# An Analysis of 5 Million OpenPGP Keys

Birger Schacht and Peter Kieseberg*
*JRC Blockchains & Institute of IT Security Research*
*St. Pölten University of Applied Sciences, St. Pölten, Austria*
{is161323, peter.kieseberg}@fhstp.ac.at

Received: July 11, 2020; Accepted: September 2, 2020; Published: September 30, 2020

**Abstract**

OpenPGPG is a well-known environment for email encryption, data signing, authentication and key certification with a long-standing history. Commonly, research regarding OpenPGP focuses on the web of trust and cryptography related aspects. However, there are a lot of other properties of OpenPGP keys that have not been analyzed until now. In this work, we analyze a set of 5 million OpenPGP keys with respect to algorithms used and selection of internal parameters. Furthermore, we analyze connections to third party software, as well as related aspects of the keys. The major contribution lies in analyzing these properties, to visualize trends of OpenPGP usage over the last 20 years and to analyze the evolution of OpenPGP since its beginnings. This provides an insight which can be useful for further decision making regarding OpenPGP and the adoption of public key cryptography in general. In addition, plotting the evolution of public key properties can help find anomalies. Looking at the details of the keys over time makes it possible to see if recommendations regarding key characteristics have an effect on real world use, which in turn might give feedback on new recommendations. The analysis of OpenPGP keys also allows to investigate, how long it takes for changes in default settings of popular software packets to reach the users.

**Keywords**: PGP, key exchange, web of trust

## 1 Introduction

OpenPGP is a standard for data encryption, data signing, authentication and key certification. In 1986 Phil Zimmerman started working on a software called Pretty Good Privacy (PGP) which he released under a copyleft license in June 1991 ([1], [2, p. 98]). PGP got a lot of attention and subsequently was widely distributed around the globe, in spite of U.S. regulations that forbid the export of cryptographic systems using keys larger than 40-bit. In the following years, PGP was improved and in 1994 researchers at the Massachusetts Institute of Technology (MIT) submitted the PGP packet format as an internet standard to prevent diverging developments of European and U.S. based PGP implementations. The standard was published as RFC 1991 [3] in 1996. In September 1997 an OpenPGP Working Group (OpenPGP WG) was formed at the Internet Engineering Task Force (IETF) [4]. The working group released RFC 2440 [5] in 1998, which defined the OpenPGP Message format and obsoleted the original RFC 1991. RFC 2440 was subsequently replaced by RFC 4880 [6] in 2007. In 1997 Werner Koch, a programmer from Germany, began writing the *GNU Privacy Guard* (GnuPG). Soon after the intial *GnuPG* release the draft for RFC 2440 was published and Koch released a first OpenPGP draft compliant version of GnuPG in July 1998 [7].

However, OpenPGP is not only used for email communication. OpenPGP signatures allow to sign software packages to verify the integrity of the software. The distributed version control system *git* can

*Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 11(3):107-140, Sept. 2020
DOI:*10.22667/JOWUA.2020.09.30.107*

*Corresponding author: Matthias Corvinus-Straße 15, A-3100 St. Pölten, Austria, Tel: +43-6603126291

use OpenPGP signatures to verify the integrity of individual commits. The widely used remote management software *openssh* can use OpenPGP authentication keys as a means of authenticating against a server. Although the last version of the OpenPGP standard was released in 2007, the OpenPGP working group is still extending and adapting the standard, to integrate technological progress and new research in cryptography. In 2009, RFC 5581 [8] introduced the Camellia block cipher to OpenPGP. In 2012, RFC 6637 [9] defined an extension to include support for public keys that are based on Elliptic Curve Cryptography (ECC). Since November 2015 the OpenPGP WG is working on an draft of a new version of the the OpenPGP standard [10], informally called RFC 4880bis. The draft updates the fingerprint format, gets rid of key ids (key ids are truncated versions of the key fingerprint), deprecates MD5, SHA1 and RIPEMD160 and standardizes the use of elliptic curve cryptography (among other things).

The overview of related work shows that most of the research on OpenPGP keys looks at connections between OpenPGP keys through signatures (also referred to as the *Web of Trust*, or WOT) as a social graph. This is an important topic, because the security properties of the Web of Trust are tightly linked to the strength of the connections between the nodes in this graph. Some of the studies also touch the topics of algorithm distribution and in [11] the authors specifically analyzed the vulnerability of keys. However, there are a lot of other properties of OpenPGP keys that have not been analyzed until now. The goal of this research is to look at these properties, to visualize trends of OpenPGP usage over the last 20 years and to analyze the evolution of OpenPGP since its beginnings. This provides an insight which can be useful for further decision making regarding OpenPGP and the adoption of public key cryptography in general. In addition, plotting the evolution of public key properties can help find anomalies. Looking at the details of the keys over time makes it possible to see if recommendations regarding key characteristics have an effect on real world use, which in turn might give feedback on new recommendations. The analysis of OpenPGP keys also allows to investigate, how long it takes for changes in default settings of popular software packets to reach the users. The information in the User ID packets can be used to analyze the relationship of OpenPGP users to email providers, moreover, the content of the User ID packets can give indications of possible improvements in user interfaces. The percentage of keys with user attribute packets could be a sign of their usefulness. Looking at the keydata, we can evaluate how many keys have expiration dates set. Patterns might give an indication about software products and their distribution.
In this work, we therefore provide the following contributions:

- We conduct intensive data exploration on OpenPGP keys and provide a trend analysis, e.g. regarding the adoption of public key algorithms.
- The trend analysis also gives a good indication on the speed of adopting new techniques and the out-phasing of obsolete (or even insecure) methods, as well as trends.
- We identify correct and incorrect usage of OpenPGP, as well as intensive links to end-user software.

This work is structured as follows:

- Section 2 describes the OpenPGP packet format and the keyserver system used to distribute OpenPGP keys, while in Section 3 we give an overview on the related work.
- Section 4 outlines the data collection and analysis approach.
- The following sections gives an overview on the results and findings, split into three main focus areas:
  - Section 5 describes the work done on the public key packets of both primary and subkeys in the data set.

  - Section 6 goes into the analysis of the UserID and User Attribute packets of the OpenPGP keys.
  - Section 7 analyzes the signature packets that are part of the OpenPGP keys.

- Section 9 provides a conclusion of the findings of the previous chapters.

## 2 Background

Figure 1 gives a short overview on the internals of PGP and OpenPGP. The ”OpenPGP Message Format” (previously ”PGP Message Exchange Formats”) is a standard defined by the IETF in RFC 4880 (which obsoletes the older RFC 1991 and 2440). There are numerous software applications and software libraries implementing the standard, PGP being the initial software and GnuPG being nowadays the most widespread one. These applications enable users to handle various types of OpenPGP data, like signatures, encrypted messages and keys. OpenPGP keys come in pairs, a public and a private key, where a public key component can have subkeys and different properties (bound to the key by self signatures). The key itself and the subkeys can have different capabilities. To enable a trust network, public keys are often signed by keys from other keyholders. Those signatures are attached to the public key.

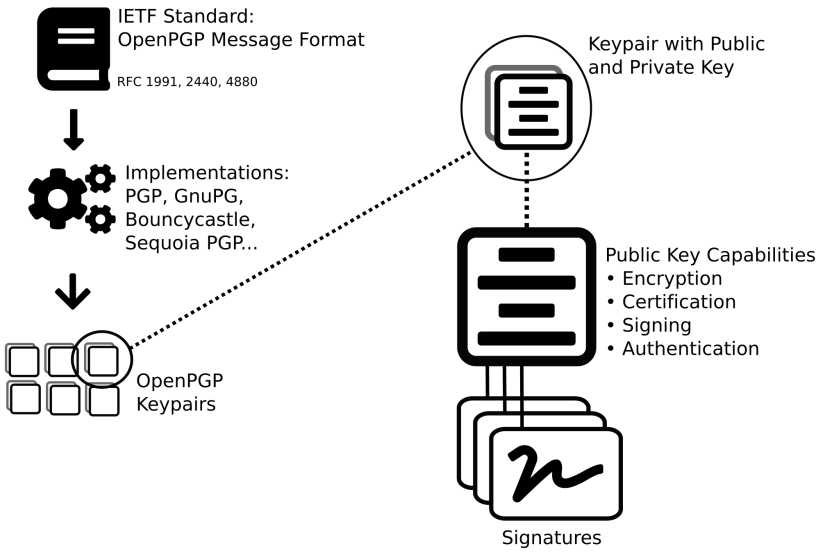



Figure 1: OpenPGP concept

### 2.1 The OpenPGP data format

OpenPGP public keys, like any OpenPGP data, consist of packets. An OpenPGP packet is a chunk of data that has a *tag* attached to it, specifying its meaning. A packet can also contain other packets, for example a Compressed Data Packet (Tag 8) contains a compressed set of OpenPGP packets. Every OpenPGP packet consists of a header and a body, where the first octet of the header defines the tag of the package, the rest of the header specifies its length. The most common packets in OpenPGP public keys are Public Key Packets (Tag 6), Public Subkey Packets (Tag 14), User ID Packets (Tag 13), User Attribute Packets (Tag 17) and Signature Packets (Tag 2), all of which we will subsequently analyze.

## 2.2 Public Key Packets

Public Key Packets and Public Subkey Packets have the same format: Every public OpenPGP key has one primary key, sometimes referred to as *top level* key or *master* key. A primary key can have multiple sub-keys and it indicates the bond to a sub-key using a Signature Packet. The OpenPGP specifications define two versions of OpenPGP public key packets, version 3 and version 4 (there is actually also version 2, but it has the same format as a version 3 public key packet except for the version number). Version 3 packets are deprecated and were only used from PGP 2.6 up to PGP 5.0. A version 3 Public Key Packet contains the version number 3, the time the key was created, the validity of the key in days, the algorithm ID for the public key generation, as well as the key material itself, which consists of series of multi-precision integers. For version 3 Public Key Packets the standard defines RSA as the solely possible public key algorithm. A version 4 Public Key Packet has nearly the same structure but contains the version number 4 and does not contain the validity of the key, which is specified using a Signature Packet containing the key expiration time.

According to RFC 4880bis, the draft for the update to RFC 4880, it is planned to introduce a version 5 public key packet, that is similar to the version 4 but adds a field for the count of the key material, to make parsing easier.

## 2.3 User ID Packets and User Attribute Packets

The User ID Packet contains UTF-8 text that most often represents a user id (by convention the name and email address of the key holder), but, theoretically, could contain any data. A public key can contain multiple User ID Packets (i.e. if a person wants to use the key for multiple email addresses). A variation of the User ID packet is the User Attribute packet, which was defined in RFC 4880 to store types of data different than text. A User Attribute packet contains one or more attribute subpackets, which consist of a header defining the subpacket length and the subpacket type and a body containing the subpacket specific data. There is only one type of subpacket defined at the moment, which is Type 1, defining an Image Attribute Subpacket containing a JPEG image. To certify that the User ID Packet or the User Attribute Packet belong to the top level key, such a packet is followed by OpenPGP Signature Packets.

## 2.4 Signature Packets

Signature Packets describe a binding between some data and a specific public key. Like public key packets there are version 3 and version 4 signatures, but according to RFC 4880, current implementations should only generate type 4 signatures [6, ch. 5.2].

Besides a field denoting the version, a version 4 signature packet also contains the signature type (i.e. is it a signature on a binary document or a signature binding a subkey to a primary key), the ID of the public key algorithm used and the ID of the hash algorithm used. This is followed by the count of the hashed subpackets and then the list of hashed subpackets, the count of the unhashed subpackets and then the list of unhashed subpackets, the left 16 bit of the signed hash value and then the signature itself, consisting of one or more multi-precision integers (which are algorithm specific). The subpackets of a Signature Packet contain data describing the signature. The header of a signature subpacket contains the length of the signature sub packet and its type, while the value of the signature subpacket type may be between 0 and 32 or 100 to 110 for private or experimental use. Typical signature subpacket types are the creation time subpacket, the issuer fingerprint subpacket or a subpacket with key flags, that hold information on how to use the key.

A signature that is generated by the primary key being signed is called a self signature. Self Signatures usually have the signature Type 13 and they are used to define properties of an OpenPGP key. The

modification of the expiration date of a key for example is set by adding a new self signature containing a subpacket with a new key expiration time (in number of seconds after the key creation).

Following the self signatures are signatures from other keys, also known as *certifications*. Signatures that certify other keys sometimes have an expiration date or contain a Policy URI sub packet, to give information about the key signing policy of the key owner. The procedure of certifying other keys is the basis for the Web of Trust.

## 2.5 Algorithms used in OpenPGP

OpenPGP nowadays allows the use of a multitude of different algorithms, for public key components RSA, El Gamal, DSA, ECDSA and EdDSA are available, for the symmetric components IDEA, Triple DES, CAST5 and AES can be selected, and the MD5, SHA-1 and RIPE-MD hash functions are available.

## 2.6 The SKS keyserver network

For the distribution of OpenPGP keys, so-called *keyservers* are used: A user can upload the key to one of the keyserver which enables other users to search the keyservers and download the public key material. According to [12], the first keyserver was developed by Brian A. LaMacchia in 1994 at MIT. The users interacted with the keyserver using an email interface, for example by submitting public key material by email or by sending in the command to search for a public key. A couple of years later, in 1997, the PGP Public KeyServer Project was started by MIT student Marc Horowitz. With newer implementations of keyservers, it is possible to request or publish keys via a web interface or using client applications that accesses the keyserver using the OpenPGP HTTP Keyserver Protocol (HKP, [13]). In addition key material can also be modified, for example if certifications are added to a key (if the key expiry date gets extended or if a key gets revoked) the updated key material can be uploaded to the keyserves.

In the early 2000s, Yaron M. Minsky from Cornell University started writing the SKS keyserver, an OpenPGP keyserver that synchronizes the data with its peers [14]. The idea behind these servers is that they synchronize in order to make all certificates available in all keyservers [15]. Because of the design of the SKS keyserver network, which aims at censorship resistance, once a key is published, it is not possible to remove the key data from the network. Furthermore the submission of key data is not authenticated, which means anyone can add additional signature packets to a public key and upload these signatures to the keyserver network. In spring 2019, this led to an attack on OpenPGP keys of prominent figures of the OpenPGP ecosystem, which were spammed with between 55.000 and 150.000 signatures [16].

Therefore alternative key distribution mechanisms have being evaluated. In 2019, a non distributed keyserver was set up, which strips third-party signatures during upload and only publishes UserID packets when users consent to it [1]. In addition, a multitude of other approaches have been devised (see e.g. [17]), also utilizing infrastructure based on blockchain technology [18]. It is also possible to store OpenPGP certificates in DNS, as specified in RFC 7929 [19] or use a Web Key Directory as described in a draft for an OpenPGP Web Key Directory standard [20].

Due to the fact that the SKS keyservers do not delete any keys and the network collected key data for more than 10 years, the keys on these servers provide access to over 5 million public keys. This collection of keys is publicly available to make it easier for new keyservers to initialize the keyserver data store and join the network. By downloading this data set, parsing and analyzing the keys and their attributes, this research provides a valuable insight in evolution of a widely used security standard.

[1] https://keys.openpgp.org/about/privacy

# 3 Related Work

A very early statistical analysis of the Web of Trust was done by Neal McBurnett in [21] in 1997: Back then only about 5.700 public keys and about 100.000 signatures were part of his data set. In [22], Čapkun et al examined the PGP ecosystem from the late 90s and describe it as a *self-organized system* and compare the PGP certificate graph to small-world characteristics. Boguñá et al analysed the Web of Trust in 2004 [23], working with 191.548 keys acquired in July 2001 and filtered the signature graph for bidirectional signatures. This resulted in a graph with 57.243 keys, where the largest connected sub network contained 10.680 keys. Their analysis focused on the degree distribution, clustering, correlations, and community aggregation in social networks. In [24], the authors looked at the Web of Trust as a large directed graph, with the keys as vertices and the signatures as edges. Their data set contained approximately 24.000 keys and they used the user id when sorting the keys, especially the domain part of the email address in the user id fields. Other authors provided an overview over the key submissions in [25]: They looked at 3.526.080 primary keys with the overall majority being version 4 keys

In [26], the authors employed graph analysis to explore security related issues of the Web of Trust. They worked with a data set of 2.7 million keys and 1.1 million signatures. Their examination found 240.283 so-called *Strongly Connected Components* (SCC) with the largest SCC (LSCC) consisting of about 45.000 keys. When looking at the distances between keys and GnuPG's limit on a certification path (which is 5 steps from one key to another), their findings indicate that a very large fraction of the keys in the LSCC can be sufficiently verified by the signature chains within GnuPG's restrictions. Their research also focuses at cryptographic algorithms used in the Web of Trust. The findings list for example 398.849 occurrences of SHA1 and 41.700 occurrences of MD5.

In [27], Barenghi et al in 2015 analyzed 3.8 million keys. They note that the number of revoked keys is rather small, and the number of expired ones is almost negligible. In particular, they ascertained that 99.6 % of the primary public keys do not have an expiration date set, which may be ascribed to the optional nature of the expiration date field. The research also examines possible key material issues which might lead to compromise and it gives an overview of the amount of affected keys. In particular the RSA key material is tested for outdated key sizes, prime moduli and common primes. Another research-focus are the employed asymmetric key cryptosystems for OpenPGP signatures. The paper states that ECDSA signatures account for a negligible portion of the signatures and explains this by the only recent introduction of ECDSA in OpenPGP.

In [28], the authors do not look at all the keys on the key servers or the Web of Trust, but at the much smaller, and curated, key ring of the OpenPGP keys of Debian Developers. This key ring was analysed over the span of 6 years, between 2009 and 2016. Their researched focused on providing a graphic representation of the key ring at various points in time with the idea to get better insight in growth and evolution over time. A focus of the research was also a migration of the Developers keys away from 1024 bit keys to at least 2048 bit keys that started in 2010.

In [11], the authors searched the key servers for OpenPGP signatures that use the same $r$ value in DSA signatures (a duplicate $r$ value is a known weakness of DSA and ECDSA - it can occur, when two DSA signatures use the same $k$ value, which should always be unique), but they found only one signature with that weakness. Furthermore, they also looked for RSA keys that share their prime factors and found a couple of keys that matched. Table 1 gives an overview on the relevant work heavily related to our work in this paper.

| Author | Title | Graph based |
|---|---|---|
| McBurnett [21] | Pgp web of trust statistics | No |
| Čapkun et al [22] | Small worlds in security systems: An analysis of the pgp certificate graph | Yes |
| Boguñá et al [23] | Models of social networks based on social distance attachment | Yes |
| Jörgen Cederlöf [24] | Dissecting the leaf of trust | Yes |
| Kristian Fiskerstrand [25] | Openpgp key statistics | No |
| Ulrich et al [26] | Investigating the openpgp web of trust | Yes |
| Barenghi et al [27] | Challenging the trustworthiness of pgp: Is the web-of-trust tear-proof? | No |
| Wolf et al [28] | Progression and forecast of a a curated web-of-trust: A study on the debian project's cryptographic keyring | No |
| Hanno Böck [11] | A look at the pgp ecosystem through the key server data | No |

Table 1: Related Work

# 4 Approach

The analysis in this work is split into three parts, every part addressing a different attribute of OpenPGP keys, one for public key packets, one for user ID and user attribute packets and one for signature packets. As noted above, to make it possible for new keyserver operators to join the network, various keyserver dumps (a keyserver dump is a collection of files containing concatenated OpenPGP public keys) are provided online and regularly updated. This work is based on a database dump from June 2019, shortly before the attack on the OpenPGP ecosystem, in which the keyservers where flooded with thousands of bogus certifications [16]. The acquired archive was around 12GB in size and contained 5,499,675 keys.

After retrieving the data from the database, the data was parsed using the `pgpdump` Python library and written to CSV files. The goal of this step was to extract the needed information from the base64 encoded strings before analysis, thus the rather computation intensive part of parsing was done only once and not every time the analysis steps were executed.

For the analysis of the data, the interactive computing environment "Jupyter Notebook" together with the Python data analysis library `pandas` was used. For every step of the research, the parsed data was first loaded into pandas to basically do a quantitative analysis. Then clusters of data points with common attributes were analysed over time and correlations with historical data were identified. For the temporal analysis, the data set had to be cleaned from obviously faked packets, which is indicated in the respective sections of the research. Following, another quantitative analysis of characteristics was made on the clusters and the results of the clusters were compared with each other. The sub-clusters were also plotted over time, to compare developments. Plotting over time also allowed to find spikes in the evolution of data points, which were further investigated.

# 5 Results for Public Key Packets

As described in Section 2, an OpenPGP Key consists of OpenPGP packets of different types. Public Key Packets adhere to the Public Key Packet Format as described in RFC 1991 [3, ch. 6.6], RFC 2440 [5, ch. 5.5.2] or RFC 4880 [6, ch. 5.2.2]. They contain information about the version of the packet, the time of

creation, the public-key algorithm and the key material itself and in packets of version 2 and 3 also the validity of the key. A public key packet can have Tag 6 set if it is a primary or top-level key and Tag 14 if the public key packet contains information about a sub key which is associated with a primary key.

## 5.1 Primary Key Public Key Packets

Of the 5,499,675 OpenPGP keys in the data set 5,499,572 of the public key packets of the primary are recognized as usable keys by the Python `pgpdump` library. The relatively small number of keys that could not be properly parsed were discarded from this chapter of the analysis. The vast majority of the Public Key Packets are Version 4 Packets, but there are also some Version 3 and even Version 2 Public-Key Packets in the data set. Table 2 shows the distribution of versions. These numbers resemble the results Kristian Fiskerstrand made in his analysis of OpenPGP keys in 2014, where he calculated that the majority were Version 4 (94.74%) keys, Version 3 keys homing in at 4.73% and Version 2 keys at 0.53%. [25].

| Version | Quantity | Percentage |
|---|---|---|
| 2 | 18608 | 0.34 |
| 3 | 167318 | 3.04 |
| 4 | 5313646 | 96.62 |

Table 2: Distribution of OpenPGP versions

Subsection 2.5 lists the asymmetric algorithms, OpenPGP Public Key Packets can use as well as their associated ID numbers. 7 of the public-key packets have an unknown algorithm ID and 36 have an algorithm ID out of the private/experimental range. Because the goal of this research is to analyse real world use of OpenPGP keys, and keys with experimental or unknown algorithms usually cannot be used, these keys were removed from the data set. This leaves 5,499,529 in the data set.

## 5.2 Version 2 Packets

Before being able to study the development of the OpenPGP versions over time, the data set has to be cleaned of key packets with obviously incorrect or faked timestamps. The data set was acquired on 20th June 2019, so any public-key packets having creation timestamps after this date were removed from further analysis. The first RFC to describe a public key packet was RFC 1991, which was published in 1996 and listed 2 and 3 as possible version numbers. The RFC states about PGP versions, that ”versions 2.6 and 2.7 conform to this specification. Version 2.3 conforms to this specification with minor differences.” [3, ch. 1]. According to the book ”PGP Pretty Good Privacy”, which was published in 1995, the first release of PGP 2.0 was in fall 1992 [2, p. 103], so it is safe to assume that there are no legit public-key packets with Version 2 (or above), that were created earlier than that date. Therefore any public-key packets with a creation date before 1.1.1992 were excluded.

Figure 2 shows the creation of Version 2 Public-Key Packets since 1992. In 1994 there is a spike in key generation. This can be explained by the release of a RSA implementation by the company RSA Data Security, Inc. (RSADSI), which held the patent for the public-key cryptography algorithm RSA. When Phil Zimmermann released PGP in 1991, he implemented the RSA algorithm without having licensed the patent from RSADSI and so it was not legally possible to distribute or use PGP in the US. But in March 1994 RSADSI released version 2.0 of RSAREF, a reference implementation of RSA which allowed the use of the patent in noncommercial programs. Together with academics from MIT, Zimmermann adapted PGP to use the RSAREF implementation and MIT began to function as the official distributor of PGP. This version was then released at the end of May 1994 as PGP 2.6 and despite export restrictions

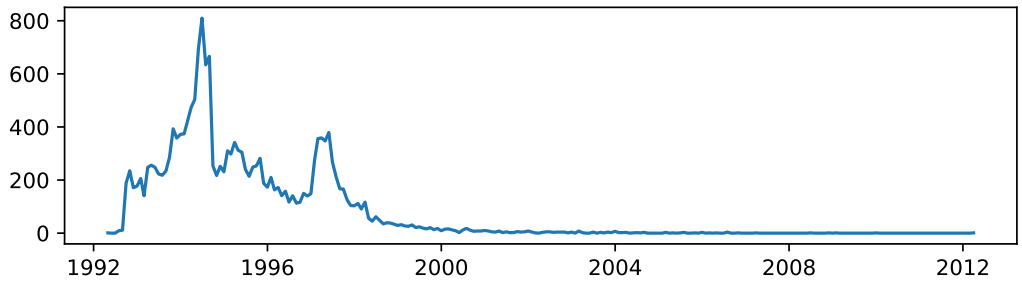



Figure 2: Creation of Version 2 PGP Keys since 1992

soon found its way on to European FTP servers. Because it was the first version of PGP which was not infringing patents, this led to an increased amount of new PGP keys in the following months.

When Zimmerman wrote Version 2 of PGP, he only implemented the RSA algorithm. RFC 1991 is a description of the message format of PGP packets and therefore only standardizes the use of RSA [3, ch. 6.6]. This also means that all Version 2 PGP keys are RSA keys.

Looking at the distribution of key lengths of the RSA modulus of Version 2 Public Key Packets in Figure 3, a clear majority of 1024-bit keys can be seen. According to Bruce Schneiers book ”Applied

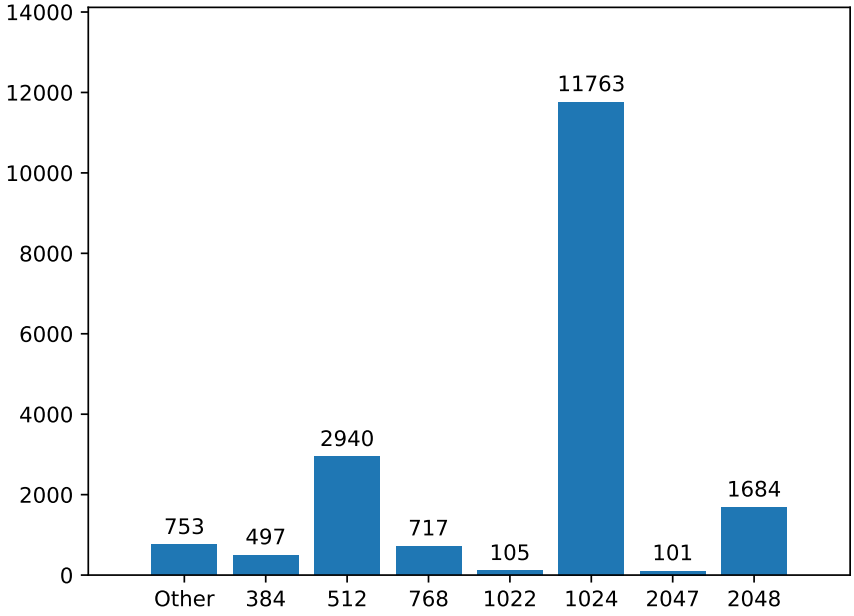



Figure 3: Distribution of keylength of Version 2 RSA public-key packets

Cryptography” a 1024-bit modulus was deemed sufficient by an expert committee at a European Institute for System Security workshop in 1991[29, p. 473]. In his book ”PGP Pretty Good Privacy” Simson Garfinkel depicts a key generation dialog from PGP version 2.6, which provides the options to create 512-, 768- or 1024-bit keys and he recommends to create a 1024 bit key (in this dialog 1024 bit is being described as to be ”Military” grade, slow and of highest quality) [2, p.163]. Both the estimates of experts

and the predefined options surely influenced the user choice of key lengths.

### 5.3 Version 3 Packets

When Zimmermann and the team from MIT adapted PGP in 1994 to use the RSAREF implementation, they had to do so under one condition: RSADSI wanted a guarantee that users had to upgrade to the RSAREF based release of the software. The deal was, that PGP would start to produce packets with version number 3 after September 1st 1994 [30]. This design decision is observable in Figure 2, where after the spike of key creation a sudden drop in September 1994 appears. Nonetheless, one can also see that it took until the end of the 1990s to get rid of software that generated Version 2 keys. The move to Version 3 packets is reflected in Figure 4 as well, which illustrates the evolution of Version 3 PGP keys. Version 3 keys were, like Version 2 keys, only RSA keys [5, ch. 5.2.2]).

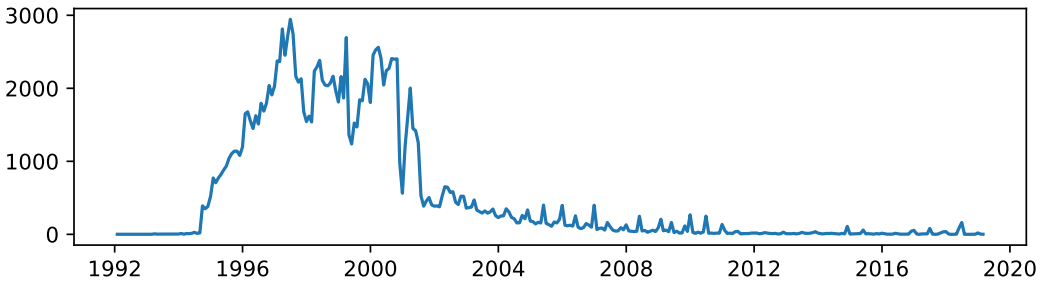



Figure 4: Creation of version 3 keys since 1992

When comparing the distribution of key lengths in version 3 public-key packets (Figure 5) with the distribution in Version 2 public-key packets (Figure 3), a move to bigger keys is noticeable. There are nearly as many 2048-bit keys as 1024-bit keys (44.85% 1024-bit keys versus 42.17% 2048-bit keys). In addition, the number of keys with a modulus less than 1024 is smaller, but people were already starting to use 4096-bit keys.

A side by side comparison of the chronological development of 2048-bit keys and 1024-bit keys in Figure 6 shows, that while 1024-bit lost its popularity at the end of the 1990s, 2048-bit keys got more favored until around 2002, a time when version 3 keys started to get less and less used altogether.

The validity of public keys with version 2 and version 3 public-key packets is defined by a number in the public-key packet representing the number of days the key should be valid from its creation. Setting this should be good practice, as no key should be used indefinitely and it should expire automatically [29, p. 161 ff]. Still, of the 185,926 keys in the data set that consist of Version 2 or Version 3 Public Key Packets, 171,947 have *no* key expiration set (92.48%). From the remaining keys, most have the validity set to 1 year, Table 3 lists the 5 most used validities. All of those keys are expired by now.

| Validity (in days) | Count |
|---|---|
| 365 | 3072 |
| 366 | 1391 |
| 1 | 458 |
| 31 | 424 |
| 30 | 236 |

Table 3: Most used validities of version 2 and version 3 public keys

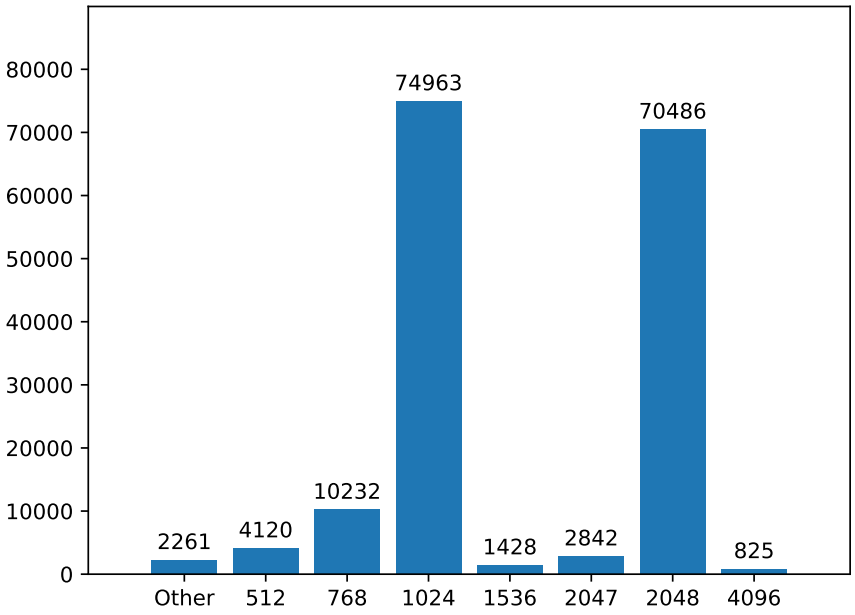



Figure 5: Distribution of key lengths of version 3 RSA public-key packets

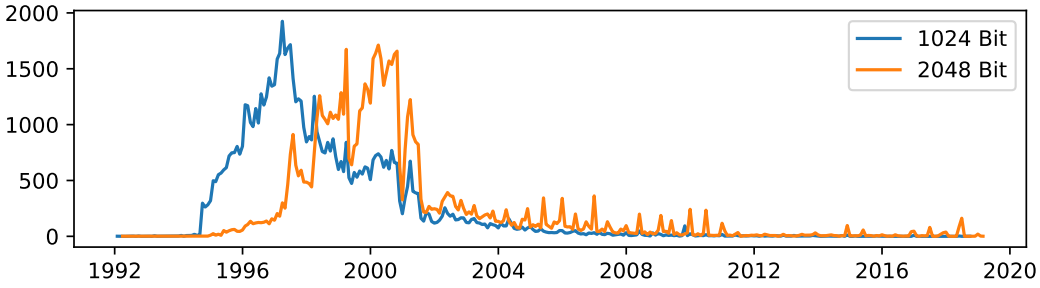



Figure 6: Comparison of version 3 1024-bit versus 2048-bit public-key packets

## 5.4 Version 4 Packets

In September 1997 a working group at IETF was initiated to work out the first *OpenPGP* standard which in November 1998 was published as RFC 2440 [5]. This was the first standard that defined OpenPGP packets with Version 4 and it was the first time that RSA was not the only Public Key algorithm, in fact supporting RSA was not even a requirement for implementations. NIST on the other hand had proposed DSA for the Digital Signature Standard and made it a federal standard (FIPS 186 [31]) in 1994. Thus RFC 2440 lists DSA as public-key algorithm with packet ID 17 and it also lists ElGamal, and this one both for encryption-only (packet ID 16) and for encryption and signing (packet ID 20) [5, ch. 9.1]. The RFC states that implementations must implement DSA for signatures and ElGamal for encryption [5, ch. 9.1] and regarding the length of DSA keys implementations the standard enforces a minimal key size of 768 bits, but recommends 1024 bit keys for long-term use. [5, ch. 12.6]. RSA keys on the other hand should not have a key size less than 768-bit too [5, ch. 12.4] and ElGamal keys should not be less than

768-bit, but for long term security the standard recommends 1024-bit or longer [5, ch. 12.5].

The first software to produce Public Key Packets with Version 4 was PGP 5. PGP 5 was one of the successor releases of Zimmermans PGP and was released in May 1997 by the company PGP Inc. For this analysis of OpenPGP Version 4 Public Key Packets, any packets created before 1997 were removed, because their creation date was most certainly faked.

Figure 7 shows the development of Version 4 public-key packets over the last 22 years.

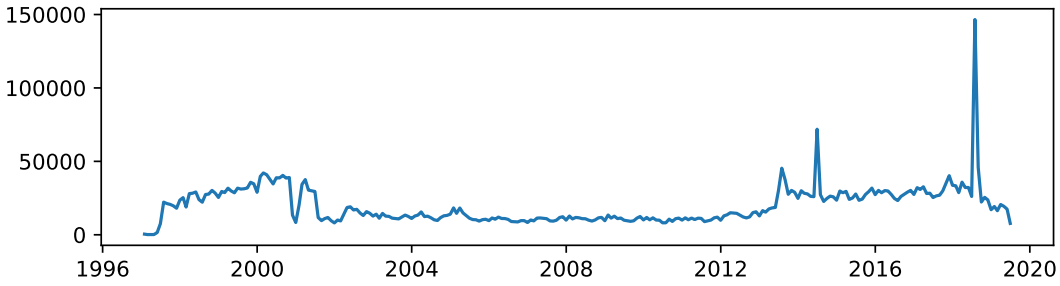



Figure 7: Creation of Version 4 Keys since 1997

There are a couple of significant spikes in Figure 7: The first spike is in the year 2013, during June. This was the time when the Guardian started publishing the documents Edwards Snowden leaked and specifically wrote about the PRISM program to collect material including the content of emails. [32]. Snowden also confirmed in an interview with The Guardian, which was published 2013, that properly implemented strong crypto systems work against the NSA and that the major problem was weak endpoints [33]. Later it was revealed that Snowden also tried to use OpenPGP to communicate with the journalists Glenn Greenwald and Laura Poitras [34]. It seems likely, that the news coverage of the surveillance efforts of the NSA and the Five Eyes led to an increase in usage of OpenPGP software and thus an increase in new keys.

Another spike of new keys happened on 16th of June 2014. The amount of keys with this creation date can be attributed to the Evil32 research project [35]. The goal of this experiment was to prove that the ongoing use of short key IDs was not secure anymore, because short key IDs are not collision resistant and that instead the fingerprints should be used as identifiers. RFC4880 describes the Key ID as the low-order 64 bits of the fingerprint and the version 4 fingerprint as ”the 160-bit SHA-1 hash of the octet 0x99, followed by the two-octet packet length, followed by the entire Public-Key packet starting with the version field” [6, ch. 12.2]. In addition to the key ID there is the short key ID, which is the lowest order 32 Bit of the fingerprint. Short key IDs are not standardized in the RFC, but were used in a lot of implementations to identify keys. To prove their point, the researchers chose to create a copy of the strongset, which is the LSCC of the keys in the Web of Trust. Their copy of the strongset had the same UserIDs and the same short key IDs as the keys in the real strongset. Of the 23,844 keys the Evil32 project created, 21,532 have 16th of June 2014 as their creation date.

The biggest spike happened in July 2018, more precisely on 21st and 22nd of July. On May 25 2018 the GDPR became effective and the rise in discussions around and awareness of data protection and privacy lead to some users criticizing the SKS keyserver network for not being GDPR compliant. At the end of May 2018, a user with the nickname *yakamo* published a proof of concept program called keyserver-fs [2] that uses UID Packets of OpenPGP keys to store arbitrary data (by using multiple UIDs which can each have up to 2048 characters). In an article, the author explained that they wrote the

[2]https://github.com/yakamok/keyserver-fs

software to highlight how easy PGP Key-Servers can be abused [3]. On July 21st and 22nd 2018, someone published more than 100,000 keys that seem to be created by some fork of the *keyserver-fs* tool. The keys use a different format for the data stored in the UID but they have some similarities with the keys created by *keyserver-fs* (they are all RSA keys with 1024-bits key length. They all have one or more UserIDs that consist of strings of uppercase letters of 30 characters. The last of these UserIDs ends with a string formatted like an email address).

In December 1999 the OpenPGP Working Group at IETF started working on an update to RFC 2440. This update took until November 2007, when it was published as RFC 4880. Though there were no mentionable changes in the format of public-key packets, the list of public-key algorithms changed insofar that ElGamal signatures were not longer permitted and package Tag 20, which before was assigned for ElGamal signatures, now became a reserved tag. This was a consequence of Bleichenbachers paper "Generating ElGamal Signatures Without Knowing the Secret Key" [36].

| Name | Quantity | Percentage |
|---|---|---|
| DSA | 2674573 | 50.41 |
| RSA | 2617867 | 49.34 |
| RSA Sign-Only | 5583 | 0.11 |
| EdDSA | 4299 | 0.08 |
| ECDSA | 2556 | 0.05 |
| Elgamal Encrypt or Sign (depr.) | 1009 | 0.02 |
| Elgamal (Encrypt Only) | 10 | 0.0 |
| ECDH | 2 | 0.0 |

Table 4: Distribution of algorithms in OpenPGP Version 4 Public-Key Packets

Table 4 shows the distribution of algorithms of Version 4 OpenPGP Public Key Packets. RSA with algorithm ID 1 and DSA with algorithm ID 17 are used the most, together they amount to 99.75% of the keys. When looking at the development of RSA and DSA public-key packets over time in Figure 8, a segmentation is noticeable.

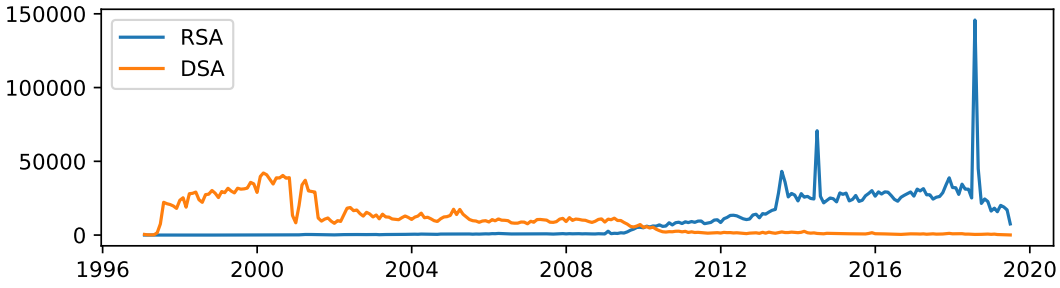



Figure 8: Comparison between development of RSA and DSA OpenPGP Public-Key Packets Version 4

DSA keys were created from the time when PGP 5 was released in the 90s until around 2010. Around the same time, RSA keys started to get more used and then more or less replaced the DSA Keys. This can be attributed to a couple of factors: first there was the already mentioned patent on RSA which ran out in September 2001. GnuPG first implemented RSA key generation with version 1.0.7, which was released in April 2002 and it changed its default to RSA 2048 in 2009 with version 2.0.13 [37]. Second, there were rising security concerns with DSA. The original FIPS 186 [31] only standardized DSA with

[3] https://medium.com/@mdrahony/are-pgp-key-servers-breaking-the-law-under-the-gdpr-a81ddd709d3e

maximal 1024-bit, which was considered to be to short, as computational resources improved. In NISTs "Recommendation for Key Management - Part 1: General", which was released 2007, a key size of 1024-bit is only recommended for a security lifetime through 2010 [38]. Another factor was that in the original Digital Signature Standard (DSS) publication, DSA was standardized together with SHA-1, which was a popular hash function for a long time, but in 2005 the first weaknesses in the algorithm were published. A third factor that facilitated the adoption of RSA as preferred Pubic-Key Algorithm was the increased use of smart cards, which had much better support for RSA than DSA.

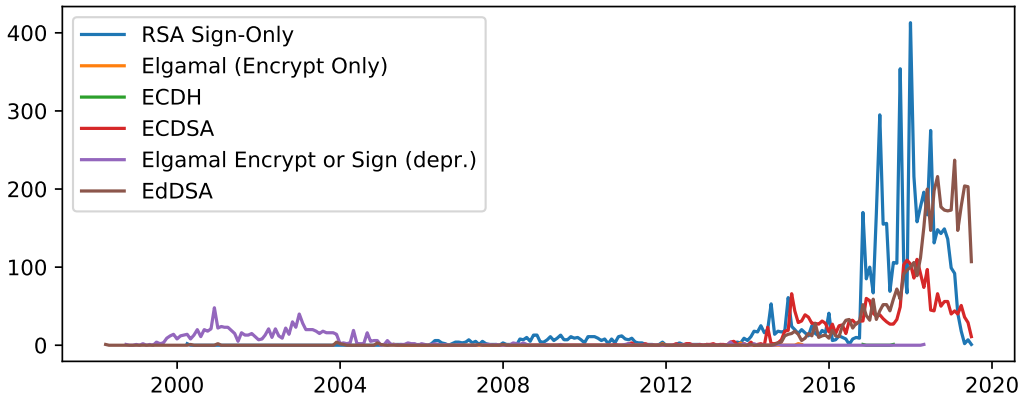



Figure 9: Comparison of development of Version 4 Public-Key Packets with uncommon algorithms

When looking at development over time of the not so common algorithms in Figure 9 the first thing to notice is that keys with algorithm ID 16 (ElGamal (Encrypt or Sign)), did get used at the beginning of the 2000s, but not for a very long time - it was only used by around 1000 keys and that only until 2005. In 2002 GnuPG decided to hide the option to create ElGamal (Encrypt or Sign) keys and make it only possible to create such keys using expert mode. Considering that Bleichenbachers attack on ElGamal signatures was released in 1996, it is remarkable that these keys were still generated and it is beneficial that RFC 4880 forbid implementations to generate such keys. Figure 9 also shows an increase in both ECDSA and EdDSA keys.

In 2008 PGP Corporation started to work on an extension of RFC 4880, with the goal of defining Elliptic Curve Cryptography for OpenPGP. The draft was approved in 2012 as RFC 6637 [9] and defined ECDH with algorithm ID 18 and ECDSA with algorithm ID 19. The draft for the next OpenPGP standard, RFC 4880bis [10], incorporates these extensions of the algorithm list and adds the Edwards-Curve Digital Signature Algorithm (EdDSA) as defined in RFC 8032 [39]. In GnuPG the support for RFC 6637 was implemented in 2011, but only released in 2014. As usual with software, it took some time until the release reached the users - for example the Debian Linux distribution only shipped GnuPG with ECC support in 2017 with the release of Debian 9.0. Another fact is that for users a full switch to ECC based keys makes only sense if all the communication parties involved also have support for ECC on the software side. This is why an increase in ECC based keys is only observable in the last few years.

Last but not least there are a lot of public-key packets created since around 2014 with algorithm ID 3, which is RSA Sign-Only, with a big increase since 2017. This is noteworthy, since these keys are deprecated and RFC 4880 clearly states that implementations should not create such keys [6, ch. 13.5]. Of the 5,583 RSA Sign-Only keys, 2,966 keys have a UserID packet of `noone@proofmode.witness.org`. ProofMode is an Android Camera app created by the Guardian Project that allows to share media files and which supports chain of custody needs using OpenPGP signatures [4]. 507 of the RSA Sign-Only

[4]https://guardianproject.info/apps/org.witness.proofmode/

keys have the string `OnionMail Server` as part of their UserID. OnionMail is a mail server run via the TOR network supporting encrypted and anonymous mail [5]. What ProofMode and OnionMail have in common is that they are both written in Java and rely on the Bouncy Castle library (resp. Spongy Castle, an Android version of the Bouncy Castle library), a widespread collection of cryptographic APIs for Java. When doing an online search for "Bouncy Castle RSA key generation" one tends to stumble over the "Bouncycastle PGP Cookbook", a blog from 2013 that provides a code example for creating an RSA key with Java [6]. The code uses both `RSA_SIGN` and `RSA_ENCRYPT` for the algorithm identifier, which correlates to the algorithm IDs 3 and 2. Both ProofMode[7] and OnionMail make the same mistake when generating RSA keys and it seems likely that also other Java programmers have based their code on the "Bouncycastle PGP Cookbook".

When looking at the bit sizes of the DSA and RSA keys in Figure 10, the difference between DSA and RSA is evident, as nearly all the DSA keys are 1024-bit keys. When FIPS 186, which allowed larger DSA keys, was adopted, RSA was already gaining ground, which meant that DSA with 2048 bits or even more didn't really get used. Regarding RSA, there is more fragmentation. As mentioned above, GnuPG defaults to 2048 bit RSA keys since 2009. Enigmail, one of the most widespread graphical interfaces for GnuPG, changed the default to 4096 in 2014 [40]. The big share of 1024 bit RSA keys can be attributed for a good deal to the aforementioned keys uploaded in July 2018. Overall, there is a trend towards bigger RSA keys. Figure 11 shows how 4096 RSA keys started to replace 2048 RSA keys in 2015.

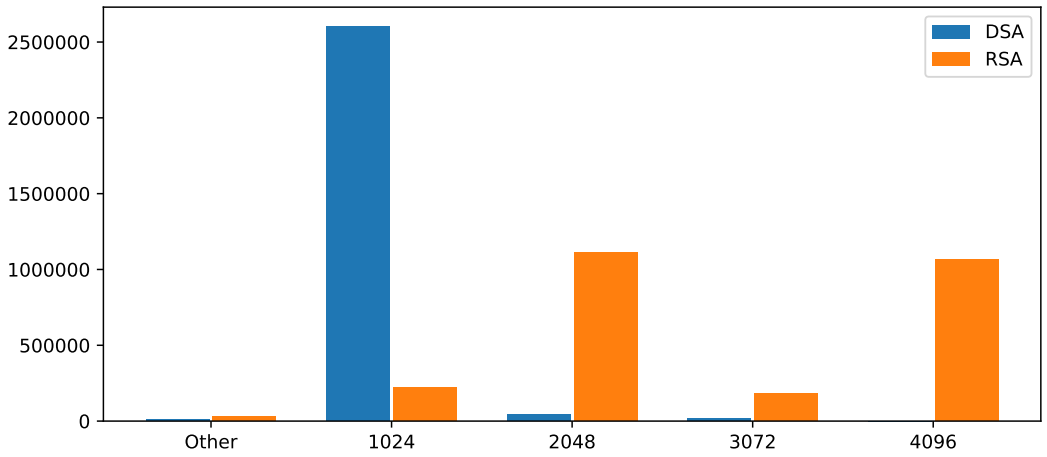



Figure 10: Distribution of Bitsizes for Public-Key Packets Version 4 RSA and DSA

## 5.5 Public Key Subkey Packets

RFC 2440 recommends to use different keys for encryption and signing. In fact, this was one of the motivating forces behind the V4 key format with separate signature and encryption keys" [5, ch. 13]. Of the 5,499,675 keys in the data set, 4,932,754 contain one or more subkeys, totalling to 5,286,449 subkeys. One key with 280 subkeys is the key with the most subkeys, but most of the keys (94.24%) have exactly one subkey. That is not surprising, as for a long time the default was to generate a primary

[5] http://en.onionmail.info/what.html
[6] https://bouncycastle-pgp-cookbook.blogspot.com/2013/01/generating-rsa-keys.html
[7] The bug in proofmode was reported at https://github.com/guardianproject/proofmode/issues/65

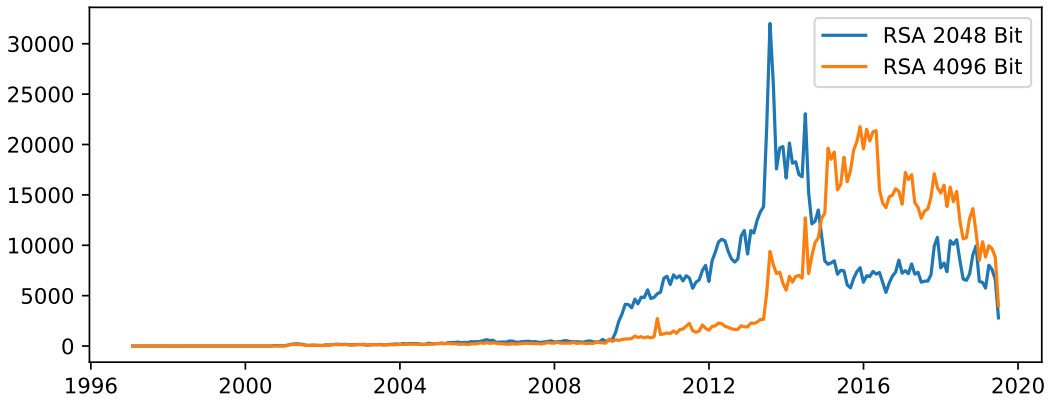



Figure 11: Creation of 2048 and 4096 Bit RSA keys over time

key for signature operations and a subkey for encryption.

Of all the Subkey Key Packets, 1 has an unknown algorithm ID and 3 have an algorithm ID from the experimental range. After discarding Subkey Key Packets with either experimental or unknown algorithm IDs, as well as Subkey Key Packets with a creation time before the 1st of January 1997 (when subkeys were standardized) and after 20th of June 2019 (when the data set was acquired), there are 5,283,149 keys left in the data set. Figure 12 shows the evolution of subkey creation since 1997.

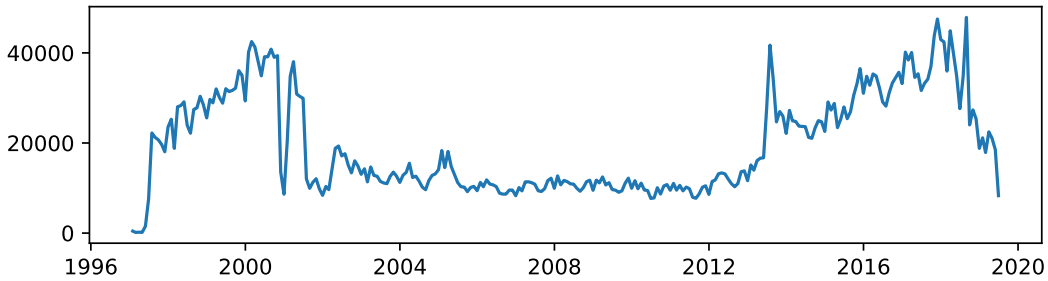



Figure 12: Creation of subkeys since 1997

One can see a similarity to Figure 7, which is due to the fact that most of the times subkeys and primary keys are created together. Whats missing are the two main spikes that represented the creation of the Evil32 keys and the keyserver-fs attack. From the Evil32 keys only 104 of the 23,844 keys were created with subkeys and the keyserver-fs attack did not use subkeys at all. Table 5 shows the distribution of algorithms in the Public Key Subpackets (4 subkeys with an unknown algorithm were omitted). Similar to the distribution of primary keys, there are two public-key algorithms that are used the most, namely RSA and ElGamal (Encrypt-Only). ElGamal (for encryption) is one of the public-key algorithms (together with DSA) implementations must implement according to RFC 2440 [5, ch. 9.1]. Figure 13 shows the evolution of ElGamal subkeys versus RSA subkeys in the last 22 years. One can see the resemblance to Figure 8. This is due to the fact that usually DSA primary keys were generated with ElGamal subkeys for encryption and RSA primary keys were created with RSA subkeys.

Looking at the not so common algorithms in Figure 14, one can see another similarity to primary keys, namely the rise of elliptic curve cryptography. Analogous to the increase of ECDSA and EdDSA in Figure 9, the subkeys show a rise in ECDH keys. This development is also due to RFC4880bis [10], which lists RSA and ECDSA as a MUST requirement for signatures and RSA and ECDH as a MUST

| Name | Quantity | Percentage |
|---|---|---|
| Elgamal (Encrypt Only) | 2691273 | 50.94 |
| RSA | 2566497 | 48.58 |
| DSA | 7787 | 0.15 |
| ECDH | 7347 | 0.14 |
| RSA Encrypt-Only | 5355 | 0.1 |
| EdDSA | 2593 | 0.05 |
| ECDSA | 1919 | 0.04 |
| Elgamal Encrypt or Sign (depr.) | 366 | 0.01 |
| RSA Sign-Only | 12 | 0.0 |

Table 5: Distribution of algorithms in OpenPGP Version 4 Public-Key Subpackets

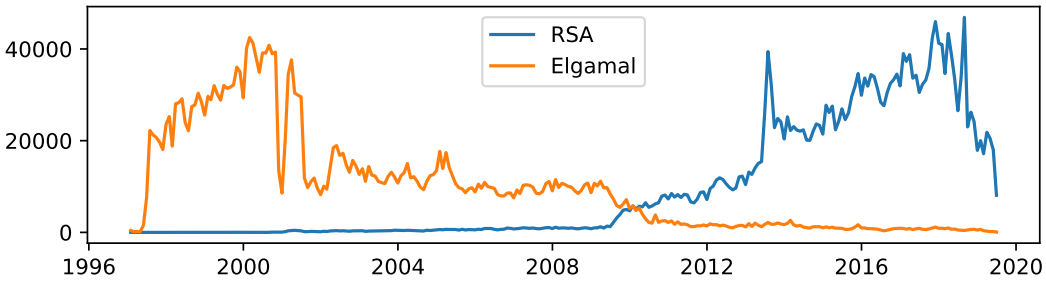



Figure 13: Development of RSA and ElGamal OpenPGP Public Key Subkey Packets

requirement for encryption. EdDSA is listed in RFC 4880bis only as a SHOULD requirement.

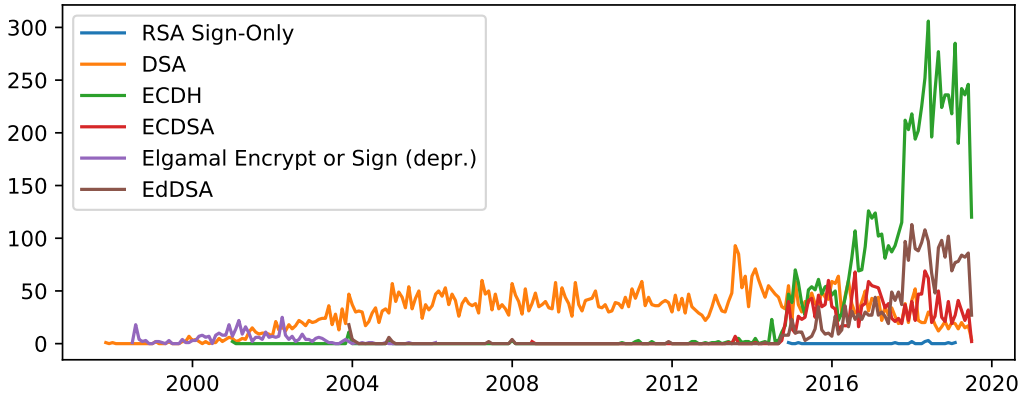



Figure 14: Comparison between development of not so common Algorithms in OpenPGP Public-Key Subkey Packets

Regarding the bit sizes shown in Figure 15 ElGamal shows a clear majority of 2048-bit keys. 2048 bits was the default bit for ElGamal subkeys when generating an DSA/ElGamal key pair with GnuPG for a long time. With RSA keys, the bit size of the keys is nearly evenly distributed between 2048- and 4096-bit keys. Figure 16 shows that RSA with 2048-bit keys was more common until around 2015 and then RSA with 4096-bit keys started to get used more. GnuPG defaults to RSA 2048, but recommends using elliptic curve algorithms in case more security is required [8]. Revision 3 of NIST 800-57 that RSA

[8] url="https://www.gnupg.org/faq/gnupg-faq.html

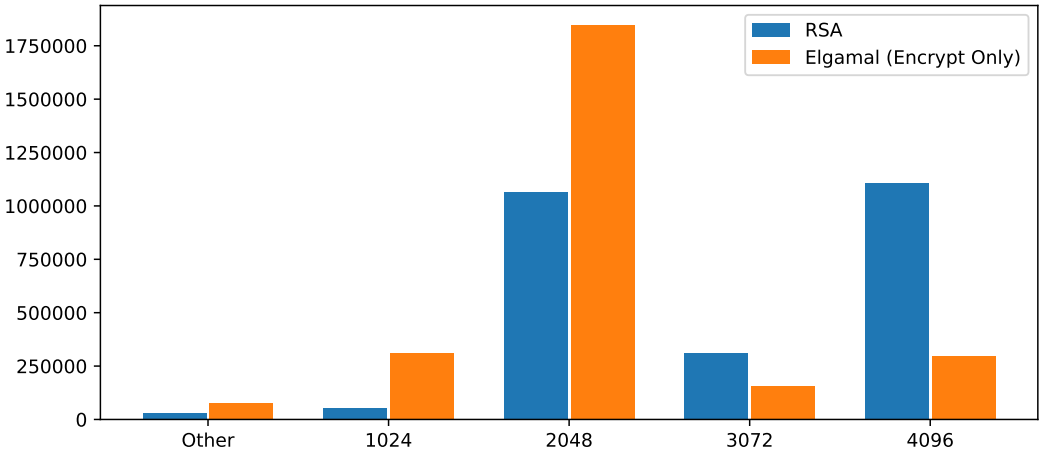



Figure 15: Distribution of bit sizes for Public-Key Subpackets Version 4 RSA and ElGamal

2048 bit is comparable with 112 bits of security, which, according to the publication, is acceptable from "2014 through 2030" [41]. The Debian Project on the other hand prefers 4096 bit keys.

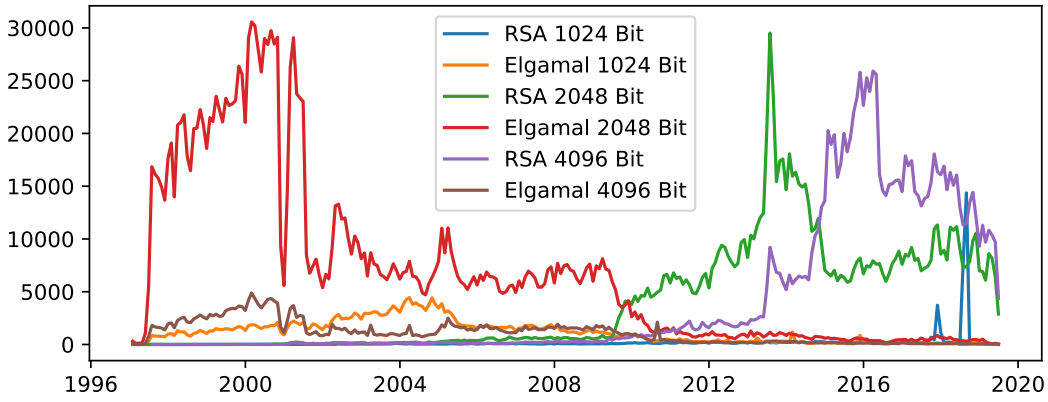



Figure 16: Evolution of bit sizes for Public-Key Subpackets Version 4 RSA and ElGamal

# 6 Results for User ID and User Attribute Packets

## 6.1 User ID Packets

To tie an OpenPGP key to an identity a User ID packet can be used. The User ID packet is part of the PGP specification since RFC 1991 [3, ch. 6.7] and the description only changed a little in wording in OpenPGP RFC 2440 [5, ch. 5.11] and RFC 4880 [6, ch. 5.11], where it is defined as a packet with Tag 13. The main difference between RFC 1991 and the newer RFCs 2448 and 4880 is that the latter specify the content of the packet to be UTF-8 text and the former defined it as printable ASCII. RFC 4880 states that by convention, the User ID Packet "includes an RFC 2822 [RFC2822] mail name-addr, but there are no restrictions on its content" [6, ch. 5.11].

Before taking a deeper look at the User ID packets in the data set and doing analysis, obvious fake User ID packets have to be removed. Overall, the data set contains 5,499,599 public keys with User ID packets. As described in Section 5.4, there have been attacks against the keyserver network in which the User ID packet was abused to store data. 113,366 keys were identified as part of the attack by comparing the UserID Packet with format the aforementioned keyserver-fs tool produces.These keys were removed, because they would distort further analysis of the User ID packets. Furthermore, the data set contained 1,542 keys with User ID packets that are bigger than 8,192 Byte (together around 500 Megabytes) and did not contain usable data. These UserID packets were removed before the analysis as well, reducing the set to 5,386,224 remaining keys. A vast majority of 92.42% of the OpenPGP keys have only one User ID associated with them. Figure 17 shows the distribution of the number of UIDs. 6,080,288 User ID Packets (97.29%) contain an email address.

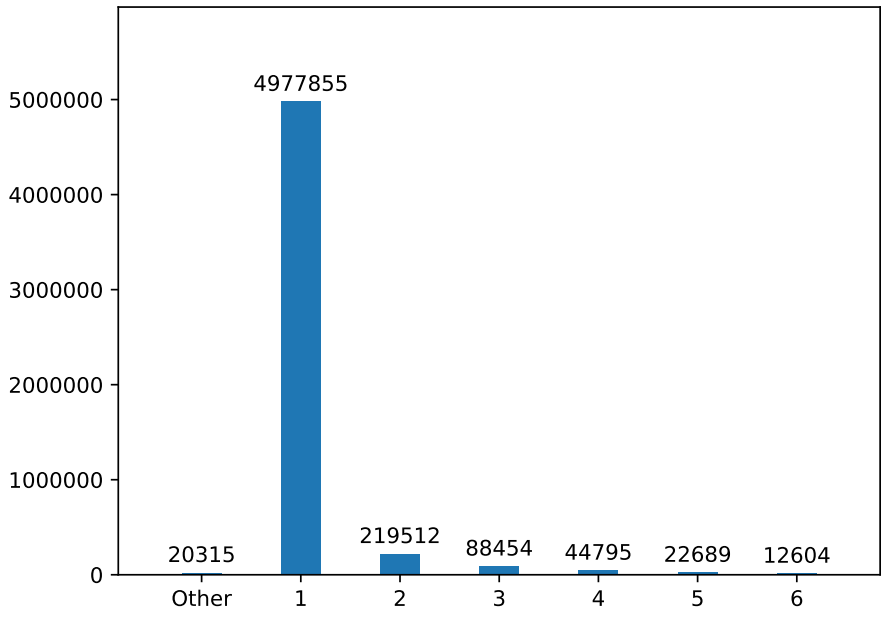



Figure 17: Overview of number of UIDs associated with OpenPGP keys

### 6.1.1 User ID Packets with Email Addresses

Of the 97.29% UIDs with email addresses, a big part (13.99%) are from the email service gmail.com. This should not be a surprise, with gmail.com being the biggest email service provider with more than 1.5 billion users [9]. Figure 18 shows an overview on the domains of email addresses in UID packets. Most of the used email providers are well-known and part of global businesses, with the exception of riseup.net, protonmail.com and tellfinder.com: Riseup.net is an Seattle based volunteer-run email provider with focus on privacy, Protonmail.com is also an email service with a focus on privacy, run by Proton Technologies AG, a company based in Switzerland. It shouldn't surprise, that among users of a privacy focused email providers there is a relatively large part of OpenPGP users. Tellfinder.com

[9] https://twitter.com/gmail/status/1055806807174725633/photo/1

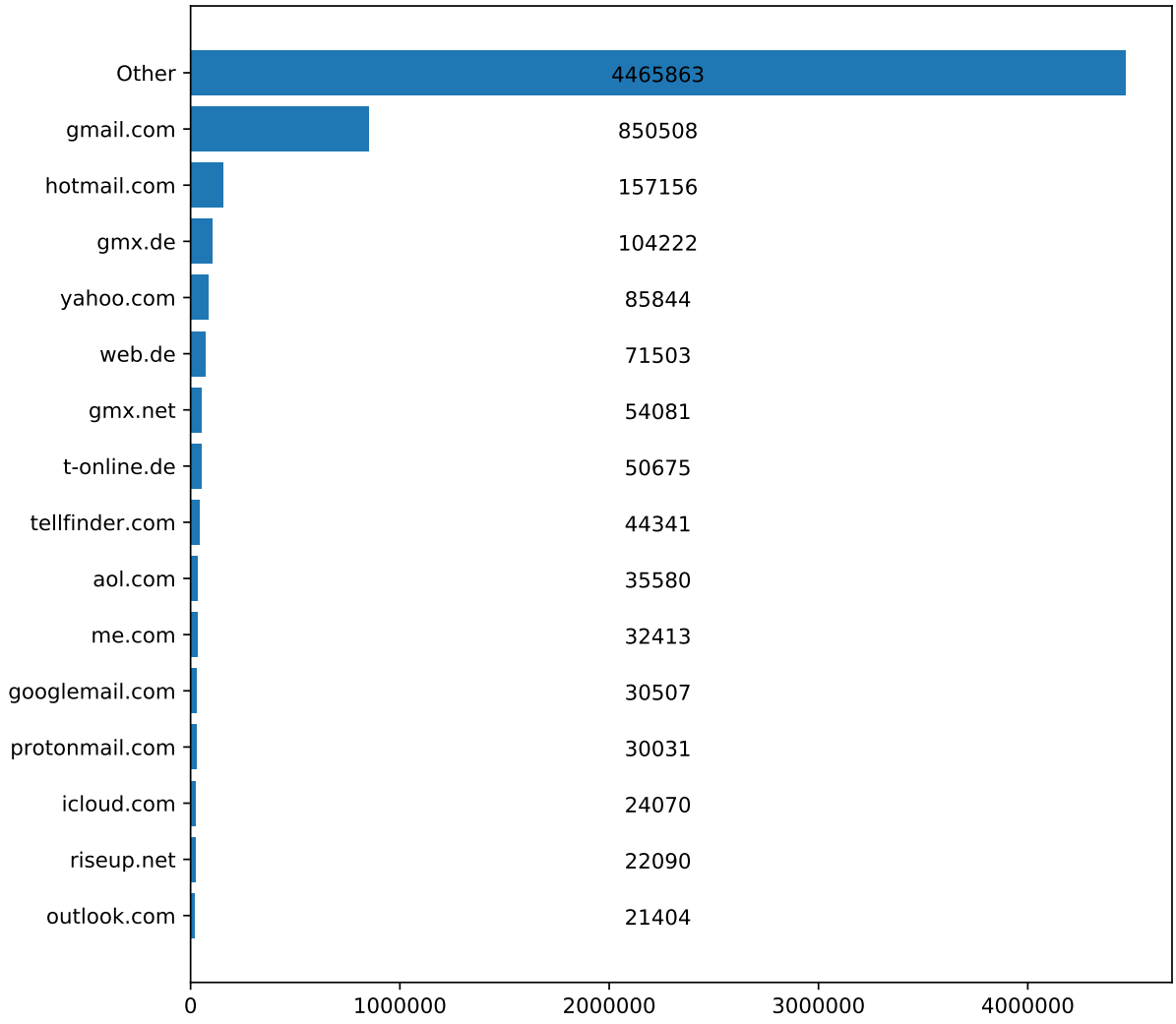



Figure 18: Overview of domains parts in email addresses in UID packets

on the other hand is not an email provider at all, but a software product by the Toronto based software company Uncharted designed to explore domain-specific web crawls. Whats interesting about these User ID Packets is, that all of them, except two, have the UID `TellFinder Page Archiver - Signing Key <support@tellfinder.com>`. Apparently the software uses OpenPGP keys to sign data. One of the keys with another UID String has the UID `TellFinder Page Archive - Root Key <support@tellfinder.com>` and all but 16 of the `TellFinder Page Archiver` keys are signed by this root key, which might be a sign of some kind of Certificate Authority.

Looking at the top level domains (TLD) of the domain parts of the email addresses in User ID Packets as listed in Figure 19, one can see that the `.com` is used in a big majority of keys (2,707,270). The `.de` TLD comes second, which is mostly due to the email provider `web.de` which is also listed as one of the major domains (71503 keys with an email address ending in `.web.de` out of 868,593 addresses ending in `.de`). Other bigger TLDs are `.net` with 579,989 keys, `.org` with 249,132 keys and `.edu` with 167,808 keys.

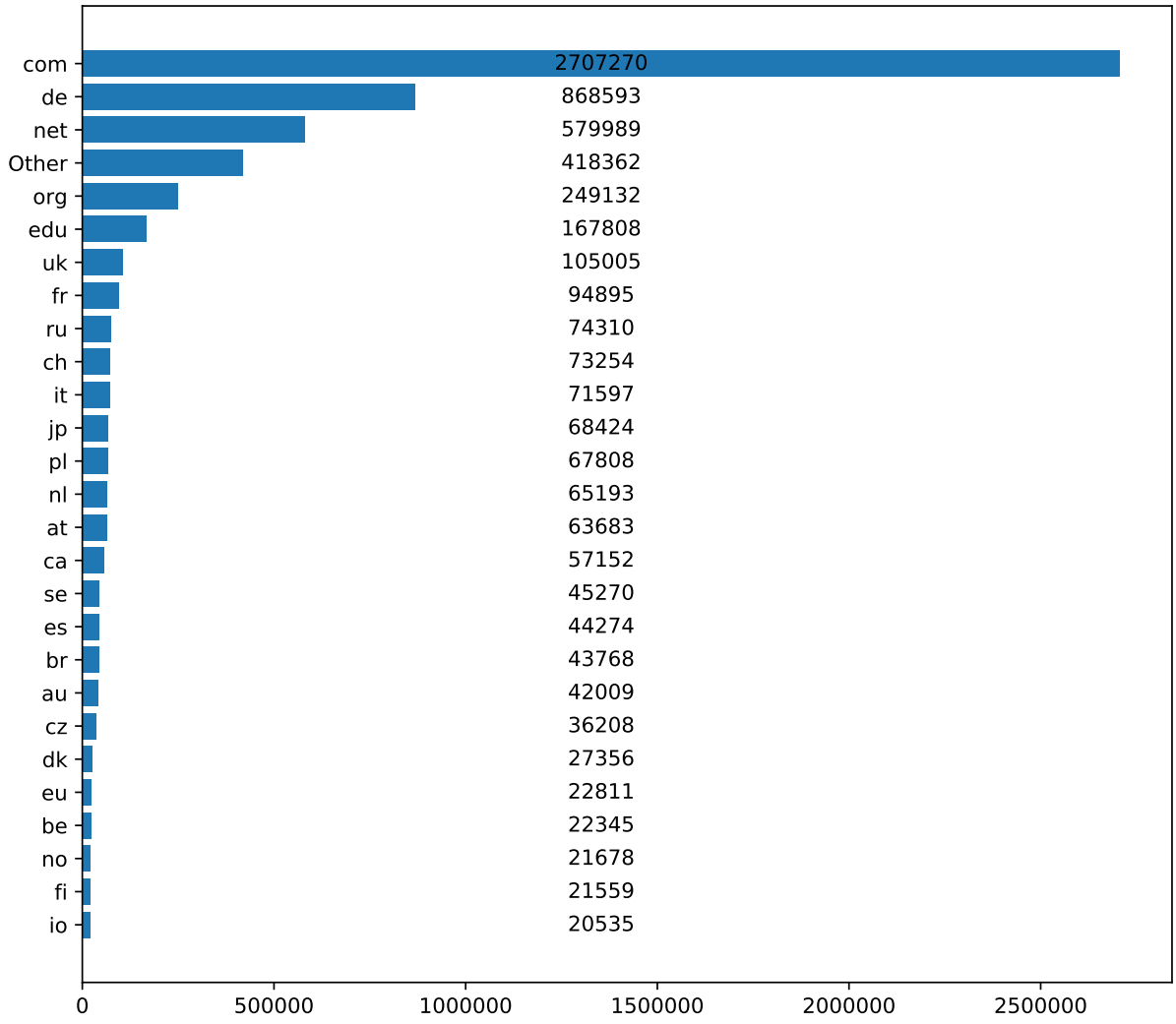



Figure 19: Overview of Top Level Domain in email address parts of UID packets

**User ID Packets and Software** The number of and the content of the User ID packets can also disclose information about the use and spread of software. The ”schleuder” mailing list manager [10] for example generates OpenPGP keys with 3 UIDs, with two of them having the suffixes `-request` and `-owner` added to the local part of the email address. Using these characteristics, it is possible to find 223 keys of schleuder lists in the data set, distributed over 60 domains. Another OpenPGP related software is *keybase*, a platform to map social media accounts to encryption keys. Every keybase user has a unique username on the platform and a lot of users also link their OpenPGP key to their profile by putting a URL to the profile (in the form of `https://keybase/username`) in the User ID packet of their key. The data set contains 4,829 User IDs containing a keybase URL.

**User ID Packets with Comment Fields** For quite some time, tools used to generate OpenPGP keys asked the user to provide a comment in a separate input field, which then was included in the the UserID string. This feature was controversial and in 2013 a blog post by Daniel Kahn Gillmor titled ’OpenPGP

[10] https://schleuder.org

User ID Comments considered harmful' [11] made the point that these user interface elements led to users entering text that does not contain any useful information or information that was already present in some other part of the key. Kahn Gillmor listed multiple examples of User ID comment types that are uncalled for, for example users simply entering `none`, users describing the parameters of the cryptographic material in the comment or using the comment field to repeat information that was already part of the name or email address component of the User ID field.

Of the User ID packets in the data set, 6,048,157 adhere to the RFC2822 format. 810,377 of those contain a comment field. In 2,296 User ID packets, the comment field contains `none` as comment and 2,539 have `no comment` as comment, which confirms Gillmors analysis. Using the `difflib` python library, it is possible to use pattern matching to compare the comments with other elements of the User ID field, like name or local or domain part of the email address. 13,133 comments are exactly the same as the name, 3,346 are the same as the domain and 18,246 comments are similar to the local part of the email address. Some users also tend to add information about the cryptographic primitives of the key in the comment field of the User ID packet. 709 keys contain the term `RSA`, like *This is my RSA key.*, *Super RSA key* or *SUSE GPG RSA 4096*. The same is the case with the term `DSA` which is present in 439 keys, with comments like *No expiry - 3072 DSA & El Gamal* or *Gmail DSA 2048-bit sign & encrypt*. This information should not be part of the User ID of a key and is already present in the attributes of the key itself. [12]. By now, the comment field has been removed from most of the OpenPGP related software.

### 6.1.2 User ID Packets without Email Addresses

Although the RFCs mention that a User ID packet usually contains an email address and a name, the standards don't force this format. In fact, some use cases of OpenPGP certificates have been developed, where the User ID packet must contain other data. One example would be the Monkeysphere project that aims at extending OpenPGP's web of trust to securely identify servers, as well as each other [13]. Using the scripts of the monkeysphere project, one can for example generate an OpenPGP public-key from an OpenSSH RSA server certificate. This public-key can then be published on the keyservers and users can authenticate a server using the OpenPGP web of trust. It is also possible for the server to authenticate SSH clients using the monkeysphere project and client-side authentication is also available for HTTPS. When creating the server key, one passes an identifier for the server, which consists of a *scheme* (e.g. ssh or https) and a fully-qualified hostname (and port) [14]. Indeed, of the 169,567 UIDs without an email address, 1,805 contain the scheme `ssh://` and 644 contain the scheme `https://`.

## 6.2 User Attribute Packets

In addition to the User ID packet, RFC 4880 also defines a User Attribute packet with Tag 17 [6, ch. 5.12]. Only 79,513 keys in the data set contain a User Attribute Packet, that's 1.445%. The User Attribute packet can contain one ore more attribute subpackets, but only one type of attribute subpackets is defined, which is the Image Attribute Subpacket, thus, this packet type is mostly used for storing images.

Of the keys in the data set that have a User Attribute Packet, 78,675 contain image data. All of the images are JPEG [42] images, which is the only image type RFC 4880 standardizes. 6 of the pictures contain images in the JPEG Multi Picture Format. The biggest picture is 6000x4000 pixels, the mean width of the pictures is 197.58, the mean height is 211.3. Of the remaining packets, 823 contain text that starts with the string `openpgpid+token:` which then in most cases is followed by an https URL.

[11] https://debian-administration.org/users/dkg/weblog/97
[12] https://debian-administration.org/users/dkg/weblog/97
[13] https://web.monkeysphere.info
[14] https://manpages.debian.org/buster/monkeysphere/monkeysphere-host.8.en.html

These packets were generated by the Android app `OpenKeychain` and the URI scheme is described in the IETF draft "Linked Identities for OpenPGP" [43]. The purpose of those URIs is to point to a *linked identity* that links the key to a resource on the internet in a mutual and verifiable way [43]. By now the draft expired and also the OpenKeychain app removed the code for handling such packets.

3 of the remaining user attribute packets contain an encrypted data packet that is encrypted for the key with the key id `A276228EC2F6A01C`, which has the user id `Crypto Message Board`. The three keys and the key the data is encrypted to all are 512 bit RSA keys and have a similar creation timestamp.

# 7 Results for Signature Packets

Signatures create a binding between some data and public key material. In the OpenPGP standards, a signature packet is an OpenPGP packet with Tag 2. RFC 1991 defined signature packets with the Versions 2 and 3, RFC 2440 and RFC 4880 dropped the Version 2 and added a Version 4 signature packet definition. The versions 3 and 4 have different formats - similar to how the public key packet changed from version 3 to version 4. In version 3 the creation time of the signature and the key id of the signer still were part of the signature itself. A version 4 signature packet on the other hand could contain "Signature Subpackets" and both the creation time of the signature and the key id of the signer were moved to signature sub packets with Type 3 and Type 28.

The data set contains 21,229,327 signature packets. Those are distributed over signatures on primary keys (391,873), signatures on sub keys (5,597,721), signatures on user id packets (15,028,832) and signatures on user attribute packets (210,901). As usual with OpenPGP packets, the data set contains signatures with fake timestamps. Similar to public key packets, all signature packets with a creation timestamp before 1992 and after the date the dump was made (2019/06/20) were removed before analyzing the data. The data set contains 863,113 valid Version 3 signatures and 20,331,436 valid Version 4 signatures, which is 95.79%.

## 7.1 Version 3 Signature Packets

Figure 20 shows the creation of Version 3 signatures since 1994. Similar to the Version 3 public key

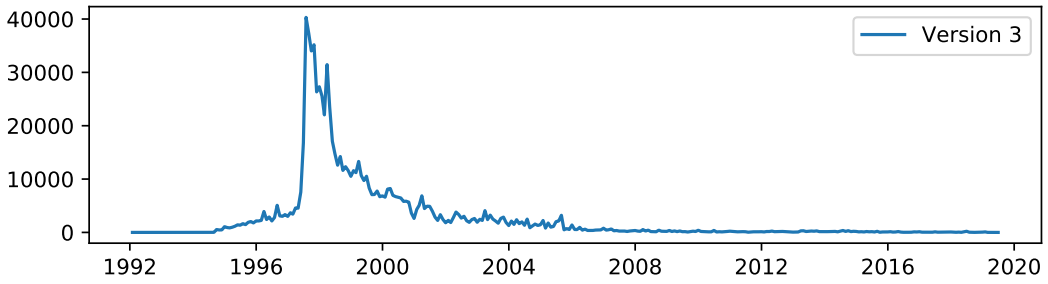



Figure 20: Creation of Version 3 signature packets

packets in Section 5.3, the first Version 3 signature packets were generated in September 1994. There was a peak around 1997, but then Version 3 signature packets were soon replaced by Version 4 packets. The Version 3 signature packets used RSA and DSA as public key algorithms, with DSA being in the majority (466,369 DSA signatures versus 396,715 RSA signatures - there are also 29 RSA Sign Only signatures). Figure 21 shows a comparison of Version 3 RSA versus DSA signatures. One can clearly see DSA keys starting to be used when RFC 2440 was standardized and made DSA keys the default. Nonetheless, RSA signatures were also very common with Version 3 signatures, probably because there

were already a lot of RSA keys from the years before the OpenPGP standard. Furthermore PGP 2.x, which was already widely distributed, used RSA as public key algorithm.

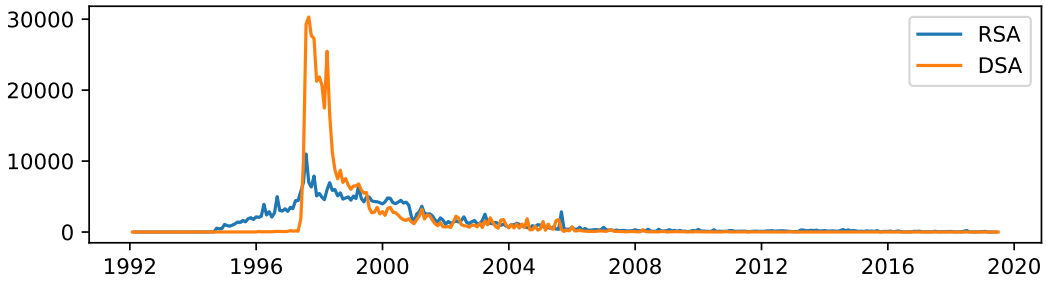



Figure 21: Creation of Version 3 signature packets RSA vs. DSA

As to hash algorithms, RFC 2440 lists MD5, SHA-1, RIPE-MD/160 and MD2 and specifies that implementations must implement SHA-1 and should implement MD5. RFC 4880 on the other hand deprecates MD5 and adds SHA256, SHA384, SHA512 and SHA224.

Figure 22 shows the distribution of the used hash algorithms in Version 3 packets. One can see a majority of SHA-1 versus MD5, which was the default hash algorithm used in PGP 2.6.x.

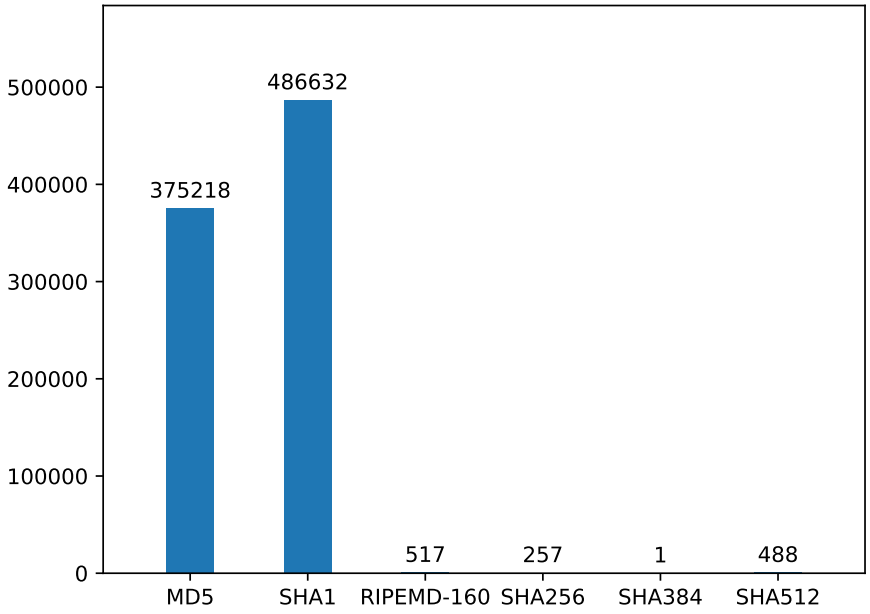



Figure 22: Distribution of hash algorithms in Version 3 packets

When looking at the use of hash algorithms over time, one can see a clear resemblance to the usage of public key algorithms in Version 3 signature packets. In fact, there are only 20.374 RSA signatures that use SHA1 as hash algorithm and no DSA signatures using MD5 - this should not be a surprise, because in the original Digital Signature Standard specification SHA-1 (back then simply called SHA) was the only hash algorithm listed. On the other hand, chapter 5.2.2 of RFC 2440, which describes the Version 3 signature packet format, states that ”[w]ith RSA signatures, the hash value is encoded as described in PKCS-1 section 10.1.2”. PKCS-1 described the RSA encryption standard and was published

as RFC 2313 in March 1998 [44]. It only defines MD2, MD4 or MD5 as *message digesting* (hashing) algorithms.

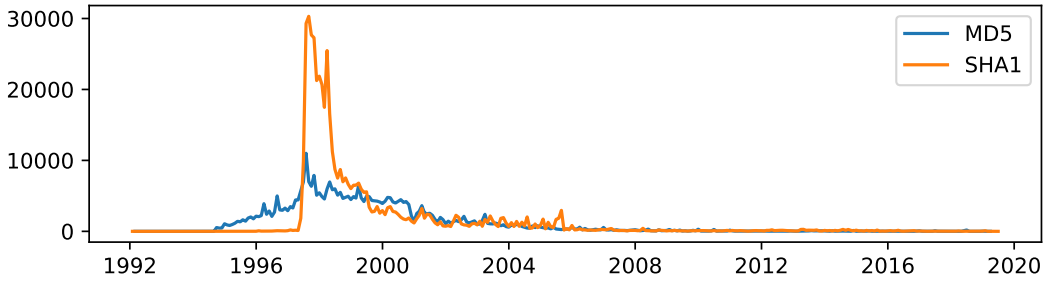



Figure 23: Evolution of hash algorithms in Version 3 packets

## 7.2 Version 4 Signature Packets

The Version 4 signature packet format was first defined in RFC 2440. Similar to the analysis of the Version 4 public-key packets, all signature packets with a creation timestamp before 1997 were removed. Figure 24 shows the evolution of Version 4 signature packets over the last 22 years. The plot exhibits the same spikes as Figure 7, namely the Evil32 research project in June 2014 and the spamming of the keyserver network through the keyserver-fs tool in July 2018. In this case the spikes are a lot bigger though, because with the Evil32 research, the generated public keys also imitated all the inter-key signatures of the strong set. In the case of the attack from July 2018, the amount of spam signatures to spam keys is even bigger, because they store information in the User ID packets and bind the User ID packets to the public key with a self signature. There are 113,366 spam keys with an average of 12.04 User ID and signature packets bound to them.

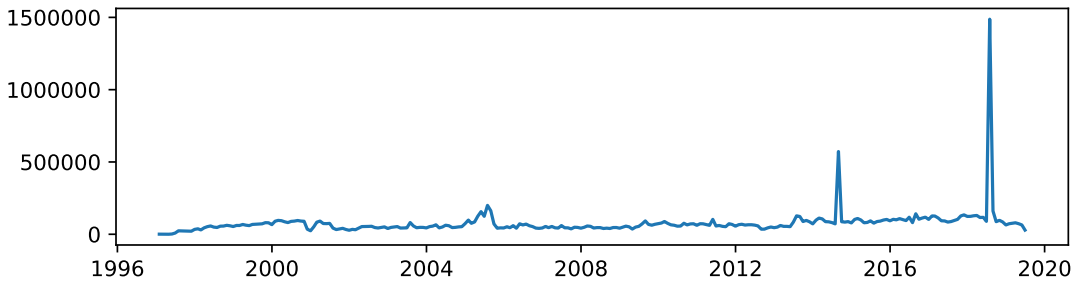



Figure 24: Evolution of Version 4 signature packets

To get a better understanding of how the Version 4 signing packets developed over time, the keys of the Evil32 research and the keys from the spam attack in 2018 were removed from the data set, which results in a more fine grained graph in Figure 25.

In addition to the spikes another similarity with the public key packets is the distribution of algorithms used, as Figure 26 shows. Most of the signature packets use RSA and DSA as algorithms, only 0.31% use ECDSA or EdDSA. Similar to the situation with public key packets, as shown in Figure 8, DSA signatures are used until around 2010 and then RSA signatures became more common.

Regarding the hash algorithms there is a different distribution than with Version 3 signature packets, as Figure 27 shows. MD5 is now nearly gone, SHA-1 is by far the most used algorithm and two of the

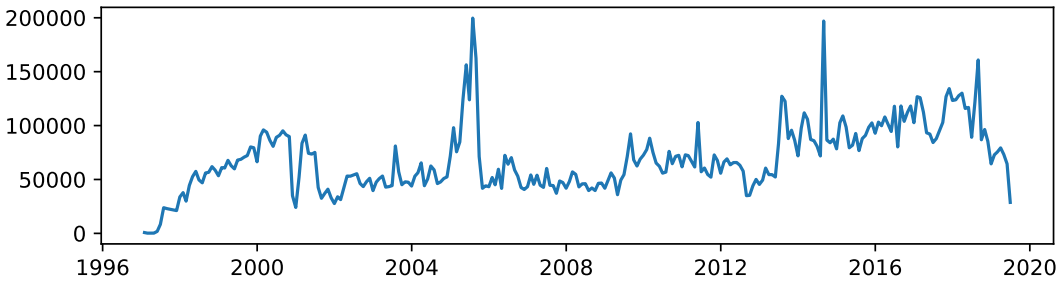



Figure 25: Evolution of Version 4 signature packets (cleaned)

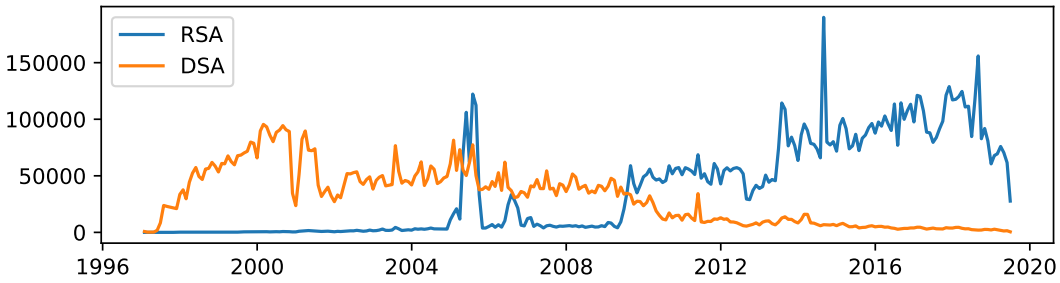



Figure 26: Evolution of Version 4 signature packet algorithm

newer SHA-2 hashing algorithms are also used by 27.8% of the signatures. The vast majority of SHA-1 signatures can be attributed to DSA being the most common signature algorithm, as Figure 26 shows. DSA was originally, as described above, only specified with SHA-1 as hash algorithm, only in 2009 with the release of a new Version of the Digital Signature Standard, FIPS 186-3 [45], NIST stopped to mention SHA-1 directly but started to refer to the FIPS PUB 180-3. FIPS PUB 180 defines the Secure Hash Standard (originally only SHA-1) and since Version 2 [46] (released in 2002) this standard also describes SHA-256, SHA-384 and SHA-512 and in FIPS PUB 180-3 SHA-224 was added [47].

As to RSA, PKCS-1 was updated in 1998 and now recommended, next to MD2 and MD5, also SHA-1 as hash algorithm [48]. Another update of that standard in 2003 added SHA-256, SHA-384 and SHA-512 [49]. GnuPG started default to SHA-1 in 2002 and switched to SHA-256 in 2009 [37]. Figure 28 shows a decrease of SHA-1 usage since around 2015 and an increase for both SHA-256 and SHA-512 at around the same time.

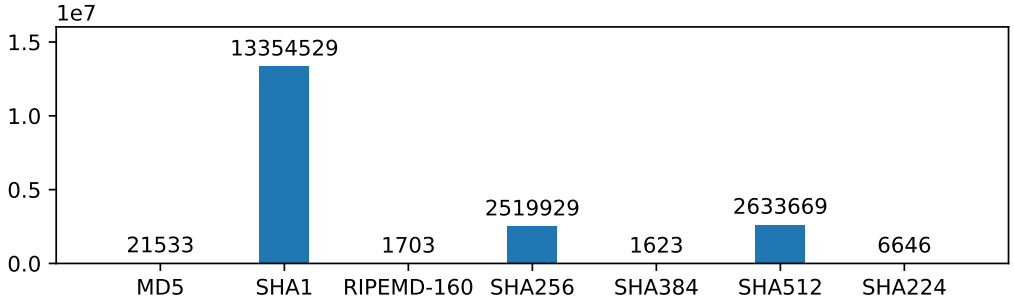



Figure 27: Distribution of Version 4 signature packet hash algorithms

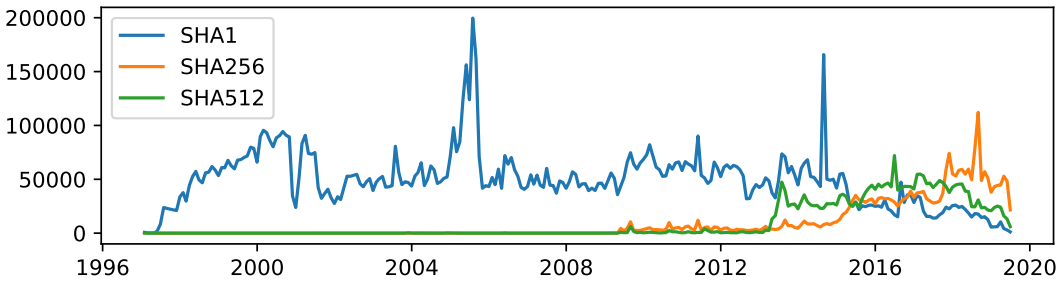



Figure 28: Evolution of Version 4 signature packet hash algorithms

### 7.3 Signature Subpackets

With Version 4 of signature packets, it became possible to add additional information to a signature using signature subpackets. RFC 2440 defines 21 signature subpacket types [5, ch. 5.2.3.1] and RFC 4880 added 3 types to that list [6, ch. 5.2.3.1]. Table 6 lists the subpacket types defined in RFC 4880.

| ID | Type | ID | Type |
|---|---|---|---|
| 2 | Signature Creation Time | 21 | Preferred Hash Algorithms |
| 3 | Signature Expiration Time | 22 | Preferred Compression Algorithms |
| 4 | Exportable Certification | 23 | Key Server Preferences |
| 5 | Trust Signature | 24 | Preferred Key Server |
| 6 | Regular Expression | 25 | Primary User ID |
| 7 | Revocable | 26 | Policy URI |
| 9 | Key Expiration Time | 27 | Key Flags |
| 10 | Placeholder for backward compatibility | 28 | Signer's User ID |
| 11 | Preferred Symmetric Algorithms | 29 | Reason for Revocation |
| 12 | Revocation Key | 30 | Features |
| 16 | Issuer | 31 | Signature Target |
| 20 | Notation Data | 32 | Embedded Signature |

Table 6: Signature subpacket types listed in RFC4880 [6, ch. 5.2.3.1]

One type of subpacket has to be part of each signature is the subpacket that stores the creation time of the signature (subpacket Type 2). Another signature subpacket that stores a time value is the Signature Expiration Time subpacket (subpacket Type 3), which holds the validity period of the signature in seconds after the signature creation time. 12.04% of the signatures have an expiration set, which is nearly double the percentage of keys with expiration dates. One reason for this discrepancy is, that there are/were some attempts to establish certification authorities in the OpenPGP ecosystem. The certifications of those CAs usually were only valid for some period. One of these certification authorities is the *PGP Global Directory Verification Key*. The PGP Global Directory [15] is a service that sends a verification email to any address of a given key and signs the key if the verification is successful. This also explains Table 7, which lists the five most used validity periods. The most used expiration time by far is 14 days, which is the expiration time of the signatures of the PGP Global Directory Verification Key. Another CA is the community driven certificate authority cacert.org. Cacert also publishes a "Certification Practice Statement" which defines that the expiration for signatures on certificates for assured members

[15] https://keyserver.pgp.com/vkd/GetWelcomeScreen.event

is 24 months [16].

| Validity (in days) | Count |
|---|---|
| 14 | 1969853 |
| 1461 | 71138 |
| 366 | 23352 |
| 7 | 13189 |
| 365 | 12308 |

Table 7: Most used validity of signature packets

Signature subpackets can also be used to publish preference lists of symmetric algorithms. In RFC 1991 the only symmetric algorithm defined was IDEA [3, ch. 6.4.1] and it was the only cipher used in PGP 2.0 until 2.6. Both RFC 2440 and RFC 4880 on the other hand state that implementations must implement Triple-DES. In addition to Triple DES RFC 2440 states that implementations should support IDEA (for backwards compatibility with PGP) and CAST5. RFC 4880 later replaced the should-requirement of IDEA with AES-128.

The signature subpacket containing symmetric algorithm preferences can be used by communication partners to find out which algorithms the software of the messages recipient supports besides the defaults. The data set contains 6,685,984 signature subpackets that define a list of preferred symmetric algorithms. Figure 29 shows the distribution of the most preferred symmetric algorithms. There is a clear preference for AES256 with 67.87%, with CAST5 second (27.01%) and AES128 third (4.0%).

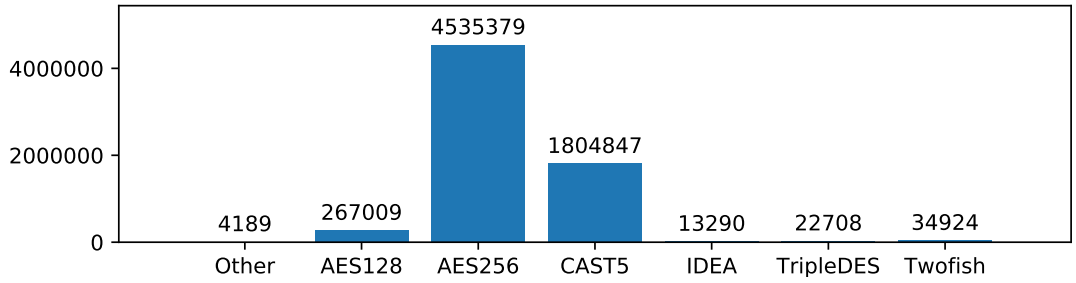



Figure 29: Distribution of most preferred symmetric algorithm

When NIST announced in 1997 that they would start a process of looking for a successor of DES, the Advanced Encryption Standard. This also had an effect on implementations of OpenPGP. Although it was not yet clear, which of the submitted algorithms wwas going to be chosen to be AES, RFC 2440 already reserved symmetric algorithm IDs for AES with 128, 192 and 256 bit [5, ch. 9.2]. Figure 30 shows the evolution of the preferred symmetric algorithms since 1998. CAST5 was mostly popular until 2004, but in the early 2000s, when the Rijndal algorithm was selected as the Advanced Encryption Standard, first AES128 and later AES256 gained popularity and AES256 is the most preferred symmetric algorithm ever since. GnuPG for example started to support AES in Version 1.0.4, which was released in October 2000.

Another preference that can be set using a signature subpacket is the preferred hash algorithm (Type 21). After removing the known fake keys and signatures with obviously fake timestamps, the data set contains 4,684,374 subpackets declaring a preferred hash algorithm. The distribution of the most

[16] http://www.cacert.org/policy/CertificationPracticeStatement.html

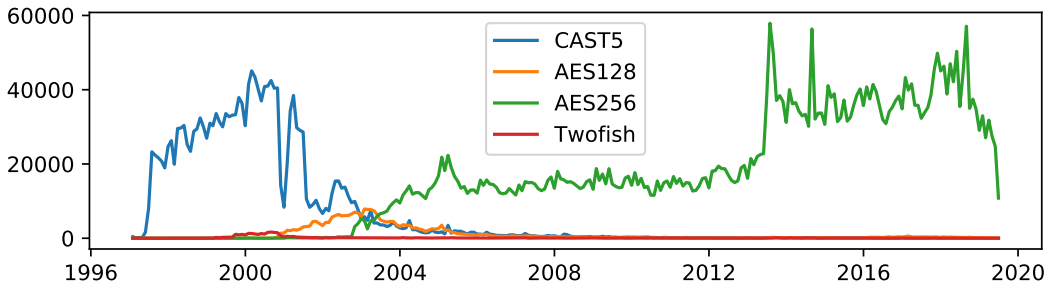



Figure 30: Evolution of most preferred symmetric algorithm

preferred hash algorithm, as listed in Figure 31, show a similar preference to SHA based algorithms as the distribution of signature algorithms in Figure 27.

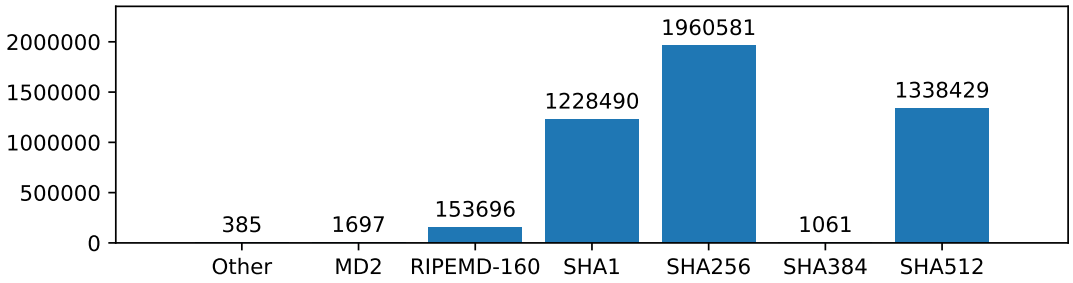



Figure 31: Distribution of most preferred hash algorithm

The evolution of the preferred hash algorithm over the last 20 years in Figure 32 shows RIPEMD-160 being on the top of the preference list in the early 2000s, but then being replaced by SHA-1. Around 2010, 3 years after the release of RFC 4880, first SHA256 and a few years later also SHA512 replaced SHA-1.

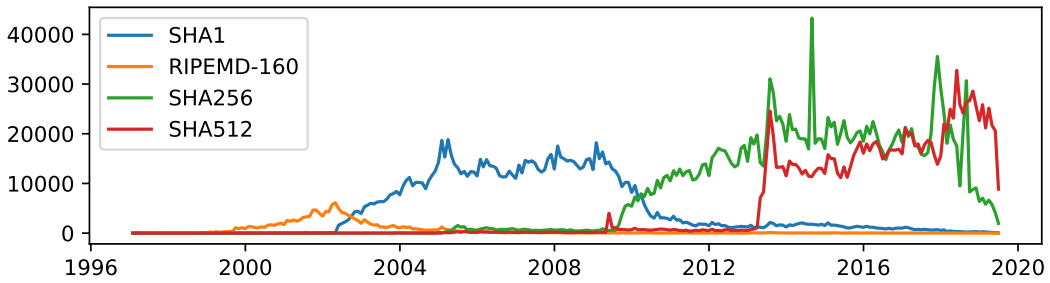



Figure 32: Evolution of most preferred hash algorithm

## 8 Future Work

As described in Section 2, the OpenPGP data format standardizes a lot of attributes of OpenPGP keys and OpenPGP signatures. This research looked into some of them, but there are still a lot of characteristics

that might shed light on the evolution of the use of OpenPGP keys. Just the signature subpackets provide a vast trove of information, for instance an interesting approach would be to look at subpacket Type 26, which allows to add a URL pointing to a policy statement or subpacket Type 24, which allows to define a preferred key server. A deeper analysis of the User ID packets could yield more useful findings for user interface research in security applications. Regarding the algorithms in use, one could take a specific look at RSA keys and their prime factors.

Another interesting approach would be to implement a solution to regularly produce analyses of OpenPGP keys, to be able to spot abnormalities and distinctive features early on. Similar to the Tor Projects "Tor Metrics" [17], which publishes metrics about users, servers, traffic and performance of the Tor network, or the OrNetRadar [18] which is used to monitor events related to Tor relays, one could create a continuing stream of data about OpenPGP keys.

Furthermore, regarding the complexity of the OpenPGP standard, it might be worthwhile to implement an abstraction layer. OpenPGP is designed to satisfy various use cases, from software signatures to encrypted backups. But the vast majority of users only use OpenPGP as a solution for email encryption. A simplified version of OpenPGP might make it easier to adapt the standard and respond to new found vulnerabilities or new regulations.

# 9 Conclusion

OpenPGP is a very complex packet format that has seen many changes. It is used for a diverse set of applications, from encrypting emails to singing software releases. This analysis shows, that regulatory developments have a major impact on cryptographic solutions, though looking at the development of OpenPGP packets over time, one can see that in some cases it takes a lot of time to get rid of *legacy* algorithms (i.e. the research shows the creation of ElGamal signatures until nearly ten years after a successful attack). Another interesting fact is the case of implementations creating keys with deprecated algorithm types. It is noteworthy that even if software libraries implement the correct algorithms, a public code sample using the library the wrong way can lead to software products creating OpenPGP packets with deprecated algorithm types. This topic has also been researched by Morteza Verdi et al in "An Empirical Study of C++ Vulnerabilities in Crowd-Sourced Code Examples" which states that "[t]he reuse of crowd-sourced code snippets can facilitate rapid prototyping. However, recent research shows that the shared code snippets may be of low quality and can even contain vulnerabilities" [50]. In the case of deprecated algorithm types this could be prevented by deprecation warnings in software libraries or if the libraries did not even implement the deprecated algorithm types. Another solution would be to be more strict in the OpenPGP standard and change it to prohibit implementations from generating packets with deprecated algorithm types. The research also shows, that through big data analysis of public keys information about software solutions can come to light. The case of the TellFinder keys demonstrates that from studying the public keys that have similar User ID packets, one can draw conclusions about companies or software. Looking at the User ID packets without email addresses and the distinctive format of the keys generated by the *monkeysphere* scripts shows that the content of the packets can reveal information about solutions used in tech infrastructure. Both the analysis of the public key packets and the analysis of the signature packets show that expiration dates on cryptographic primitives is not very common, in spite of IT security education, awareness trainings and literature on cryptography emphasizing the importance of a limited validity of encryption keys. A consequence of this might be to change user interfaces to automatically make keys only valid for a predefined number of years. Something else that could help would be to make it easier to extend the validity of keys. The

[17] https://metrics.torproject.org/

[18] https://nusenu.github.io/OrNetRadar/

study of the hash algorithms reveals that it took a long time to get rid of SHA-1 as hash algorithm, even though it was deprecated by NIST in 2011. Only in the last 5 years SHA-2 based algorithms became more common than SHA-1. A downside of the OpenPGP standard might be that its complexity makes updates to the standard take a long time. The current specification is still RFC4880, which lists SHA-1 as a MUST requirement for implementations, although there have already been multiple attacks on SHA-1, just recently with the first collision for full SHA-1 [51].

## Acknowledgments

This research was funded by the Josef Ressel Center for Blockchain Technologies & Security Management (BLOCKCHAINS). The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

## References

[1] P. Zimmermann, "Why i wrote pgp," 1991, https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html [Online; accessed on September 15, 2020].

[2] S. Garfinkel, *PGP: pretty good privacy*. O'Reilly Media, Inc., 1995, p. 85ff.

[3] P. Zimmermann, D. Atkins, and W. Stallings, "PGP Message Exchange Formats," IETF RFC 1991, August 1996, https://rfc-editor.org/rfc/rfc1991.txt.

[4] C. Breed, "Openpgp working group announcement," September 1997, https://mailarchive.ietf.org/arch/msg/openpgp/jW3kdoHwoOG3kdo9Nlt-VGaVzn4 [Online; accessed on September 15, 2020].

[5] H. Finney, R. L. Thayer, L. Donnerhacke, and J. Callas, "OpenPGP Message Format," IETF RFC 2440, November 1998, https://tools.ietf.org/html/rfc2440 [Online; accessed on September 15, 2020].

[6] H. Finney, L. Donnerhacke, J. Callas, R. L. Thayer, and D. Shaw, "OpenPGP Message Format," IETF RFC 4880, November 2007, https://tools.ietf.org/html/rfc4880 [Online; accessed on September 15, 2020].

[7] W. Koch, "A short history of the gnu privacy guard," December 2007, https://lists.gnupg.org/pipermail/gnupg-announce/2007q4/000268.html [Online; accessed on September 15, 2020].

[8] D. Shaw, "The Camellia Cipher in OpenPGP," IETF RFC 5581, June 2009, https://tools.ietf.org/html/rfc5581 [Online; accessed on September 15, 2020].

[9] A. Jivsov, "Elliptic Curve Cryptography (ECC) in OpenPGP," IETF RFC 6637, June 2012, https://tools.ietf.org/html/rfc6637 [Online; accessed on September 15, 2020].

[10] W. Koch, brian m. carlson, R. H. Tse, and D. Atkins, "OpenPGP Message Format," IETF Internet-Draft, September 2019, https://tools.ietf.org/html/draft-ietf-openpgp-rfc4880bis-08 [Online; accessed on September 15, 2020].

[11] H. Böck, "A look at the pgp ecosystem through the key server data." *IACR Cryptology ePrint Archive*, vol. 2015, p. 262, March 2015.

[12] S. Yamane, J. Wang, H. Suzuki, N. Segawa, and Y. Murayama, "Rethinking openpgp pki and openpgp public keyserver," *CoRR*, vol. cs.CY/0308015, August 2003.

[13] D. Shaw, "The OpenPGP HTTP Keyserver Protocol (HKP)," IETF Internet-Draft, March 2003, https://datatracker.ietf.org/doc/html/draft-shaw-openpgp-hkp-00 [Online; accessed on September 15, 2020].

[14] J. Clizbe, "Announcement of sks," November 2011, https://lists.gnu.org/archive/html/sks-devel/2011-11/msg00009.html [Online; accessed on September 15, 2020].

[15] M. Pini, "PEAKS: adding proactive security to OpenPGP keyservers," April 2018, https://www.politesi.polimi.it/bitstream/10589/140111/3/Tesi_MattiaPini.pdf [Online; accessed on September 15, 2020].

[16] D. K. Gillmor, "Openpgp certificate flooding," June 2019, https://dkg.fifthhorseman.net/blog/openpgp-certificate-flooding.html [Online; accessed on September 15, 2020].

[17] D. K. Gillmor, "Abuse-Resistant OpenPGP Keystores," IETF Internet-Draft, August 2019, https://datatracker.ietf.org/doc/html/draft-dkg-openpgp-abuse-resistant-keystore-04 [Online; accessed on September 15, 2020].

[18] A. Yakubov, W. Shbair, and R. State, "BlockPGP: A Blockchain-based Framework for PGP Key Servers," in *Proc. of the 2018 6th International Symposium on Computing and Networking Workshops (CANDARW'18), Takayama, Japan*. IEEE, November 2018, pp. 316–322.

[19] P. Wouters, "DNS-Based Authentication of Named Entities (DANE) Bindings for OpenPGP," IETF RFC 7929, August 2016, https://tools.ietf.org/html/rfc7929 [Online; accessed on September 15, 2020].

[20] W. Koch, "OpenPGP Web Key Directory," IETF Internet-Draft, November 2019, https://datatracker.ietf.org/doc/html/draft-koch-openpgp-webkey-service-08 [Online; accessed on September 15, 2020].

[21] N. McBurnett, "Pgp web of trust statistics," 1997, http://bcn.boulder.co.us/~neal/pgpstat/ [Online; accessed on September 15, 2020].

[22] S. Čapkun, L. Buttyán, and J.-P. Hubaux, "Small worlds in security systems: an analysis of the pgp certificate graph," in *Proc. of the 2002 workshop on New security paradigms (NSPW'02), Virginia beach, Virginia, USA*. ACM, September 2002, pp. 28–35.

[23] M. Boguñá, R. Pastor-Satorras, A. Díaz-Guilera, and A. Arenas, "Models of social networks based on social distance attachment," *Physical review E*, vol. 70, no. 5, pp. 056 122:1–8, 2004.

[24] J. Cederlöf, "Dissecting the leaf of trust," 2004, http://www.lysator.liu.se/~jc/wotsap/leafoftrust.html [Online; accessed on September 15, 2020].

[25] K. Fiskerstrand, "Openpgp key statistics," 2014, https://blog.sumptuouscapital.com/2014/01/openpgp-key-statistics/ [Online; accessed on September 15, 2020].

[26] A. Ulrich, R. Holz, P. Hauck, and G. Carle, "Investigating the openpgp web of trust," in *Proc. of the 16th European Symposium on Research in Computer Security (ESORICS'11), Leuven, Belgium*, ser. Lecture Notes in Computer Science, vol. 6879. Springer, 2011, pp. 489–507.

[27] A. Barenghi, A. Di Federico, G. Pelosi, and S. Sanfilippo, "Challenging the trustworthiness of pgp: Is the web-of-trust tear-proof?" in *Proc. of the 20th European Symposium on Research in Computer Security (ESORICS'15), Vienna, Austria*, ser. Lecture Notes in Computer Science, vol. 9326. Springer, September 2015, pp. 429–446.

[28] G. Wolf and V. G. Quiroga, "Progression and forecast of a curated web-of-trust: A study on the debian project's cryptographic keyring," in *Proc. of the 13th IFIP International Conference on Open Source Systems (OSS'2017), Buenos Aires, Argentina*. Springer, May 2017, pp. 117–127.

[29] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*. John Wiley & Sons, 1995.

[30] M. Johnson, "Ziffwire article on pgp 2.6 (fwd)," July 1994, http://mailing-list-archive.cryptoanarchy.wiki/archive/1994/07/d3a9ba3ba6890c50eb6b947372e80894d89aec70e3c371f1b40d29b687cc8266/ [Online; accessed on September 15, 2020].

[31] U. D. O. C. I. of Standards and Technology, "Federal information processing standards publication 186," May 1994, https://archive.org/details/federalinformati186nati [Online; accessed on September 15, 2020].

[32] G. Greenwald and E. MacAskill, "Nsa prism program taps in to user data of apple, google and others," June 2013, https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data [Online; accessed on September 15, 2020].

[33] G. Greenwald and E. Snowden, "Edward snowden: Nsa whistleblower answers reader questions," June 2013, https://www.theguardian.com/world/2013/jun/17/edward-snowden-nsa-files-whistleblower [Online; accessed on September 15, 2020.

[34] M. Lee, "Ed snowden taught me to smuggle secrets past incredible danger. now i teach you." 2014, https://theintercept.com/2014/10/28/smuggling-snowden-secrets/ [Online; accessed on September 15, 2020].

[35] R. Klafter and E. Swanson, "Evil 32: Check your gpg fingerprints," 2014, https://evil32.com [Online; accessed on September 15, 2020].

[36] D. Bleichenbacher, "Generating elgamal signatures withouth knowing the secret key," in *Proc. of the 1996 Internation Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'96), Saragossa, Spain*, ser. Lecture Notes in Computer Science, vol. 1070. Springer, May 1996, pp. 10–18.

[37] W. Koch, "Gnupg - release notes," July 2017, https://gnupg.org/download/release_notes.html [Online; ac-

cessed on September 15, 2020].

[38] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management, part 1: General (revised)," March 2007, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r2007.pdf [Online; accessed on September 15, 2020].

[39] S. Josefsson and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)," IETF RFC 8032, January 2017, https://rfc-editor.org/rfc/rfc8032.txt [Online; accessed on September 15, 2020].

[40] P. Brunschwig, "changed default key size for key generation to 4096 bits," April 2014, https://gitlab.com/enigmail/enigmail/commit/2a84f99395555169f275f6d3e31331932bc07de6 [Online; accessed on September 15, 2020].

[41] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management, part 1: General (revised)," https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r3.pdf [Online; accessed on September 15, 2020], July 2012.

[42] E. Hamilton, "Jpeg file interchange format," September 1992, https://www.w3.org/Graphics/JPEG/jfif3.pdf [Online; accessed on September 15, 2020].

[43] V. Breitmoser, "Linked Identities for OpenPGP," IETF Internet-Draft, April 2015, https://datatracker.ietf.org/doc/html/draft-vb-openpgp-linked-ids-01 [Online; accessed on September 15, 2020].

[44] B. Kaliski, "PKCS #1: RSA Encryption Version 1.5," IETF RFC 2313, March 1998, https://tools.ietf.org/html/rfc2313 [Online; accessed on September 15, 2020].

[45] U. D. O. C. I. of Standards and Technology, "Federal information processing standards publication 186-3," June 2009, https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf [Online; accessed on September 15, 2020].

[46] N. I. o. S. Information Technology Laboratory and Technology, "Federal information processing standards publication 180," August 2002, https://csrc.nist.gov/csrc/media/publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf [Online; accessed on September 15, 2020].

[47] N. I. o. S. Information Technology Laboratory and Technology, "Federal information processing standards publication 180," October 2008, https://csrc.nist.gov/csrc/media/publications/fips/180/3/archive/2008-10-31/documents/fips180-3_final.pdf [Online; accessed on September 15, 2020].

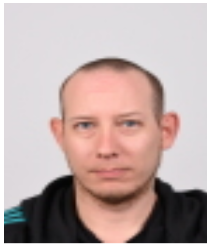
[48] B. Kaliski and J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2.0," iETF RFC 2437, October 1998, https://tools.ietf.org/html/rfc8032 [Online; accessed on September 15, 2020].

[49] J. Jonsson and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1," IETF RFC 3447, February 2003, https://tools.ietf.org/html/rfc3447 [Online; accessed on September 15, 2020].

[50] M. Verdi, A. Sami, J. Akhondali, F. Khomh, G. Uddin, and A. K. Motlagh, "An empirical study of c++ vulnerabilities in crowd-sourced code examples," *IEEE Transactions on Software Engineering*, vol. Early Access, September 2020.

[51] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full sha-1," in *Proc. of the 37th Annual International Cryptology Conference (CRYPTO'17), Santa Barbara, California, USA*, ser. Lecture Notes in Computer Science, vol. 10401. Springer, August 2017, pp. 570–596.

## Author Biography



**Birger Schacht** received his bachelor degree in IT Security at the St. Pölten University of Applied Sciences in Austria. His research interests focus on network security and data protection in professional environments.

**Peter Kieseberg** heads the "Josef Ressel Center for Blockchain Technologies & Security Management," as well as the "Institute of IT Security Research" at the St. Pölten University of Applied Sciences, Austria, and has worked for more than 10 years in the field of IT Security Research. He is co-organizer of the Cross Domain Conference for Machine Learning and Knowledge Extraction (CD-MAKE) and founder and chair of the International Workshop on Security of Mobile Applications (IWSMA), which is taking place for the ninth time in 2020. Peter's research interests mainly focus on issues surrounding the foundations of blockchains regarding non-cryptocurrency applications, as well as privacy and data protection in data driven environments. He is a senior member of the IEEE and chair of the Austrian IEEE SMC chapter, as well as member of the ACM.