, full name. After fetching the user information with the *GitHub* API, we queried the crypto file contributors in Listing 6.

```
select distinct FileAuthorCrypto.UserId from FileAuthorCrypto.UserId
inner join GithubNames on FileAuthorCrypto.UserId = GithubNames.UserId
where Author like CONCAT(' \%', GithubNames.UserName, '\%') or
Author like CONCAT('\%', FullName, '\%') or Author like CONCAT('\%', Mail, '\%')
group by FileAuthorCrypto.UserId.UserId
```

Listing 6: Query crypto file contribution

In summary, the pipeline returned 1,000 *Stack Overflow* crypto developers, where 522 had a *GitHub* account. These 522 developers contributed to 23,633 repositories where only 812 repositories contained crypto APIs. The 812 crypto repos provided us with 2,404 crypto files.

4
Results and Discussion

In this chapter, we report the obtained results and discuss our findings in detail

4.1 Results

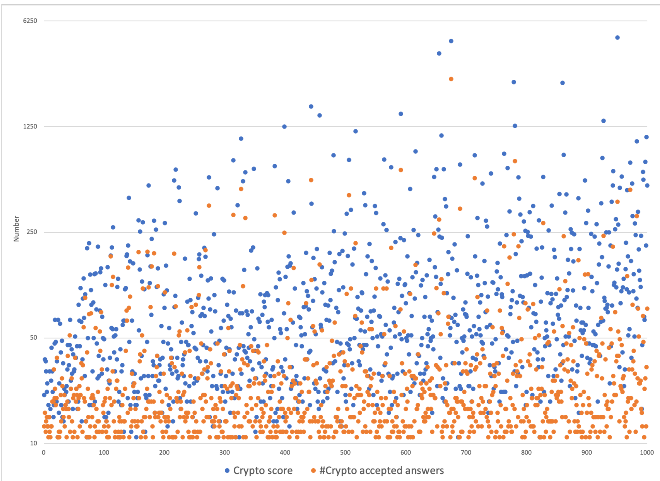


Figure 4.1: *Stack Overflow* crypto score and number of crypto answers

Figure 4.1 shows the distribution of the crypto score and the crypto accepted answers of the 1,000 *Stack Overflow* crypto users. We defined the crypto score term based on the total number of votes that a user received by contributing to crypto accepted answers (40 crypto-tags). In particular, an accepted answer is specified by the asker. Overall, the crypto users had on average a crypto score of 138, whereas the average of the crypto accepted answers is 38. The majority of the crypto developers' accepted answers were between 25 and 250, and such developers' crypto scores were between 10 and 50. According to the scatter plot, we observe that the total number of crypto accepted answers of developers is commonly fewer than the total number of their crypto scores. However, a high number of a crypto score does not necessarily convey that all the crypto accepted answers of a developer received proportional upvotes.

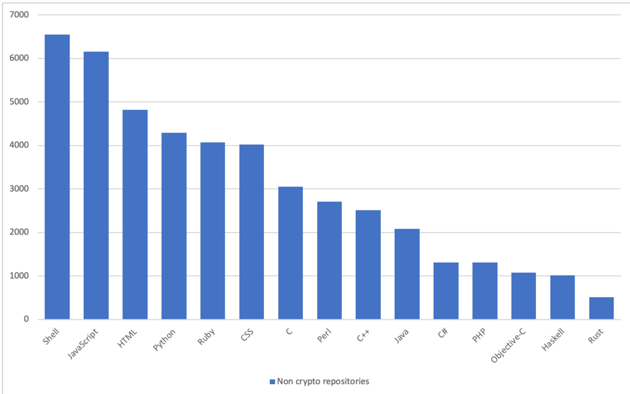


Figure 4.2: Crypto users' top 15 programming languages

Figure 4.2 shows the top 15 languages used in repositories of the users (*i.e.*, 522). Shell was the most used language with 6,556 repositories, whereas Rust is the least used. We decided to study seven programming languages, which are commonly used for application programming (see Figure 3.5).

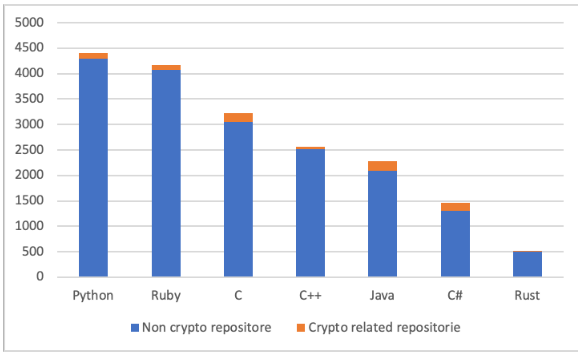


Figure 4.3: Crypto amount of programming languages

We checked all the repositories and found that 189 developers out of 522 developers contributed to at least one crypto file.

Figure 4.3 illustrates the seven selected languages for cryptography, where we distinguished between crypto repositories and repositories without crypto API usage. Overall, crypto APIs were most used in the Java language with a total of 195 crypto repositories, whilst Rust was the least used with only five crypto repositories. Regarding the proportional relationships of the crypto repositories, C# was the most popular language with 10.8% crypto repositories and again Rust was the least popular language with only 1.2% crypto repositories.

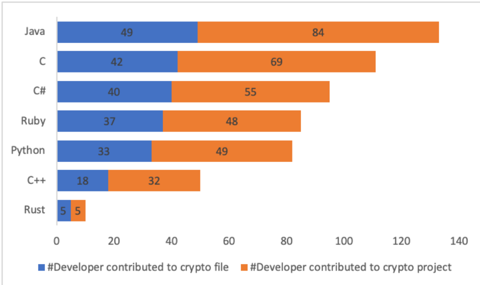
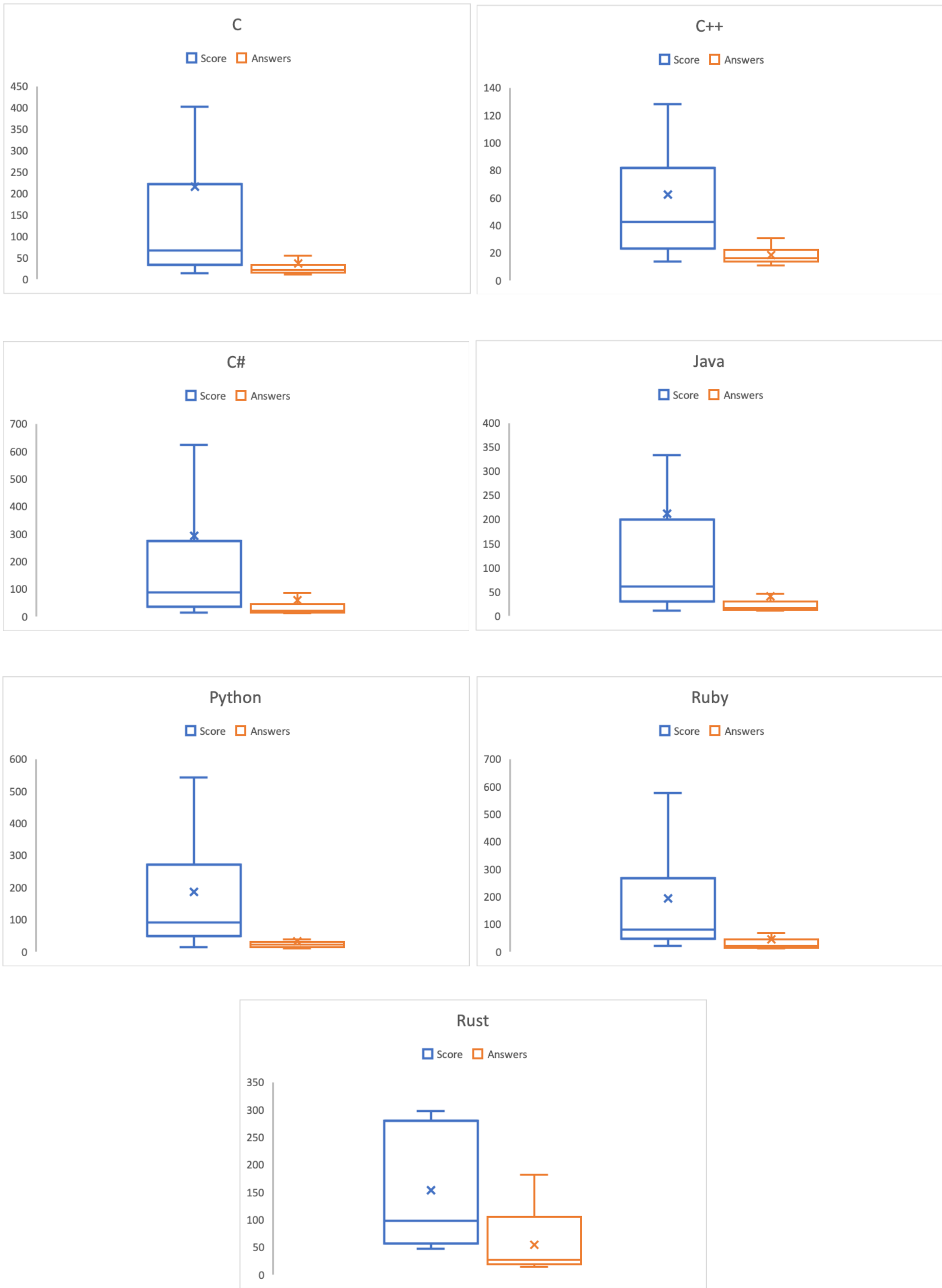


Figure 4.4: Crypto file contribution

Figure 4.4 illustrates the seven selected crypto languages for the 189 crypto developers who contributed to at least one crypto file. The bar chart compared the developers who contributed to crypto files with the developers who contributed to crypto repositories. Java has the largest number of developers who contributed to crypto files and projects. Rust, with only five crypto developers, has the least number of developers who contributed to crypto files and crypto projects. Out of the total 522 *Stack Overflow* crypto developers, 189 have contributed to a crypto file and 343 had at least one repository where others used crypto APIs. In the following, the crypto file contributors are compared based on their crypto activities for each language.

Figure 4.5: Crypto file contributors per language



The seven box-plots show the crypto activities (*i.e.*, crypto score and crypto accepted answers) of the crypto file contributors (*i.e.*, 189). The interquartile range of the box-plots overlap with one another, therefore the distribution of the crypto activities is similar.

Table 4.1: Crypto contributors per language

Language	#Contributors	avg. #file contribution	avg. #accepted answers	avg. #score
C	42	16.3	37	215.2
C#	40	11.4	58.4	292.4
C++	18	23.7	18.4	62.6
Java	49	11.2	40.1	211.7
Python	33	5	32.2	186.3
Ruby	37	2.8	45.7	193.5
Rust	5	3.4	54.8	154.2

Table 4.1 explains the number of contributors based on each language, and their average number of crypto activities.
The crypto contributors of the C# language (*i.e.*, 40 users) had on average the highest number of crypto accepted answers (*i.e.*, 58.4). In C++ the crypto contributors (*i.e.*, 18 users) had on average the smallest number of crypto accepted answers (*i.e.*, 18.4). Hence, the C# crypto contributors had the highest crypto score (*i.e.*, 292.4), whereas C++ had the lowest crypto score. The more accepted answers one had, the more likely it is to get a higher crypto score. Surprisingly the developers with the lowest crypto score and the lowest number of crypto accepted answers (*i.e.*, C++) had the highest number of crypto file contributions. Ruby had on average the lowest number of crypto file contributions (*i.e.*, 2.8).
Based on the seven crypto languages, crypto contributors had visible differences in their crypto activities and their number of crypto file contributions. Consequently, it was of great interest to us to analyze the data with statistical methods.

4.2 Discussion

We analyze the data with the Mann-Whitney U test to observe whether different groups of developers differ in terms of crypto activities (crypto score, reputation, and the number of crypto accepted answers) on *Stack Overflow* and number of crypto file contributions on the *GitHub*.

We compared the data from seven perspectives and looked for significant difference. First, we compared the crypto activities of the 189 crypto contributors with the 332 users who did not contribute to crypto files. Thereafter, we looked only into the 189 crypto contributors. We compared them based on their *Stack Overflow* crypto activities as well as their *GitHub* crypto contribution.

4.2.1 The Mann-Whitney U test

We used the Mann-Whitney U test, which is a nonparametric test, to investigate whether two independent samples were selected from populations having the same distribution. In other words, a randomly selected value of crypto activities from the first group population is considered to be equal to a randomly selected value of the second group population. The Mann-Whitney U test provides two hypotheses, the null hypothesis being met if there is no significant difference between the two groups, which is the case when the calculated p-value is greater than the significance value alpha of 0.05. The alternative hypothesis H1 is accepted otherwise, which means that there is a significant difference between the two groups.

We selected the Mann-Whitney U test, as the data meet the assumptions of the test. We have independent data sets, which can be split into independent groups. The data is not normally distributed. Furthermore, the independence of observations is met, which means that there is no relationship between the observations in each group of the independent data sets or between the groups themselves.

4.2.2 Crypto contributors v.s. users without crypto contributions

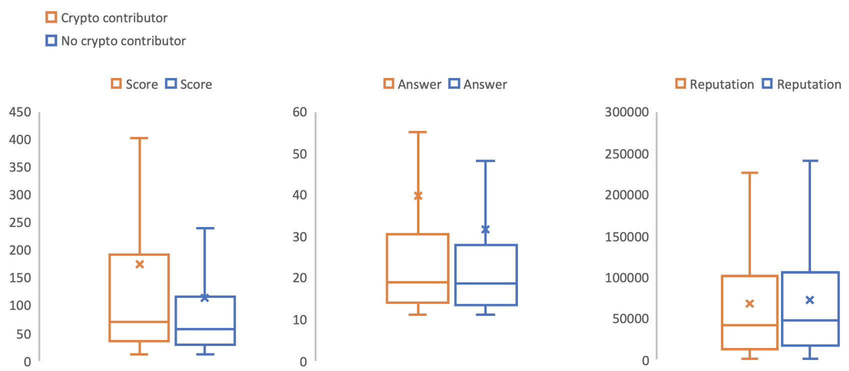


Figure 4.6: Crypto activity

In the first group, we compared the crypto activities between the crypto contributors and the users without crypto contributions. In the crypto score, we observe that the distribution of crypto activities is almost similar. Crypto contributors have on average a crypto score of 173.5, whereas users without crypto contributions have an average crypto score of 113. At the same time, the median values are similar. Hence, it seems as if crypto contributors have a higher crypto score. Regarding the number of crypto accepted answers, the box-plots are distributed equally. On average contributors with no crypto contribution have 31.8 crypto accepted answers. The median values of crypto accepted answers for crypto contributors is 19, and for users without crypto contributions 18.5. Furthermore, the distribution of the reputation looks similar. Crypto contributors have on average a reputation of 67,223.6, whereas users without crypto contributions have an average reputation of 72,215.

In terms of visual analyses, we do not see noticeable differences in the crypto activities between the crypto contributors and users without crypto contributions. The following statistical analyses with the Mann-Whitney U test will illustrate if there is a statistical difference.

Table 4.2: Crypto contributors versus users without crypto contribution

Category	P-value	Accepted hypothesis
Score	0.25	H0
Reputation	0.29	H0
Crypto accepted answers	0.46	H0

Table 4.2 shows that all p-values of the crypto activities are greater than the significance value alpha of 0.05, therefore, the null hypothesis is accepted for the crypto activities. In conclusion, there is no significant difference in terms of *Stack Overflow* crypto activities between crypto contributors and users without crypto contributions.

4.2.3 Crypto contributors

The data analysis focuses only on the 189 crypto contributors. We use the median values (*i.e.*, crypto contribution and crypto activity) to split the 189 crypto contributors into high and low groups.

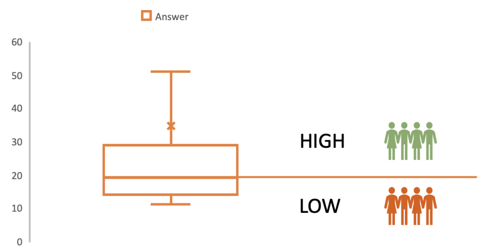


Figure 4.7: Crypto contributors' number of accepted answers

Crypto Accepted Answers: We first used the median number of crypto accepted answers (*i.e.*, 19) to split the crypto contributors into two groups. We compared the number of crypto file contributions for these two groups. The null hypothesis is accepted with a p-value of 0.14. As a result, there is no difference in the number of crypto file contributions based on the number of crypto accepted answers.

In other words, whether someone has a high or low number of crypto accepted answers does not affect the number of crypto file contributions or vice versa.

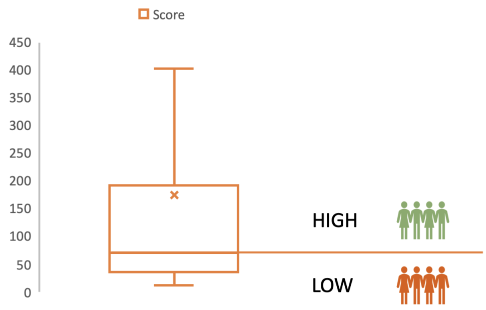


Figure 4.8: Crypto contributors' score

Crypto Score: We split the crypto contributors at the median crypto score of 69. Then, we compared the number of crypto file contributions to these groups. The null hypothesis is accepted with a p-value of 0.64, which conveys that there is no difference in the number of crypto file contributions based on the crypto score.

The number of crypto file contributions are not affected by whether someone has a high or low crypto score.

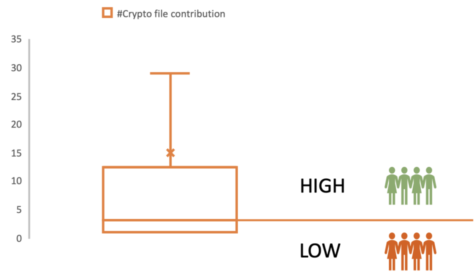


Figure 4.9: Crypto contributors' number of file contributions

Number of Crypto File Contribution: We used the median value of three crypto files to split the developers into two groups.

Table 4.3: Number of crypto file contribution

Category	P-value	Accepted hypothesis
Score	0.75	H0
Reputation	0.74	H0
Answers	0.26	H0

In the comparison of crypto activities, all categories have p-values greater than the significance value alpha of 0.05. Therefore, the null hypothesis is accepted for crypto activities. The number of crypto activities is not affected, whether someone has a high or low number of crypto file contributions.

Crypto Activity: Unlike previous tests, we combined the three factors of crypto activities and checked whether that affects the number of crypto file contributions.

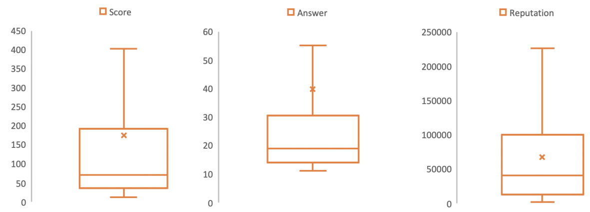


Figure 4.10: Crypto contributors' activities

The null hypothesis is accepted for the number of crypto file contributions with a p-value of 0.39. The number of crypto file contributions are not affected, whether someone has a high or low number of crypto activities

5

Threats to Validity

We discuss possible threats that might affect the validity of this work. We focused on the largest two platforms to study crypto activity and the contribution of users. However adding other online sources such as crypto Stack Exchange or GitLab could afford more data, leading to a more realistic conclusion. We collected 1,000 developers from *Stack Overflow* who contributed to crypto accepted answers, but we only found 522 of those developers on *GitHub*. We believe that other techniques lead to finding more *GitHub* links [2,3,4,22]. Moreover, there might be users with private crypto repositories, which cannot be studied. We looked only into a single *GitHub* account of a *Stack Overflow*-developer, whereas a developer might use multiple accounts. We only studied repositories whose programming language was among the selected languages. Therefore, we have not yet analyzed many repositories in other languages. More importantly, the diversity of crypto libraries in each programming language is debatable. Still, the list of crypto libraries in each language could be increased. We used the git blame command to fetch contributors of a crypto file. Nevertheless, we examined neither commit history nor the lines where crypto APIs were used. Hence, there are possibilities that developers who contributed to crypto files did not contribute to any crypto APIs. All these steps would help this study to project more realistic conclusions.

6
Conclusion

Profiling cryptography developers cross online communities has not previously been studied. To this end, we built a five-stage pipeline to extract crypto developers on *Stack Overflow*, to find their *GitHub* page and identify crypto repositories of such developers, and finally, extract contributors of crypto files.

In view of the visual and statistical analyses, we did not find any significant difference between crypto contributors and users without crypto contributions regarding their *Stack Overflow* crypto activities (crypto score, reputation, and number of crypto accepted answers). In other words, despite a high number of crypto accepted answers, a developer does not necessarily contribute to many crypto files on *GitHub*. The same applies to the crypto score, where a high crypto score does not lead to a higher number of crypto file contributions. Likewise, the higher number of crypto contribution on *GitHub* does not necessarily mean that a developer has a high number of crypto activities on *Stack Overflow*.

Employing more programming languages, crypto libraries, and users may constitute the object of future studies.

7 Acknowledgement

I thank my supervisor, Mohammadreza Hazhirpasand, who has contributed an invaluable assistance.

A

Anleitung zum wissenschaftlichen Arbeiten

The Anleitung consists of the conference paper “Profiling Cryptography Developers”. M. Hazhirpasand, S. Ali, and O. Nierstrasz. Profiling Cryptography Developers Planned for submission to the 28th edition of the International Conference on Software Analysis, Evolution, and Reengineering (SANER’21)

Bibliography

[1] J. Guo, S. Xu, S. Bao, and Y. Yu. 2008. Tapping on the potential of Q&A community by recommending answer providers. In proceedings of the 17th ACM *Conference on Information and Knowledge management.* ACM, 921–930.

[2] W. Mo, B. Shen, Y. Chen, and J. Zhu. 2015. Tbil: A tagging based approach to identity linkage across software communities. *In Software Engineering Conference (APSEC)*, 2015 Asia-Pacific. IEEE, 56–63

[3] S. Liu, S. Wang, F. Zhu, J. Zhang, and R. Krishnan. 2014. Hydra: Large-scale social identity linkage via heterogeneous behavior modeling. In proceedings of the 2014 ACM *SIGMOD International Conference on Management of Data*. ACM, 51–62.

[4] X. Zhang, T. Wang, G. Yin, C. Yang, Y. Yu, and H. Wang. 2017. DevRec: A Developer Recommendation System for Open Source Repositories. *In International Conference on Software Reuse.* Springer, 3–11

[5] B. Vasilescu, V. Filkov, and A. Serebrenik. 2013. StackOverflow and GitHub: Associations between software development and crowdsourced knowledge. *2013 International Conference on Social Computing (SocialCom). IEEE, 188–195.*

[6] W. Huang, W. Mo, B. Shen, Y. Yang, and N. Li. 2016. CPDScorer: Modeling and Evaluating Developer Programming Ability across Software Communities. In SEKE. 87–92.

[7] R. Saxena, N. Pedanekar. 2017. I Know What You Coded Last Summer: Mining Candidate Expertise from GitHub Repositories. ACM, 299–302.

[8] R. Venkataramani, A. Gupta, A. Asadullah, Basavaraju Muddu, and Vasudev Bhat. 2013. Discovery of technical expertise from open source code repositories. *In Proceedings of the 22nd International Conference on World Wide Web*. ACM, 97–98.

[9] C. Hauff and G. Gousios. 2015. Matching GitHub developer profiles to job advertisements. *In Proceedings of the 12th Working Conference on Mining Software Repositories.* IEEE Press, 362–366.

[10] Z. Zhao, Q. Yang, D. Cai, X. He, and Y. Zhuang. 2016. Expert Finding for Community-Based Question Answering via Ranking Metric Network Learning. In IJCAI. 3000–3006

[11] J. Yan, H. Sun, X. Wang, X. Liu, and X. Song: Profiling Developer Expertise across Software Communities with Heterogeneous Information Network Analysis, 2018

[12] J. Zhang, M. S Ackerman, and L. Adamic. 2007. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th International Conference on World Wide Web*. ACM, 221–230.

[13] J. Liu, Y. Song, and C. Lin. 2011. Competition-based user expertise score estimation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 425–434.

[14] M. Bouguessa, B. Dumoulin, and S. Wang. 2008. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge discovery and data mining.* ACM, 866–874.

[15] Z. Zhao, L. Zhang, X. He, and W.Ng. 2015. Expert finding for question answering via graph regularized matrix completion. IEEE Transactions on Knowledge and Data Engineering 27, 4 (2015), 993–1004.

[16] G. Zhou, S. Lai, K. Liu, and J. Zhao. 2012. Topic-sensitive probabilistic model for expert finding in question answer communities. In *Proceedings of the 21st ACM International Conference on Information and knowledge management*. ACM, 1662–1666

[17] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen. 2013. CQArank: Jointly model topics and expertise in community question answering. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM, 99–108.

[18] M. Zhou and A. Mockus. 2010. Developer fluency: Achieving true mastery in software projects. In Proceedings of the eighteenth ACM *SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 137–146.

[19] H. Ying, L. Chen, T. Liang, and J. Wu. 2016. EARec: leveraging expertise and authority for pull-request reviewer recommendation in GitHub. *In Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering*. ACM, 29–35

[20] Y. Xiong, Z. Meng, B. Shen, and W. Yin. 2017. Mining Developer Behavior Across GitHub and StackOverflow. In SEKE. 578–583.

[21] Y. Fu, H. Sun, and L. Ye. 2017. Competition-aware task routing for contest based crowdsourced software development. *In Software Mining (SoftwareMining), 2017 6th International Workshop on*. IEEE, 32–39.

[22] E. Kouters, B. Vasilescu, A. Serebrenik, and M. GJ van den Brand. 2012. Who's who in Gnome: Using LSA to merge software repository identities. In Software Maintenance (ICSM), 2012 28th IEEE *International Conference on Software Maintenance.* IEEE, 592–595.

[23] S. Melnik, H. Garcia-Molina, and E. Rahm. 2002. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In Data Engineering, 2002. Proceedings. 18th *International Conference on Data Engineering.* IEEE, 117–128.

[24] C. Shi, X. Kong, P. S Yu, S. Xie, and Bin Wu. 2012. Relevance search in heterogeneous networks. In Proceedings of the 15th *International Conference on Extending Database Technology*. ACM, 180–191.

[25] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in CSCW. ACM, 2012, pp. 1277–1286

[26] A. Sajedi Badashian, A. Esteki, A. Gholipour, A. Hindle, and E. Stroulia. Involvement, contribution, and influence in GitHub and *Stack Overflow*. In CSSE, 2014

[27] M. Hazhirpasand, M. Ghafari, S. Krüger, E. Bodden and O. Nierstrasz, "The Impact of Developer Experience in Using Java Cryptography," 2019 ACM/IEEE *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Porto de Galinhas, Recife, Brazil, 2019, pp. 1-6, doi: 10.1109/ESEM.2019.8870184.

[28] Capiluppi, Andrea & Serebrenik, Alexander & Singer, Leif. (2012). Assessing Technical Candidates on the Social Web. IEEE Software. 30. 10.1109/MS.2012.169.

[29] Yang, Di & Martins, Pedro & Saini, Vaibhav & Lopes, Cristina. (2017). *Stack Overflow* in Github: Any Snippets There?

[30] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in Android applications," in Proceedings of the 2013 ACM *SIGSAC Conference on Computer Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 73–84.

[31] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in Android applications," in Proceedings of the 2013 ACM *SIGSAC Conference on Computer Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 73–84.

[32] S. Shao, G. Dong, T. Guo, T. Yang, and C. Shi, "Modelling analysis and auto-detection of cryptographic misuse in Android applications," 2014, pp. 75–80.

[33] Shrestha, Amendra & Kaati, Lisa & Johansson, Fredrik. (2013). Detecting multiple aliases in social media. 10.1145/2492517.2500261.

[34] Baltes, Sebastian & Diehl, Stephan. (2018). Usage and Attribution of Stack Overflow Code Snippets in GitHub Projects. Empirical Software Engineering. 10.1007/s10664-018-9650-5.