A Probabilistic Trust Model for GnuPG

Jacek Jonczy, Markus Wüthrich, and Rolf Haenni
Institute of Computer Science and Applied Mathematics
University of Berne, Switzerland
jonczy@iam.unibe.ch

Abstract

Trust networks are possible solutions for the key authenticity problem in a decentralized public-key infrastructure. A particular trust model, the so-called Web of Trust, has been proposed for and is implemented in the popular e-mail encryption software PGP and its open source derivatives like GnuPG. In this paper, we investigate the drawbacks and weaknesses of the current PGP and GnuPG trust model, and we propose a new approach to handle trust and key validity in a more sophisticated way. A prototype of our solution has been implemented and tested with the current GnuPG release.[1]

1 Introduction

Due to the rapid emergence and constant evolution of various distributed systems and applications in large, inherently insecure networks, methods and techniques to establish information security play an increasingly crucial role. One of the most fundamental challenges is the problem of establishing a secured channel between two users of the network. For this, classical single-key cryptography requires the two users to previously exchange a common secret key over a secure channel, which is impracticable for large or even global networks.

With the advent of public-key cryptography, the keys to exchange are *public*, i.e. the channel used for the exchange is no longer required to be secure. At first sight, this seems to be an ideal solution for the key exchange problem, but an important subproblem remains, namely to ensure that a public key actually belongs to its supposed owner. We will refer to it as the *key validation problem*. Note that spoofing another's identity is easily possible in any of several ways, i.e. the key validation problem is anything but trivial, particularly when the two users involved have never met and know nothing about each other.

As a solution for the key validation problem, *public-key infrastructures* (PKI) have been proposed and implemented in many different ways. One type of PKI requires one or several central authorities responsible for issuing *digital certificates* for public keys. Such a certificate is an unforgeable warranty for the binding between the involved public key and its owner.[2] It is of crucial importance for the reliable operation of such a centralized PKI that the certificate authorities are fully *trustworthy*.

At the other end of the conceptual range are PKIs, which avoid central certificate authorities entirely. The most prominent example of such a decentralized PKI is a distributed trust model called *Web of Trust*. It is used in PGP, GnuPG, and other OpenPGP-compatible systems. The basic concept of this particular model goes back to Zimmermann's first PGP release in the early 90ies, and since then it has not changed much [13, 16]. In this paper, we will refer to it as the *PGP trust model*, as suggested in [1].

Distributed trust models allow any user in the network to issue certificates for any other user.[3] The issuers of such certificates are called *introducers*, who can make them publicly available, typically by uploading them to *key servers*, from which they are accessible to other users. Someone's personal collection of certificates is called *key ring*. In this way, responsibility for validating public keys is delegated to people you trust. In comparison with a centralized PKI, this scheme is much more flexible and leaves trust decisions in the hands of individual users. These trust decisions are finally decisive for a user to validate public keys (i.e. to accept them as authentic) on the basis of the given local key ring.

In this paper, we will first give a short overview of the PGP trust model. The main goal is to point out some of its inherent weaknesses and deficiencies. To overcome these difficulties, we will then propose a more flexible PGP trust model, in which we propose to see the key validation problem as a two-terminal network reliability problem in a corresponding stochastic graph [11]. This view is similar to the one proposed in [6], but it requires less theoretical background knowledge. In the last part of this paper, we describe the prototype implementation of this model in GnuPG.

2 The PGP Trust Model

The PGP trust model has some particular characteristics. First of all, (only) three levels of trust are supported: *com-*

[1]This research is supported by the Hasler Foundation, project no. 2042, and the Swiss National Science Foundation, project no. PP002–102652.

[2]Strictly speaking, a certificate is a warranty for the binding between the involved public key and a *description* of the owner [9]. Such a description can consist of a single attribute (name, first name, birth date, e-mail address, etc.) or a combination thereof. In the PGP context, this description is called *user ID* and typically consists of an e-mail address.

[3]In the PGP jargon, issuing a certificate is called *signing a key*, and certificates are therefore called *signed public keys* or simply *signatures*.

plete trust, *marginal trust*, and *no trust*.[4] The owner of the key ring, who needs to manually assign these trust values for all other users, automatically receives full trust.[5]

When a user places trust in an introducer, implicitly it means that the user possesses a certain amount of confidence in the introducer's capability to issue valid certificates, i.e. correct bindings between users and public keys. This is the general intuition, but the actual meaning of the three trust levels in PGP is not defined explicitly.

Key Validation in PGP. Based on such trust values, the PGP trust model suggests to accept a given public key in the key ring as *completely valid*, if either

(a) the public key belongs to the owner of the key ring,
(b) the key ring contains at least C certificates from completely trusted introducers with valid public keys,
(c) the key ring contains at least M certificates from marginally trusted introducers with valid public keys.

To compensate for the above-mentioned ambiguity of the trust levels, the PGP trust model allows the users to idividually adjust the two *skepticism parameters* C (also called COMPLETES_NEEDED) and M (also called MARGINALS_NEEDED). In general, higher numbers for these parameters imply that more people would be needed to conspire against you. The default values in PGP are $C = 1$ and $M = 2$, and $C = 1$ and $M = 3$ in GnuPG. If a given key is not completely valid according to the above rules, but if at least one certificate of a marginally or completely trusted introducer with a valid public key is present, then the key attains the status *marginally valid*. Otherwise, the key is considered to be *invalid*.[6] The distinction between *marginally valid* and *invalid* keys is often neglected, so will we in the sequel.

An example to illustrate the PGP trust model is the *certificate graph* shown in Fig. 1. An arrow from X to Y represents a certificate issued by X for Y. Gray circles indicate complete trust (A, E, G, J, L, N), gray semicircles indicate marginal trust (C, D, F, M), and white circles indicate no trust (B, H, I, K, O). The results of the key validation are shown for the $C = 1$ and $M = 2$. Completely valid public keys are represented by nested circles (A, B, C, D, E, H, I, J, N, O). Note that all public keys with a certificate issued by A, the owner of the key ring, are completely valid.

How Trustworthy is the PGP Trust Model? The PGP trust model has both advantages and drawbacks. An important advantage is the simplicity of the above-mentioned evaluation rules. This leads to a very efficient evaluation algorithm, which performs on a given key ring in time linear to its size. Another advantage is the above-mentioned adjustability of the skepticism parameters C and M, which allows the users to express their own policy regarding the threshold of his confidence in the PGP key validation mechanism.[7]

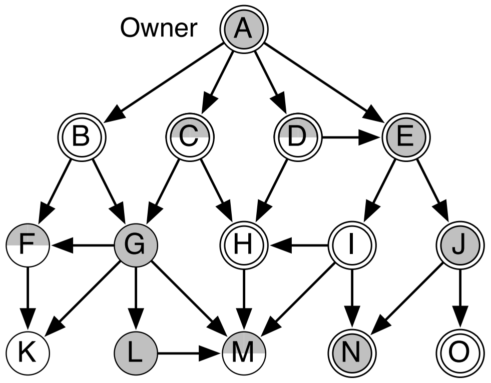

Figure 1. Example of a PGP key ring.

A major deficiency of the PGP trust model is that the limited levels of trust in PGP is clearly insufficient to reflect possible varying opinions about an introducer's trustworthiness.[8] In real life, it may well be that among two marginally trustworthy introducers one of them is twice more trustworthy than the other. Unfortunately, the PGP trust model does not support such distinctions.

A similar problem arises from the limited levels of validity, which does not always allow to properly separate quite different situations within a certificate graph. As an example, consider two different keys with an unequal number of certificates, all from marginally trusted introducers. If in both cases the number of certificates is beyond the threshold M, according to the third key validation rule, the keys will be rated equally as *completely valid*. This conclusion does not measure up to the fact that more positive evidence is available for the validity of the key with the greater number of certificates.

Another quite severe problem arises from the pragmatic nature of the third key validation rule. As demonstrated in [6, 10], this can lead to quite counter-intuitive conclusions. Consider the two key rings shown in Fig. 2. On the left hand side, there are two certificate chains of length 3 from A to B, each of them containing one completely trusted and one marginally trusted introducer. On the right hand side, the situation is very similar, except that there are more than two (possibly infinitely many) certificate chains of length 3 from A to B, in which the order of the two introducers is reversed. In such a situation, one would clearly expect B to have a higher degree of validity in the second case, but the PGP trust model tells us to accept B in the first and reject B in the second case (for the default values $C = 1$ and $M = 2$).

A similar type of problem arises from the fact that the PGP trust model does not take into account the possibility of people controlling multiple public keys. As a consequence,

[4] It is common to separate *unknown trust* from *no trust*, but this has no significance for the PGP key validation algorithm.

[5] This particular type of full trust is also called *implicit* or *ultimate* trust.

[6] A more general way of defining the validity of public keys is by means of the so-called *key legitimacy* $L = c/C+m/M$, where c and m denote the number of certificates from completely resp. marginally trusted introducers with valid keys [14]. Then a key is completely valid for $L \geq 1$, marginally valid for $0 < L < 1$, and invalid for $L = 0$.

[7] Another adjustable PGP parameter is CERT_DEPTH, which defines the maximum certification chain length. As in the example of the previous subsection, this parameter is often ignored.

[8] It is interesting to know that the OpenPGP message format specification reserves an entire octet to store trust values: "*The trust amount is in a range from 0–255, interpreted such that values less than 120 indicate partial trust and values of 120 or greater indicate complete trust. Implementations should emit values of 60 for partial trust and 120 for complete trust*" [3].

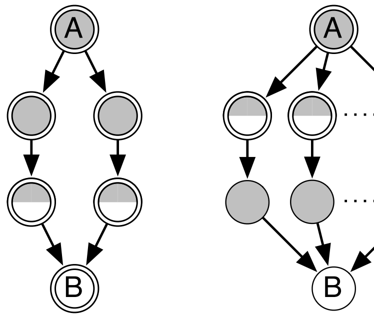

Figure 2. Counter-intuitive PGP key validation.

it could well happen that a key with certificates from two marginally trusted and apparently different users is considered to be valid (for $M = 2$), but in reality they are issued by the one and the same person. This is a problem of invisible dependencies, which could easily be exploited by malicious users to make people accept non-existing bindings between users and keys [10].

To conclude this subsection, let us mention two features of more sophisticated trust models, which are not available in the PGP trust model. The first one is the ability to issue *recommendations* (i.e. certificates relative to somebody's trustworthiness as a reliable introducer) or other higher level statements in the sense of Maurer's multi-level trust model [12].[9] The second missing feature is the support of negative, mixed, or weighted statements as proposed in [7].

3 Probabilistic Key Validation

The overcome some of the above-mentioned deficiencies of the PGP trust model, we propose to translate the key validation problem into an appropriate *network reliability problem* [4]. Network reliability problems are well-studied in *reliability theory*, and they have many applications in network design and other areas.

In a general setting, the starting point is a network represented as a directed stochastic graph, where vertices are subject to independent failures with given probabilities and arcs (directed edges) are perfectly reliable.[10] The problem then is to compute the probabilities that the network provides an operating connection between two, some, or all vertices. Only the directed *two-terminal problem*, which is also called *source-to-terminal* or *s,t-connectedness problem*, is relevant for the particular application of this paper. This is the problem of computing the probability of establishing at least one operating network path from a vertex s (the source) to another vertex t (the terminal).

Formulating Trust-Based Key Validation as a Network Reliability Problem. If we intuitively map the s,t-connectedness problem to the trust-based key validation context, we get the following setting: the graph represents the key ring, vertices are introducers (resp. their public keys), arcs are certificates, and failure probabilities are (negated) trust values assigned to introducers. In other words, a trust value is now understood as somebody's *probability* of being a reliable introducer.[11] This allows us to specify trust values on an infinitely fine scale between 0 (no trust) and 1 (complete trust).

The example in Fig. 3 depicts the subgraph obtained from the key ring of Fig. 1, if one is concerned with the validity of K's public key only. The PGP trust values are replaced by respective probabilities: 0.9 for complete trust, 0.5 and 0.6 for marginal trust, and 0.1 for no trust. Note that the trust value assigned to K has no impact on its own key validation.

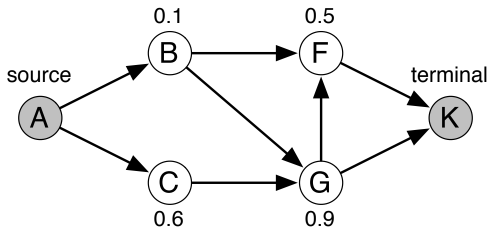

Figure 3. A PGP key ring as reliability network.

The translation into a network reliability problem offers us now a broad range of computational techniques to solve the key validation problem. We will first use the above example to illustrate a simple but not very efficient method, and then give a short overview of more advanced techniques.

Computing Network Reliabilities. Consider again the network of Fig. 3, in which the key validation problem consists in A's attempt to validate K's public key. For this, there must be at least one complete certificate path from A to K, and if no such path exists, the validity of K's public key cannot be established from A's point of view. In the example of Fig. 3, there are five certificate paths from A to K, namely $\{A, B, F, K\}$, $\{A, B, G, K\}$, $\{A, C, G, K\}$, $\{A, C, G, F, K\}$, and $\{A, B, G, F, K\}$. Note that the last two paths are not minimal and therefore irrelevant for the overall network reliability. From the remaining three paths, we can also omit the source A (who's trust value is 1 by default) and the terminal K (who's trust value has no impact).[12] Finally, we obtain the following set of minimal paths:

$$minpaths_K = \{\{B, F\}, \{B, G\}, \{C, G\}\}.$$

To calculate the network reliability for a connection from A to K, which will be our measure for the validity of K's

[9]The OpenPGP message format specification foresees the possible inclusion of higher level certificates such as recommendations: "*Signer asserts that the key is not only valid, but also trustworthy, at the specified level. Level 0 has the same meaning as an ordinary validity signature. Level 1 means that the signed key is asserted to be a valid trusted introducer,* [...]. *Level 2 means that the signed key is asserted to be trusted to issue level 1 trust signatures, i.e. that it is a "meta introducer". Generally, a level n trust signature asserts that a key is trusted to issue level n–1 trust signatures*" [3].

[10]In a directed stochastic graph, it is always possible to replace vertex failures by corresponding arc failures, and vice versa [4]. Furthermore, every undirected network can easily be transformed into a directed one.

[11]We do not specify whether these probabilities are interpreted as frequencies or as subjective degrees of belief.

[12]This is a slight deviation from the classical s,t-connectedness problem, where every vertex in the path is relevant, including the source and the terminal.

public key, we have to compute the probability of the set $minpaths_K$. The probability of a single path is simply the product of its (stochastically independent) trust values, and for the overall probability of the set $minpaths_K$, we can apply the so-called *inclusion-exclusion* formula:

$$\begin{aligned}P(minpaths_K) &= P(\{B,F\}) + P(\{B,G\}) + P(\{C,G\}) \\ &- P(\{B,F,G\}) - P(\{B,F,C,G\}) - P(\{B,C,G\}) \\ &+ P(\{B,F,C,G\}) = 0.581.\end{aligned}$$

This result is the probabilistic measure we propose for the validity of K's public key. Depending on A's own validation policy, e.g. by specifying a validity threshold $\lambda \in [0,1]$, the key may be accepted as valid or not. This result is very different from the PGP scenario in Fig. 1, where K's public key is considered invalid (except for $C = 1$ and $M = 1$).

Our proposal for a probabilistic evaluation of trust networks solves some of the deficiencies of the PGP trust model mentioned in Section 2. First of all, it eliminates the limited levels of trust and validity, which leads to an increased overall flexibility. At the same time, it solves the problem of counter-intuitive conclusions in situations like the one shown in Fig. 2. This improves both the robustness and the coherence of the results.

Other Network Reliability Methods. Applying the inclusion-exclusion formula to a set of minimal paths is an exact, but not a very efficient solution for the s,t-connectedness or other network reliability problems. Reliability theory provides a variety of other techniques, with a general distinction between *exact* and *approximate* methods.

Most exact methods start by either enumerating *complete states*, *pathsets*, or *cutsets*. These enumeration methods are often combined with *reduction techniques* (e.g. series and parallel reductions), *decomposition techniques* (e.g. Shannon's decomposition, BDDs, d-DNNFs, cd-PDAGs), or so-called *sum-of-disjoint-products* algorithms. Note that all exact methods inherently suffer from an exponential worst-case complexity.

Approximate methods are either lower and upper bound estimations or sampling algorithms (e.g. Monte-Carlo, importance sampling). The method implemented in GnuPG is a Monte-Carlo sampling algorithm. For a complete and detailed discussion of network reliability techniques, we refer to the literature [2, 4].

4 Implementation

This section is devoted to the implementation of the probabilistic trust model in GnuPG.[13] We will give an overview of the most important changes as well as a brief description of the implemented algorithms. Furthermore, we will show by an example how to use GnuPG with its new functionality.

Modification of the GnuPG Source. For the implementation of the new trust model, we have used GnuPG v.1.4.5.[14] All affected source files can be found within the `g10` folder of the GPG source archive. The following list shows all modified or extendend files with a brief description of their functionalities. For more details, we refer to [15].

`gpg.c` : main function and parsing of arguments passed on the command line.

`options.h` : data structures to store the options passed on the command line.

`tdbio.c`, **`tdbio.h`** : definition of a trust database file structure as well as all input/output related functions.

`trustdb.c`, **`trustdb.h`** : algorithms for public key validation.

`pkclist.c` : functions related to primary keys.

`tdbdump.c` : implementation of commands for the import and export of owner trust values.

`keyedit.c` : implementation of the **`--edit-key`** command.

The most important changes affect the `trustdb.c` source file. It contains now the algorithms of our probabilistic key validation method and is the main entry point for updating the trust database. There are two major extensions: (1) an algorithm for the computation of minimal certificate paths, and (2) an algorithm for the computation of the key validity based on these paths.

The first algorithm is implemented as a breath-first search (BFS) in the certificate graph. After the entire graph has been searched, all minimal certificate paths are stored for each key. The choice of a BFS algorithm has two reasons. First, the key validation according to the original PGP trust model is also implemented as a BFS, which allows to reuse some code for the new algorithm. Second, by performing a BFS (instead of a depth-first search), we avoid the production of many non-minimal paths. For further information about this procedure, we refer to [15].

To overcome the problem of the exponential worst-case complexity of exact methods, we implemented the second algorithm as a Monte-Carlo sampling process, from which we obtain approximate solutions. The main advantage of this method is that it performs well and is easy to implement, see [15] for details. Another possible implementation is described in [8].

Example Usage. Consider again the example in Fig. 3. A's key ring contains public keys of B, C, F, G, and K. Our goal now is to validate these keys using the extended GnuPG implementation. We will illustrate the necessary steps in form of respective GnuPG command line instructions.

Let us first have a look at the generic command for executing the GnuPG program. For general information about the possible parameters **`options`**, **`command`**, and **`args`**, we refer to the GNU Privacy Handbook [5] or to the gpg man page.[15]

```
gpg [options] command [args]
```

In order to switch to the probabilistic key validation model, the option **`--trust-model`** must be set at the beginning. Note that this option is only necessary the first time **`gpg`** is

[13]All project related information, including download and installation instructions can be found under *http://leeloo.unibe.ch/∼mwuethrich/bachelor.*

[14]This is the most recent stable version on September 15th, 2006.

[15]http://www.gnupg.org/(en)/documentation/manpage.en.html

called within a session.

```
gpg --trust-model probabilistic [options] command [args]
```

The command `--edit-key` allows us to edit key information. For example, by supplying a user's email address as identifier, we can specify trust levels:

```
gpg --edit-key K@foo.bar trust
```

As a result of this command, the following dialog to enter a trust value between 0 and 1 shows up. A precision up to six decimal places is taken into account, i.e. more exact values are rounded.

```
[...]
Please decide how far you trust this user to correctly
verify other user's keys (by looking at passports,
checking fingerprints from different sources, etc.)

Enter a number between 0.0 and 1.0
  0.0 means you don't trust this person at all
  1.0 means you fully trust this person
  m = back to the main menu

Your decision?  0.75
```

After this, the key ring is reevaluated and the corresponding key validities in the trust database are updated. With the following command, the trust database can also be updated manually:

```
gpg --update-trustdb
```

For our sample key ring, the output looks as follows:

```
gpg:  sampling trials set to 2000, validity threshold
set to 0.6, probabilistic trust model

E53E6AC8:A <A@foo.bar> trust:1.000 valid:1.000 ok
7866C82B:C <C@foo.bar> trust:0.600 valid:1.000 ok
85ADC1BB:B <B@foo.bar> trust:0.100 valid:1.000 ok
8694CBC0:F <F@foo.bar> trust:0.500 valid:0.586 not ok
D6BAAD0F:G <G@foo.bar> trust:0.900 valid:0.640 ok
36A66930:K <K@foo.bar> trust:0.000 valid:0.581 not ok

gpg:  next trustdb check due at 2006-09-15
```

Two supplementary options are available for the new trust model, namely:

`--sampling-trials` This option specifies the number of trials within the sampling algorithm used for computing the key validity. The default value is set to 1000. A higher value means more accurate results, but also increased efficiency.

`--validity-threshold` This option replaces the original skepticism parameters used in GnuPG. Any key whose validity is equal or higher than the specified threshold will be accepted as valid (marked with `ok`). Note that a key with a value below the threshold may still be authentic, but the given key ring does allow us to prove it. The default value of the threshold is 0.5.

For the above trust database output, we used 2000 sampling trials and a validity threshold of 0.6.

5 Conclusion

The main contribution of this paper is the proposal for a probabilistic trust model for GnuPG. The key validation problem has been transformed into a directed two-terminal network reliability problem. As a result, several weaknesses of PGP's trust model are eliminated. The most important improvement comes from the gradual trust values, which then result in gradual levels of validity. Our new model also avoids counter-intuitive scenarios like the one shown in Fig. 2. To conclude, we think that the proposed key validation method is a reasonable, flexible, and useful enhancement of the existing GnuPG functionality. At the moment, it is not officially included in the GnuPG software, but we hope it will at some future time.

References

[1] A. Abdul-Rahman. The PGP trust model. *EDI-Forum: the Journal of Electronic Commerce*, 10(3):27–31, 1997.

[2] M. O. Ball, C. J. Colbourn, and J. S. Provan. Network reliability. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 673–762. Elsevier, 1995.

[3] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *RFC 2440: OpenPGP Message Format*. IETF Network Working Group, 1998.

[4] C. J. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, New York, USA, 1987.

[5] M. Copeland, J. Grahn, and D. A. Wheeler. *The GNU Privacy Handbook*. The Free Software Foundation, 1999.

[6] R. Haenni. Using probabilistic argumentation for key validation in public-key cryptography. *International Journal of Approximate Reasoning*, 38(3):355–376, 2005.

[7] J. Jonczy and R. Haenni. Credential networks: a general model for distributed trust and authenticity management. In A. Ghorbani and S. Marsh, editors, *PST'05: 3rd Annual Conference on Privacy, Security and Trust*, pages 101–112, St. Andrews, Canada, 2005.

[8] J. Jonczy and R. Haenni. Implementing credential networks. In K. Stølen, W. H. Winsborough, F. Martinelli, and F. Massacci, editors, *iTrust'06, 4rd International Conference on Trust Management*, LNCS 3986, pages 164–178, Pisa, Italy, 2006.

[9] R. Kohlas, J. Jonczy, and R. Haenni. Towards precise semantics for authenticity and trust. In *PST'06, 4th Annual Conference on Privacy, Security and Trust*, Toronto, Canada, 2006.

[10] R. Kohlas and U. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In H. Imai and Y. Zheng, editors, *PKC'2000, Third International Workshop on Practice and Theory in Public Key Cryptography*, LNCS 1751, pages 93–112, Melbourne, Australia, 2000. Springer.

[11] G. Mahoney, W. Myrvold, and G. C. Shoja. Generic reliability trust model. In A. Ghorbani and S. Marsh, editors, *PST'05: 3rd Annual Conference on Privacy, Security and Trust*, pages 113–120, St. Andrews, Canada, 2005.

[12] U. Maurer. Modelling a public-key infrastructure. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *ESORICS, European Symposium on Research in Computer Security*, LNCS 1146, pages 324–350. Springer, 1996.

[13] W. Stallings. *Protect Your Privacy, a Guide for PGP Users*. Prentice Hall, 1995.

[14] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 3rd edition, 2003.

[15] M. Wüthrich. GnuPG and probabilistic key validation. Bachelor thesis, IAM, University of Berne, Switzerland, 2006.

[16] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1994.