11-22-2013

Rule-based conditional trust with OpenPGP.

Andrew Jackson

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Department of Computing Security

Master of Science in
Computer Security and Information
Assurance

Rule-based conditional trust with OpenPGP.

By

Andrew Jackson

November 22, 2013

R·I·T

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Department of Computing Security

Master of Science in
Computer Security and Information Assurance

Thesis Approval Form

Student Name: Andrew Jackson
Thesis Title: Rule-based conditional trust with OpenPGP.

Thesis Committee

	Name	Signature	Date
Chair	Bill Stackpole	_____	_____
Member	Bo Yuan	_____	_____
Member	Pete Lutz	_____	_____

Contents

List of Figures

1 Abstract:

This thesis describes a new trust model for OpenPGP encryption. This trust model uses conditional rule-based trust to establish key validity and trust. This thesis describes "Trust Rules" that may be used to sort and categorize keys automatically without user interaction. "Trust Rules" are also capable of integrating key revocation status into its calculations so it too is automated. This thesis presents that conditional trust established through "Trust Rules" can enforce stricter security while reducing the burden of use and automating the process of key validity, trust, and revocation.

2 Introduction:

This thesis wishes to develop a new trust model for OpenPGP encryption. The current 4880 RFC [4] that describes OpenPGP allows for two types of trust models: a three-tiered hierarchy certificate authority, PKI type of structure, and a "Web of Trust" [48]. Each of these models either depends on a group to establish trust or a trusted introducer, but neither establish conditional rules for trust establishment. Such conditional rules would allow key validity to be based on particular user-established conditions being satisfied. This thesis develops a system that uses user-developed rules to conditionally assign trust to OpenPGP keys. These "Trust Rules" allow users to develop stricter validation rules based specifically on their own criteria, thereby increasing their security and confidence in the use of OpenPGP. For

example, a person could establish that a key is only valid if it is signed by two or more keys from a group of 5 keys, which are explicitly chosen by the end user. Many different trust models could be developed if a rule-based conditional trust model were established. This model would allow each group or individual to create their own "Trust Rules" to establish trust and key validity.

Additionally, with recent high profile compromises of certification authorities [40], such as DigiNotar, certification authorities issuing certificates for the purpose of man-in-the-middle attacks [40], and the code signing of the Stuxnet virus [7] [32] we can see how any trusted certification authority or their associated registration authority can be a single point of failure in the traditional PKI infrastructure. The use of certification authorities or any single entity to establish key validity or trust would appear to be very problematic potentially leaning towards outright distrust. Add to that recent news reports of National Security Agency (NSA) data collection [18] [19] [17] [43] and one can quickly see the potential need for encryption and potentially even a distrust towards established cloud services, email provides, and social networks.

3 History & Terminology:

In this next section we will outline the history associated with our thesis topic and some terminology will be explained. Before we proceed we need to

establish a base line of terminology. We will try to use already established definitions and terms when and where possible.

3.1 Public Key Infrastructure:

The modern Public Key Infrastructure (PKI) that we know now started out in 1988 with the first edition of standards being published. These were published by the International Organization for Standardization and International Electrotechnical Commission (ISO/IEC) as ISO/IEC 9594:1990 and also by the International Telegraph and Telephone Consultative Committee (CCITT), now known as the International Telecommunication Union (ITU), published as ITU-T X.500 (1988) Series of Recommendations. These standards set forth an electronic directory service to be used as a general-purpose directory store of information. Employee information such as phone numbers, office numbers, email, certificates, and other information could be stored within the directory. There have been 5 editions of the standards with the 5th being published as ISO/IEC 9594:2005 and ITU-T X.500 (2005)[27].

Within the standard are the documents ISO/IEC 9594-8 and ITU-T X.509 [29] which respectively outline the use of "Public-Key and Attribute Certificate Frameworks" (called "Authentication Framework" before 4th edition). These documents along with The Internet Engineering Task Force (IETF) Request for Comment (RFC) RFCs 3280 [23], 4325 [41], 4630 [24], are what define the format and structure of the Public Key Infrastructure. The IETF goal is to "...develop a profile to facilitate the use of X.509 cer-

tificates within Internet applications for those communities wishing to make use of X.509 technology. Such applications may include WWW, electronic mail, user authentication, and IPsec" [23].

Many people use the Public Key Infrastructure everyday without ever giving it any thought. This lack of thought was a goal of the IETF when they described the respective RFCs. The procedures, protocols, and policies are designed so that the user has minimal interaction with PKI. The general public's exposure to most PKI is through the utilization of Secure Socket Layer (SSL) communication to buy products online through retail websites or check email. A lesser few use PKI to authenticate and login to VPNs, workstations, and other enterprise systems. These SSL connections and PKI authentications are secured with X.509 Certificates, signed by digital signatures and issued by Certificate Authorities (CAs) such as Verisign or other trusted CAs.

This thesis will examine the following X.509 aspects: X.509 Certificates, Certification Authority, and Certificate Revocation Lists.

3.2 X.509 Certificates:

A X.509 Certificate, as defined in RFC 3280 [23], has the following fields: signatureAlgorithm, signatureValue, version, serialNumber, signature, issuer, validity, subject, subjectPublicKeyInfo, uniqueIdentifier, extensions. Each of these values may be a sequence of values which describes attributes associated with that field.

- **signatureAlgorithm** specifies the algorithm to use in calculating the signature.
- **signatureValue** is the actual value of the certificate signature.
- **version** specifies what version number of X.509 this certificate belongs.
- **serialNumber** is the unique number assigned to the certificate by the certificate authority (CA).
- **signature** specifies the algorithm to use in calculating the signature. MUST be the same as signatureAlorithm.
- **issuer** is who issued the certificate as a X.501 Name Type.
- **validity** is the date in which the certificate is valid.
- **subject** identifies to whom the certificate has been issued.
- **subjectPublicKeyInfo** holds the public key of the subject.
- **uniqueIdentifier** is a unique number used to provide uniqueness if a subjects name is reused.
- **extensions** provide a mechanism to associate additional attributes with certificates.

The X.509 Certificate binds the public key of a user or system to the identifying information within the certificate by being digitally signed by a Certification Authority. This digital signature ensures that no tampering of the

certificate is possible, because only the issuing CA will have the corresponding private key that was used to sign the X.509 Certificate.

3.3 Certification Authority:

A Certification Authority (CA), also referred to as a Certificate Authority, is a trusted source that issues public-key certificates. It does this by digitally signing public keys and relevant identifying information, such as the subject, subjectPublicKeyInfo, and other X.509 Certificate fields. The information held within the certificate is thereby bound to the respective private key of the subject. The digital signature reliably prevents tampering of the public-key certificate because only the CA may alter the data as it alone has the corresponding private key. Once public-key certificates are issued they may be publicly distributed freely or published in an electronic directory service ie: X.500.

There are two important criteria for Certification Authorities to evaluate before issuing a public-key certificate. These have been defined in the X509 Recommendations [29] as follows:

- A certification authority shall be satisfied of the identity of a user before creating a certificate for it.
- A certification authority shall not issue certificates for two users with the same name.

To facility these requirements the CA must take precautions to prevent falsifications of identity, public key information, and resolve duplicate names.

The first requirement can easily be satisfied given enough information can be confirmed by the certification authority. Common forms of identity such as a government issued photo identification card and/or Passport may be used to confirm ones identity. For web based entities, such as a Web Server, ownership of the web servers domain name may be established before issuing a certificate. For both end user and web server certificates the public key information needs to be securely transmitted to the certificate authority. If this is not done the certification authority may end up signing the wrong public key. If this occurs the web server or user might not be able to read any secure communications and it could allow an attacker to read all of the communications that were thought to be secure.

The second requirement ensures that no user can pose as another user by duplicating the name. The name or subject field of a public-key certificate (X.509) holds a distinguished name that identifies the entity. A distinguished name is defined in X.501 [28] as "... that name which consists of the sequence of the relative distinguished name of the entry which represents the object and those of all of its superior entries (in descending order)."

As an example, the subject (Distinguished name) of the certificate issued to ipay.rit.edu is: "C = US, ST = New York, L = Rochester, O = Rochester Institute of Technology, OU = Information Technology Services, CN = ipay.rit.edu" This can easily make sense when it is known that C=

Country, ST= State, L= Location, O= Organization, OU= Organizational Unit, CN= Common Name. The distinguished name (DN) tells us the location and entity that requested the certificate. The final Common Name must match the domain name or (URl) of the web server that is being visited, in this case ipay.rit.edu.

3.4 Certificate Revocation List:

Certificate Revocation Lists (CRL) provide a way for a Certification Authority to revoke a public-key certificates. These CRLs are established as endpoints via http, ldap, or other network communication specifications to update end users of the status of a public-key certificate. Certificates normally have an expiration date after which they are no longer valid, but if a certificate needs to be revoked for another reason the Certification Authority may use the CRL to do so.

There are a few reasons why a Certification Authority may need to revoke a certificate. A Certification Authority may revoke for the following reasons: The user's private key may be compromised, the user is no longer certified by the authority, or the certification authority's certificate has been compromised. Each of the following cases would lead to a Certificate Revocation List being updated with the relevant public-key certificate identified as revoked or invalid.

3.5 PGP:

"Pretty Good Privacy" (PGP) was written by Philip Zimmermann in 1991 [47] [48]. The history of PGP is well known and can be explored more thoroughly by the reader in Michael W. Lucas's book "PGP & GPG: Email for the Practical Paranoid" [33] or from Phil Zimmermann's direct accounts in "Why I wrote PGP" [47]. The short version of the history is that Mr. Zimmermann wanted to allow individuals to send communications and exchanges securely via encryption without the possibility of government mandated back doors or key escrows. PGP allows any person the ability to send email or other communications knowing that the only person able to read the communique would be the addressed recipient.

For our purposes it is noteworthy to know the differences between PGP, GPG, and OpenPGP. PGP or "Pretty Good Privacy" is the original specification as developed by Phil Zimmermann, which later established the PGP Corporation. Referring to PGP should be taken as talking about PGP the corporation, however many use it interchangeably with the RFC or OpenPGP. GPG or "Gnu Privacy Guard" is a software package that reimplements the PGP standard and later, when developed, the OpenPGP standard. Finally, OpenPGP is the RFC standard that GPG and PGP software implement enabling them to be OpenPGP compliant.

The OpenPGP standard is defined in RFC 4880 [4] and previously in RFC 2440 [5] & RFC 1991 [2]. This standard defines how all OpenPGP clients should format, process and communicate OpenPGP messages.

The OpenPGP specification differs from the X509 specification in many ways. The key specification, trust model, allowed algorithms all have different options and specifications. For this thesis the major differences should be noted through the use of different trust models and key verification of trust. OpenPGP uses a "Web of Trust" versus X.509's hierarchical tree model. OpenPGP more recently has extended its specification to include a hierarchical trust model, but the "Web of Trust" is probably the most used trust model with OpenPGP. The "Web of Trust" consists of every individual who uses the OpenPGP standard regardless of software package. These users make up the web and form connections or edges of the web by digitally signing other users OpenPGP keys. Each user then may elect to trust some other user's digital signature for verification of the public-key certificate or key of a third user. This OpenPGP trust takes place instead of implementing X.509 Certification Authorities. Each user may act as their own Certification Authority. A user's verification and certification of other's keys may be extended to others that trust them. With all users certifying OpenPGP keys it creates a graph or web of trust connections between all users.

3.6 OpenPGP Format:

An OpenPGP message or certificate is constructed from a number of records or entries that are called packets. These packets should not be confused with network packets or other types of packets. These OpenPGP packets are chunks of data that are tagged with specific meaning. The following are the

types of OpenPGP packets according to RFC 4880 [4]:

- Public-Key Encrypted Session Key Packets
- Signature Packet
- Symmetric-Key Encrypted Session Key Packets
- One-Pass Signature Packets
- Key Material Packet
- Compressed Data Packet
- Symmetrically Encrypted Data Packet
- Marker Packet
- Literal Data Packet
- Trust Packet
- User ID Packet
- User Attribute Packet
- Symmetric Encrypted Integrity Protected Data Packet
- Modification Detection Code Packet

An OpenPGP message, keyring, certificate, or other types of OpenPGP objects are made up of these packets. Packets may hold other OpenPGP

packets, also known as subpackets, within. Some packets are never to be exchanged with other users, but are only for internal software data stores e.g. Trust Packets. Each packet is made up of a packet header followed by the packet body. Not all combinations of OpenPGP packets form valid objects. Some packets may be included multiple times where as others can only be included once and must be the last packet in the series, e.g. the Modification Detection packet. Finally, some packets may be digitally signed by the private key of the user to bind the packet data held within said packet to the users key, there by establishing a trusted connection to the signed packet.

Next, this thesis will describe a selection of OpenPGP packets that are of particular interest in regards to this thesis and will explore their workings and format.

3.6.1 Key Material Packet:

The Key Material Packet contains the actual key material, private or a public key, depending on the type of Key Material Packet. There are four variants of Key Material Packets which are:

Public-Key Packet: Format Described below.

Public-Subkey Packet: Public-Subkey Packet has the exact same format as a Public-Key packet, but describes a subkey.

Secret-Key Packet: A Secret-Key packet contains all the information as

described in a Public-Key packet, including the public-key material, but also includes the secret-key material appended at the end of all the public-key fields.

Secret-Subkey Packet: Secret-Subkey Packet has the exact same format as a Secret-Key packet, but describes a subkey.

The preceding Key Material Packets depend on the structure of the Public-Key Packet so, this thesis will explore its format and use. The format of a version 4 Public-Key OpenPGP packet [4], contains :

- A one-octet version number (4).
- A four-octet number denoting the time that the key was created.
- A one-octet number denoting the public-key algorithm of this key.
- A series of multiprecision integers comprising the key material.

By utilizing the four types of packets previously mentioned OpenPGP creates keys and subkeys. These subkeys are signed by the primary key or primary user identity to establish a trust relationship between them. A user of OpenPGP can utilize subkeys to increase their security. By using subkeys it allows a person to keep their primary key in a secure location and use subkeys in insecure locations. These subkeys may be revoked, if compromised, by the primary key. It is then possible to have multiple subkeys in multiple locations all used independently of each other, but tied together

in common by the primary key. The user is guaranteed that no adversary can sign keys or establish new subkeys as the primary key would be required. It should be noted that the use of more than one encryption subkey would be problematic as the sender of an encrypted communique would have to pick the correct subkey needed for encryption. For this reason it is advised to only use one encryption subkey at a time, but you may produce new encryption keys at any interval that you wish. If a subkey is ever compromised the user can revoke the subkey and still retain any collected digital signatures on the primary key. Again, despite using different keys the recipient of a message can establish a trust path to the subkey by verifying that the digital signature on the subkey matches with the primary key.

3.6.2 Signature Packet:

A Signature packet associates a public key with some data via a digital signature thereby binding the data to the public key. The most common signature type is a signature on a file, a block of text, or a signature that is used to certify a User ID. The signatures on User IDs are what make up the "Web of Trust" within OpenPGP. Signatures on text are used to verify the sending origin of a message and provide modification protection. The body of a version 4 Signature packet contains:

- One-octet version number (4).
- One-octet signature type.

- One-octet public-key algorithm.
- One-octet hash algorithm.
- Two-octet scalar octet count for following hashed subpacket data.
- Hashed subpacket data set (zero or more subpackets).
- Two-octet scalar octet count for the following unhashed subpacket data.
- Unhashed subpacket data set (zero or more subpackets).
- Two-octet field holding the left 16 bits of the signed hash value.
- One or more multiprecision integers comprising the signature.

There are 26 Signature Packet subpackets that assert different information about an OpenPGP key attribute or the signature packet itself. Subpackets are made up of objects like Signature Creation Time, Key Expiration Time, Trust Signature, Signer's User ID, Key Server Preferences, Preferred Key Server, and Reason for Revocation just to name a few. This thesis will next examine a handful of signature subpackets.

3.6.3 Trust Signature:

Just as in a more traditional PKI structure OpenPGP allows for a Certification Authority type structure through the use of "Trusted Introducers." These trusted entities operate in much the same manner as a regular Certification Authority by verifying the identity of the person or entity. Unlike

X.509 Certification Authorities, however, the trusted introducer does not create an X509 Certificate for the entity, but produces an OpenPGP Signature packet with a Trust Signature subpacket. When others encounter an OpenPGP key signed by the trusted entity they assume the OpenPGP key is valid even if they have not verified this to be true themselves.

A Trust Signature packet is very simple, it has one octet which describes the "level" (depth) and one octet which describes the amount of trust. The signer uses the Trust Signature to assert the trustworthiness of another at a specified level. Level 0 has the same meaning as an ordinary validity signature. Level 1 means that the signed key is certified to be a trusted introducer, with the 2nd octet specifying the degree of trust. Level 2 means that the signed key can be trusted to issue level 1 trust signatures. This continues on as higher levels are introduced, with a level *N* trust signature asserting that a key is trusted to issue level *N-1* trust signatures. The trust amount is an integer from 0-255 and is interpreted so that values less than 120 indicate partial trust and values greater than 120 indicate complete trust. In actuality, when implementing the OpenPGP specification most applications produce values of 60 for partial trust and 120 for complete trust.

So, with the introduction of Trust Signatures it is possible to create a PKI structure by first creating a OpenPGP key (A) that is a "Trusted Introducer" of level 2 signatures and completely trusted. Then, use key (A) to create 3 new OpenPGP keys (B, C, D) with Trust Signatures of level 1 and completely trusted. Now, B,C, and D may sign end users keys and if a user trusts the

original OpenPGP key A, they will also accept signatures from B, C, and D as valid and thereby trust any key that B, C, and D have certified.

3.6.4 Preferred Key Server:

Like X.509 Certificates OpenPGP keys may be stored in online directory stores or Key Server as they are called with OpenPGP. There are many key servers throughout the world, so if a user has a particular favorite or preference in which key server they would like to use they may signify this by using a Preferred Key Server sub packet signature. Like other sub packets the Preferred Key Server packet attaches an attribute to the OpenPGP key. The Preferred Key Server contains only one field which is a string in the form of a Uniform Resource Identifier (URI). This URI denotes the key server that the user prefers be used by others for updates. Keys with multiple User IDs may have a preferred key server for each User ID. Other users update a key by introducing Trust Signatures or other types of signature packets upon a key. These new signatures need to be distributed to others via the key server. Without an update mechanism the new signatures could not be distributed and used to recalculate the "Web of Trust" by adding new signatures or edges.

3.6.5 Key revocation:

As mentioned previously OpenPGP keys may be revoked by their owners. This revocation is established by a revocation signature on the key being

revoked. This revocation can happen for a number of reasons and is denoted by the "Reason for Revocation" signature subkey. The Reason for Revocation subkey begins with one octet containing the reason for revocation followed by a string specifying the reason. The first field has the following format.

- 0 - No reason specified
- 1 - Key is superseded
- 2 - Key material has been compromised
- 3 - Key is retired and no longer used
- 32 - User ID information is no longer valid
- 100-110 - Private Use

It should also be noted that signatures as well as keys may be revoked. Signatures made on another person's key may be revoked by the signatory at anytime. Like key revocation revoked signatures carry a reason for revocation as well.

4 Literature Review:

In this section this thesis will look at previous research on the topic of encryption, OpenPGP, User Interface design in encryption software, Trust model, and new encryption protocols. In evaluating OpenPGP and other encryption schemes there seems to be two major approaches.

1. Improving End-user software ease of use.

2. Improving protocol mechanisms and architecture.

With our approach this thesis presents a mix of both ease of use and mechanisms. This thesis has defined a new trust model which would need application component development and integration. And at the same time this thesis has carefully tried to keep from defining an entire new protocol and have worked within the OpenPGP specifications [4]. Our trust model has incorporated current research and ideas on establishing trust. As well as how trust is involved with software trust calculations, which this thesis will examine in more detail. As stated previously this thesis does not wish to change the OpenPGP message format [4], but only change how the trust is calculated internally in software packages.

4.1 Ease of Use:

OpenPGP is an open standard, as documented in RFC [4], however there are many implementations of that standard in software. PGP software is the oldest of such software. PGP's software shortcomings have been well established in the review of PGP 5.0 titled "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0" by Whitten and Tygar [45]. The researchers start with a premise that 90% of all security failures are based on configuration errors. Also, that software security user interface standards must be held to a higher standard than traditional non-security related software. This higher

standard should be used to prevent end users from entering security configuration errors. They suggest that security user interface requirements are such that there is a need to develop domain specific standards. The usability standards for normal software are not sufficient for security products. They defined usable security software as usable "...if the people who are expected to use it:

1. are reliably made aware of the security tasks they need to perform;
2. are able to figure out how to successfully perform those tasks;
3. don't make dangerous errors;
4. are sufficiently comfortable with the interface to continue using it."

To evaluate their hypothesis they used two separate evaluation methods. The first was a direct analysis method called cognitive walkthrough, and the second a laboratory user test.

In a cognitive walkthrough the researchers try to evaluate the software from the standpoint of a "novice" user. With this in mind they evaluate the learnability of the software. Through a cognitive walkthrough Whitten and Tygar identified numerous design issues and improvements for PGP 5.0. The design issues are identified as Visual Metaphors, Key Servers, Key Management Policy, Irreversible Actions, Consistency, and Too much Information. Visual Metaphors has to do with the use of images or icons to denote end user operations. In using Visual Metaphors a software developer should be careful

to avoid introducing confusion by not fully realizing how an image or icon may be perceived by the end user. Care should be taken to adjust images to accurately depict the operations. When using Key Servers they felt the user needed to be more informed about how the operation was taking place. The lack of a Visual Metaphor for this operation left users unaware of operations taking place on remote servers versus their local computers. Accordingly they felt the user interface of PGP 5.0 did not make this clear. Further, the Key Management Policy was not identified within the user interface, but only was available through the users manual. They note that "validity" and "trust" are shown to the user, but their meanings are not made obvious to the user. Next, Irreversible Actions need to be properly identified to the user. Irreversible actions are those that if taken by a user will not be able to reconstructed after their affects have taken place. These actions include things such as: Deleting the private key, accidentally publicizing a key, and accidentally revoking a key. Consistency within the software should be kept to create a clear mental picture. The use of "encryption" and "encoding" should not be interchanged, as PGP 5.0 does. The substitution of such words may confuse the end user as to what really is happening. Finally, according to Whitten and Tygar, PGP 5.0 displays too much information. They would reduce the amount of information that is displayed by default to the user. They suggest hiding fields such as the Key Length and Key Creation Time from users, but display this information via a properties page. Their reasoning is that more advance users will seek out this information if they need it,

but it confuses novice users.

For the laboratory user test Whitten and Tygar setup a scenarios in which the test subject is a member of a political campaign. For reasons of security, communications are secured with PGP encryption and digitally signed. Test subjects were then given a task of emailing a proposed itinerary for their candidate to the campaign manager and fourth other team members. This task would require the test subject to perform the following operations: Generate a Key Pair, acquire the team member's public keys, publish their public key, sign the itinerary, encrypt the itinerary to the recipients of the email, and finally send the email. During this test limited interaction was allowed between the test subject and the monitor, however the monitor did provide informative replies via email posing as the campaign manager and other political volunteers. There were widespread problems with the test subjects as they tried to complete their given task. Many could not create keys, send, or verify encrypted email within a 90 minute deadline. Three participants actually emailed the secret itinerary in plaintext without encryption. Many participants in the research were confused and unable to send, sign, and/or encrypted email to the research conductors. In fact one person was so confused as the nature of OpenPGP that they create public/private keys for each individual they wished to communication with. Based on the user test, the researchers felt that it supports their theory that the user interface design for PGP 5.0 is not usable for people who do not already have knowledge of PGP.

Whitten and Tygar make the following suggestion for an improvement over current security user interface design. Establish a clear and accurate conceptual model of the security to the user as quickly as possible. They express the need for the model to be as small as possible to allow for quick and accurate integration by the user while maintaining security. They suggest that through the use of appropriate visual metaphors, warning messages, wizards and other interactive tools that one would be able to increase user understanding and use of security software. The exact use and design of such an interface has been left open for future research.

The short comings of encryption software continued to be evaluated by Garfinkel and Miller, but this time with newer software and protocols, such as Outlook and S/MIME [15]. In "Johnny 2: A User Test of Key Continuity Management with S/MIME and Outlook Express" they present the user testing of Key Continuity Management (KCM). KCM ignores the standard X.509 certificate chain, instead it uses only the public key to identify email from each sender. The use of only a public key with no certificate chain is similar to the operation of SSH. The server information is cached on the first connection and is then verified on each subsequent connection. KCM follows this paradigm by caching S/MIME certificates and associating the originating email with that certificate when first received and then uses it on each subsequent email sent and received there after. The authors note that this system is not as secure as having a third party certificate authority, nor does it establish a responsible party for policy violations. Given these

short comings however, they express hope in KCM's increased likelihood of adoption and its ability to scale to multiple organizations.

Garfinkel & Miller's study is a continuation on the work done by Whitten and Tygar [45] with what they call a "radical reinterpretation" of the results. Garfinkel & Miller put forth that the key certification model used by PGP was the underlying problem in Whitten and Tygar's research, not PGP 5.0 or user training. To remedy the problem Garfinkel and Miller used KCM and tried to mimic Whitten and Tygar's research method for their laboratory user tests. In Garfinkel & Miller's user tests they introduce attacks on KCM into the scenario to test how the users respond.

With the introduction of KCM the authors also described a new user interface that would allow key continuity to be displayed to the user via different background colors. They described a system that would display a green border around messages that have a good signature from a person known to use signatures. The border would be displayed as gray if an email is received from a person that normally signs their emails. A yellow border is shown when a new signature and a new email are recognized. Lastly, a red border is shown when an email is received from a known sender with a signature that does not match the one recorded.

The subjects of the user test were divided into three categories: No KCM, Color, and Color+Briefing. The No KCM group had the KCM system disabled and a gray border around all messages. The Color group had the KCM enabled as normal. And the Color+Briefing group had the KCM enabled as

normal and received an additional briefing. Using the No KCM as a baseline the researchers then show the improvement of KCM against three different kinds of attacks. They were called the "New Key Attack", the "New Identity Attack" and the "Unsigned Message Attack". The "New Key Attack" tried to introduce a new key for an already established email identity (Certificate and Email pair). The "New Identity Attack" tried to establish a new identity for a known person using a new email address and subsequently a new certificate. The "Unsigned Message Attack" attempted to entice the user to respond to an unsigned email for a known identity.

The results of the tests showed that KCM users were much better at identifying the "New Key Attack" versus the No KCM group with a dramatic improvement shown in the Color+Briefing group. In the "New Key Attack" the researchers sent an attack email for a known good email, but with a new key. This message was then colored red by the KCM. For the "New Identity Attack" KCM did not show significant improvement over the No KCM test group. For the "New Identity Attack" the researchers used an attacker email that was established via Hotmail and was misspelled. KCM identified this message with a yellow border. Despite these clues to possible attack many test subjects rationalized that the key and the email address were different because a home computer was being used. Only two subjects were identified as noticing the misspelling of the name, and only one of them used this information to confirm their decision to not send secret information to the attacker. Finally, the "Unsigned Message Attack" KCM was a "success" for

both the Color group and the Color+Briefing group. However, in follow-up interviews it was found that most test subjects did not use KCM as much to identify the possible attack, as they did the requested recipient's email address. The email address for the attack was a Hotmail address. Many of the test subjects did not trust the security of Hotmail, thus the NO KCM group was not as susceptible to this attack as the researchers initially thought.

Garfinkel & Miller conclude that KCM could improve security but is not the silver bullet for the email security problem. There remains the open question regarding establishment of identity trust relationships, to which KCM does not provide a solution. However, the authors feel that KCM adds to the development of technologies that could be implemented to secure email and provide for more widespread adoption of S/MIME.

The use of visual cues in security software has continued to be researched by Kapadia [31]. Kapadia identified how the use of visual cues to denote security can be beneficial or a threat to security. In the research on these visual cues it was outlined how an attacker might be able to exploit the relationship or lack of with regards to key trust. Kapadia outlined how an attacker could get a trusted third party to verify their key, then proceed to communicate with the victim, the victim may assume that the communication is secured via strongly trusted certification authorities, but when in fact they have third party CAs. Since, clients do not display the certification authority by default the victim has no way of knowing that the attacker is not using a strongly trusted authority. Kapadia, in this case was arguing for more verification

of third party CAs as well as pointing out a possible security flaw. In our system the Tagged keys are separated from the Trusted keys, and Tagged Keys show the Trust Rules that triggered them. These rules should allow our system to not succumb to the attack depicted by Kapadia. The end user should notice that the Trust Rule did not trigger the same rule as the rest of the group/community if even at all. The hope would be that an end user would notice the different tags that were assigned and then inquire as to why key X does not trigger the group's previously establish Trust Rules.

In another paper by Garfinkel [14] visual references were noted as helping end users and a comparison of different email clients and their interfaces was conducted. Garfinkel showed how users of AOL email can visually identify email sent within the network versus email that originated outside of the AOL network. This difference provides the user with clues that the origin of the email has been authenticated. Garfinkel also identified Outlook's small certificate and Apple's Mail Security header as reasons that allow easier use of S/MIME or X.509 Certificates. This thesis will explore further research into the use of Trust Rules in an email client in Section 7 on Future Research.

4.2 Trust Models:

In developing our Trust Model and our associated Trust Rules research on Trust Models was evaluated. The research on Trust Models mainly described new mechanism or protocols to exchange and establish trust.

Our primary Trust Rule described establishes that an individual trusts

at least K authentication servers or Certificates Authorities will be truthful out of N servers was described by Chen [6]. In the paper Chen described a situation where you may only minimally trust a group of authentication servers, but you trust that at least N servers will be truthful. Chen then established a protocol to exchange this trust between users and servers. Instead of establishing new protocols this thesis would use PGP signatures as an exchange mechanism.

Many trust models focus on distributed protocols in establishing their community. These distributed models rely on the end points to establish trust without the need for a central server. This kind of distributed trust was also described by Adbul-Rahman [1], but in developing their trust mechanism they used a recommendation protocol. Abdul-Rahman suggested a new recommendation protocol separate from trust calculations. This thesis does not wish to develop a new protocol to establish trust, as this thesis feels signatures are sufficient to denote a recommendation of trust and/or validity.

In considering distributed trust mechanisms some have looked to the communities rather than the end users to establish this trust. This segmentation of peers into communities was described by Ravichandran [38]. Ravichandran specifically studied peer-to-peer interactions, reputation systems, delegation, and proposed a new "Eigen Group Trust" model as an improvement over Eigen Trust. In Eigen trust [30], a peer would calculate the global trust value by obtaining reputation values from all peers in the network, but Ravichandran segments the network into groups before calculating trust. By using

"Eigen Group Trust" it reduces the calculations and removes the requirement that the user have a global view of the network before calculating trust relationships. Our trust model can introduce similar ideas by exchanging Trust Rules between communities and groups. While one can not export the trust or signatures that Trust Rules have established there is nothing that prevents one from distributing the Trust Rules themselves. The Trust Rules can be formed and discussed among a community as to what the best or most secure rule set should be. Once in agreement the Trust Rules could be distributed to the members. These Trust Rules while developed by the group community still rely on the end user to implement. One could conceive of rules being established between different groups and communities that would allow for interchange of keys and trust.

While many of the former research has looked into trust and its distribution, distrust is as equally important. The distribution of trust and distrust was studied in Guha's paper on "Propagation of Trust and Distrust" [20]. The paper's goal "is to propose and analyze algorithms for implementing" a web of trust at the micro (single website) or macro level (entire internet). Guha puts forth that their work is the first to incorporate distrust into computational trust propagations. Firstly, they formalize the mathematical algorithms for trust propagation, such as direct propagation, cocitation, transpose trust, and trust coupling. Secondly, then turn to distrust propagation methodologies, identifying; Trust only, One-step distrust, and propagated distrust. Thirdly, they look at two different iterative propaga-

tion algorithms; Eigen Trust [30] and Weighted linear combinations (WLC). Fourthly, they turn to different types of rounding, since we will establish a boolean outcome from trust graph established utilizing real number value pairs for trust/distrust. From the trust algorithms, the iteration method, distrust propagation, and rounding cases (three each) there results in $3^4 = 81$ experimental schemes to evaluate. Finally, utilizing the Epinion's website dataset Guha sets about to analyze each of the experimental schemes to see which one best predicts the trust or distrust between nodes in the epionion's dataset. The best performing results were seen from the one step distrust with Eigen Trust [30] iterations.

In establishing trust we need to ensure that we have identified the correct individual or entity. Cheng's research [8] is focused on the sybil attack. The Sybil attack was formerly known as "pseudospoofing" is the process of creating multiple identities or pseudonyms by on entity. The term "sybil" was introduced by John Douceur [12] and based on the book [42] after the same name which is about the treatment of Sybil Dorsett who suffered from multiple personality disorder, now known as dissociative identity disorder. Cheng presents the notion of what constitutes sybilproofness and considers how to prevent falsely raising ones reputation utilizing fake links between sybils. Cheng does not address "badmouthing" or the case in which sybils are used to degrade the reputation of other non-sybil nodes, however they suggest that there is no reputation function which can guard against all badmouthing strategies.

4.3 Reputation Systems:

The distribution of trust and delegation of duties can allow for credibility and reputation in a small group to then be extended further as the interconnects to other communities are established. Group dynamics and reputation in an online community have been studied by Resnick [39] in evaluating eBay's reputation system. Resnick identified three criteria for reputation systems:

1. Provide information that allows participants to distinguish between trustworthy and non-trustworthy agents.

2. Encourage participants to be trustworthy.

3. Discourage participation from those who are not trustworthy.

The first criteria is satisfied with OpenPGP participants recognizing trustworthy agents through the signatures on the agents key in question. A non trustworthy individual should not be able to obtain key signatures from other people or trusted third parties. The second criteria is enforced by each end user refusing to communicate with a key of unknown trustworthiness or validity. Without manual verification or other signatures on said key the user has no way of knowing the key holders identity is truthful and correct. The third criteria is satisfied when a user revokes their signature from a users key. The possibility of key signature revocation should persuade individuals to continue to act in a trustworthy manner.

Group dynamics and reputation have also been studied by Huynh [25] in which "Certified Reputation" was described as a new form of third-party

certification of reputation. This thesis present the use signatures by group members as a form of reputation or certification, since key validity could have been already established through the use of Trust Rules. If a member of the group has turned out to become untrustworthy members could revoke their signatures, and then if a user re-enters the group members may re-sign that members key.

It would also be permissible to establish a separate revocation mechanism as defined by the group into the Trust Rule. This trust revocation could be by the initial certifying parties, the key holder, or another third-party that is strictly for key revocations. When developing the Trust Rule for the community it can be decided that any user that is found untrustworthy will have their key signed by the communities revocation keys. To prevent undue revocation by the group it is possible to establish a group of keys that must be a part of the signature before complete revocation takes place. The use of K of N total signatures for revocation can be used just the same as when the certifying Trust Rule was developed and may in fact be incorporated into the Trust Rule itself by placing a NOT around the revocation signatures. The negation will allow the Trust Rule to succeed only if the revocation keys are not present. This is the opposite of normal trust evaluations, but can be accomplished using the NOT operator. This type of revocation status being incorporated into the trust metric was outlined by Bicakci [3].

4.4 OpenPGP Web of Trust Analysis:

The original analysis of the Web of Trust was done in 1996 by Neal McBurnett [35] shortly after the creation of OpenPGP. He subsequently updated his findings in 1997 as the Web of Trust grew [34]. In 1996 he found that there were 2047 keys in the 'Strongly Connected' set of keys or strong set. That is to say, keys which have a path connecting them to every other key in the set. These paths are established by key signatures and do not take into account any type of trust modeling. In 1997 the strong set included 3100 keys. After identifying the largest strong set of OpenPGP keys he analyzed the distance or number of hops which separate each key. This would be the degree in network graph theory terminology. The mean shortest distance (MSD) was calculated to be 5.98188 in 1996 and 6.061 in 1997. The maximum shortest path in 1996 and 1997 was 21. McBurnett also created a listing of keys within the strong set sorted by MSD. An interesting note, is that this analysis only took 30 minutes on a Sun Sparc 1000.

Analysis continued in 2001 by Drew Streib [44] however the website is no longer available, but was able to be accessed by the Internet Archive Wayback Machine [11]. Streib analyzed 1,461,786 keys and found the strong set to be 10,828 keys strong. The average MSD was calculated at 6.6741. Streib continue the analysis until April 2002 when there was 1,945,876,437 keys total and 12,285 keys in the strong set. The source code used to do the analysis, keyanalyze, was released under GPL open source terms.

Drew Streib's research points to more resent research by Jason Harris [22]

utilizing the keyanalyze source code, however again the website is offline and this time the Internet Archive does not have any useful records.

In 2004, Jrgen Cederlf conducted more research and analysis of the web of trust, but this time it was more graphical [**?**]. Cederlf graphed the web of trust as a group matrix such that, "A dot in column x, counting from left, and row y, counting from the top, means that key number x has signed key number y. A red dot means a signature of level 0 or 1, a blue dot level 2 and a green level 3. Signatures which are not on the primary User ID are represented by a darker dot." The image generated appear as a leaf when keys are sorted by MSD and can be seen in figure 1.

Henk P. Penning continued the analysis, which continues today, and is probably the most complete with many interesting graphs [36]. The most recent graphs have been included in appendix A.10. While unclear when the research began, the graphs present data from 2003 to current. All recent data shows that the strong set is growing and that the MSD to all keys is decreasing. The strong set contains approximately 50,000 keys. The number one ranked key, Peter Palfrader, has a MSD of 3.5832. One interesting piece of data from Penning's research is the average degree or signatures per key within the strong set is approximately 10. That means that the average user has, on average, 10 signatures. There are of course many users which have hundreds of signatures which then act as trust anchors

Penning's graphs were updated with an idea from Matthew Wilcox to use the data from Streib and Harris and graph the data over time. These

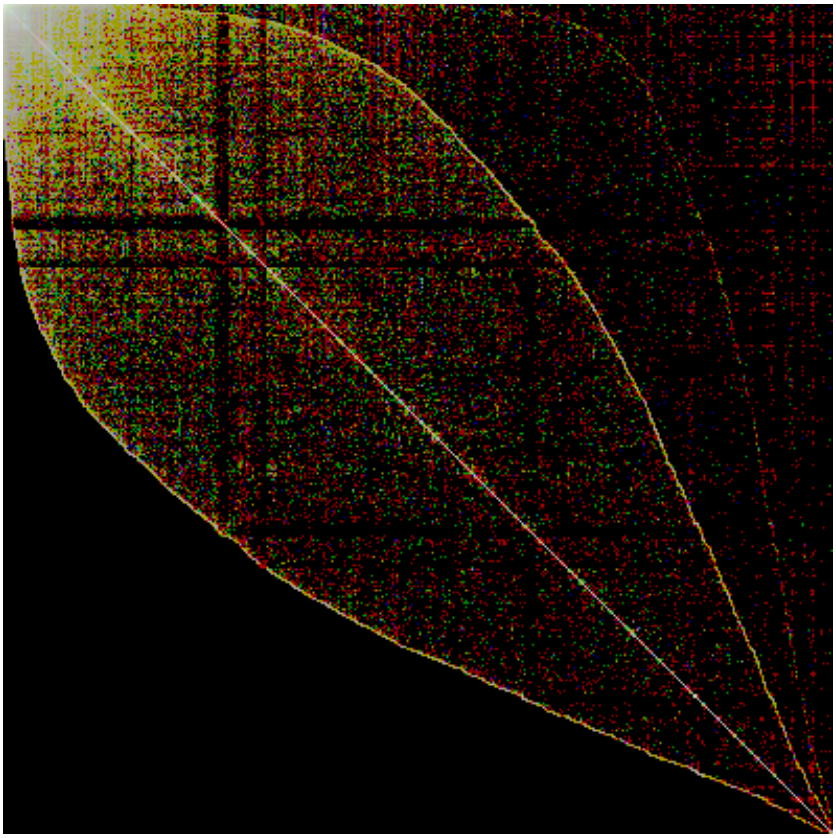

Figure 1: OpenPGP Keys graphed as matrix, sorted by MSD.

graphs were similar to a stock index or NASDAQ or if your from the UK, the FTSE 100, commonly called the Footsie. The footsie graphs by Wilcox showed the change in strong set over time and the change in top 50 keys, top 1000 keys to name a few data points. The graphs have been included in appendix A.11. Subsequently penning's graphs were updated to show the footsie idea and are also included in appendix A.10.

Our own analysis of the web of trust top 50 keys is included in appendix A.5. We show the change in the top 50 keys from 2008 to 2013. The top 50

keys are fairly stable, with the following key statistics: 6% No Change, 22% New, 28% Upward Moving, 44% Downward Moving. The downward moving keys are mainly a result of new keys introduced into the top 50. Most of the old top 50 keys remained in the top 50 even though their place may have changed. In analyzing the new keys in the top 50 we find that the keys are vary strongly connected with most keys with over 200 cross signatures.

- AAE6022E, Karlheinz Geyer has 1030 cross signatures.
- BAB58229, Marcus Frings has 526 cross signatures.
- 4743206C, Joachim Breitner has 259 cross signatures.
- 88C7C1F7, Steve McIntyre has 269 cross signatures.

4.5 Miscellaneous Research:

Gutmann [21] conducted a survey in which respondents chose email address, DNS, or IP address as a better identifier than a X.500 Distinguished Name. Respondents to the survey selected a repository presence check versus a more traditional CRL. Respondents also chose HTTP access over LDAP as an access method. For these reasons and possibly easier application development this thesis chose OpenPGP. OpenPGP also does not have the rigid structure associated with its PKI/X.509 counterpart.

In a more recent article [13], Stephen Farrell, suggests that we continue with our current PKI and not reinvent things until something better comes

along. He outlines the reinvention and alternatives over the years; incorporating XML and JSON into PKI, Simple PKI (SPKI), XML Key Management Specification (XKMS), and DANE (DNS-based Authentication of Named Entities). All of which have yet to replace our X.509 based PKI system. Farrel then proceeds to outline two new pieces of technology that he speculates will allow clients to use their own PKI. First, he calls for a key-registration service which would register client public keys for authentication to web services. Second, a strategy to bind different keys from different devices to the same user account to allow for device mobility.

In contrast to Perrin's paper [37] on the use of "cryptoIDs" this thesis presents that the use of current OpenPGP key fingerprints as a mechanism to key distribution can be used. Perrin also wished to establish a more long-lived identifier or "cryptoID" under which public/private key pairs would be created. This description comes very close to how OpenPGP can create a long lived parent key with shorter lived subkeys used for encryption and signing. The user can establish a long lived key through the parent key and its collected signatures. Perrin used "cryptoIDs" certificates instead of X.509 or OpenPGP and established their own format. Also, in regards to fingerprints and hashing mechanisms the particular implementation can be updated much the same as how the encryption algorithms are updated to prevent attacks.

Finally, there is a social stigma associated with encryption. Gaw [16] has studied how many people see encryption and has identified that many see

it as a burden and for the paranoid. By infusing encryption into a community and group dynamics this thesis would hope to make it easier and remove the paranoid stigma attached to using encryption. Through the use of easier-to-use applications this thesis would remove the burden and extra steps associated with encryption. This thesis presents a trust model to remove some burden in verification of keys through automatic key tagging with the use of Trust Rules.

5 Description of Trust Model:

In developing a new trust model there are a few requirements that this thesis wishes to take into consideration. First, the trust model should work with the current "Web of Trust" OpenPGP model as well as develop its own mechanism for establishing trust. The Web of Trust has been well established in user's minds and is a good model for securely establishing communications. The end user should maintain their responsibility in establishing how and who they choose to trust. Second, our new trust model should maintain and, in most cases, increase security from the Web of Trust model, while being resilient and resistant to attacks. Third, our new trust model should be easy for end users to use. Crypto-systems have historically been hard to use, so this new trust model should be developed with usability in mind.

In addition to the requirements this thesis also has developed this trust model with the following assumptions.

1. Trusted Introducer and/or Certification Authorities should not be trusted solely on their own word, but should be confirmed through others assertions as well.

2. Any single Trusted Introducers and/or Certification Authorities may make mistakes, can be compromised, or act without regard to a specific user.

3. The only truly trusted party is that of the end user to which the keyring belongs– Only that user's certifications can be fully trusted.

4. Key material will most likely be exposed, compromised, attacked and will need to be renewed or revoked at some point in the future.

5.1 OpenPGP Integration:

Integration in OpenPGP is a requirement, because it is a well established crypto-system that many people use for securing their communications. The likelihood of creating and gathering new users to a new crypto-system would be problematic and only serve to fracture the crypto-system landscape. By building on what others have already done this thesis seeks to leverage their contributions and implement our new trust model within the already established protocols. This thesis has already discussed the OpenPGP specifications with some detail, so it will now focus on what would be needed to implement the trust model within OpenPGP. This thesis does not wish to

change the OpenPGP specification nor does it think it is needed. Our current trust model can be implemented without changes to the OpenPGP RFC 4880 Standard [4].

5.2 Trust Packets:

OpenPGP RFC 4880 [4] allows for a "Trust Packet" that is used internally by software implementations. These trust packets should be ignored on input to programs and also should not be exported as output outside of the local keyring by programs. By using these packets this thesis presents a new trust model. This trust model can be developed without fear of users tainting the current Web of Trust. Additional trust models may be developed by end users to fit their security requirements. Because the Trust Packets can not be exported it allows only the current end user to establish trust with them and allows the program to use these packets for management of our trust model. The RFC leaves the format of the Trust Packets to the software package implementation so, this leaves us room to insert our trust model.

This thesis will use the Trust Packet to store a tag or folder marker to denote which "Trust Rule" was satisfied when the trust calculations were run. This tag will serve to quickly establish the corresponding Trust Rule with the key without recalculating its trust. This will serve as a caching mechanism so that the trust model can safely operate on numerous keys without overloading the application with new trust calculations. This is not to say that the tags are cached indefinitely, but only when the program does

not upload or download new or updated keys. Whenever a key update is performed by the program key trust calculations must be preformed again on the updated keys to verify the trust rule corresponding to applied tags are still valid. For example, if the key has been updated its trust calculations must be run again to confirm the validity of the tag or tags associated with it. When keys are updated a key may have been revoked, may contain more signatures, or signatures may have been revoked on that key by their signators, so this recalculation of trust rules must take place.

A tag will consist of a Name Field, formatted as string of 255 bytes in length, plus an ID Field consisting of two octets that will specify the unique identifier. This ID Field will be used to distinguish the tag on keys and in Trust Rules. Keys may be tagged by more than one tag at a time with each tag having its own corresponding Name and ID Fields.

Our Trust Model will use the following format for the The Trust Packet:

- One octet consisting of the version number. (1)
- One octet stating the number, in bytes, of the tag or tags that follow.
- One or more octets consisting of tag IDs.

5.3 Trust Rules:

This thesis will use "Trust Rules" to determine when and if a key should be trusted. These Trust Rules establish clear rule-based conditions that must be satisfied for a key to be valid and its identity trusted. This type of trust

is different from the trust established in a trusted introducer or a friends signature on other's key. Our trust model is only used in confirming trust in an identity not for calculating or introducing trust to or through others. All trust calculations using our trust model are kept internal and should not be the basis for signing another's key. Key signatures should be evaluated the same as if using the Web of Trust, in that you manually confirm a key. The signing of keys should be maintained through the normal methods via manual verification of key IDs, key fingerprints, government IDs, key signing parties, etc.

Trust Rules can be thought of as filters or tags that sort email. The Trust Rules operate on OpenPGP keys and tag them when a rule is satisfied. The criteria for satisfying a rule can be as simple as a key having a particular signature to more complex conditional operations. Conditional Trust Rules may calculate the trust from a group of keys, relative age of key, number of keys, etc.

Trust Rules are created by key value pairs, operators: =,<,>, AND, NOT, OR and built-in functions: GROUP, CONTAINS, BEGINSWITH, ENDSWITH. The key can be any OpenPGP field and the value is any valid input for that respective field. A Trust Rule may be made of one or more expressions. Each of these expressions have the form of: Expression = 'OpenPGP field' Operator OR Keyword 'value'. A Trust Rule may combine multiple expressions via logical operators to create larger more complex expressions.

5.3.1 Operators:

- **Equals "=" :** Is the comparison operator in which a string or numerical comparison is done on the key and its value pair. A successful comparison will result in a true condition.

- **Less than "<" :** Is the less than operator in which a numerical comparison is done either integer or date wise comparison. A successful comparison (of lesser value) will result in a true condition.

- **More than ">" :** Is the more than operator in which a numerical comparison is done either integer or date wise comparison. A successful comparison (of more value) will result in a true condition.

- **AND :** Is the AND operator which performs a logical conjunction on two expressions which outputs the result. Both expressions are required to be true for a true output.

- **NOT :** Is the NOT or Negative operator which performs the negation of an operation. Such that the result of an operation is the direct opposite from what it should have been. True becomes false and false becomes true.

- **OR :** Is the OR operator which performs a logical disjunction on two expression which outputs the results. Both expressions are required to be false for a false output. Where as, any true input results in a true output.

5.3.2 Functions:

Functions may be used anywhere an expression would be allowed.

- **GROUP:** is used to allow the evaluation of a group of values with a threshold for valid (successful) evaluation. It has the format of GROUP(key,threshold) {item1, item2, item3, ... itemN} where threshold is an integer value that must be met for the evaluation to be true and item1 through itemN are a list of values to compare against.

- **CONTAINS:** is used to do a sub string comparison search on a field. CONTAINS returns true if a sub string match has been found within the parent search string. Its format is CONTAINS(key,subString) where key is the OpenPGP field name and subString is the search string that if found within key will return true.

- **BEGINSWITH:** is used to search a key value for a sub string at the beginning of a string. The search is anchored at the beginning of the parent string and only returns true if the parent string starts with and matches the sub string specified. Its format is BEGINSWITH(key,subString) where key is the OpenPGP field name and subString is the search string that is anchored at the beginning.

- **ENDSWITH:** is used to search a key value for a sub string at the end of a string. The search is anchored at the end of the parent string and only returns true if the parent string ends with and matches the

sub string specified. Its format is ENDSWITH(key,subString) where key is the OpenPGP field name and subString is the search string and anchored at the end.

5.3.3 Example Trust Rules:

To tag all Version 4 OpenPGP keys it can be written as:

```
Version 4 Keys:  Version Number = 4
```

To tag all RIT keys within our key ring it can be written as:

```
RIT Keys:  ENDSWITH(User ID, ‘‘rit.edu’’)
```

To tag all RIT keys signed by 3 signatures out of a group of 5 keys it can be written as:

```
RIT Group:
GROUP(Issuer,3){4E3F1504,C74E9DC5,B6D2CB01,10E8C93F,E9DC5B6D,194BC24A}
AND
ENDSWITH(User ID, ‘‘rit.edu’’)
```

Notice the use of the User ID. This would then require that the key being evaluated also belong to an rit.edu entity and not allow any keys which only satisfy the GROUP clause to succeed. Also, for display purposes this thesis has used Key IDs, when in implementation the use of key fingerprints or the keys themselves should be used to prevent collisions within rules.

5.4 Resistance:

In developing new Trust Rules for use in our new model this thesis makes sure that the rules are resistant against attacks from outsiders and eavesdroppers. It may be possible for an attacker to setup an entire network of Certification Authority Servers, End User Keys, Signatures, etc. so that they could insert their mistaken OpenPGP keys into a good system. To prevent this, the thesis has proposed a system and trust model that is resistant to false claims while still allowing collaboration. Such a system should be setup to verify claims through more than one party and utilize a system that rewards good parties and punishes bad parties.

5.4.1 Web of Trust:

The Web of Trust is resistant to brute attacks such as the previous example as long as each single user verifies keys manually. If a user starts to sign keys indiscriminately and does not verify the identity or the key holder they degrade the value of their own signature. This degradation of signatures can eventually affect the entire web, however, the person would hopefully obtain a reputation of not carefully verifying keys. This reputation mechanism is mostly social and would not strictly insure that our good web of trust is maintained. It is feasible that an entire network of false keys may be setup and interconnected that is entirely false. This false network could appear to be a valid and good behaving part of the web with interconnected key signatures, but is in fact all run by an attacker.

While manual verification can be burdensome, it is vital to maintaining a proper Web of Trust. This verification however, sometimes is impossible given the miles that may exist between communicating entities. So, to forgo this manual verification one can use third-party verifiers or "Trusted Introducers" to perform this verification. This however, has the affect of allowing the Trusted Introducer free reign over what they claim to be valid. Once trusted a Certification Authority or Trusted Introducer is free to sign any key they wish, which in fact could be an attackers key. Our trust model can combat this side affect by requiring, if desired, that more third parties verify the identity and certification claims of these Trusted Introducer or Certification Authorities.

5.4.2 Our Trust Model:

Through the use of multiple trusted third parties this thesis wishes to mitigate one or more rogue certification authorities from polluting the signature web with false claims. These third party authorities are chosen by the end user and inserted into a Trust Rule with the GROUP function so that a subset of them must agree before tagging the key as valid. By using an internal non-exportable trust packet the thesis prevents the user from automatically signing keys without the manual verification that should be required. Signatures by end users should denote verification of identity by said user, not the successful evaluation of a user's established Trust Rule(s). Users are free to create their own set of Trust Rules based on their group interactions

and own personal security requirements. By not allowing the exportation of Trust Signatures and the automatic signing of keys this thesis presents that our trust model can maintain the resistance to attack that the Web of Trust has while increasing the ease of use.

5.5 Resilience:

Any crypto-system or trust model needs to be resilient in the face of attackers. The need to identify and recover from attacks is paramount, since attacks and the possibility of compromise should be expected. The ability to deal with and recover from an attack is as vital as the system's ability to resist an attack. Our trust model allows for the recovery from an attack the same way as the Web of Trust deals with attacks, through the revocation of signatures and keys.

5.5.1 Web of Trust:

If a user has signed a key with their signature and they no longer believe the key represents or belongs to the certified person claimed by the key the said signature may be revoked by the user. (One should wonder how carefully they verified the identity of the person to begin with before signing the key.) In this case the user has stopped certifying the other users identity and key pairing.

There are other reasons that a signature may be revoked, such as an employee leaving a business which included an email address and associated

OpenPGP key. The revocation of the key in this case does not state that the key identity is in question only that the person in question no longer maintains a persona with that company. If not revoked it may appear that the person still maintains an email address and works for said company.

If a user feels their own keys may have been compromised they can revoke the entire OpenPGP key. This could be very bad for the user as they will lose all the accumulated signatures on that OpenPGP key and will have to collect them again, that is have others sign the new key. In the case of a key compromise the revocation should state as such with a "Key material has been compromised" signature.

5.5.2 Our Trust Model:

While our model uses the same revocation technique as the Web of Trust it does have some intricacies when a trusted third party has been found to be non-trust worthy. Lets look at what would normally happen if a Trusted Introducer was found not to be so trustworthy. First, the user would revoke their signature and remove the Trusted Introducer from their trusted list. Next, all the trust calculations would need to be recalculated. After these calculations any key that was only verified by the now distrusted party would need to be removed from the trusted section of the keyring.

With our trust model the revocation of a trusted certifier would also include a recalculation of trust, but the revocation of a signature would only be required if the user signed the key. A signature of trust is not needed

in our trust model, but can be used to further enhance security, only the inclusion in a Trust Rule is required. Obviously the Trust Rules must be updated to exclude the untrustworthy certifier.

5.6 Usability:

This thesis needs to make sure its new trust model is easy to use. In software testing of PGP version 5.0 Whitten and Tygar [45] found that many users could not create keys and other associated OpenPGP tasks in a respectable time frame. End users can become easily confused and need easy to understand visual clues. In developing our Trust Rules interface this thesis has used a common window type that users should already be familiar with. The window is based on Mozilla Thurderbird's [9] email filter window. A mock-up of a proposed window for adding tags is shown in Figure 2.

In Figure 2 this thesis presents an example of a Trust Rule that would match all keys that end with "rit.edu" thus matching all RIT email. These keys will then be tagged with RIT. While this example is somewhat simple the idea of being able to sort and classify keys in an automatic fashion allows us to quickly and easily setup our Trust Rules and then go about our business of using the keys in communication. Once Trust Rules have been defined this should reduce the amount of time spent by the end user analyzing keys and the signatures that certify them. This manual process can now be carried out by the Trust Rules with keys tagged as the user has decided. The user only has to import the new key into their keyring and the Trust Rules will

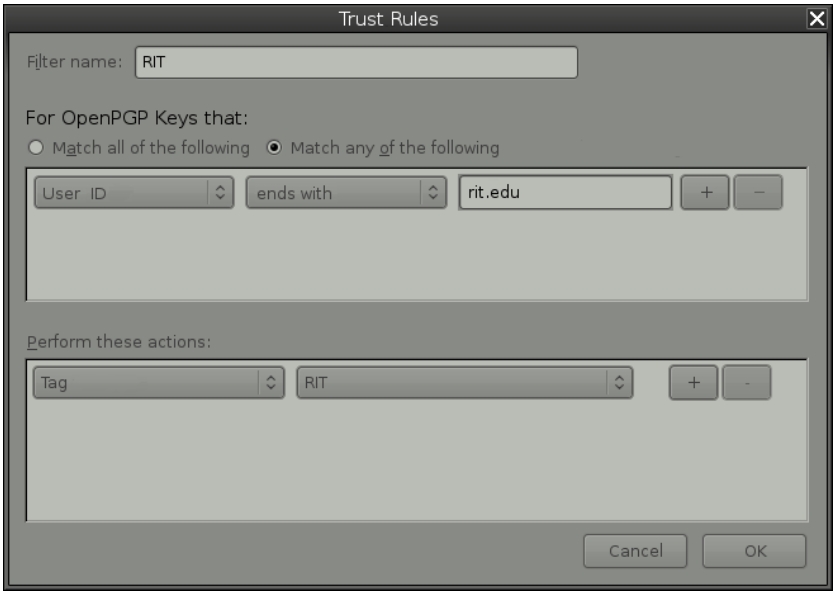

Figure 2: Mock-up of adding a Trust Rule.

be evaluated and the key tagged automatically.

In establishing the relationship between tagged keys and non tagged keys this thesis presents the need to keep it clear in the end users mind, otherwise their assumptions might be misplaced. The interface should make it clear that tagged keys are not the same as Trusted Keys which have been verified and signed by the user. Tags allow for automatic sorting and classification of keys into groups. The tags or groups might have increased security associated with them, but that may not always be the case. Since the user is capable of establishing any Trust Rule that they wish, it may be that they have added Trust Rules that do not have any additional security benefit. The decision of establishing more trust in tagged keys should be left to the end user and how they implement their Trust Rules. Each end user should know if a Trust

Rule is for security or nearly for sorting and classification. With that in mind this thesis is treating all tagged keys as a separate case from Trusted Keys, but not grouping them in with "Other Collected Keys" as some software packages call non trusted keys. This distinction should be clear after looking at Figure 3 which shows a mock-up of how this thesis see Tagged Keys being integrated with current software packages.

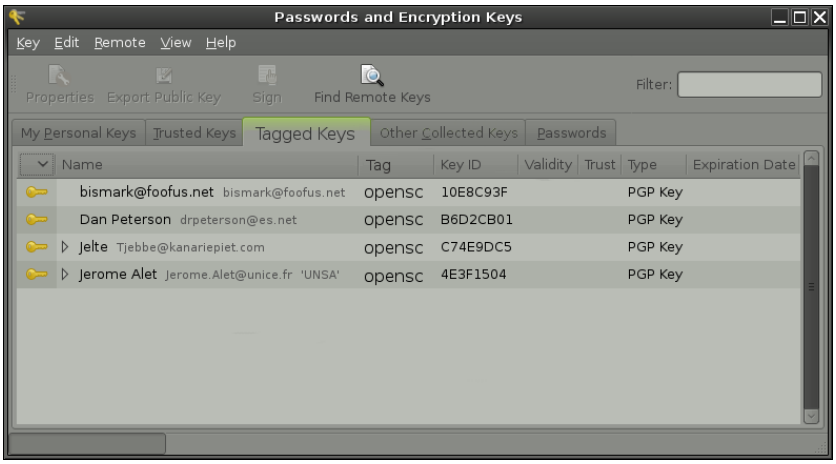

Figure 3: Mock-up of Key Management using Tags.

To denote the distinction that this thesis would wish to keep between Trusted Keys, Other Collected Keys, and Tagged Keys another tab has been created for the Tagged Keys as shown in Figure 3. This mock-up was derived from the encryption manager Seahorse version 2.20.1 [10]. Notice that a "Tagged Keys" tab to the already available tabs. It also tagged the keys as "opensc" because they belong to opensc mailing list participants. With the presented mock-up this thesis hopes it provides an easy to use interface for the end user to interact with.

6 Evaluation:

In this section the thesis will evaluate our new trust model through user case studies and later with some quantitative analysis. This thesis presents where Trust Rules makes sense for the end user and how it can simplify the key verification process.

6.1 User Case Study:

6.1.1 Rochester Institute of Technology:

Using RIT as an example, RIT could automatically sign one PGP key for all authenticated users through an online system that integrates into the campus email system or other online services, such as myRIT. Next, each department could sign a user's key if certain requirements are met such as: physical presentation of RIT ID, membership in said department, RIT ID Number, RIT username, and OpenPGP key fingerprint. The RIT ID should be used to confirm the persons identity, and also match with the person's ID picture with the picture on their OpenPGP key. After this all users in the department and the rest of RIT would know that:

1. The user is a current RIT student/faculty/staff.
2. The user is a member of a particular department.
3. The user's picture is correct.
4. The user's public key has been manually confirmed.

The key would only be considered valid if two signatures are on the key: one by RIT and one by the department to which the person belongs. The model that this thesis has described above requires two signatures to be valid. This simple model would prevent an attacker at any one department from falsely verifying keys, because they must first be signed by RIT.

6.1.2 Evaluation of Model:

In evaluating the model and how it works for this case or class of organization this thesis has evaluated how the model compares to other trust models. First, with X.509 Certificates it is not possible to prevent or limit key signatures once a Certification Authority has been trusted. All signatures issued by a CA will be trusted. So, our model is more flexible than X.509. Second, in comparison to OpenPGP Web of Trust this thesis finds that there would be a reliance on one person to establish the trust of keys. This reliance reduces security because of the possibility of compromise and it also goes against our established criteria for a new trust model. X.509 does have the ability to distribute Certifications Authority roles to multiple entities, but each can sign and establish new keys completely independent of each other. Our model as described with the established Trust Rules would allow for independent operation while maintaining central RIT control over the key distribution and signing.

6.1.3 Social Networking:

Using MySpace, Xanga, bebo, flickr, FaceBook, or any other social networking site as an example this thesis presents a secure mechanism for key exchange and verification of identity. The website itself could sign keys of individual users thereby verifying the digital pseudonym. After which friends and associates can verify the information in person. These contacts may be new friends found through the website. Community friends can meet to verify identity and once a threshold of key signatures has been established the group or community Trust Rule would be evaluated and make the person a member in good standing with the community. These networks are already established and can provide a number of contacts with which a user may communicate. By having the community develop the rules and sign keys it provides incentive for new users to join in with secure communications and lowers the bar for new secure pathways to develop. If an individual had to approach each user and verify their key it would be a daunting task, but with the community undertaking this task it distributes the load and allows for quicker keyring creation. This social networking model would mostly verify digital pseudonyms, but can be mixed with other trust models, since the model itself does not prevent multiple rules being run simultaneously.

6.1.4 Evaluation of Model:

In this form the Trust Rules are developed by the group or community that is going to use them. The specifications of the group need to be satisfied. In

this case there is a high barrier to enter, because of cost. The cost of setting up a Certification Authority or other type of structure for each community would be prohibitive. Our trust model, however allows for each group to have their own rules and to operate completely independently of each other.

6.2 Quantitative Analysis:

6.2.1 Number of Signatures:

In analyzing our trust model this thesis has picked a particular Trust Rule to analyze. While other Trust Rules may be developed the best general purpose Trust Rule this thesis presents is the N of K Servers Trust Rule. The thesis will evaluate this Trust Rule with a purposed threshold where $N = 10$ and $K = 15|20$ servers. Thus the Trust Rule will have the following format:

```
Trusted Keys:  GROUP(Issuer,10){keyID1, keyID2, keyID3,
                keyID4, keyID5,..., keyID15|20}
```

This should provide the right amount of servers for there to be competition among the authentication severs while maintaining a small enough number to be feasible. The threshold is chosen so that a large percentage of servers must agree before a key is verified as valid. For increased security one only has to increase the value of N to a larger percentage of K.

To analyze our chosen Trust Rule this thesis evaluates the number of signatures versus the number of total users in a given population. For our Trust Rule with the threshold for the GROUP function set at 10 the maximum sig-

natures any user should be required to get would be 10. This maximum is shown in Figure 4 where it has been graphed and used for our analysis. Next, this thesis compared our model with those of other types of trust system, such as X.509 Certificates and the more traditional Web of Trust. The Web of Trust worst case scenarios is that each user has to obtain every other users signature. That is, the total number of signatures is equal to the total number of users. The Web of Trust is graphed in Figure 4 as well. The Web of Trust best case scenarios is one where every person in the web knows enough people to connect themselves with everyone else in the network through just one hope. Or put differently, if I know 10 people and those 10 people know another 10 people we have just connected 100 people while allowing myself to only know the first 10 people directly. This relationship through friends is established with a limit of knowing the next person only through one other person. The relationship can not be a distance greater than one person away. This best case Web of Trust relationship happens to be $S = \sqrt{U}$ where S is the number of signatures required and U is the number of total users. This best case is of course a best case and not very probable in the real world, because each person would need to know distinctly different people and never overlap, otherwise the number of signatures required by each person would have to be increased because of duplication. Finally, this thesis has graphed X.509 Certificates by showing that only one signature or certification is required. By having Certification Authorities integrated into Operating Systems, Web Browsers, and Email clients it reduces the certifications down to

only one as long as an already trusted CA is chosen.

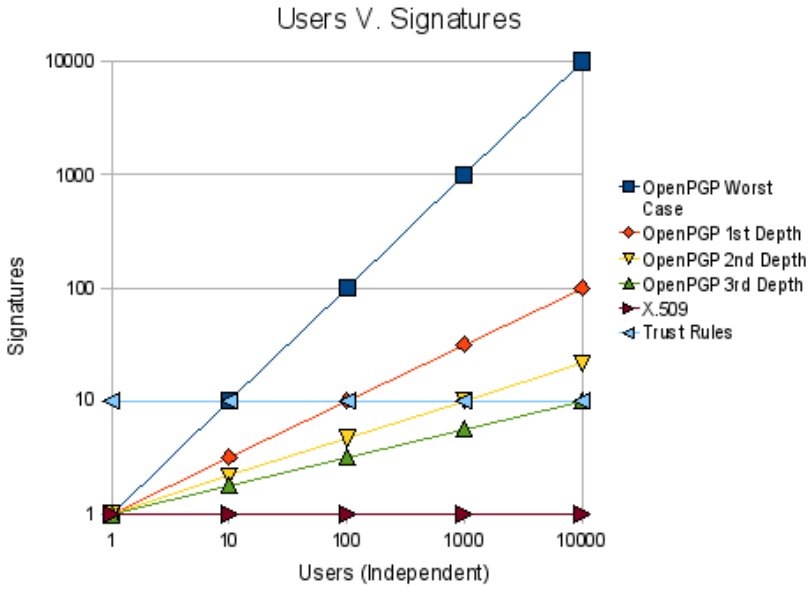

Figure 4: Number of Users Versus the Number of Signatures.

Now that this thesis has presented the different trust models graphed in Figure 4 it is possible to analyze the trust models and evaluate each by comparing their intersections, and respective slopes. First this thesis presents that all of the trust models start with one signature where as the Trust Rules starts with ten required signatures. As users are added to the network/web each model requires more and more signatures, except the X.509 trust model and Trust Rules. The X.509 trust model incorporates Certification Authorities certificates into the host computer via the Operating System or User Application Programs. Because of this X.509 only requires one signature from an already trusted CA to be almost one hundred percent accepted.

Trust Rules does not require more signatures because this thesis has decided to use fifteen to twenty certification authorities to establish our trust in an end user's identity. These same CAs will be used regardless of the number of users in the network.

Next, this thesis presents that the slopes and associated number of signatures differ between the worst case and best case scenarios of the Web of Trust model. As stated previously, the worst case is $S = U$ and the best case is $S = \sqrt{U}$ where S is the number of signatures and U is the number of users. Any slope that is less than that of the best case scenarios can be considered an improvement by reducing the total number of required signatures. That being said, the security of such a system might be questioned if the number of required signatures is substantially lower. There may be a compromise between ease of introduction into the encryption system and sustained security over time as new users are added.

In evaluating where our Trust Rules model with its current rules may be used, it appears that it would be a good model when the number of users if over ten. Even though it does not surpass the Web of Trust best case until over one hundred users our Trust Rules would probably be a good choice. This is because the best case scenarios is less likely to appear in the real world. It is more reasonable to assert that the required number of signatures is between ten and one hundred.

7 Future Research:

7.1 X.509:

More research needs to be done into using X.509 with Trust Rules. This thesis chose OpenPGP because of its lack of rigid structure and reliance in a hierarchical tree and because there was a clear path to implementing our Trust Rules within software packages with minimal specification and application changes. This does not forgo the X.509 Trust Model from implementing Trust Rules or another type of conditional rule-based system, but it appears at this time that there would be substantial work involved in application, specification, and social change. There is already a widely used system developed around X.509 certificates, so that would further complicate the development of a new trust model for that system. The trust model would more likely need to make better business sense economically than it would security wise to be adopted. It still would be possible and interesting to see some type of conditional rule-based trust model implemented in X.509.

7.2 Additional Trust Rules:

This thesis has explored a very limited scope with regards to the many possible Trust Rules that could be developed. Many different Trust Rules need to be developed and evaluated to test their security additions or subtractions. Because of the flexibility Trust Rules can very easily be written poorly in a manner that would harm the end users security. Such a Trust Rule would be

counting the number of signatures on a given key. If such a Trust Rule were put into place it could easily be bypassed by an attacker.

Our current Trust Rule has a slope of zero and does not scale as the number of users scales. While this can be a feature it can also be a security weakness, as each Certification Authority is trusted with more and more verifications. To combat this reliance on any one Certification Authority this thesis would need to develop a Trust Rule that scales some, but one that does not have a slope greater than OpenPGP Web of Trust. There could also be a Trust Rule that steps up the threshold when a given number of keys is trusted through that rule. As more keys are trusted through a given rule that rule's threshold could automatically be increased so that the total amount of trust in any set of Certification Authorities is reduced.

7.3 Email Integration:

A natural extension to Trust Rules would be to integrate the tags feature into an email client so that it can be used for sorting and other email related tasks. The tags could be used to automatically bypass spam filtering allowing an email to go directly to a specific folder based on the tag that was applied to the email.

8 Conclusion:

This thesis presents that "Trust Rules" can provide a service that until now has been missing in most cyrptosystems. The use of user established rules provides the user with greater flexibility and the ability to decide their own security requirements. This thesis sees conditional trust through the use of rule-based matching and key identification as a natural evolution from manual key verification. Through the use of tags the end user may easily organize and catalog their keys to enhance their security landscape. These rules can be very simple and provide organization only, or they may be very complex and incorporate multiple Certification Authorities and automatic key revocation. In either case this thesis find the reduction in manual steps taken through automatic signature key review and verification appealing. This thesis presents that general purpose "Trust Rules" as outlined will work for most users in establishing key validity and trust. For those more advanced or more security conscious users a mixing of "Trust Rules" and Web of Trust manual verification is possible. Our trust model does not prevent continued use of the Web of Trust, but merely makes additions to user interactions to make some processes more automated. This thesis presents that the automated tag feature would enhance any user's experience so that they may organize their keyring more effectively. Overall, this thesis presents "Trust Rules" with tagging as a valid option regardless of the user's security preferences.

References

[1] Alfarez Abdul-Rahman and Stephen Hailes. A distributed trust model. In *NSPW '97: Proceedings of the 1997 workshop on New security paradigms*, pages 48–60, New York, NY, USA, 1997. ACM.

[2] D. Atkins, W. Stallings, and P. Zimmermann. PGP message exchange formats. RFC 1991, Internet Engineering Task Force, August 1996.

[3] Kemal Bicakci, Bruno Crispo, and Andrew S. Tanenbaum. How to incorporate revocation status information into the trust metrics for public-key certification. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1594–1598, New York, NY, USA, 2005. ACM.

[4] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. RFC 4880, Internet Engineering Task Force, November 2007.

[5] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP message format. RFC 2440, Internet Engineering Task Force, November 1998.

[6] Liqun Chen, Dieter Gollmann, and Chris J. Mitchell. Authentication using minimally trusted servers. *SIGOPS Oper. Syst. Rev.*, 31(3):16–28, 1997.

[7] T.M. Chen and S. Abu-Nimeh. Lessons from stuxnet. *Computer*, 44(4):91–93, 2011.

[8] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, P2PECON '05, pages 128–132, New York, NY, USA, 2005. ACM.

[9] Contributors. Mozilla thunderbird. `http://www.mozilla.com/thunderbird/`, 1998-2008.

[10] Contributors. Seahorse. `http://www.gnome.org/projects/seahorse/`, 2002-2008.

[11] Various Contributors. Internet archive, wayback machine. `http://archive.org/web`, 1996.

[12] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag.

[13] Stephen Farrell. Not reinventing pki until we have something better. *IEEE Internet Computing*, 15(5):95–98, 09 2011.

[14] Simson L. Garfinkel, David Margrave, Jeffrey I. Schiller, Erik Nordlander, and Robert C. Miller. How to make secure email easier to use. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 701–710, New York, NY, USA, 2005. ACM.

[15] Simson L. Garfinkel and Robert C. Miller. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *SOUPS*

'05: Proceedings of the 2005 symposium on Usable privacy and security, pages 13–24, New York, NY, USA, 2005. ACM.

[16] Shirley Gaw, Edward W. Felten, and Patricia Fernandez-Kelly. Secrecy, flagging, and paranoia: adoption criteria in encrypted email. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 591–600, New York, NY, USA, 2006. ACM.

[17] Glenn Greenwald. Front: Exclusive: How nsa can see 'nearly everything you do online': Secret tool searches email, chat and social media use, Aug 01 2013. Name - NSA International Inc; Copyright - (Copyright , Guardian Newspapers Limited, Aug 01, 2013; Last updated - 2013-08-01.

[18] Glenn Greenwald. Front: Exclusive: Us orders phone firm to hand over data on millions of calls: Top secret court ruling demands 'ongoing, daily' data from verizon, Jun 06 2013. Name - National Security Agency; NSA International Inc; Copyright - (Copyright , Guardian Newspapers Limited, Jun 06, 2013; Last updated - 2013-06-06.

[19] Glenn Greenwald and Ewen MacAskill. Front: Revealed: how us secretly collects private data from aol, apple, facebook, google, microsoft, paltalk, skype, yahoo and youtube: Files prove existence of undercover operation codenamed prism, Jun 07 2013. Name - National Security Agency; Apple Computer Inc; NSA International Inc; Federal Aviation

Administration; Copyright - (Copyright , Guardian Newspapers Limited, Jun 07, 2013; Last updated - 2013-06-08.

[20] R. Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM.

[21] Peter Gutmann. Pki design for the real world. In *NSPW '06: Proceedings of the 2006 workshop on New security paradigms*, pages 109–116, New York, NY, USA, 2007. ACM.

[22] Jason Harris. Keyanalyze statistics. `http://keyserver.kjsl.com/~jharris/ka/`.

[23] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 3280, Internet Engineering Task Force, April 2002.

[24] R. Housley and S. Santesson. Update to DirectoryString Processing in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 4630, Internet Engineering Task Force, August 2006.

[25] Trung Dong Huynh, Nicholas R. Jennings, and Nigel R. Shadbolt. Certified reputation: how an agent can trust a stranger. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous*

agents and multiagent systems, pages 1217–1224, New York, NY, USA, 2006. ACM.

[26] ISO/IEC / ITU-T. Security architecture for open systems interconnection for ccitt applications, March 1991. ISO/IEC ISO 7498-2, ITU-T Rec. X.800.

[27] ISO/IEC / ITU-T. Information technology open systems interconnection the directory: Overview of concepts, models and services, August 2005. ISO/IEC 9594:2005, ITU-T Rec. X.500.

[28] ISO/IEC / ITU-T. Information technology open systems interconnection the directory: Overview of concepts, models and services, August 2005. ISO/IEC 9594-2 , ITU-T Rec. X.501.

[29] ISO/IEC / ITU-T. Information technology open systems interconnection the directory: Public-key and attribute certificate frameworks, August 2005. ISO/IEC 9594:2005-8, ITU-T Rec. X.509.

[30] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[31] Apu Kapadia. A case (study) for usability in secure email communication. *Security & Privacy Magazine, IEEE*, 5(2):80–84, March-April 2007.

[32] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Security Privacy, IEEE*, 9(3):49–51, 2011.

[33] Michael W. Lucas. *PGP & GPG: Email for the Practical Paranoid*. No Starch Press, 2006.

[34] Neal McBurnett. Pgp web of trust statistics. `http://bcn.boulder.co.us/~neal/pgpstat/19961206/`, 1996.

[35] Neal McBurnett. Pgp web of trust statistics. `http://bcn.boulder.co.us/~neal/pgpstat/`, 1997.

[36] Henk P. Penning. Analysis of the strong set in the pgp web of trust. `http://pgp.cs.uu.nl/plot/`, 2013.

[37] Trevor Perrin. Public key distribution through "cryptoids". In *NSPW '03: Proceedings of the 2003 workshop on New security paradigms*, pages 87–102, New York, NY, USA, 2003. ACM.

[38] Ajay Ravichandran and Jongpil Yoon. Trust management with delegation in grouped peer-to-peer communities. In *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 71–80, New York, NY, USA, 2006. ACM.

[39] Paul Resnick and Richard Zeckhauser. Trust among strangers in internet transactions: Empirical analysis of ebay' s reputation system. *Advances in Applied Microeconomics*, 11:127–157, 2002.

[40] S.B. Roosa and S. Schultze. Trust darknet: Control and compromise in the internet's certificate authority model. *Internet Computing, IEEE*, 17(3):18–25, 2013.

[41] S. Santesson and R. Housley. Internet X.509 Public Key Infrastructure Authority Information Access Certificate Revocation List (CRL) Extension. RFC 4325, Internet Engineering Task Force, December 2005.

[42] Flora Rheta Schreiber. *Sybil*. Henry Regnery Company, Washington, DC, USA, 1973.

[43] Mathew J. Schwartz. Nsa prism: Inside the modern surveillance state. *Informationweek - Online*, Jun 10 2013. Copyright - Copyright 2013 CMP Media LLC. All rights reserved. No part of the report or the data or information included therein may be reproduced, republished or redistributed without the prior written consent of CMP Media LLC; Last updated - 2013-06-11.

[44] Drew Streib. keyanalyze - analysis of a large openpgp ring. `http://dtype.org/keyanalyze/`, 2001.

[45] Alma Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *SSYM'99: Proceedings of the 8th conference on USENIX Security Symposium*, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[46] Matthew Wilcox. The footsie web of trust analysis. http://www.parisc-linux.org/~willy/wot/footsie/, 2008.

[47] Philip Zimmermann. Why I Wrote PGP. http://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html, 1991.

[48] Philip R. Zimmermann. *The official PGP user's guide*. MIT Press, Cambridge, MA, USA, 1995.

A Appendix

A.1 Terminology

The International Telecommunication Union (ITU) Rec. X.800 & International Organization for Standardization ISO 7498-2 [26] defines terms which are important to this thesis and are outlined below.

Asymmetric (encipherment): (e.g. public key) encipherment, in which knowledge of the encipherment key does not imply knowledge of the decipherment key, or vice versa. The two keys of such a system are sometimes referred to as the "public key" and the "private key".

Authentication exchange: A mechanism intended to ensure the identity of an entity by means of information exchange.

Authentication information: Information used to establish the validity of a claimed identity.

Confidentiality: The property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

Credentials: Data that is transferred to establish the claimed identity of an entity.

Cryptography: The discipline which embodies principles, means, and methods for the transformation of data in order to hide its information

content, prevent its undetected modification and/or prevent its unauthorized use.

Data origin authentication: The corroboration that the source of data received is as claimed.

Decipherment: The reversal of a corresponding reversible encipherment.

Decryption: See decipherment.

Digital signature: Data appended to, or a cryptographic transformation (see cryptography) of a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery e.g. by the recipient.

Encipherment: The cryptographic transformation of data (see cryptography) to produce ciphertext.

Encryption: See encipherment.

Key: A sequence of symbols that controls the operations of encipherment and decipherment.

Key management: The generation, storage, distribution, deletion, archiving and application of keys in accordance with a security policy.

Repudiation: Denial by one of the entities involved in a communication of having participated in all or part of the communication.

Password: Confidential authentication information, usually composed of a string of characters.

Peer-entity authentication: The corroboration that a peer entity in an association is the one claimed.

Privacy: The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Signature: See digital signature.

Symmetric (encipherment): (i.e. secret key) encipherment, in which knowledge of the encipherment key implies knowledge of the decipherment key and vice versa.

The following terminology is defined in ITU-T X.509 [29]:

Certificate revocation list (CRL): A signed list indicating a set of certificates that are no longer considered valid by the certificate issuer. In addition to the generic term CRL, some specific CRL types are defined for CRLs that cover particular scopes.

Certificate validation: The process of ensuring that a certificate was valid at a given time, including possibly the construction and processing of a certification path, and ensuring that all certificates in that path were valid (i.e., were not expired or revoked) at that given time.

Certification authority (CA): An authority trusted by one or more users to create and assign public-key certificates. Optionally the certification authority may create the users' keys.

CRL distribution point: A directory entry or other distribution source for CRLs; a CRL distributed through a CRL distribution point may contain revocation entries for only a subset of the full set of certificates issued by one CA or may contain revocation entries for multiple CAs.

Hash function: A (mathematical) function which maps values from a large (possibly very large) domain into a smaller range. A "good" hash function is such that the results of applying the function to a (large) set of values in the domain will be evenly distributed (and apparently at random) over the range.

Key agreement: A method for negotiating a key value on-line without transferring the key, even in an encrypted form, e.g., the Diffie-Hellman technique.

Private key: (In a public key cryptosystem) that key of a user's key pair which is known only by that user.

Public-key: (In a public key cryptosystem) that key of a user's key pair which is publicly known.

Public-key certificate (PKC): The public key of a user, together with some other information, rendered unforgeable by digital signature with

the private key of the certification authority which issued it.

Public key infrastructure (PKI): The infrastructure able to support the management of public keys able to support authentication, encryption, integrity or non-repudiation services.

Simple authentication: Authentication by means of simple password arrangements.

Strong authentication: Authentication by means of cryptographically derived credentials.

Trust: Generally, an entity can be said to "trust" a second entity when it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects. This trust may apply only for some specific function. The key role of trust in this framework is to describe the relationship between an authenticating entity and an authority; an entity shall be certain that it can trust the authority to create only valid and reliable certificates.

RFC 4880 defines the following terminology:

Keyring: A keyring is a collection of one or more keys in a file or database. Traditionally, a keyring is simply a sequential list of keys, but may be any suitable database.

A.2 Top 50 OpenPGP Keys, 2008

Data Collected from: 2008-04-13 Dataset available at http://keyserver.kjsl.com/ka/

Rank	Key ID	Name	MSD
1	94C09C7F	Peter Palfrader	3.5555
2	68FD549F	Martin Michlmayr	3.6357
3	C82E0039	Peter Palfrader	3.6473
4	F081195D	Matthias Bauer	3.6479
5	E263FCD4	Kurt Gramlich	3.6562
6	75BE8097	Florian Lohoff	3.6574
7	248AEB73	Rene Engelhard	3.6629
8	3F3E6426	Guido Guenther	3.6668
9	74E0B766	Andreas Mueller	3.6762
10	607559E6	Benjamin Hill (Mako)	3.6793
11	307D56ED	Nol Kthe	3.6855
12	00D8CD16	Alexander Schmehl	3.6890
13	797EBFAB	Enrico Zini	3.6978
14	F2CF01A8	Bdale Garbee	3.7071
15	C99870B1	Benjamin Hill (Mako)	3.7282
16	BC7D020A	Alexander Wirt	3.7318
17	C158CCED	Florian Lohoff	3.7325
18	CD15A883	Alexander Schmehl (private)	3.7337
19	258D8781	Michael Bramer	3.7388
20	7E7B8AC9	Joerg Jaspert	3.7436
21	19C9B6BA	Maximilian Wilhelm (uni)	3.7437
22	2BE16D01	Moray Allan	3.7484
23	5706A4B4	Simon Richter	3.7521
24	0F7A8D01	Norbert Tretkowski	3.7522
25	BD8B050D	Roland Rosenfeld	3.7528
26	9B7C328D	Luk Claes	3.7532
27	9ED101BF	Michael Banck	3.7646
28	3E8DCCC0	Martin Wuertele	3.7706
29	3FCC2A90	Amaya Rodrigo Sastre	3.7743
30	E10F502E	Marcus Frings	3.7764
31	EE0977E8	Jens Kubieziel	3.7790
32	1BF8DE0F	Roland Stigge (ernie)	3.7826
33	29499F61	Sam Hocevar	3.7849

34	A0ED982D	Christian Brueffer	3.7931
35	5A35FD42	Christoph Ulrich Scholler (FNB)	3.7968
36	5D64F870	Martin Zobel-Helas	3.7975
37	29F19BD1	Dr. Michael Meskes	3.7994
38	46F3212D	LaMont Jones	3.7994
39	44779E18	Fabio Massimo Di Nitto	3.8021
40	253E58E3	Noah Heusser	3.8042
41	9C67CD96	Torsten Veller	3.8086
42	58510B5A	Christoph Berg	3.8090
43	D98502C5	Elmar Hoffmann	3.8099
44	8A724E45	Stefan Roehrich	3.8112
45	DD934139	Patrick Feisthammel	3.8123
46	5B0358A2	Werner Koch	3.8124
47	95FECA34	Volker Gueth	3.8126
48	969457F0	Joost van Baal	3.8150
49	CF3401A9	Elmar Hoffmann	3.8185
50	6D8ABE71	Christoph Berg	3.8250

A.3 Top 50 OpenPGP Keys Graph

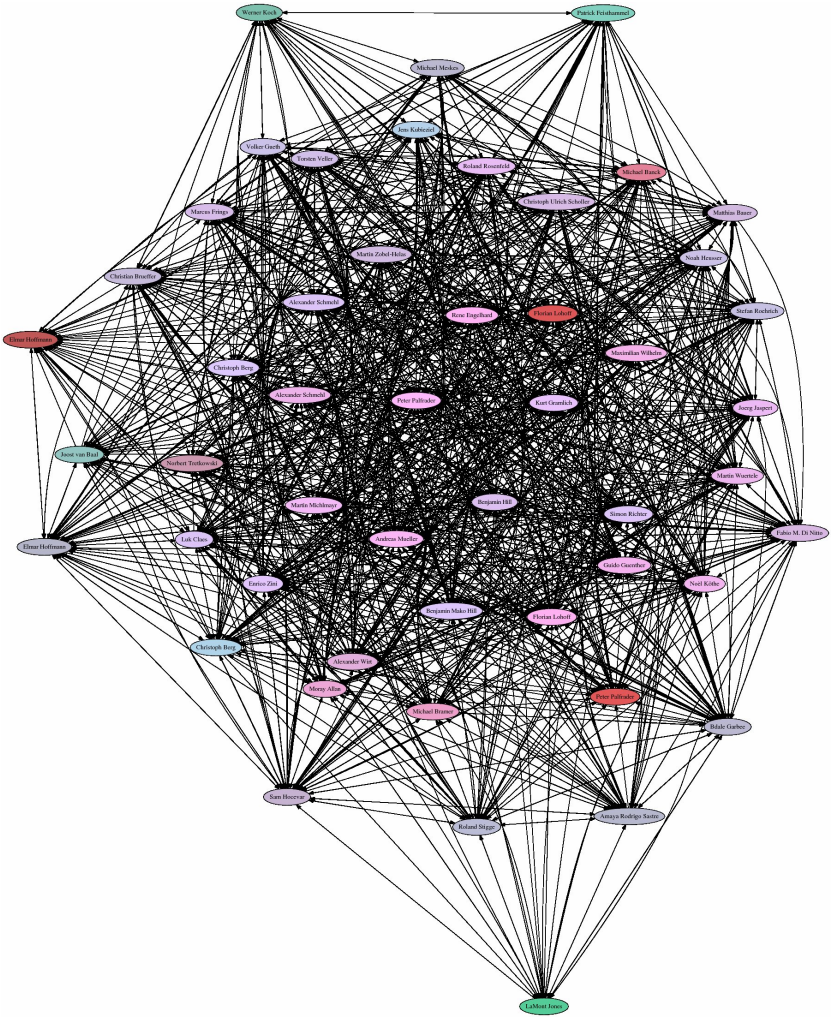

Figure 5: Graph of Top 50 OpenPGP Keys ranked by MSD. 2008

A.4 Top 50 OpenPGP Keys, 2013

Data Collected from: 2013-08-25 Dataset available at http://pgp.cs.uu.nl/doc/top_50.html

Rank	Key ID	Name	MSD
1	94C09C7F	Peter Palfrader	3.57031
2	607559E6	Benjamin Hill (Mako)	3.64598
3	C82E0039	Peter Palfrader	3.64762
4	65D0FD58	CA Cert Signing Authority (Root CA)	3.65037
5	68FD549F	Martin Michlmayr	3.67061
6	AAE6022E	Karlheinz Geyer (TUD)	3.67124
7	307D56ED	Nol Kthe	3.69334
8	E263FCD4	Kurt Gramlich	3.69883
9	248AEB73	Rene Engelhard	3.70122
10	C99870B1	Benjamin Hill (Mako)	3.70254
11	F2CF01A8	Bdale Garbee	3.70501
12	797EBFAB	Enrico Zini	3.71041
13	BAB58229	Marcus Frings (Work)	3.72188
14	E10F502E	Marcus Frings	3.72582
15	BD8B050D	Roland Rosenfeld	3.73317
16	74E0B766	Andreas Mueller	3.74222
17	3F3E6426	Guido Gnther	3.74547
18	F081195D	Matthias Bauer	3.74649
19	75BE8097	Florian Lohoff	3.75182
20	BC7D020A	Alexander Wirt	3.75615
21	9B7C328D	Luk Claes	3.76101
22	7E7B8AC9	Joerg Jaspert	3.77075
23	19C9B6BA	Maximilian Wilhelm	3.78249
24	2BE16D01	Moray Allan	3.78292
25	58510B5A	Christoph Berg	3.79384
26	D98502C5	Elmar Hoffmann	3.79597
27	4743206C	Joachim Breitner	3.80009
28	88C7C1F7	Steve McIntyre	3.80134
29	5706A4B4	Simon Richter	3.80223
30	9ED101BF	Michael Banck	3.80293
31	29F19BD1	Michael Meskes	3.80393
32	9C67CD96	Torsten Veller	3.80644
33	EE0977E8	Jens Kubieziel	3.81234

34	CF3401A9	Elmar Hoffmann	3.81279
35	C0143D2D	Christian Perrier	3.81391
36	5D64F870	Martin Zobel-Helas	3.81935
37	1880283C	Anibal Monsalve Salazar	3.82135
38	C158CCED	Florian Lohoff	3.8221
39	969457F0	Joost van Baal	3.82294
40	EC152942	Gerfried Fuchs	3.82378
41	0F7A8D01	Norbert Tretkowski	3.82437
42	8501C7FC	Sebastian Harl	3.825
43	5A35FD42	Christoph Ulrich Scholler (FNB)	3.82564
44	98016DC7	Josef Spillner	3.83135
45	A0ED982D	Christian Brueffer	3.83323
46	258D8781	Michael Bramer	3.83393
47	253E58E3	Noah Heusser	3.83535
48	29499F61	Sam Hocevar	3.83644
49	7244970B	Kurt Roeckx	3.83852
50	44779E18	Fabio M. Di Nitto	3.84032

A.5 Top 50 OpenPGP Keys Compared

Comparison of 2013 data to 2008 to note changes in the top 50 keys sorted by MSD. Key statistics: 6% No Change, 22% New, 28% Upward Moving, 44% Downward Moving.

Rank	Name	MSD	Change
1	Peter Palfrader	3.57031	No Change
2	Benjamin Hill (Mako)	3.64598	↑ 10th
3	Peter Palfrader	3.64762	No Change
4	CA Cert Signing Authority (Root CA)	3.65037	NEW
5	Martin Michlmayr	3.67061	↓ 2nd
6	Karlheinz Geyer (TUD)	3.67124	NEW
7	Nol Kthe	3.69334	↑ 11th
8	Kurt Gramlich	3.69883	↓ 5th
9	Rene Engelhard	3.70122	↓ 7th
10	Benjamin Hill (Mako)	3.70254	↑ 15th
11	Bdale Garbee	3.70501	↑ 14th
12	Enrico Zini	3.71041	↑ 13th
13	Marcus Frings (Work)	3.72188	NEW
14	Marcus Frings	3.72582	↑ 30th
15	Roland Rosenfeld	3.73317	↑ 25th
16	Andreas Mueller	3.74222	↓ 9th
17	Guido Gnther	3.74547	↓ 8th
18	Matthias Bauer	3.74649	↓ 4th
19	Florian Lohoff	3.75182	↓ 6th
20	Alexander Wirt	3.75615	↓ 16th
21	Luk Claes	3.76101	↑ 26th
22	Joerg Jaspert	3.77075	↓ 10th
23	Maximilian Wilhelm	3.78249	↓ 21st
24	Moray Allan	3.78292	↓ 22nd
25	Christoph Berg	3.79384	↑ 42nd
26	Elmar Hoffmann	3.79597	↑ 43rd
27	Joachim Breitner	3.80009	NEW
28	Steve McIntyre	3.80134	NEW
29	Simon Richter	3.80223	↓ 23rd
30	Michael Banck	3.80293	↓ 27th
31	Michael Meskes	3.80393	↑ 37th

32	Torsten Veller	3.80644	↑ 41st
33	Jens Kubieziel	3.81234	↓ 31st
34	Elmar Hoffmann	3.81279	↑ 43rd
35	Christian Perrier	3.81391	NEW
36	Martin Zobel-Helas	3.81935	No Change
37	Anibal Monsalve Salazar	3.82135	NEW
38	Florian Lohoff	3.8221	↓ 17th
39	Joost van Baal	3.82294	↑ 48th
40	Gerfried Fuchs	3.82378	NEW
41	Norbert Tretkowski	3.82437	↓ 24th
42	Sebastian Harl	3.825	NEW
43	Christoph Ulrich Scholler (FNB)	3.82564	↓ 35th
44	Josef Spillner	3.83135	NEW
45	Christian Brueffer	3.83323	↓ 34th
46	Michael Bramer	3.83393	↓ 19th
47	Noah Heusser	3.83535	↓ 40th
48	Sam Hocevar	3.83644	↓ 33rd
49	Kurt Roeckx	3.83852	NEW
50	Fabio M. Di Nitto	3.84032	↓ 39th

Footnotes: Analysis of "NEW" Keys – AAE6022E, Karlheinz Geyer has 1030 cross signatures. BAB58229, Marcus Frings has 526 cross signatures. 4743206C, Joachim Breitner has 259 cross signatures. 88C7C1F7, Steve McIntyre has 269 cross signatures.

A.6 OpenPGP Keys 950-1000 Ranked by MSD

Data Collected from: 2008-04-13 Dataset available at http://keyserver.kjsl.com/ka/

Rank	Key ID	Name	MSD
950	4F0BEABB	Klaus J. Mueller	4.2779
951	BF85AB31	Matthias Kalle Dalheimer	4.2779
952	4FC59E44	Alexander Schremmer	4.2785
953	632C74BF	Julian Baeume	4.2785
954	C81115B1	Sebastian Jaenicke	4.2787
955	DC426429	Frank Thomas	4.2788
956	D7FA4512	Joerg Schmitz-Linneweber	4.2790
957	89754606	Rick van Rein (business)	4.2792
958	D1813CED	Achim Dreyer (signing key)	4.2794
959	B83A8797	Sven Lankes	4.2795
960	58536791	Andrew Tridgell	4.2798
961	7CDC44F3	Karl Deutsch	4.2802
962	DF118AF1	Havard Eidnes	4.2802
963	5E642B40	Frank Matthie	4.2803
964	9DED2AA5	Moritz Muehlenhoff	4.2805
965	7DFF8533	peter honeyman	4.2820
966	F681E4CE	Jan Schmidle	4.2820
967	56BA5951	Jan Willem Knopper	4.2824
968	6248BA12	Israel Herraiz	4.2826
969	293697C2	Florian Reitmeir	4.2827
970	7032F238	Jon Dowland	4.2831
971	44DD7643	Georg Nikodym	4.2832
972	6F268727	Nathan Lutchansky	4.2849
973	E3046DF3	Jos De Graeve	4.2849
974	1242A6F2	Simon Hausmann	4.2850
975	B98F8E89	Darryl Ross	4.2850
976	A1EE761C	Pierre Habouzit	4.2854
977	1AAAC2A4	Andreas Mller (Student)	4.2854
978	F2D58DB1	DFN-PCA, CERTIFICATION	4.2857
979	890B15B2	Alberto Garcia Gonzalez	4.2858
980	7A588C62	Mark Knox	4.2862
981	627CCF95	Bas Wijnen	4.2866
982	5D54A300	Jeff Snyder	4.2869

983	E12469C1	Ruediger Weis	4.2875
984	7FFA98B4	Jean-Francois PARIS (certificat)	4.2876
985	DA4A1116	Bernhard E. Reiter	4.2879
986	EC63E6B7	Jos De Graeve	4.2879
987	00292B81	Nathalie Weiler	4.2880
988	F61F73F8	Bastian Venthur	4.2882
989	49E2CF4C	Paul Mackerras	4.2883
990	EB9CDAD5	Eduardo Marcel Macan	4.2884
991	73FAAFF8	Christophe Mutricy	4.2885
992	09D9E662	Jonathan Kleinehellefort	4.2885
993	DDAF6454	Bernhard Walle	4.2890
994	5E35DB91	Stephan Rutten	4.2890
995	13A9EA7C	Ellis Whitehead	4.2892
996	6D742669	Luca Capello	4.2894
997	9DFFAAD4	Ludovic Brenta	4.2895
998	B65C0BE9	Ralf Meyer	4.2897
999	97B3E98E	Matthias Reich	4.2902
1000	57EF3E4F	Michael Clark	4.2904

A.7 OpenPGP Keys 950-1000 Graph

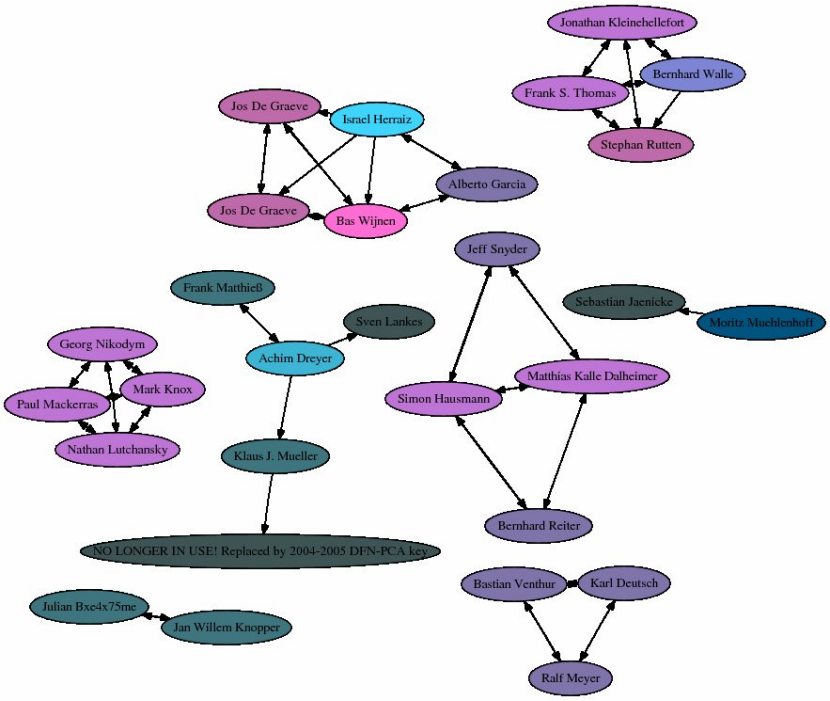

Figure 6: Graph of OpenPGP Keys 950-1000 ranked by MSD.

A.8 OpenPGP Keys 99950-100000 Ranked by MSD

Data Collected from: 2008-04-13 Dataset available at http://keyserver.kjsl.com/ka/

Rank	Key Fingerprint	MSD
99950	412955B6 E1718941	8.2315
99951	C86D3413 991C6A37	8.2316
99952	CF7C9D61 F6B815D5	8.2316
99953	7AC02409 3393825D	8.2318
99954	AAA50CDB AC72D729	8.2318
99955	3F4B27A3 131CE23D	8.2332
99956	FB43C837 44EBF755	8.2332
99957	15B6C4E3 5B980B91	8.2333
99958	5C3D8D98 363A8D16	8.2333
99959	67FDD15A 6990EC20	8.2336
99960	C12A3E91 1060E131	8.2336
99961	F977B8B1 AE242AF4	8.2336
99962	F5F858F3 40ACFE10	8.2341
99963	4462C983 4A43FD91	8.2344
99964	1D347C60 3022C2C4	8.2349
99965	6AC7682D 0CA023CD	8.2349
99966	E44D7F40 260E7CD9	8.2349
99967	77393AA4 F33F7925	8.2350
99968	9177F43A 8902B779	8.2351
99969	5DBDDFBD 88F93E1B	8.2354
99970	71210ED5 98EC95E9	8.2362
99971	A0118797 212AC8DE	8.2362
99972	C7595F95 4A056405	8.2362
99973	FC88DE4D 2B26ED5A	8.2362
99974	0AF213DC 2D6B0399	8.2367
99975	3C805297 97793FAD	8.2367
99976	8E7B0A86 8B6EF60C	8.2367
99977	A8F23271 7BB2AEC0	8.2367
99978	AF9E3FAB B0D1AC83	8.2367
99979	B0AA9F2F BC975EBD	8.2367
99980	D67F988B 4BE0704F	8.2367
99981	DCB89031 A969CE49	8.2367
99982	FA658070 63327EE2	8.2367

99983	FFB18E1E 88C674B6	8.2367
99984	722A7604 45DECC8F	8.2370
99985	4939AD4B 58BDA1AA	8.2373
99986	81ACDE39 3E39B075	8.2373
99987	DCC4A93D 67E7FA7F	8.2373
99988	1D82AF3B B97B40E8	8.2376
99989	3845EA0A 9EF4EF9A	8.2376
99990	59F1952F 10A5F68E	8.2376
99991	8C2D81FD 190206B1	8.2376
99992	4852C81A EFE9177A	8.2380
99993	C07B1894 0A64A546	8.2380
99994	E0D429B9 68883D62	8.2380
99995	F33EC76B 2C5D876B	8.2380
99996	12F0D1A0 86077445	8.2386
99997	587D4B19 85745BB9	8.2386
99998	C97333BC 0E23E41E	8.2388
99999	CE8C4228 1B89134B	8.2388
100000	69164034 45E6F247	8.2390

A.9 OpenPGP Keys 999950-100000 Graph

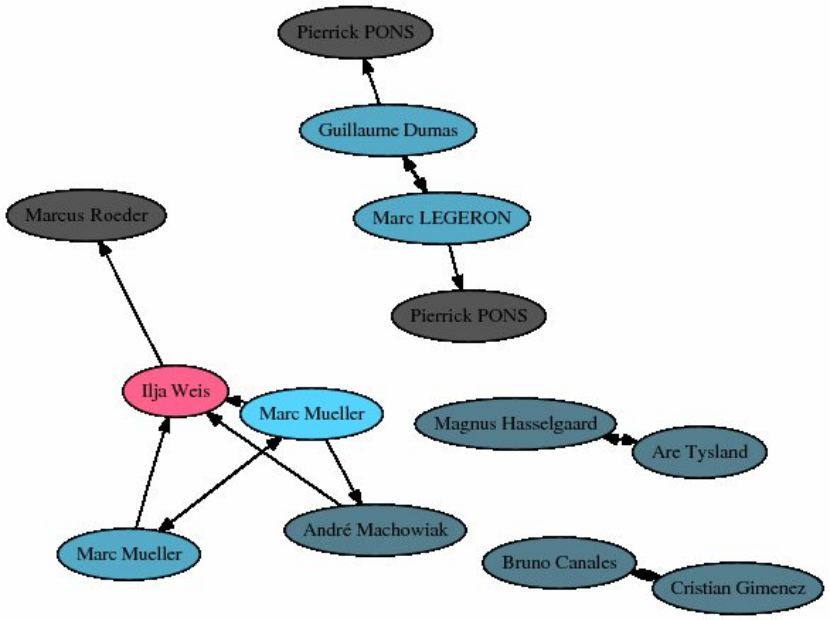

Figure 7: Graph of OpenPGP Keys 999950-100000 ranked by MSD.

A.10 PGP Web of Trust Key Analysis

Key analysis data by Henk P. Penning [36] Data collected - August 2013

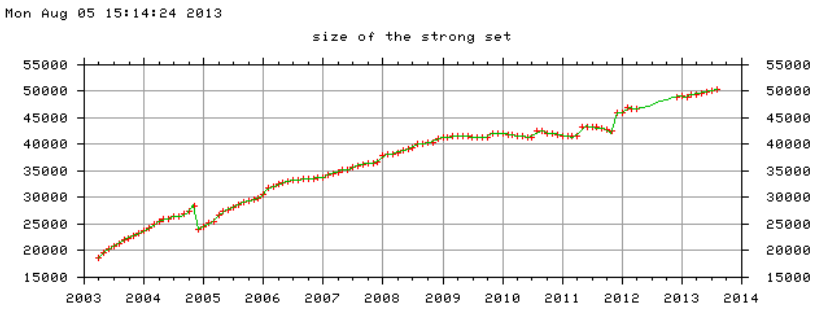

Figure 8: Size of Strong Set.

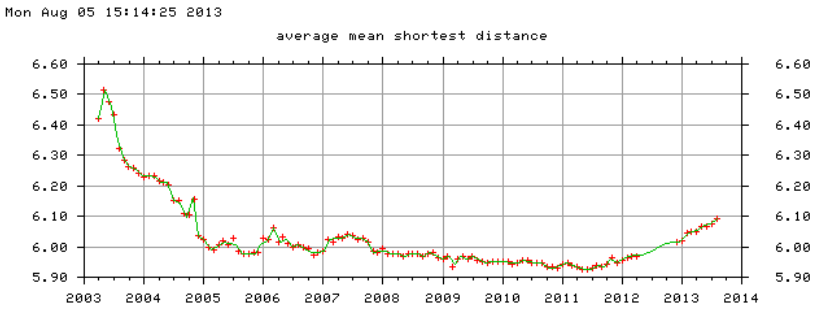

Figure 9: Average Mean Shortest Distance.

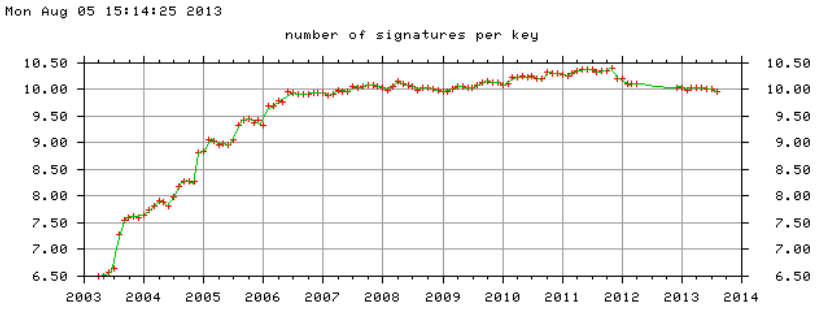

Figure 10: Average Degree (Signatures per Key).

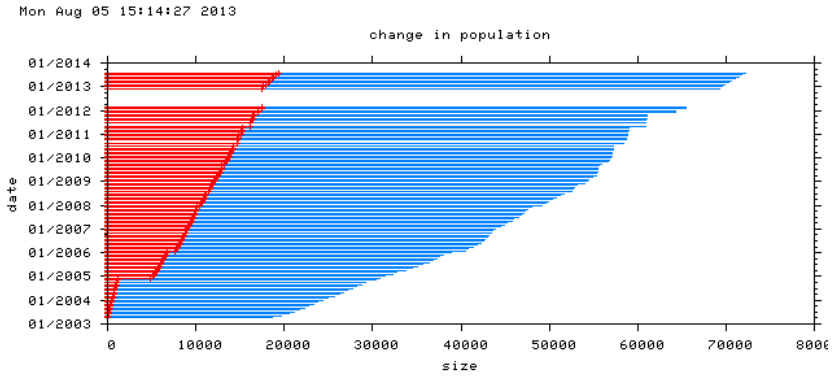

Figure 11: Population change of strong set.

A.11 The Footsie Web of Trust analysis

Key analysis data by Matthew Wilcox [46] Data collected - August 2013

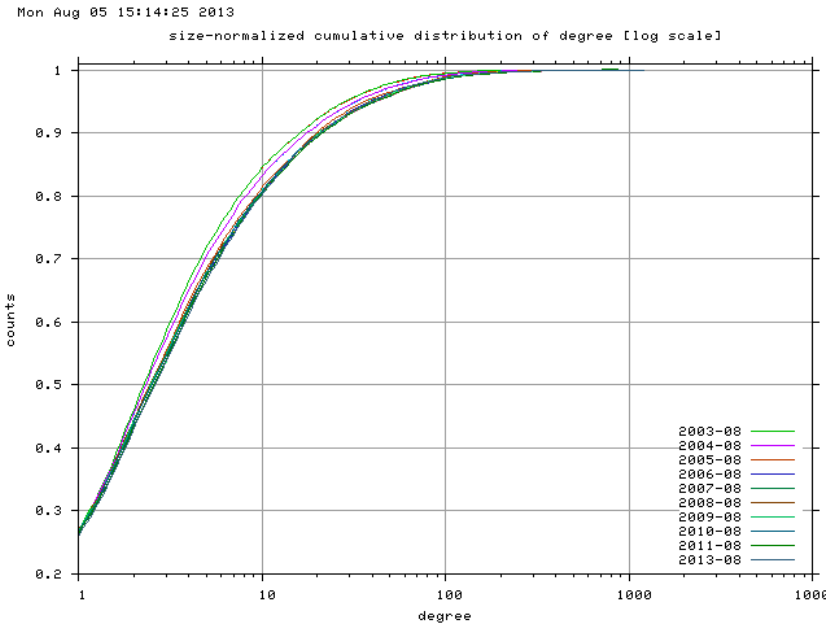

Figure 12: Degree distribution over time.

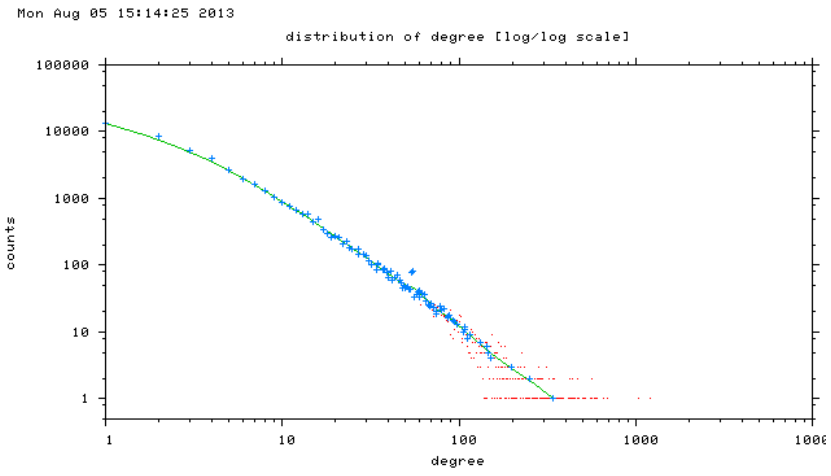

Figure 13: Degree distribution for July 2013.

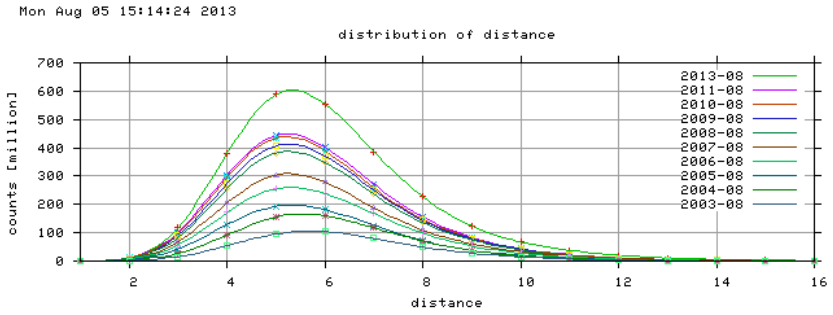

Figure 14: Distance distribution.

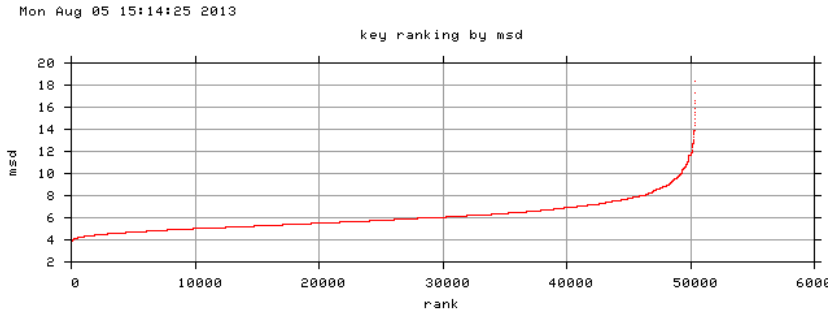

Figure 15: MSD vs rank.

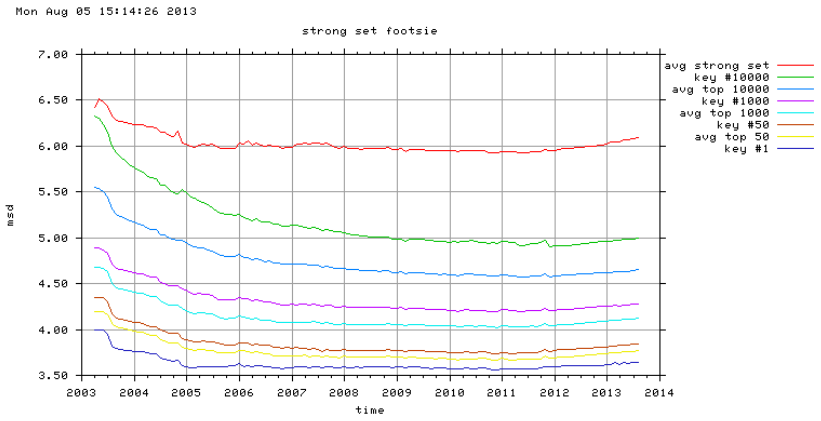

Figure 16: Strong Set "Footsie" Index.

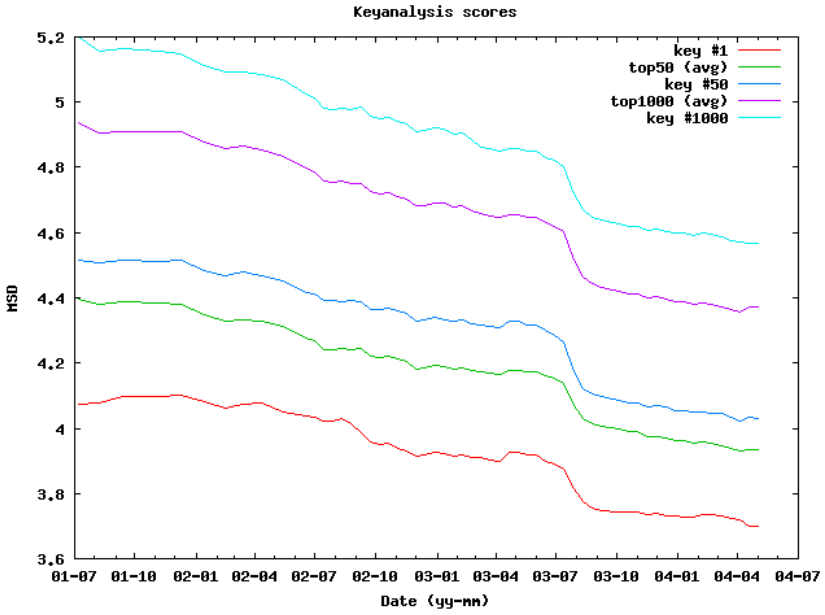

Figure 17: Original Strong Set "Footsie" Index.

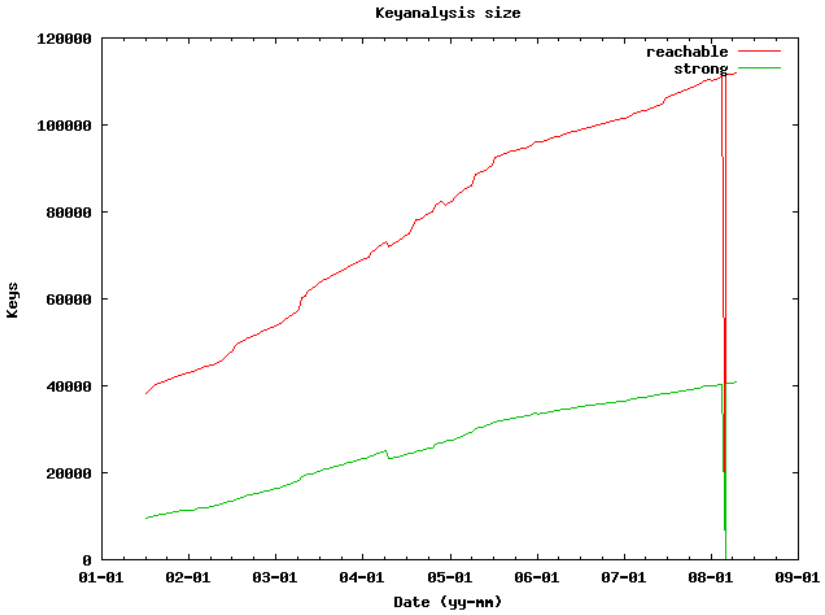

Figure 18: Strong Set vs Reachable Set key size.