Maximilian Eibl, Martin Gaedke. (Hrsg.): INFORMATIK 2017,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 2437

Real or Fake? Large-Scale Validation of Identity Leaks

Fabian Maschler, Fabio Niephaus and Julian Risch[1]

Abstract: On the Internet, criminal hackers frequently leak identity data on a massive scale. Subsequent criminal activities, such as identity theft and misuse, put Internet users at risk. Leak checker services enable users to check whether their personal data has been made public. However, automatic crawling and identification of leak data is error-prone for different reasons. Based on a dataset of more than 180 million leaked identity records, we propose a software system that identifies and validates identity leaks to improve leak checker services. Furthermore, we present a proficient assessment of leak data quality and typical characteristics that distinguish valid and invalid leaks.

Keywords: Data quality, Data profiling, Leak validation, Identity leaks

1 Identity Leaks

Large databases of Internet services are targets of hacking attacks. Often, such stolen data is traded in the Dark Web or made public to achieve fame. Subsequent to data breaches, criminals can use personal data, such as email addresses and credit card information, or usernames and passwords for identity theft and other crimes. In response to identity leaks and their threat to users, there are leak checker services, such as the HPI Identity Leak Checker (ILC)[2] or HIBP[3]. With the help of these services, users can easily find out whether their identity data is contained in any published leaks. To provide such a service, leak checkers search and import leak data from the Internet automatically and continuously. Often, they go for quantity and collect leak data unrevised, to quickly enlarge their dataset.

This fully automatic approach comes at a cost: Leak checker databases often contain a lot of false positives, which are invalid entries with no corresponding real identity. Moreover, sometimes hackers pretend to have successfully hacked a service and then spread fake data to harm the reputation of a certain service or to gain attention. If there are no validity checks, also fake leaks end up in leak checker databases. However, leak checkers need clean databases to provide reliable results to their users. To improve the quality of those leak databases and reduce the number of false positives, it is necessary to validate all entries.

[1] University of Potsdam, Hasso-Plattner-Institute, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany, {fabian.maschler, fabio.niephaus, julian.risch}@student.hpi.uni-potsdam.de

[2] https://sec.hpi.de/leak-checker/

[3] https://haveibeenpwned.com/

doi:10.18420/in2017_248

We present a software system that automatically validates entire leak files with millions of entries, as well as single identity records. We cross-validate our approach on 900 manually labeled leak files and achieve an F_1-Measure of 85 %. Experiments on a database with more than 180 million leaked identities detect 10 % of the identities as invalid.

2 Related Work

Jäger et al. propose a leak checker service and explain its functionality [Ja15]. They describe a manual process of collecting leak data from hacker forums, the Dark Web, and file hosting providers or BitTorrent. A basic analysis of password distributions is conducted, which has been previously done by Wang et al. limited to Chinese passwords [Wa15]. Butler et al. focus on tracing a leak and identifying its surface point [BWP16]. The authors address large service providers, such as PayPal, and present concepts to automatically gather leaked data to inform and protect customers. Involving attacked service providers, Nixon summarizes several ideas on vetting leaks [Ni14]. According to that, password policies or measures for uniqueness and existence of usernames and email addresses are useful for the validation of leak data. Other ideas include checking the formatting of credit card numbers or analyzing the distribution of real names. Unfortunately, Nixons summary is missing an evaluation of the described ideas with real data. With our work, we fill this gap with experiments on a real-world dataset and moreover extend the list of effective and efficient validity checks. Recently, our results have been incorporated into the ILC and build the basis for ongoing identity leak research of Jäger and Graupner et al. [Gr16, Ja16].

3 Leak Validation

Our software system decomposes identity leak validation into two subtasks: (i) Validation of a leak file in its entirety based on features of the full dataset and (ii) Validation of a single identity and its username, email address, etc. in detail. Table 1 lists all proposed validation methods.

Tab. 1: Validators for (i) leak files and (ii) single identities.

Validator	Description
(i)	
MLBasedValidator	Validation based on supervised machine learning
LeakDistributionValidator	Analysis of character n-gram distribution
(ii)	
ExistenceValidator	Filter records with empty fields
GenericValidator	Filter special characters with regular expressions on usernames
EmailValidator	Filter general email format (Apache)
EmailSpecificationValidator	Filter provider-specific email regular expressions
MXRecordValidator	Filter domains without MX Record
SMTPValidator	Filter unknown email addresses at SMTP server

3.1 Validation of Entire Leak Files

To identify valid identity leak files, we propose a machine learning approach (MLBasedValidator) based on several features of the entire leak file.

Quantitative Features. The feature set includes, but is not limited to:

- text length of the entire leak
- average line length
- the number of occurrences of special characters or words such as
  - '@' to estimate the number of email addresses
  - '*CREATE TABLE*' or '*PRIMARY KEY*' to identify SQL statements
  - '*qwert*', '*123456*' or '*password*' to represent common passwords

Furthermore, we propose regular expressions to identify HTML tags, email addresses, and common password hashes. Regarding common password hashes, we focus on *sha1*, *sha256*, *sha512* and *md5*. Although leaked password hashes can be a security risk, not every leaked password hash can be cracked and misused. However, even salted password hashes can be guessed if the corresponding password hints are known, as seen in the Adobe leak [Kr13]. Passwords that do not occur in any dictionary are also subject to this risk. In our work, we use hashes as a feature for leak validation, rather than analyzing password strength and security risk of leaked hashes.

Qualitative Features. Typical leak files follow a tabular structure. To measure how similar a leak file is to a CSV-file like tabular structure, we propose a similarity measure for consecutive lines. Two lines are similar if both lines contain at least one common delimiter symbol, such as '*:*', '\t', or '*;*'. Further, the frequency of at least one of these symbols must be the same for consecutive lines.

In computational literature analysis, the task of authorship attribution is to attribute the authorship of a text to the correct author, out of a set of candidates. We propose to transfer features that represent the writing style of an author to features for the different characteristics of valid and invalid leak data. A prevalent feature for authorship attribution are character n-grams, which are tuples of n consecutive characters. Houvardas et al. extract character n-grams and map their frequencies in a histogram for authorship attribution [HS06]. We transfer this technique to leak analysis and define only two classes of leaks: each leak file is either valid or invalid. We choose the Adobe leak as a reference/ground truth dataset for valid leak files, since it is an international leak file with a huge number of identities and thus suits best for general conclusions. Considering the local part of all email addresses, we analyze single character, 2-gram, and 3-gram distributions. As a result, we extract three histograms that represent the relative frequency of each n-gram in email address local parts of the Adobe leak.

To validate a leak file, we count character n-grams in this leak and compare their relative frequencies to our reference frequency distribution. If the difference of the relative frequency is higher than a parameter *epsilon*, this n-gram deviates. A higher *epsilon* allows more variation of n-grams. A second parameter *coverage* specifies, how many n-grams can deviate at most for a leak to be considered valid. We count only those n-grams, which appear in the expected **and** the observed distributions. For higher percentages of *coverage*, less leaks are classified valid, since they have to be more similar to the reference leak.

If the n-gram distribution of a leak is classified invalid, it is too different from the reference and we invalidate all identities of that leak. Because n-gram distributions are not reliable for small leak files and they might differ from the reference distribution even for valid leak files, we skip this check for small leak files.

With this *LeakDistributionValidator*, we can successfully detect randomly generated leak files in which the distribution of n-grams differs from the expected distribution. However, leaks of online services in a specific country differ from more general or international leaks. Thus, the leaks which are classified as invalid by this check should be manually inspected before all identities are invalidated. Furthermore, using multiple valid leaks to extract reference distributions would make the analysis more robust.

3.2 Validation of Single Leak Identities

A leak that is classified valid, can still contain a great amount of invalid identities. In this section, we propose methods to filter out single fake identities.

Generic Validations. When checking the validity of an identity, there are some general checks that can be performed. The easiest validation, implemented with the *Existence-Validator*, invalidates identities missing required information, such as username or email address. Furthermore, a username usually cannot contain special characters, such as control sequences or white space. The *GenericValidator* implements this specification with configurable regular expressions. In a similar fashion, email addresses are subject to a well-defined format, described in RFC5322 and extended by RFC6531 [Re08, YM12]. Our *EmailValidator* performs these general checks with regular expressions and an email address validator provided by Apache Commons[4]. Depending on the dataset available, other generic checks, such as credit card validation, can be performed in a similar way.

Provider-Specific Email Address Validation. Hackers often try to attack services that are used by millions of users. The biggest data breaches usually contain many email addresses from different email providers. However, the majority of email addresses is from popular email providers, such as Gmail, Yahoo or Outlook. These providers allow only certain characters to be in the local part of email addresses. For example, Gmail allows only letters (a-z), numbers (0-9), and a period (.) to be in the local part. The length of such local parts is limited between 6 and 30 characters and they must start and end with a letter or a number. Gmail uses @gmail.com and @googlemail.com in some countries and allows users to define address aliases, which contain a plus (+). All this information can be used to validate a given Gmail address. For this purpose, we compiled specifications in the form of regular expressions for 29 popular email providers. For example, the Gmail specification is the following:

```
^[a-z0-9]([a-z0-9.+]){4,28}[a-z0-9]@g(oogle)?mail\.com$
```

The *EmailSpecificationValidator* checks, if the domain of an email address matches any of the specified regular expressions. In this case, the address also has to match the specified regular expression for the local part. Otherwise, the identity is declared invalid and needs no further checking.

MX Record Validations. Checking email address specifications is already effective to reveal invalid entries. A more advanced technique is to check an email's domain for MX

[4] https://commons.apache.org/proper/commons-validator/apidocs/org/apache/commons/validator/routines/EmailValidator.html

records. An MX record is a DNS entry and specifies the mail servers that handle incoming emails for a given domain. If no MX record is present, there is no way an email can be sent to this domain. Thus, the email address is invalid. This is the case, if a domain has been deleted or if an email address has been generated randomly.

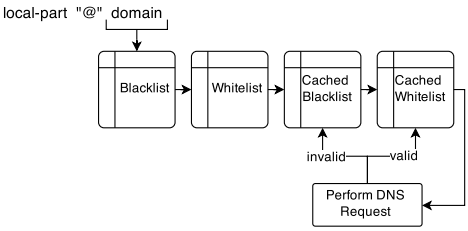

Fig. 1: MX Record Check Process

The *MXRecordValidator* verifies the existence of a valid MX record for a given email address domain. Performing DNS requests is comparatively costly in terms of time. Analyzing large datasets would result tremendous runtimes. To speed up validation, our implementation uses caching and multi-threading. Figure 1 visualizes the process of checking a domain. First, we check if a domain is present in a blacklist or whitelist. The blacklist contains domains of popular providers for disposable email addresses, whereas the whitelist contains domains of popular email providers such as Gmail or Yahoo. The *MXRecordValidator* then checks, if a domain is in a cached list. The cache is split into a cached blacklist and a cached whitelist. A DNS request is performed only if a domain is not contained in any of these lists. Thereby, each domain is checked only once and added to the cached blacklist or whitelist afterwards. This keeps the number of DNS requests as low as possible.

SMTP Validations. Although, the Simple Mail Transfer Protocol (SMTP) was first defined in 1982, it is still used to deliver emails today [Cr82]. RFC5321 specifies all commands that are required to send an email via SMTP [Kl08]. To find out, if an email account exists on the server, we pretend to send an email to that address, but before sending email data, we quit the process. As seen in Listing 1, the mail server replies with code 250 if a recipient is valid or with 554 if it is unknown. Therefore, this is used to check for the existence of an email address.

RFC5321 compliant mail servers allow 100 recipients [Kl08]. Hence, we can validate 100 email addresses per connection to an SMTP server. Email providers usually use more than one mail server for redundancy purposes. Thereby, more than one connection to each email provider can be established to further speed up the validation process. We have implemented this test and made it available through *SMTPValidator*. Unfortunately, performing this check is even a lot more time-consuming than an MX record check.

```
220 mail.domain.de ESMTP
HELO domain.de
250 mail.domain.de
MAIL FROM: <sender@domain.de>
250 2.1.0 Ok
RCPT TO: <valid@domain.de>
250 2.1.5 Ok
RCPT TO: <invalid@domain.de>
554 5.7.1 <invalid@domain.de>: Recipient address rejected: user unknown
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

List. 1: SMTP communication example to validate the existence of an email address

4 Large-Scale Leak Validation

Hackers usually capture thousands of login and account details in data breaches. For this reason, databases for identity leak checkers are likely to contain millions of entries. To deal with this huge amount of data, we use the producer-consumer pattern and implement two different threads that communicate via a blocking queue. A producer thread fetches one entry after another from the database and adds it to the queue. A consumer thread does more complex operations, such as validating multiple regular expressions or performing SMTP checks. For this reason, our system uses a single producer thread and multiple consumer threads, as shown in Figure 2. This separation parallelizes the validation of a huge amount of identities and speeds up the process.

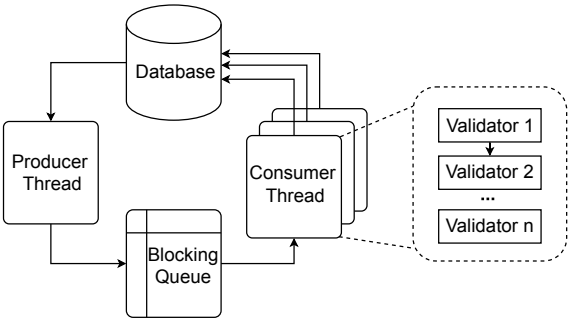

Fig. 2: Producer-consumer architecture for parallel validation of identities.

All identities are analyzed and validated in the consumer threads, each performing a list of validations. The order of validations should filter identities early to minimize processing time. For example a consumer thread could process generic validations, email specification check, MX record checks, SMTP checks, and character n-gram distribution analysis.

5 Evaluation

In this section, we evaluate our work on the analysis of leak data quality and identification of fake identities. Therefore, we evaluate our proposed methods on a sample of the ILC database. An overview of the dataset, for example the number of accounts, domains or common passwords, is available online[5].

The evaluation starts with evaluation leak files validation and continues with the analysis of single leak identities. Here, we distinguish between generated data, a sample of the ILC database and a full analysis with a subset of methods.

5.1 Leak File Validation

For the quantitative analysis described in Section 3.1, we use supervised machine learning with decision tree classifiers. Decision trees allow very time-efficient classification, which is important for our use case with many leak files to check. We evaluate our model with a 5-fold cross-validation on a balanced test set of 450 valid and 450 invalid identity leak files from May 2014. These files stem from leak posting websites, such as `www.leakedin.com` and have been manually labeled as valid or invalid. Valid identity leak files contain real identities including email addresses, passwords and usernames. Invalid identity leak files consist of fake identities or do not contain identities at all, such as log files, configuration files, HTML files, or link lists.

In our experiments, a random forest classifier with 30 trees, each constructed while considering 5 random features out of all 15 features, performs best. As a result, we achieve an F_1-Measure of 85 %, with identical results for precision and recall. While we found no other features or classifier models to improve this further, we were able to reduce the number of used features, still achieving a high F_1-Measure. A subset of only 3 features consisting of '@' signs, *md5* hashes, and the similarity of consecutive lines, achieves an F_1-Measure of 81 % using an alternating decision tree.

5.2 Identity Validation

For the evaluation of single identity validation, we consider a sample of the ILC database comprising several leak files, containing almost 1.5 million identities with email addresses and usernames. We discovered 725 909 identities with duplicate email address in the entire dataset (inter-leak). Analyzing each leak separately, there are overall only 26 644 identities with duplicate email addresses (intra-leak). We assume, the latter results from services that use usernames instead of email addresses for authentication or do not check for duplicate accounts. Comparing the large difference between intra- and inter-leak duplicates, we

[5] `https://sec.hpi.uni-potsdam.de/leak-checker/statistics`

assume that the database contains identical leaks, probably imported from different sources. Since such a non-fuzzy de-duplication can be easily achieved with appropriate database constraints, we do not further discuss this kind of invalid data.

Table 2 shows the results of our approach using all validators except the *SMTPValidator* for time-efficiency reasons. The *MXRecordValidator* uses 329 whitelisted domains and 778 blacklisted temporary email providers.

Tab. 2: Results on a sample of the ILC database with a runtime of 90.92 s. In total, 1 455 230 entries were checked, which represent 100 % in the table.

Description	Absolute Value	Relative Value %
LeakDistributionValidator	69 791	4.80
ExistenceValidator	828 363	56.92
GenericValidator	35 859	2.46
EmailValidator	23 719	1.63
EmailSpecificationValidator	54 303	3.73
MXRecordValidator	43 536	3.00
Blacklisted domain	9 945	0.68
Total fake data	237 153	16.30

Our software identifies 16.30 % of the dataset as invalid data. As it turns out, even the Adobe leak contains a fair amount of invalid entries and so do many other valid leaks. In fact, 56.92 % of all identities have been identified as incomplete by the *ExistenceValidator*, though this does not qualify them as invalid. Apparently, even large online services do not validate all user-related information and several leaks are incomplete.

Not all web services verify email addresses by sending an email, therefore a higher number of email addresses do not meet their provider's specifications. Quite a few email addresses have domains with no or an invalid MX record for probably the same reason. With the *MXRecordValidator* it takes only 91 s to process 1.5 million identities with all domains cached but 15 min with an empty cache. The cache reveals that 18 143 domains are blacklisted and 79 492 entries are whitelisted. This sums up to almost 98 000 distinct DNS requests, which equals just 6.71 % compared to the total number of identities. The user defined blacklisted domains from temporary email providers result in flagging 9 945 (0.68 %) identities as invalid.

Moreover, all identities from three leak files have been marked invalid, with the *LeakDistributionValidator*. Indeed, these leaks look suspicious, because most of the email addresses seem to be randomly generated while skimming over the leak. For example, we identify suspicious patterns of common firstname-lastname combinations. This result should still be handled carefully, since very specific leaks deviate from the expected distribution. For

example, leaks of Chinese messaging services and e-mail providers[6] use telephone numbers for registration. For this reason, there is a higher frequency of numbers in such leak files.

To evaluate the robustness of this method, we introduce a fake leak created with GenerateData[7]. Although, all usernames and email addresses are well-defined and do not appear suspicious at first sight, this leak is detected invalid. We tested different parameter configurations of the *LeakDistributionValidator* and find that for longer n-grams the parameters have to be much smaller to detect invalid leaks. Although the analysis takes more time with longer n-grams, the results are more robust and can be fine-tuned this way. We recommend to either manually double-check detected leaks or to use these results to downgrade the leak's relevance and then indicate to users that this leak might not be valid.

Finally, we run parts of our software on the full dataset of the ILC database. To allow a reasonable runtime on this huge dataset, we enable only a subset of validators. Table 3 shows the corresponding results.

Tab. 3: Number of flagged (invalid) entries per validator on the full ILC database with a runtime of 46 min. In total, 186 952 311 entries were checked, which represent 100 % in the table.

Description	Absolute Value	Relative Value %
GenericValidator	4 349 267	2.33
EmailValidator	3 242 407	1.73
EmailSpecificationValidator	3 852 816	2.06
Total fake data	11 444 490	6.12

Compared to results of the smaller sample, the relative frequencies are quite similar. However, the total number of invalid identities is proportionately larger, because the full dataset contains almost 187 million identities. The subset of validators identifies more than 11 million fake identities in about 46 min runtime. Extrapolating the results from the smaller sample to the entire ILC database, all proposed validators would identify approximately 10 % of the entries as invalid.

To conclude, we give some examples of detected invalid identities. Table 4 exemplarily shows a few usernames and email addresses that were flagged as invalid. We allow letters of any language, numbers, dash, period, underscore, and '@' in usernames, so these examples are evident. Some invalid email addresses look like mismatches, typing errors or even hacking attempts. In Table 5, there are email addresses with invalid email specifications. We have double-checked them manually and they indeed cannot be valid addresses according to the corresponding email providers. Furthermore, Table 5 lists a few examples for email addresses with invalid domains. In fact, these domains are either missing an MX record or do not exist at all.

[6] `http://www.qq.com/` and `http://www.163.com/`

[7] `http://generatedata.com/`

Tab. 4: Examples for invalid usernames and malformed email addresses.

Invalid usernames	Malformed email addresses (non RFC5322 compliant)
??Frank??	aqwa242@comcast,net
Ø¨Ø³Ø§Ù§Ù... Ù...Ù^Ø³Ù%ه	anonymous3000
.::009::.	<body onload="document.write('hacked...
winner 12378	non g 1982
(vitimin-z)	taqwa65@yahoo.comr

Tab. 5: Examples for email addresses that violate providers' specifications or that have domains with no or only an invalid MX record.

Addresses violating provider's specification	Addresses domains with no or invalid MX records
mon.ino.(...)r.beto.ia@gmail.com	big@foo.bar
asdf@gmail.com	kashif@alrahmahsoft.com
lad@yahoo.com	tra1@ac.com
c@msn.com	asjklddfasjkl@dfassfddfas.com
star.zhang@163.com	sd@ut.yu

6 Conclusions and Future Work

In this work, we study the problem of identity leak validation and propose a general approach to validate entire leak files, as well as single leak identities. Based on supervised machine learning, we classify leak files as valid or invalid and on a more detailed level, we analyze how usernames and email addresses differ in real and fake leaks. We present a time-efficient implementation for validating email addresses based on MX records and analyze character n-gram distributions to identify generated leaks. Experiment results demonstrate that our approach is suited for real-world leak checker databases with more than 180 million leaked identities. We reveal approximately 10 % of the data in the examined ILC database as fake identities or mismatched information.

Interesting future work includes the analysis of other attributes of leaked identities, such as password hashes. Checking individual passwords for password policy compliance or the frequency distribution of simple and hard passwords could be next steps. However, password policies might change over time and the frequency distribution of passwords can be service-dependent. In the Adobe leak, for example, many passwords include the word '*adobe*'. The analysis of frequency distributions is also interesting for other information that is not devised by users, such as real names, email addresses containing real names, or birthdates. For example, the frequency distribution of births is far from uniform, with the highest peak nine months after Christmas.

Acknowledgment

This work has its origins in the Master's seminar *Dark Web Monitoring and Analysis of Leak Data* at the Hasso Plattner Institute at the University of Potsdam. We would like to thank our seminar supervisors David Jäger and Hendrik Graupner for numerous discussions on this topic.

References

[BWP16] Butler, Blake; Wardman, Brad; Pratt, Nate: REAPER: an automated, scalable solution for mass credential harvesting and OSINT. In: APWG Symposium on Electronic Crime Research (eCrime). IEEE, pp. 1–10, 2016.

[Cr82] Standard for the format of ARPA Internet text messages, RFC 822 (Internet Standard). http://www.ietf.org/rfc/rfc822.txt, Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148.

[Gr16] Graupner, Hendrik; Jaeger, David; Cheng, Feng; Meinel, Christoph: Automated Parsing and Interpretation of Identity Leaks. In: Proceedings of the ACM International Conference on Computing Frontiers. CF '16, ACM, New York, NY, USA, pp. 127–134, 2016.

[HS06] Houvardas, John; Stamatatos, Efstathios: N-gram feature selection for authorship identification. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications. Springer, pp. 77–86, 2006.

[Ja15] Jaeger, David; Graupner, Hendrik; Sapegin, Andrey; Cheng, Feng; Meinel, Christoph: Gathering and Analyzing Identity Leaks for Security Awareness. In: International Conference on Passwords. Springer, pp. 102–115, 2015.

[Ja16] Jaeger, David; Graupner, Hendrik; Pelchen, Chris; Cheng, Feng; Meinel, Christoph: Fast Automated Processing and Evaluation of Identity Leaks. International Journal of Parallel Programming, pp. 1–30, 2016.

[Kl08] Simple Mail Transfer Protocol, RFC 5321 (Draft Standard). http://www.ietf.org/rfc/rfc5321.txt.

[Kr13] Adobe Breach Impacted At Least 38 Million Users. http://krebsonsecurity.com/2013/10/adobe-breach-impacted-at-least-38-million-users/, Last accessed on 2017-04-28.

[Ni14] Vetting Leaks: Finding the Truth when the Adversary Lies. http://www2.deloitte.com/content/dam/Deloitte/au/Documents/risk/deloitte-au-risk-vetting-leaks-081214.pdf, Last accessed on 2017-04-28.

[Re08] Internet Message Format, RFC 5322 (Draft Standard). http://www.ietf.org/rfc/rfc5322.txt, Updated by RFC 6854.

[Wa15] Wang, Ding; Cheng, Haibo; Gu, Qianchen; Wang, Ping: Understanding passwords of Chinese users: Characteristics, security and implications. CACR Report, Presented at ChinaCrypt, 2015.

[YM12] SMTP Extension for Internationalized Email, RFC 6531 (Proposed Standard). http://www.ietf.org/rfc/rfc6531.txt.