H. Langweg, M. Meier, B.C. Witt, D. Reinhardt (Hrsg.): Sicherheit 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 55

Hashing of personally identifiable information is not sufficient

Matthias Marx[1], Ephraim Zimmer[1], Tobias Mueller[1], Maximilian Blochberger[1], Hannes Federrath[1]

Abstract: It is common practice of web tracking services to hash personally identifiable information (PII), e. g., e-mail or IP addresses, in order to avoid linkability between collected data sets of web tracking services and the corresponding users while still preserving the ability to update and merge data sets associated to the very same user over time. Consequently, these services argue to be complying with existing privacy laws as the data sets allegedly have been pseudonymised. In this paper, we show that the finite pre-image space of PII is bounded in such a way, that an attack on these hashes is significantly eased both theoretically as well as in practice. As a result, the inference from PII hashes to the corresponding PII is intrinsically faster than by performing a naive brute-force attack. We support this statement by an empirical study of breaking PII hashes in order to show that hashing of PII is not a sufficient pseudonymisation technique.

Keywords: personally identifiable information; hashing; pseudonymisation

1 Introduction

The advertisement industry is driven by the need to deliver ads tailored to the audience. In order to deliver targeted ads on the users' screens, they are being tracked across the web by different companies who often also offer reconciliation services for data to increase the accuracy and the value of their data sets.

Traditionally, users were tracked with web cookies. Cookies, however, are not as reliable as other tracking methods such as browser fingerprinting [Eck10] or simply using a more stable reoccurring token to identify a user, e. g., e-mail address or telephone number, as it imposes much more work for the user to change those personal identifiers. Even MAC and IP addresses can be linked to individual users and thus can be used to track individuals [Inf16; Cun14; ME12]. With these methods, tracking companies can merge their data by matching on the token. Exposing the actual token as plain text, however, carries regulatory and economical risks. For example, exposing the user's plain e-mail address to a competitor allows them to use the e-mail address for other purposes, such as sending targeted advertisements. Valid e-mail addresses are believed to be worth around £85 [Jen12]. Furthermore, several national as well as international privacy laws and regulations, e. g., the

[1] Universität Hamburg, Sicherheit in verteilten Systemen, Vogt-Kölln-Straße 30, 22527 Hamburg, Deutschland
<firstname>.<lastname>@informatik.uni-hamburg.de

doi:10.18420/sicherheit2018_04

European General Data Protection Regulation (GDPR) [Eur16], prohibit trading or selling data which allows to identify a user without their consent. Lastly, losing plain personally identifiable information (PII) due to a data breach incurs a much higher risk of losing reputation than losing non-userlinkable data.

For these reasons, i. e., to track users in a privacy-friendly manner and protect assets, while still preserving the linkability between different data sets, companies like Google [Goo17], Facebook [Fac17], or Oracle [Ora18] try to pseudonymise personally identifiable information (PII) with an efficient cryptographic hash function. This is very often incorrectly labelled as anonymisation instead of pseudonymisation. The GDPR defines pseudonymisation as "the processing of personal data in such a manner that the personal data can no longer be attributed to a specific data subject without the use of additional information, provided that such additional information is kept separately and is subject to technical and organisational measures to ensure that the personal data are not attributed to an identified or identifiable natural person" [Eur16, §4.5]. From a technical point of view, a value can be considered a pseudonym if it is not feasible to infer the actual PII which was used to create the pseudonym.

Using cryptographic hash functions seems like a good choice to achieve pseudonymity of PII in that sense. Hash functions operate one-way, so that the resulting hash value cannot be used to calculate the input (pre-image resistance). In addition, cryptographic hash functions are collision resistant, so that it is most likely to return different hash values for different input values. Since hashes are deterministic an attacker can guess the input and then check if the resulting output is equal to a given hash value. Our hypothesis is, that PII is of low entropy which limits the pre-image space of a hash function in such a way, that it is feasible to revert the pseudonymisation even without the use of additional information as per the definition above. We show this for all possible IPv4 and MAC addresses, phone numbers, and a data set of leaked e-mail addresses. Our results indicate that even consumer-grade hardware is sufficient for revealing actual PII for the allegedly pseudonymised values.

The remainder of this paper is structured as follows. We first present other work in the area of hashing PII in Sect. 2. We then provide some background in Sect. 3 and share our data set, environment, and experimental set-up in Sect. 4. Our results are provided in Sect. 5 and discussed in Sect. 6. We finally summarise the results in Sect. 7.

2 Related Work

Narayanan and Shmatikov [NS05] and Bonneau [Bon12] argue, that structured and memorable inputs for hash functions, such as passwords or, especially relevant for this work, e-mail addresses, provide low entropy and are vulnerable to "smart dictionary" attacks. Felten [Fel12] shows that SHA-1 hashes of social security numbers (SSNs) do not provide sufficient protection. Demir et al. [Dem+17] show that attacks on hashed e-mail and MAC addresses are theoretically feasible. They perform the reconstruction of e-mail addresses from a small data set of MD5 hashes and calculate the required time to brute-force all

available MAC addresses by the benchmark results of `hashcat`. Kumar et al. [Kum+07] show that hashing of search tokens is insufficient for protecting search query logs. Tokens often contain sensitive information, such as names, that can be guessed by dictionary attacks. A data set released by AOL in 2006 was used to show that sensitive information could be derived from hashed search tokens in practice.

These related publications show, that the hypotheses about the low entropy of PII and its insufficiency serving as input of hash functions due to its limited pre-image space has been studied several times and is considered as well known. However, we extend that work by considering the most used hash functions in practise, which are believed to provide an adequate protection level of PII, on a large data set of e-mail addresses and show that it is not only theoretically, but also practically feasible to reconstruct a large number of e-mail addresses. We additionally show that it is practically feasible to revert hashes for IPv4 and currently assigned MAC addresses within minutes as well as phone numbers in appropriate time on consumer-grade hardware.

The practical analysis of our work is improved by the observation of Tatli [Tat15], that the process of generating hashes can greatly be sped up by using patterns instead of relying on wordlists or dictionaries. Additionally, the results of inverting e-mail hashes are based on the work of Polakis et al. [Pol+10] and Kumar et al. [Kum+07], who show that valid e-mail addresses can be harvested from names collected in social networking platforms such as Facebook or Twitter as well as from census data.

3 Background

IPv4 and MAC addresses, telephone numbers, and e-mail addresses represent the most reliable PII, that tracking and advertising services can use to identify individual records of their data sets, update a specific record with newly collected data about an individual, and merge data sets traded with other tracking services. The data management platform BlueKai by Oracle, for example, calls these identifiers of users “match keys” and explicitly states to use MD5 and SHA-256 hashed e-mail addresses or phone numbers as well as user’s IP addresses collected from IP headers [Ora18]. Similarly, the organisation infsoft GmbH, which is specialised in indoor-tracking, uses SHA-256 hashed MAC addresses to recognise individuals [inf18]. For this reason, our experiments focus on these four kinds of PII whose properties will be explained in this section.

Internet Protocol version 4 (IPv4) is specified in RFC 791 [Pos81]. It uses 32-bit addresses of fixed length and thus $2^{32} \approx 4.3 \cdot 10^9$ addresses exist. About 600 million addresses are reserved for special purposes, which means that nearly $3.7 \cdot 10^9$ different IPv4 addresses are worth considering when breaking hashes.[2]

[2] See RFCs 990, 1112, 1700, 1918, 2544, 3068, 3927, 5736, 5737, 6598, and 6890

Media Access Control (MAC) addresses are globally unique network interface identifiers. MAC addresses have 48 bit, allowing $2^{48} \approx 2.8 \cdot 10^{14}$ different addresses. The first 24 bit are called "Organisationally Unique Identifier" (OUI) and are controlled by the IEEE, the latter 24 bit are controlled by manufacturers. IEEE publishes a list of assigned OUIs [IEE+06], excluding secret OUIs. The list contains 24 215 entries, which leads to $2^{24} \cdot 24\,215 \approx 4.1 \cdot 10^{11}$ possible MAC addresses.

Telephone Numbers or more specifically their structure is specified by the International Telecommunication Union (ITU) [Int10]. It can be up to 15 digits and consists of a leading country code of 1-3 digits followed by the so called national significant number (NSN), which in turn consists of an area code and the subscriber number. A list of existing country codes is published by the ITU [Int16] and consists of 217 assigned unique entries at the time of writing. So the remaining national significant number of a phone number can only have a length of 14 digits in case of a one digit country code, 13 digits in case of a two digits country code, and 12 digits in case of a three digit country code, resulting in a total number of $2 \cdot 10^{14} + 44 \cdot 10^{13} + 171 \cdot 10^{12} \approx 8.11 \cdot 10^{14}$. This number can further be reduced significantly when additional information about the country code of a phone number is known. Then, the possible area codes of the country, i. e., the digits directly following the country code, and the possible length of the remaining digits, which often are limited by national numbering plans, can be taken into account.[3] Several lists of area codes and corresponding minimal and maximal lengths of the national significant number can be found at [Int17].

E-mail Addresses have a lot more entropy than the previous examples. The general structure of a global e-mail address is `local-part@domain` while the `local-part` can consist of up to 64 ASCII characters [Kle08; Res08]. Furthermore, the case sensitivity of the local-parts is discouraged [Kle08]. We can hence estimate the number of possible `local-parts` limited to loweralpha, numeric, and 20 special characters[4] to be: $(26 + 10 + 20)^{64} = 56^{64} \approx 2^{372} \approx 7.7 \cdot 10^{111}$. There is a maximum of 255 characters in the `domain` part. Additionally, RFC 2821 limits the length of e-mail addresses to 254 characters [Kle01]. The total number of possible e-mail addresses is approx. $2.2 \cdot 10^{120}$, considering the number of 330 million registered domains [Ver17].

4 Methodology

This section describes the environment, that has been used for the evaluation of the resources required to de-pseudonymise hash values that are meant to protect PII. First, the software and hardware setting for our evaluation, second the acquisition of hash values, and third the different calculations that have been performed will be explained.

[3] For example, the German area code 261 is limited to a minimal NSN of 6 digits and to a maximal NSN of 11 digits. Thus only $10^3 + 10^4 + 10^5 + 10^6 + 10^7 + 10^8$ phone numbers of the form +49 261 dd. . .d are possible.

[4] `!#$%&'*+-/=?^_`{|}~.` as recommended by the W3C [W3C17]

4.1 Environment

To de-pseudonymise hashes, we used `hashcat` v4.0.1 [Ste09], a GPU-based password recovery tool, on a desktop computer equipped with an Intel Core i5-6500 CPU at 3.2 GHz, 16 GB RAM, and a Nvidia GeForce GTX 1050 Ti graphics card with 4 GB GDDR5, running an Ubuntu 16.04. The high performance workload profile and optimised kernels for `hashcat` were enabled. The built-in benchmark of `hashcat` reports this setup to be capable of calculating $6.021 \cdot 10^9$ MD5 and $0.844 \cdot 10^9$ SHA-256 hashes per second.

4.2 Acquiring Hashes

For IP and MAC addresses as well as for telephone numbers, one million distinct addresses or numbers have randomly been generated and the hash values of the generated values have been calculated with MD5 and SHA-256.

As randomly generated e-mail addresses would not necessarily look like real e-mail addresses, real e-mail addresses have been acquired in a different way. A list of 700 million leaked credentials (e-mail address and password combinations) [Tro17] has been downloaded. This list has been cleaned up by dismissing the passwords, validating all addresses according to W3C [W3C17] and choosing one million different `.de`-addresses at random. Of these one million random e-mail addresses the MD5 and SHA-265 hash values have been calculated.

4.3 Breaking Hashes

The `hashcat` utility provides several attack types, including combinator, mask, hybrid, and rule-based attacks. Combinator attacks are two-dictionary attacks, where each word from the one dictionary is paired with each word from the second one. Mask attacks are brute-force attacks, where the search space is limited to certain word structures and character sets. A hybrid attack couples combinator and brute-force or mask attack. Rule-based attacks are dictionary attacks where each word serves as password candidate generator. The rule engine of `hashcat` could for example reverse or duplicate words or could prepend, swap, and replace characters. Multiple rules can be combined.

IPv4 Addresses For breaking the IPv4 address hashes, a combinator attack is most appropriate. Two dictionaries have been created, one for all possible left halves and another one for all possible right halves. Each dictionary has $65\,536 = 2^{16}$ entries. Both dictionaries combined result in all 2^{32} possible IPv4 addresses and require about 1 MB of storage. A single dictionary would need about 60 GB of storage. Alternatively, different rules for a mask attack could be set up.

MAC Addresses The first part of a MAC address forms the OUI, which can be found in public databases. Thus a dictionary of OUIs has been built and only the second part needed to be brute-forced, leading to the execution of a hybrid attack.

Telephone Numbers For breaking the telephone number hashes, a mask attack has been executed by acquiring lists of Indonesian, Chinese, and German country and area codes and supplementing these lists with masks depending on the lengths of the possible subscriber numbers. For example, the mask `4930?d?d?d` gives `hashcat` the instruction to probe all numbers from `49 30 000` to `49 30 999`.

E-mail Addresses Numerous attacks have been performed to break the e-mail address hashes, which can be distinguished in the way they probe the local-part of the addresses. For all attacks, lists of the 10 and 100 most common `.de`-domains extracted from an OpenPGP keyserver [PGP18] have been passed to `hashcat`.

First, several mask attacks have been executed. However, they are appropriate only for short local-parts as the runtime of those attack increases exponentially with the length of the provided mask. For one to four characters, the character set of the provided mask included every possible character, which is allowed to occur in the local-part by specification. By increasing the number of characters in subsequent mask attacks, the character set of the provided masks has been reduced.

Second, dictionary attacks with two wordlists have been executed. The first wordlist (from now on referred to as small dictionary) contained $9.7 \cdot 10^6$ first and last names that had been crawled from Facebook [Bow10]. The second wordlist (from now on referred to as large dictionary) contained approx. $3.1 \cdot 10^9$ words from dictionaries, username, and password lists [Hat13].

Lastly, `hashcat`'s rule engine has been utilised and combined with the two wordlists in order to generate several local-part candidates for each word in the wordlists. This included prepending, appending, and the duplication or deletion of single characters, and rotating, reversing, duplicating, and reflecting of words.

5 Results

In this section, the result of our attacks described in the previous section are presented.

5.1 IPv4 Addresses

Theoretical Considerations With the assumption to be able to compute $6 \cdot 10^9$ MD5 hashes (6 Giga hashes) and $844 \cdot 10^6$ SHA-256 hashes (844 Mega hashes) per second, which are the benchmark results of `hashcat` as explained in Sect. 4.1, calculating a MD5 hash

for each address in the whole IPv4 space completes in 0.72 s and calculating the same for SHA-256 hashes would take 5.09 s.[5]

Practical Results Tab. 1 shows the practical results of our combinator attacks on IPv4 address hashes. We de-pseudonymised all MD5 and SHA-256 hashes in under one minute. De-pseudonymisation of SHA-256 hashes took 5 s (20%) longer than de-pseudonymisation of MD5 hashes.

Search Space	Runtime		Recovered Hashes	Recovery Rate
	MD5	SHA-256		
$4.3 \cdot 10^9$	00:00:25	00:00:30	1 000 000	100%

Tab. 1: Practical results of combinator attacks on one million IPv4 address hashes.

5.2 MAC Addresses

Theoretical Considerations The amount of $2^{48} \approx 2.8 \cdot 10^{14}$ different MAC addresses can effectively be reduced by the 24 215 OUIs published by the IEEE (see Sect. 3). Again, assuming the ability to calculate 6 Giga MD5 hashes and 844 Mega SHA-256 hashes per second, the MAC address space can be exhaustively hashed in 1 min 8 s for MD5 and in 8 min 2 s for SHA-256.[6]

Practical Results Tab. 2 shows the practical results of our hybrid attacks on MAC address hashes. All MD5 as well as all SHA-256 hashes could be de-pseudonymised in under 15 min. De-pseudonymisation of SHA-256 hashes took approx. 10 min (three times) longer than de-pseudonymisation of MD5 hashes.

Search Space	Runtime		Recovered Hashes	Recovery Rate
	MD5	SHA-256		
$4.1 \cdot 10^{11}$	00:04:01	00:13:22	1 000 000	100%

Tab. 2: Practical results of hybrid attacks on one million MAC address hashes.

5.3 Telephone Numbers

Theoretical Considerations In order to break a hashed telephone number, one needs to produce at most $2 \cdot 10^{14} + 44 \cdot 10^{13} + 171 \cdot 10^{12} \approx 8.11 \cdot 10^{14}$ hashes (see Sect. 3). At the hashing rate of 6 Giga MD5 hashes and 844 Mega SHA-256 hashes per second, breaking all hashes of any possible number takes at most 1 d 13 h 33 min for MD5 and 11 d 2 h 55 min

[5] $2^{32}/(6 \cdot 10^9) = 0.7158$ and $2^{32}/(844 \cdot 10^6) = 5.0888$

[6] $2^{24} \cdot 24\,215/(6 \cdot 10^9) = 67.71$ and $2^{24} \cdot 24\,215/(844 \cdot 10^6) = 481.351$

for SHA-256.[7] This is the upper limit for breaking any given number. That does not seem to be out of reach for a determined attacker, but it is probably still too slow for a casual de-pseudonymisation attempt. Improvements can be made when additional information about a phone number is available. For example, if we assume the number to be of a certain origin, e. g., either German, Chinese, or Indonesian, we can limit the pre-image space to approx. 10^{12} by considering the possible area codes of these three countries, i. e., the digits directly following the country code, and the possible length of the remaining digits, which often are limited by national numbering plans (see Sect. 3). Thus, all possible German, Indonesian and Chinese telephone numbers can be hashed in 2 min 47 s, assuming a hashing rate of 6 Giga MD5 hashes per second, and can be hashed in 19 min 45 s, assuming a hashing rate of 844 Mega SHA-256 hashes per second.[8] At least theoretically, it is feasible to break the pseudonymity provided by a hashed phone number, especially if we assume previous knowledge about the received pseudonymised data set. This assumption is not far fetched, given that companies exchange data for reconciliation purposes rather than for acquiring new data subjects.

Practical Results Tab. 3 shows the practical results of our mask attacks on telephone number hashes. All MD5 and SHA-256 hashes could be de-pseudonymised. The search space of German and Indonesian telephone numbers is one magnitude larger than the Chinese search space. The runtime difference between de-pseudomisation of MD5 and SHA-256 hashes is below 7 min for a search space of size 10^{11}.

Country	Search Space	Runtime		Recovered Hashes	Recovery Rate
		MD5	SHA-256		
China	$2.3 \cdot 10^{10}$	00:07:17	00:07:12	1 000 000	100%
Germany	$4 \cdot 10^{11}$	02:28:24	02:34:16	1 000 000	100%
Indonesia	$5.8 \cdot 10^{11}$	02:45:57	02:52:42	1 000 000	100%

Tab. 3: Practical results of mask attacks on one million telephone number hashes for three selected countries.

5.4 E-mail Addresses

Theoretical Considerations The large estimated theoretical space for e-mail addresses limited to alpha-numeric and 3 special characters is already unfeasible for performing a brute-force attack, even when ignoring the `domain` part of the e-mail address (see Sect. 3). When only considering the 100 most popular e-mail domains, a total number of $7.88 \cdot 10^{117}$ addresses still exist. In the case of MD5, breaking all hashes would take unimaginable $4.16 \cdot 10^{100}$ years when assuming a hashing rate of 6 Giga hashes per second.[9]

[7] $(8.11 \cdot 10^{14})/(6 \cdot 10^{9}) = 135166.6667$ and $(8.11 \cdot 10^{14})/(844 \cdot 10^{6}) = 960900.4739$

[8] $10^{12}/(6 \cdot 10^{9}) = 166.6667$ and $10^{12}/(844 \cdot 10^{6}) = 1184.8341$

[9] $7.88 \cdot 10^{117}/(6 \cdot 10^{9})/3600/24/365 \approx 4.1646 \cdot 10^{100}$

At first sight, the cost of de-pseudonymising hashed e-mail addresses seems to be overwhelming. But that is assuming an e-mail address consisting of a randomly generated 64 character local-part rather than following a certain structure, e. g., <firstname>.<lastname>@<domain>, <firstname><year>@<domain>, <first letter of firstname>.<lastname>@<domain>, or <7 lowercase letters>@<domain>. Taking a list of the 100 most popular e-mail domains as well as a list of the 1000 most popular first names and the 1000 most popular last names, the four example e-mail structures result in an approx. number of $8 \cdot 10^{11}$ e-mail addresses.[10] With a hashing rate of 6 Giga MD5 hashes per second, we can revert any MD5 hash in 2 min 14 s. With a hashing rate of 844 Mega SHA-256 hashes it would take 15 min 48 s.[11]

Practical Results Tab. 4 gives an overview of the practical results of our mask and dictionary attacks on e-mail address hashes. We observe major differences in runtime depending on attack type, hash algorithm and number of probed domains. For both hashing functions and a combination of all attacks, probing for the top 100 domains takes approx. six times longer than probing for the top 10 domains and results in an increase of 7% of the recovery rate. In total, we de-pseudonymised approx. 43% of the hashed e-mail addresses.

For mask attacks, probing for the top 100 domains takes approx. eight times longer than probing for the top 10 domains and results in an increase of 3.3% of the recovery rate. Computing SHA-256 hashes takes approx. three times longer than computing MD5 hashes.

For small dictionary attacks, probing for the top 100 domains takes approx. six and a half times longer than probing for the top 10 domains and results in an increase of 4.9% of the recovery rate. Computing SHA-256 hashes takes approx. three times longer than computing MD5 hashes.

For large dictionary attacks, probing for the top 100 domains takes approx. five to six times longer than probing for the top 10 domains and results in an increase of 5.6% of the recovery rate. Computing SHA-256 hashes takes approx. two times longer than computing MD5 hashes.

Tab. 5 shows details of the attacks that were conducted using the top 100 domains. The custom charset `?1` consists of loweralpha (`?l`), numeric (`?d`) and special (`-_.`) characters.

6 Discussion

This section analyses the results obtained in the previous section and sets them into the broader context of PII pseudonymisation.

[10] $10^3 \cdot 10^3 \cdot 100 + 10^3 \cdot 10^4 \cdot 100 + 26 \cdot 10^3 \cdot 100 + 26^7 \cdot 100 \approx 8.0428 \cdot 10^{11}$

[11] $(8 \cdot 10^{11})/(6 \cdot 10^9) = 133.3334$ and $(8 \cdot 10^{11})/(844 \cdot 10^6) = 947.8673$

Attack Type	Search Space	Domains	Runtime		Recovered Hashes	Recovery Rate
			MD5	SHA-256		
Mask	$3.5 \cdot 10^{12}$	Top 10	00:38:34	01:57:03	166 152	16.62 %
	$3.5 \cdot 10^{13}$	Top 100	05:18:20	16:40:17	199 377	19.94 %
Small Dictionary	$1.5 \cdot 10^{12}$	Top 10	01:09:40	01:22:57	246 900	24.69 %
	$1.5 \cdot 10^{13}$	Top 100	07:58:29	09:25:29	295 881	29.59 %
Large Dictionary	$3.1 \cdot 10^{12}$	Top 10	01:35:18	02:37:38	281 714	28.17 %
	$3.1 \cdot 10^{13}$	Top 100	07:40:16	15:34:34	337 333	33.73 %
Total	$8.1 \cdot 10^{12}$	Top 10	03:23:32	06:57:38	355 155	35.52 %
	$8.1 \cdot 10^{13}$	Top 100	20:57:05	41:40:20	425 198	42.52 %

Tab. 4: Overview of the practical results of our mask and dictionary attacks on one million e-mail address hashes.

Mask (Length) / Rule	Search Space	Runtime		Recovered Hashes	Recovery Rate
		MD5	SHA-256		
`?l?d?s` (1)	$5.6 \cdot 10^{3}$	00:00:05	00:00:06	11	0.00 %
`?l?d?s` (2)	$3.1 \cdot 10^{5}$	00:00:04	00:00:05	149	0.01 %
`?l?d?s` (3)	$1.8 \cdot 10^{7}$	00:00:04	00:00:07	1 515	0.15 %
`?l?d?s` (4)	$9.8 \cdot 10^{8}$	00:00:04	00:00:07	5 781	0.58 %
`?l?d?l` (5)	$9.0 \cdot 10^{9}$	00:00:08	00:00:22	18 857	1.89 %
`?l?d?l` (6)	$3.5 \cdot 10^{11}$	00:02:33	00:09:09	49 321	4.93 %
`?l?d?l` (7)	$1.4 \cdot 10^{13}$	02:37:01	07:21:16	80 829	8.08 %
`?l` (8)	$2.1 \cdot 10^{13}$	02:38:21	09:09:05	42 914	4.29 %
Small Dictionary (SD)	$9.7 \cdot 10^{8}$	00:01:36	00:01:28	35 117	3.51 %
SD + Set of Rules	$1.9 \cdot 10^{10}$	00:02:25	00:02:45	46 306	4.63 %
SD + `?l`	$3.8 \cdot 10^{10}$	00:03:41	00:04:01	65 282	6.53 %
`?l` + SD	$3.8 \cdot 10^{10}$	00:02:55	00:03:21	51 586	5.16 %
SD + `?l?l`	$1.5 \cdot 10^{12}$	00:46:06	00:57:11	162 852	16.29 %
`?l?l` + SD	$1.5 \cdot 10^{12}$	00:39:01	00:51:15	99 423	9.94 %
`?l` + SD + `?l`	$1.5 \cdot 10^{12}$	00:40:13	00:52:43	85 221	8.52 %
SD + `?d?d?d`	$9.7 \cdot 10^{11}$	00:30:23	00:35:25	26 230	2.62 %
SD + `?d?d?d?d`	$9.7 \cdot 10^{12}$	05:12:19	05:57:20	39 073	3.91 %
Large Dictionary (LD)	$3.1 \cdot 10^{11}$	00:20:41	00:23:53	174 197	17.42 %
LD + Set of Rules	$6.2 \cdot 10^{12}$	01:49:38	03:42:47	206 804	20.68 %
LD + `?l`	$1.2 \cdot 10^{13}$	02:47:50	05:47:16	182 411	18.24 %
`?l` + LD	$1.2 \cdot 10^{13}$	02:42:07	05:40:38	110 193	11.02 %

Tab. 5: Practical results of mask, dictionary and rule-based attacks on one million e-mail address hashes and the top 100 domain. The custom charset `?1` consists of loweralpha (`?l`), numeric (`?d`) and special (`-_.`) characters.

In the previous sections we have shown that the pseudonymisation of PII by means of simple hashing can be subverted relatively easily. We elaborated on the theoretical pre-image space and were able to reduce it with knowledge of either the structure of the data or the data subjects. While we argue that this knowledge is required properly exchanging hashed PII, we acknowledge that it makes our results less applicable to someone who does not have the required knowledge.

According to our experiments with rather modest consumer-grade hardware it seems entirely feasible to break pseudonymisation of certain PII at a low cost. A determined attacker with specialised hardware may very well be able to break pseudonymisation much quicker. Furthermore, the attacks can be easily parallelised on several computers. At the time of writing, Amazon provides computing infrastructure capable of computing 450 Giga hashes per second for about 25 USD per hour [Mat17].

It is obvious that the actually achieved hash rates are much lower than the theoretically calculated values. This partly is due to the fact that not one but one million hashes have been tried to be calculated at the same time. Another reason is that for some attacks the utilisation of the graphics card was below 100 %. For dictionary attacks, words have to be copied from the host system over PCI-Express to the GPU. The GPU is busy during the copy operation and can not compute hashes which reduces the overall performance. `Hashcat`'s rule system modifies words on the GPU which effectively reduces the copy-buffer overhead [Has17]. This effect can be seen from Tab. 5.

We have observed that the advertisement industry uses cryptographic hash functions because they make it very hard to find a pre-image of a given hash. It is thus believed that hashing PII is a good way of pseudonymisation. However, if assuming a data breach in form of a third-party learning the whole list of hashed PII, the attacker may very well not be interested in reversing one particular hash, but rather as many hashes as possible. This is very similar to protecting passwords which tends to be achieved by the use of slower hash function. In 1999, the bcrypt hashing scheme was presented [PM99]. It makes brute force attacks harder by having a relatively expensive key setup phase. In 2015, the Password Hashing Competition [Wet16] awarded Argon2 [BDK16] for providing a good protection level for passwords. The setup of our experiments reported to be able to compute $4 \cdot 10^3$ bcrypt hashes per second which is six magnitudes slower than the $6 \cdot 10^9$ MD5 hashes our modest hardware can compute (cf. Sect. 4.1). An attack on a list of hashed PII will thus get more expensive when a slower hash algorithm is used. Certain attacks, however, are still feasible. For example, a list of 700 million leaked e-mail addresses could still be hashed within two days[12]. Such pre-computed result could then be stored for later lookup purposes. The data management industry could still be interested in changing their algorithms to a much less efficient algorithm in order to significantly increase an attacker's effort and costs when breaking hashes.

Another way to increase the effort of an attacker is to make the attacker break every single

[12] 700000000/4000/60/60/24 = 2.03

hash rather than a whole list of hashes. This can be achieved by prefixing each entry with a "salt" of a length of the desired protection level, e. g., eight bytes. While this allows the continued use of efficient hashing algorithms like MD5 or SHA-256, it requires a change in the format the data is exchanged and traded, because every single data entry has to carry the salt used. This also prevents data sets to be merged easily unless the two sets have used the same salts for each and every individual record. Data management providers might be relunctant to implement such a scheme due that inability of reconciliating data sets.

7 Conclusion

In this paper, experiments with rather modest hardware have been performed in order to quantify the effort required to thwart hash-based pseudonymisation of PII. The results show, that practical attacks are feasible, which complements theoretical analyses of previous work.

In more detail, the performed experiments show, that PII with relatively small pre-image space, namely IPv4 addresses, MAC addresses, and telephone numbers, are not adequately protected with simple hashing. The pseudonymisation of 43 % of hashed e-mail addresses could also been reverted within less than a day. Our results suggest that a more elaborate way of pseudonymisation is required in order to protect data subjects. In our discussion in Sect. 6, recommendations about mitigating attacks on pseudonymisation are provided. Further, the options to only provide a temporary stop-gap measure is discussed, as computing power and economic efficiency tend to increase.

References

[BDK16] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications". In: *EuroS&P*. IEEE, 2016, pp. 292–302.

[Bon12] Joseph Bonneau. "The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords". In: *IEEE S&P*. IEEE, 2012, pp. 538–552.

[Bow10] Ron Bowes. *Return of the Facebook Snatchers*. 2010. URL: `https://blog.skullsecurity.org/?p=887` (visited on 12/14/2017).

[Cun14] Mathieu Cunche. "I know your MAC address: targeted tracking of individual using Wi-Fi". In: *J. Computer Virology and Hacking Techniques* 10.4 (2014), pp. 219–227.

[Dem+17] Levent Demir et al. "The Pitfalls of Hashing for Privacy". In: *Commun. Surveys Tuts.* (99 2017).

[Eck10] Peter Eckersley. "How Unique Is Your Web Browser?" In: *Privacy Enhancing Technologies*. Vol. 6205. Lecture Notes in Computer Science. Springer, 2010, pp. 1–18.

[Eur16] European Parliament. “Regulation (EU) 2016/679 (General Data Protection Regulation)”. In: *Official Journal of the European Union* L119 (May 2016), pp. 1–88. URL: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L:2016:119:TOC.

[Fac17] Facebook. *Das Hochladen der Kundenliste*. 2017. URL: https://www.facebook.com/business/help/112061095610075 (visited on 12/14/2017).

[Fel12] Ed Felten. *Does Hashing Make Data “Anonymous”?* Federal Trade Commission. Apr. 22, 2012. URL: https://www.ftc.gov/node/605301 (visited on 12/14/2017).

[Goo17] Google. *Integrating CRM Data with Google Analytics to create AdWords Remarketing Audiences*. Jan. 2017. URL: https://developers.google.com/analytics/solutions/crm-integration (visited on 12/14/2017).

[Has17] Hashcat. *FAQ*. 2017. URL: https://hashcat.net/wiki/doku.php?id=frequently_asked_questions (visited on 12/14/2017).

[Hat13] Steven Hatfield. *My wordlist now shared*. 2013. URL: https://wp.me/p2HHRm-F (visited on 12/14/2017).

[IEE+06] IEEE Standard Association et al. *IEEE OUI and company id assignments*. 2006. URL: http://standards.ieee.org/regauth/oui/oui.txt (visited on 12/14/2017).

[Inf16] InfoCuria. *Judgment of the Federal Court of Justice in Germany (Second Chamber) in Case C-582/14: Processing of personal data*. Oct. 2016. URL: http://curia.europa.eu/juris/document/document.jsf?text=&docid=184668&doclang=en (visited on 12/14/2017).

[inf18] infsoft GmbH. *FAQ – Indoor Positioning – Data Protection*. 2018. URL: https://www.infsoft.com/technology/faq (visited on 02/14/2018).

[Int10] International Telecommunication Union. *The international public telecommunication numbering plan*. Nov. 18, 2010. URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.164-201011-I!!PDF-E&type=items (visited on 12/14/2017).

[Int16] International Telecommunication Union. *List of Recommendation ITU-T E.164 assigned country codes*. 2016. URL: http://handle.itu.int/11.1002/pub/80eefa4f-en (visited on 12/12/2017).

[Int17] International Telecommunication Union. *List of National Numbering Plans*. 2017. URL: https://www.itu.int/oth/T0202.aspx?lang=en&parent=T0202 (visited on 12/12/2017).

[Jen12] Richard Jenkings. *How much is your email address worth?* The Drum. Apr. 4, 2012. URL: http://www.thedrum.com/opinion/2012/04/04/how-much-your-email-address-worth (visited on 12/11/2017).

[Kle01] John C. Klensin. “Simple Mail Transfer Protocol”. In: *RFC* 2821 (2001).

[Kle08] John C. Klensin. “Simple Mail Transfer Protocol”. In: *RFC* 5321 (2008).

[Kum+07] Ravi Kumar et al. “On anonymizing query logs via token-based hashing”. In: *WWW*. ACM, 2007, pp. 629–638.

[Mat17] Iraklis Mathiopoulos. *Running hashcat v4.0.0 in Amazon’s AWS new p3.16xlarge instance*. Medium. Oct. 28, 2017. URL: https://medium.com/@iraklis/e8fab4541e9b (visited on 12/11/2017).

[ME12] A. B. M. Musa and Jakob Eriksson. “Tracking unmodified smartphones using wi-fi monitors”. In: *SenSys*. ACM, 2012, pp. 281–294.

[NS05] Arvind Narayanan and Vitaly Shmatikov. “Fast dictionary attacks on passwords using time-space tradeoff”. In: *ACM CCS*. ACM, 2005, pp. 364–372.

[Ora18] Oracle Corporation. *BlueKai Platform – Offline match integration*. 2018. URL: https://docs.oracle.com/cloud/latest/marketingcs_gs/OMCDA/IntegratingBlueKaiPlatform/DataIngest/offline_match.html (visited on 02/14/2018).

[PGP18] PGP Public Key Server. *Keydump*. Feb. 2, 2018. URL: http://pgp.key-server.io/sks-dump (visited on 02/02/2018).

[PM99] Niels Provos and David Mazieres. “A Future-Adaptable Password Scheme.” In: *USENIX, FREENIX Track*. 1999, pp. 81–91.

[Pol+10] Iasonas Polakis et al. “Using social networks to harvest email addresses”. In: *WPES*. ACM, 2010, pp. 11–20.

[Pos81] Jon Postel. “Internet Protocol”. In: *RFC* 791 (1981).

[Res08] Peter W. Resnick. “Internet Message Format”. In: *RFC* 5322 (2008).

[Ste09] Jens Steube. *Hashcat*. 2009. URL: https://hashcat.net/ (visited on 12/14/2017).

[Tat15] Emin Islam Tatli. “Cracking More Password Hashes With Patterns”. In: *IEEE Trans. Inf. Forensics Security* 10.8 (2015), pp. 1656–1665.

[Tro17] Troy Hunt. *Password reuse, credential stuffing and another billion records in Have I been pwned*. May 5, 2017. URL: https://www.troyhunt.com/password-reuse-credential-stuffing-and-another-1-billion-records-in-have-i-been-pwned/ (visited on 12/14/2017).

[Ver17] VeriSign, Inc. *The Domain Name Industry Brief*. Dec. 2017. URL: http://www.verisign.com/assets/domain-name-report-Q32017.pdf (visited on 02/13/2018).

[W3C17] W3C. *HTML 5.1 2nd Edition - W3C Recommendation*. 2017. URL: https://www.w3.org/TR/html5/sec-forms.html#valid-e-mail-address (visited on 12/14/2017).

[Wet16] Jos Wetzels. “Open Sesame: The Password Hashing Competition and Argon2”. In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 104.