An Introduction to PGP/GnuPG

Version 1.0

Publication

Document Revision History

Date	Version	Description
October 2023	1.0	Initial Release

This introduction has been developed to provide an introduction to the world of PGP/GnuPG. The content is based on public information and not solely the view of ETDA. It is a guideline and may be updated from time to time.

Third party sources are quoted as appropriate. ETDA is not responsible for the content of the external sources, including external websites, nor their continued availability, referenced in this introduction.

Where specific vendors or product names are given, those do not mean endorsement from ETDA, but serve as examples only.

This introduction is intended for educational and information purposes only. Neither ETDA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this introduction. All information contained herein is provided on an "As Is" basis with no warranty whatsoever. ETDA does not promise any specific result, effects, or outcome from the use of the information herein.

Written by Martijn van der Heide

[1] Creative Commons License: <https://creativecommons.org/licenses/by-nc-sa/4.0/>

Table of Contents

Preface

Communications, in terms of both the messages themselves and their transport, are increasingly conducted in electronic form and use the Internet for easy, fast, cheap, and global delivery. Such electronic interactions these days include much more than just e-mails and bulletin board postings: indeed, every aspect of our day-to-day lives have moved online, either to complement our regular offline activities (such as talking to our family and friends), or in its entirety (working from home, online gaming or ordering from shops).

All electronic communications require a certain amount of security and trust. While we have, thankfully, mostly left the era behind where infrastructure shortcomings caused significant problems to get data across without requiring several retries (do you remember modems?), there are still several things that need our attention. Who can see your messages in transit? How can you safely authenticate a user coming to your web portal? How can you make sure that the message was not altered before (or even after) arriving? How can you prove that it was you who sent the message?

Cryptography is the science of writing or reading coded messages; it is the basic building block that enables the mechanism of authentication (which establishes the identity of the sender or receiver of information, or both), integrity (which ensures that the data has not been altered in transit for at rest) and confidentiality (which ensures that only authorized entities can view the data). While digital certificates are used to attest the validity of a public key that's part of asymmetric encryption and a common format is the X.509 standard, in the CSIRT community we use a different but similar standard called PGP or GnuPG.

PGP/GnuPG uses a combination of encryption methodologies such as hashing, data compression, symmetric-key cryptography, and public-key cryptography to keep data secure. PGP/GnuPG can be used to encrypt text files, e-mails, data files, directories, and disk partitions, as well as to digitally sign objects such as e-mails and documents.

Its purposes can be defined as to protect:

1) Confidentiality

   Accomplished with encryption.

2) Integrity

   Accomplished with digital signatures.

3) Authentication

   By virtue of a Public and Private key pair, combined with the Web of Trust

4) Non-repudiation

   Accomplished with digital signatures.

Intended audience

This introduction is designed for anyone who wishes to learn more about PGP/GnuPG and its implementation into their environment.

The intended audience is CSIRT teams, but the introduction can also be used as a reference guide for any other type of organization, or individuals.

Acknowledgements

Special thanks go to all individuals in the international information security community who kindly peer reviewed this introduction.

Acronyms

This introduction uses the following acronyms:

Acronym	Term
AES	Advanced Encryption Standard
CA	Certification Authority, or Certificate Authority
CRL	Certificate Revocation List
DES	Data Encryption Standard
DSS	Digital Signature Standard
EKU	Extended Key Usage
GnuPG	GNU Privacy Guard, also abbreviated as GPG
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
MD4/5/6	Message-Digest Algorithm
OCSP	Online Certificate Status Protocol
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
RFC	Request For Comments
RSA	Rivest–Shamir–Adleman
SHA-1	Secure Hash Algorithm

Table 1: List of acronyms

1 Relevant cryptographic concepts

Before discussing PGP/GnuPG, let us introduce several cryptographic concepts first. Note that, as technology moves forward, current accepted encryption algorithm standards will become unsafe – with the ever-increasing computer power, the algorithms can be defeated over time. That is why new algorithms continue to evolve. The speed of this problem will increase exponentially when quantum computing becomes available.

1.1 Symmetric-key encryption

Symmetric-key encryption is a type of encryption where only one key (a secret key) is used to both encrypt and decrypt electronic data. The entities communicating via symmetric encryption must exchange the key so that it can be used in the encryption and decryption process, hence why this key is also called a shared secret.

By using symmetric-key encryption algorithms, data is "scrambled" into what is called ciphertext, so that it cannot be understood by anyone who does not possess the secret key to decrypt it. Once the intended recipient who possesses the key has the message, the algorithm reverses its action so that the message is returned to its original readable form.

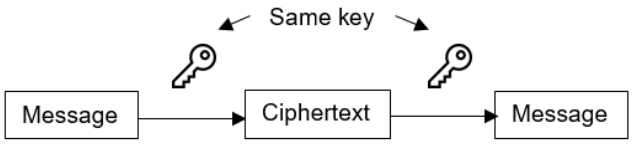

Figure 1: Symmetric-key encryption

Symmetric-key encryption is most often used for data confidentiality because most symmetric-key algorithms have been designed to be implemented in hardware and have been optimized for encrypting large amounts of data at one time. Challenges with symmetric-key encryption include:

1) changing the secret keys frequently to avoid the risk of key compromise;
2) securely generating the secret keys;
3) securely distributing the secret keys.

Symmetric-key encryption algorithms are usually better for bulk encryption. They have a smaller key length, which means less storage space and faster transmission. Due to this, asymmetric-key encryption is often used to exchange a secret session key for symmetric-key encryption such as in SSL/TLS.

Examples of symmetric-key encryption are AES, DES and 3DES, but DES is considered unsafe to use.

1.2 Asymmetric-key encryption

In asymmetric-key encryption, there is a pair of keys for one identity:

- a private key.
- a public key.

The private key must not be shared with others, while the public key can be shared with anyone.

When you encrypt a message with the public key of the recipient, then only the recipient can decrypt with their private key assuring confidentiality.

Since public keys are widely available, anyone can pretend to be the sender. If the sender first encrypts with the recipient's public key and then encrypts with their own private key, the recipient can be assured that the message originated from the sender.

Public key algorithms are rarely used for data confidentiality because of their performance constraints. Instead, public key algorithms are typically used in applications involving authentication using digital signatures and key management.

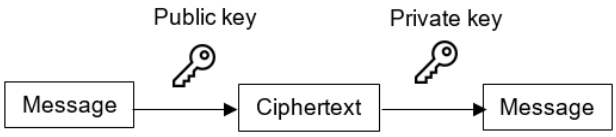

Figure 2: Asymmetric-key encryption

Examples of asymmetric-key encryption are ElGamal, RSA and Elliptic curve.

1.3 Cryptographic hash function

A hash function is a one-way cryptographic function that takes an input message of arbitrary length and outputs a fixed length code. The output is called a hash or message digest.

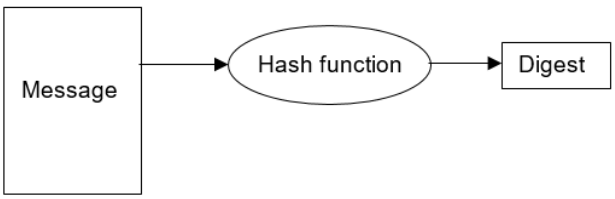

Figure 3: Hash function

A strong cryptographic hash algorithm will make it difficult to:

1) Guess the original input message that yielded the hash (pre-image resistance).
2) Find two or more input messages that yield the same hash (collision resistance).

Examples of cryptographic hash functions are MD4/5/6, SHA-1 and SHA-512, but MD4/5 and SHA-1 are considered unsafe to use.

1.4 Digital signature

A digital signature serves a similar purpose to a real-world signature. Its purpose is twofold: ensure the signer's identity and prove that the message has not been altered in any way after it left the hands of the signer.

The digital signature is generated in 2 steps:

1) Create a digest of the original message.
2) Encrypt that digest with the private key of the sender.

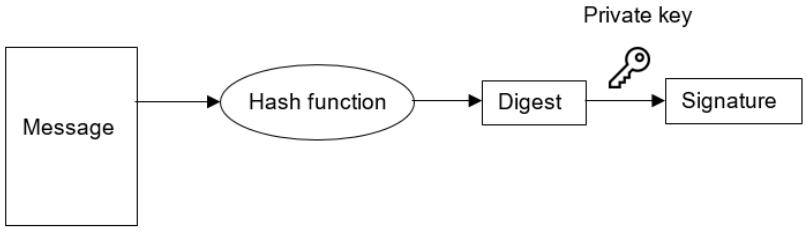

Figure 4: Digitally signing a message

This signature is then delivered together with the original message, allowing verification at the recipient's end.

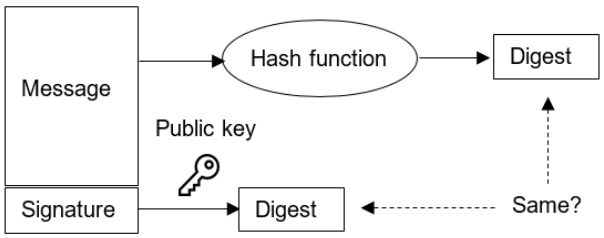

Figure 5: Verifying a digital signature

1.5 Encryption and decryption

Encryption is performed in 3 stages:

1) Create a symmetric Session Key. Only 1 Session Key is used, even if there are multiple recipients.
2) Encrypt the message with the Session Key.
3) Encrypt the Session Key itself with the Pubic Keys of all recipients.

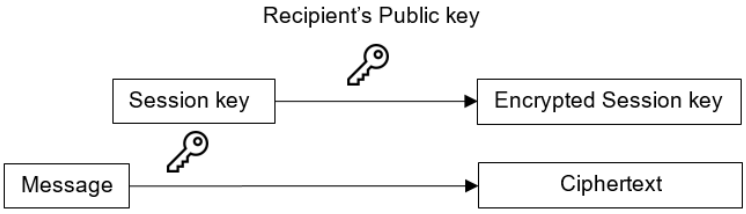

Figure 6: Message encryption

Decryption follows the reverse of the last 2 steps of the encryption process:

1) The recipient uses the Private Key to decrypt the Session Key.
2) The Session Key is then used to decrypt the message itself.

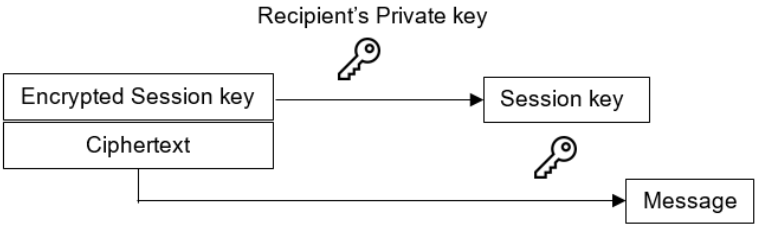

Figure 7: Message decryption

For messages that are both signed and encrypted, the signing part is performed first, then the result is encrypted.

1.6 X.509 vs. PGP/GnuPG

The purpose of both X.509 and PGP/GnuPG is the same, but the implementation is very different.

1.6.1 X.509 digital certificates

A digital certificate, also known as a public key certificate, is used to cryptographically link ownership of a public and private key pair with the entity that owns it. Digital certificates are for sharing public keys to be used for encryption and authentication.

A certificate has been digitally signed by a Certification Authority (CA). The information normally found in a certificate conforms to the X.509 v3 standard as defined in RFC 8250[2].

Certificates conforming to the standard include information about the published identity of the owner of the corresponding private key, the key length, the algorithm used, and associated hashing algorithm, dates

[2] RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile: https://www.rfc-editor.org/rfc/rfc5280

of validity of the certificate and the actions the certificate can be used for, called Extended Key Usage (EKU) extensions.

Comparing functions with PGP/GnuPG, the relevant EKUs are: Secure Email (S/MIME), Code Signing and Encrypted File System (Microsoft Bitlocker). Document Signing is distinctly different, as will be explained in chapter 1.6.2.2.

An individual may hold any number of certificates issued by any number of CAs.

1.6.2 PGP/GnuPG keys

Similarly to an X.509 digital certificate, PGP/GnuPG is used to cryptographically link ownership of a public and private key pair with the entity that owns it.

The standard for PGP/GnuPG is defined in RFC 4880[3].

PGP/GnuPG implementations conforming to the standard include information about the published identity of the owner of the corresponding private key, the key length, the algorithm used, and associated hashing algorithm, and dates of validity of the key. Any key can be used for any of the functions.

An individual may hold any number of keys.

1.6.2.1 ASCII armor

For compatibility reasons, everything that needs to be sent over an e-mail channel must be in 7-bit ASCII text format using the Base64 algorithm. Therefore, all binary (8-bit) data created by PGP/GnuPG, such as encrypted messages, signatures, and even keys themselves, must be converted to 7-bit ASCII text format.

The converted text is encapsulated in a standard header and footer text that describes what the content is intended for. The result is called ASCII armor. For example:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG vx
```

and

```
-----END PGP PUBLIC KEY BLOCK-----
```

The filename extension of ASCII armored files is typically `.asc`. Those can be opened with any text editor.

1.6.2.2 Inline vs. detached signatures

You may also have seen such ASCII armor in electronically signed e-mail messages. This is called an inline signature, as the PGP/GnuPG block is part of the same message.

[3] RFC 4880: OpenPGP Message Format: https://www.rfc-editor.org/rfc/rfc4880

Contrary to Document Signing with X.509 certificates, used to add a visual (digital) signature inside Adobe PDF or Microsoft Office files, PGP/GnuPG inline signatures cannot be used for this purpose, and would ruin the data.

Instead, the generated signature should go into a separate file, which can then be distributed together with the original document or data. This is called a detached signature and typically has a filename extension `.sig`. This purpose is also used to validate software distributions in Linux and can be compared to X.509 Code Signing.

The use and purpose of inline and detached signatures is exactly the same.

2 History of standards

2.1 PGP

PGP, or Pretty Good Privacy, was developed by Phil Zimmermann & Associates, LLC in 1991, and published by the company named PGP Inc. The rights changed hands several times, first acquired by Network Associates in 1997, then by PGP Corporation in 2002, and ultimately by Symantec Corporation in 2010, now part of Broadcom Inc., who continues to develop the PGP brand and product today.

PGP is a commercial product and uses (now expired) patented cryptographic technology and algorithms such as International Data Encryption Algorithm (IDEA) intended as a replacement for DES.

Wide adoption of PGP was not possible due to U.S. cryptographic technology export restrictions at the time, although that is no longer the case.

The first official standard for PGP was defined in RFC 1991[4] in 1996.

2.2 OpenPGP

Phil Zimmermann then began work on an open-source version of PGP encryption that employed encryption algorithms without licensing issues, by establishing an IETF OpenPGP Working Group in 1997. This work resulted in RFC 2440[5] in 1998, which has since been superseded by RFC 4880[6] in 2007 and is the standard today.

OpenPGP is a non-proprietary protocol for encrypting email communication using public key cryptography and defines standard formats for message encryption, signatures, and certificates for exchanging public keys.

A variety of popular solutions have developed following the OpenPGP standards. Examples are given in Appendix A.

2.3 GnuPG

GnuPG, known as GNU Privacy Guard or simply GPG, was developed by Werner Koch and released in 1999 as a free open-source implementation of the OpenPGP encryption standards established in RFC 4880.

It uses the GPL License and is often shipped as one of the packages in Linux distributions, as well as the backend library and engine used in several third-party products. Examples are given in Appendix A.

---

[4] RFC 1991: PGP Message Exchange Formats: https://www.rfc-editor.org/rfc/rfc1991

[5] RFC 2440: OpenPGP Message Format: https://www.rfc-editor.org/rfc/rfc2440

[6] RFC 4880: OpenPGP Message Format: https://www.rfc-editor.org/rfc/rfc4880

2.4 Compatibility

All of the various standards are fully compatible since PGP v5.x (first implementated in PGP 3), the only difference is in the algorithms they support.
Even the earlier patented IDEA has been implemented in GnuPG after the patent expired (since `libgcrypt` v1.6), although the algorithm is now no longer considered safe to use.

You can therefore use any of the products you like (Appendix A) to create and manage your keys, and use the keys in any of the other products, including those on other Operating Systems.

3 Key Life Cycle Management

3.1 Key generation

This is an important process that should be performed in a secure environment, as we are generating the Private key as well.

Your key management can be performed with any PGP/GnuPG product (see Appendix A) and the keys can then be used in any other product as well, including those on other Operating Systems. Note that even an air-gapped Raspberry Pi would suffice for your key management.

If no separate secure computer is available it is advisable to at least disconnect the computer from the network, make sure it is free from malware, and reboot it first to reduce the chance that unnecessary or unidentified processes are running before starting the key generation application.

Select the appropriate parameters:

1) Select the key length to be as long as feasible; at least 2048 bits, but 4096 or more is better.
2) Set an expiration date for the key: "no expiration" is possible, but it can always be extended.
3) Set a strong passphrase to protect the key: it can be changed at any time.
4) Prefer the RSA algorithm: avoid old algorithms such as Diffie-Hellman/Elgamal, or DSS.

When prompted, please choose to create a revocation certificate as well (see chapter 3.4).

3.1.1 Key ID and Fingerprint

The generated Key ID is 8 or 16 bytes long, depending on the age of the key. They are typically shown in hexadecimal with "0x" prepended (the hexadecimal notation mark).
For example: `0xC2FD2726E025B142`.

The Fingerprint is a 128-bit hash value of the key, typically shown in hexadecimal quadruples but without the "0x" marker in front.
For example: `D635 22D5 F6B0 AAD6 B915 84B4 C2FD 2726 E025 B142`.

Notice that the last part of the Fingerprint is always the same as the Key ID.

Key IDs and Fingerprints are not guaranteed to be unique, so also check the Identity (name and/or e-mail address) to verify the key (see chapter 4.2.1).

3.1.2 Identities

A key can have multiple identities associated, for example if you use multiple e-mail addresses and wish to use the same PGP/GnuPG key for each, rather than creating different keys for them. One identity (the first one shown) is your default identity.

Identities can easily be added or updated at any time, as well as the default identity.

However, only (limited) additions and modifications can be made; data in your key cannot be removed, only revoked. Revocation may be for individual parts, such as an identity no longer being used, or for the entire key (see chapter 3.4).

3.2 Storage and backup

The following security and continuity precautions are advised:

1) PGP/GnuPG uses 2 key ring files for storage: one for all Public keys (`pubring`) and one for all Private keys (`secring`). Please make sure that the key rings are stored securely, for example by keeping all PGP/GnuPG data directories on a USB stick that is kept in a secured and controlled location when not in use. Using a dedicated, air-gapped, system for your key management would be the safest solution.
2) Export your Private key into a file on removable media and keep it in a secured and controlled location such as a locked cabinet or a safe.
3) The passphrase should only be written down if it can be kept in a (different) secured and controlled location.
4) Move the revocation certificate that was created during the key generation process (or at any other time after) to removable media and make sure to remove it from any other location, as this allows anyone to revoke the key without any form of authentication.

3.3 Publication

Once the key is fully prepared, it can be shared with the world. You can export the Public key (never the Private key!) using your PGP/GnuPG application.

It is recommended that keys are uploaded to the public key servers (see chapter 4.2.2) to become part of the Web of Trust and easy to find for anyone who wishes to communicate securely, but this is optional.

If a key was created for your CSIRT team, it may be made available for download from the team website. The fingerprint can be shown as well for completeness.

Depending on the e-mail application you use, it may be able to automatically attach your key to every message you send.

3.4 Revocation

Keys may need to be revoked before the expiration date for several reasons, such as:

1) It has been compromised.
2) It is vulnerable due to using a key algorithm or key length that is no longer considered safe.
3) The passphrase of the Private key has been lost.
4) The e-mail address is no longer used.
5) It is no longer valid as that person no longer works for the organization.

Revocation of an entire key is done by importing a revocation certificate that was created beforehand, typically during the key generation phase. If a revocation certificate was not created yet but you still have full access to the key, one can still be created at that time.

If the reason for revocation is not having access to the key anymore, then nothing can be done, as keys cannot be removed from the key servers, only revoked. A possible mitigation for this situation is provided in chapter 6.

If your key was previously uploaded to the key servers, please remember to re-upload the revoked key as well, or no one will know about it.

As PGP/GnuPG does not have a centralized authority, revocations are more or less invisible. Applications that use keys will only see whether a key has expired, as that information is in the key, but they generally do not check with the key servers if there is any update to the key. As such, everyone who has a copy of your Public key will be unaware of its revocation unless they are manually informed (or if they happen to check).

4 Trust

It is important to understand what "trust" means in PKI. In the real world, the word is typically used to indicate trust in a person or organization; they are known to be trustworthy. However, in the PKI world, trust only means that we can depend on that a presented digital certificate or PGP/GnuPG key has been thoroughly verified to really belong to the identity that is shown in it; it says nothing about trustworthiness of the person or organization themselves, as we may not actually know them at all.

4.1 X.509: Chain of Trust

X.509 digital certificates are issued by central organizations, called Certification Authorities (CA). A Root certificate is the trust anchors of a CA, and can be stored in the certificate trust store of Operating Systems and applications such as web browsers and e-mail clients. Public Root CAs that have been thoroughly vetted are included in the certificate trust stores already, so that all certificates they issue are also automatically trusted by your Operating System and applications.

Trust in a digital Subscriber, or End Entity, certificate is verified by checking the certification path through all Intermediate certificates, back to a trusted Root certificate, installed in the trust store. This process is made easy by the fact that each certificate registered who the issuer was (its parent). PKI-aware software will also verify each certificate in the chain with the CA's Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) Responder, to make sure it has not been revoked.

The steps to be followed to verify the Chain of Trust are the same for each level (Subscriber – Intermediate(s) – Root), as also shown in figure 8:

1) Verify the certificate's integrity, by verifying the signature with the public key of the issuer.
2) Verify the validity of the certificate, with the "notBefore" and "notAfter" fields.
3) Check the revocation status of the certificate.
4) Verify the issuer's name, which must match the subject name of the issuer's own certificate.
5) Verify the constraints specified in any certificate's extensions, such as domain name, policy, the key usage (purpose) of the certificate and the path length (to make sure leaf certificates are not illegally used as Intermediate certificates).

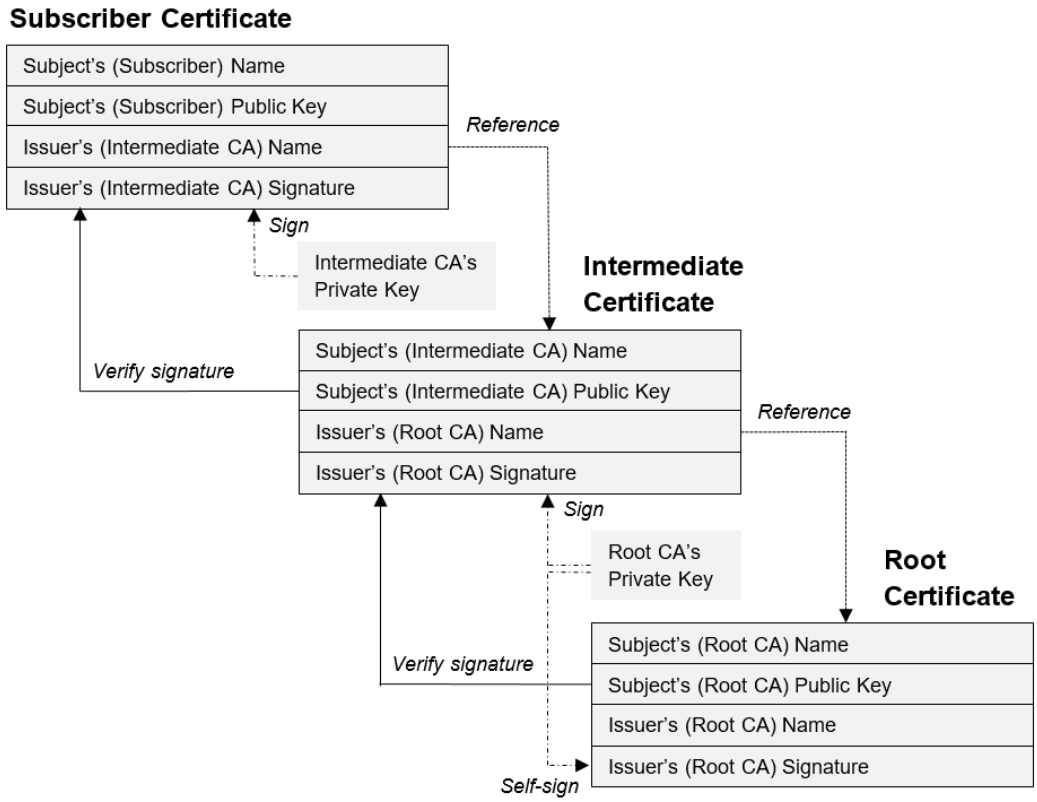

Figure 8: Chain of Trust

4.2 PGP/GnuPG: Web of Trust

Contrary to X.509 digital certificates, PGP/GnuPG does not have any central authority. Anyone can install the software, create a key, and start to use it. As the lack of a central authority means that explicit trust through a Chain of Trust cannot be established for the key, a different approach has been created for the protocol: independent key signing of Public keys to create an implicit Web of Trust, as shown in figure 9.

Bob trusts Alice and signed her key. Alice herself trusts Carol and signed her key. This creates an implicit trust path from Bob to Carol, through the signing of keys.

A Public key can be signed (cross-signing is even better) by however many people you want; the more the merrier, as this expands the Web of Trust.

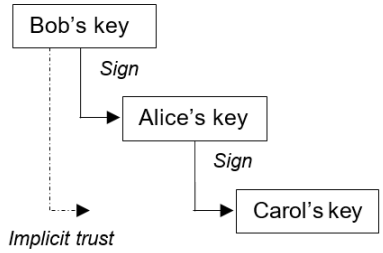

Figure 9: Web of Trust

If you receive a key from an unknown person and want to inspect the trustworthiness of that key (has it been established that the key really belongs to that person?), the implicit trust path can be followed back until you find a signer you know.

A signature is an addition to an existing key, and all signatures are independently identifiable. If you have a key in your keyring with X signatures, you can safely import the same key with Y signatures; that operation will simply add the new signatures to the key in your keyring.

Note that a key without any signatures is not necessarily a bad key; it simply does not participate in the Web of Trust (yet). Such keys could be encountered during incident response communication with a third party, or other ad hoc activities when a temporary, non-advertised key is needed for secure communication.

4.2.1 Key signing

Key signing is a process in 5 steps:

1) Verify the identity of the key holder through visual inspection with an official photo ID document such as a passport or driver's license.
2) Import the Public key to be signed into the keyring.
3) Verify that the identity (name, e-mail address) in the key matches the name of the key holder, and that the key we are about to sign is the correct one, by inspection of the key fingerprint. Note that the fingerprint is not guaranteed to be unique, so the identity should be matched as well.
4) Digitally sign that key using the PGP/GnuPG application.
5) Export and return the now-signed key to the key holder, who imports the key back into their own key ring to add the new signature.

If, at some later point in time, there are doubts about a key you signed, for example because it was compromised or lost, and the key holder cannot revoke the key for any reason (see chapter 3.3), you can revoke your own signature from the key to show you personally no longer trust it. This can be accomplished by steps 3-5 above, except you select "revoke signature" in step 4.

4.2.2 Key servers

There is a Public Key Infrastructure of key servers available throughout the Internet, all synchronized with each other. Several of them run on SKS (Synchronizing Key Server).[7] This is where Public keys can be uploaded and shared, allowing anyone who wants to securely communicate with you to find your key.

Examples of such key servers are: https://pgp.mit.edu/, http://pgp.circl.lu/, https://pgp.surfnet.nl/, http://pool.sks-keyservers.net/, https://keys.openpgp.org/ and https://keyserver.pgp.com/. Choose one with good performance from your location.

[7] SKS Keyserver Network: https://spider.pgpkeys.eu/

Using the web interface, you can simply copy & paste keys. Before uploading any key, please double-check that this is the Public key. Never expose your Private key anywhere!

Updated keys, such as additional identities, changed expiry dates, revocations, or after acquiring new signatures, can simply be re-uploaded.

4.2.3 Key signing parties

Key signing parties are social events, typically held during security conferences or seminars. Their purpose is a mass key signing opportunity to increase the Web of Trust.

To prepare, all participants send their key meta-data: name, e-mail address, and fingerprint to the organizer beforehand and bring an official photo ID document such as a passport or driver's license to the party. The organizer puts this information into a single sheet, for example a table with a row per participant and all meta-data in the columns, and prints one sheet per participant.

During the event, each of the participants in turn is subjected to verification: all other participants visually inspect the photo ID document (step 1 in the key signing process of chapter 4.2.1) and match that to the name in the key. The participant then reads the fingerprint of their key aloud, while the other participants verify that information on the sheet prepared by the organizer (step 3).

After all participants had their turn, everyone has a sheet of keys they personally verified ownership of. They can each proceed, usually on a later date, to import and sign the verified keys (step 2 and 4) and send them back signed to their key owner (step 5).

Remember the definition about trust given at the start of this chapter. You do not need to personally know all participants in the party, as the only goal is to establish that the keys that are presented really belong to their key holders.

It is also not uncommon to be prepared for the unexpected when going to a conference or seminar, and bringing several copies of the printed key meta-data along, in case there is an opportunity for individual key signing if no key signing party has been announced.

5 PGP/GnuPG in the CSIRT environment

If your team has not used cryptographic tools before, it is advisable, out of an abundance of caution, to verify with your legal counsel or local government first if there are any laws in effect that regulate the use of cryptographic tools in your country, such as approved algorithms and limits on key lengths.

5.1 A need for secure communications

Every CSIRT works with confidential information, whether it is related to their own organization, their constituents, third parties or even the world at large. There may be severe or even devastating consequences if such information would leak, such as incidents your team is handling in general, data breaches that will cause havoc if the media finds out, or information about vulnerabilities that malicious actors would be very interested in to breach your network.

CSIRT communication has many use cases, some common examples are:

- Within your team if the team is spread over multiple locations.
- Between your team and a constituent when dealing with an incident or vulnerability.
- Between your team and your National CSIRT for situational awareness.
- Between CSIRT teams or communities when sharing threat intelligence or lessons learned from an incident your team handled.
- With an external malware researcher who wants to alert your team, or work with your team to further research a piece of malware observed in your own or your constituency's infrastructure.
- With anyone on the Internet who wants to file an incident or vulnerability report.

All of the cases above probably involve confidential information. Therefore CSIRTs, as well as malware and vulnerability researchers, are hesitant to share anything unless it can be done securely.

E-mail is the most used electronic communications medium for CSIRTs, so e-mail security is a critical factor. There are several ways to secure data in transit (in addition to measures in the underlying infrastructure itself) and for e-mail those are PGP/GnuPG and X.509 S/MIME.

The CSIRT community has been using PGP/GnuPG for the past decades. It certainly has its flaws, but it works well enough, it is free and easy to set up, and it is trivial to create an ad hoc PGP/GnuPG key when a temporary one is urgently needed.

X.509 S/MIME certificates are more robust and offer the very strong Chain of Trust, but this has the drawback that obtaining an S/MIME certificate can take a lot of time while the Certification Authority verifies the certificate request. It may also be expensive to maintain a set of S/MIME certificates for your team, and you will need to convince your communication counterpart (in the use cases above) to also obtain an S/MIME

certificate. You can create your own Certification Authority[8], but this, too, can become costly due to infrastructure requirements.

5.2 Data classification

Data classification is done by the owner of the data. Apart from determining the sensitivity, classification also indicates a clear expectation of what the recipient of that data is allowed to do with it, such as sharing further or discussing it openly.

An official standard for data classification is the Traffic Light Protocol[9], which is maintained by FIRST, and used throughout the security community. It defines 4 levels: TLP:RED, TLP:AMBER, TLP:GREEN, and TLP:CLEAR. Please observe the protocol strictly when you receive any information classified with it, and when in doubt, verify with the owner of the data.

Typically, TLP:AMBER information or discussions thereof must be encrypted in transit, and TLP:RED information may not be shared over e-mail at all.

Other classification standards may be in force in your organization and must be observed as well.

5.3 Multiple keys

CSIRT teams are advised to create separate keys for the following functions:

- Team key

  One key that is used by the entire team when communicating from the team e-mail address. This team key should be widely published, such as sent to the key servers and published on the team website. See also chapter 6.2.

- Personal keys

  All team members with a responsibility for individual external communication (i.e., not using the team e-mail address) have their own personal key. At a minimum this will be the Team Representative(s), but should also include everyone who joins mailing-lists, Working Groups or Special Interest Groups related to the CSIRT work. See also chapter 6.4.

- Application keys

  Several applications hosted on the Internet/in the Cloud can use PGP/GnuPG keys for secure data exchange. Each such application should use a different key. See also chapter 5.4.

---

[8] Establishing a Certification Authority (CA): https://www.first.org/resources/guides/Establishing-a-Certification-Authority-CA.pdf

[9] Traffic Light Protocol: https://www.first.org/tlp/

All keys need to be maintained. If your team is a member of collaborative organizations, do not forget to keep your key information updated there as well. See also chapter 5.5.

5.4 Integration in the environment

Integration of your keys in your CSIRT environment will make everything easier.

Several incident ticketing systems, such as RTIR[10] and OTRS[11], can be configured to support PGP/GnuPG keys to automatically decrypt incoming e-mails and/or sign all outgoing e-mails. If supported, the Team key could be configured here.

Some applications such as MISP[12] and Cloud hosting support PGP/GnuPG for secure data exchange.

5.5 Security community Web of Trust

The cybersecurity workforce continues to grow rapidly, which is great, but one of the consequences is that it becomes impossible to know everyone, even in the same community.

Trust is an important factor in security. As discussed in chapter 4, PGP/GnuPG utilizes a so-called Web of Trust, where trust (in keys, not necessarily people) is established by signing each other's keys. We can help each other by participating in the Web of Trust, so you are specifically encouraged to do so.

If your team is or plans to become a member of collaborative organizations such as FIRST[13], APCERT[14], TF-CSIRT[15] or Trusted Introducer[16], you are invited to

- Share your PGP/GnuPG keys in your team details section.
- Sign keys of members from other teams when you meet with them.
- Go to conferences and seminars and participate in Key Signing Parties (chapter 4.2.3).

[10] RTIR: https://bestpractical.com/rtir
[11] OTRS: https://otrs.com/
[12] MISP: https://www.misp-project.org/
[13] FIRST: https://www.first.org/
[14] APCERT: https://www.apcert.org/
[15] TF-CSIRT: https://tf-csirt.org/
[16] Trusted Introducer: https://www.trusted-introducer.org/

6 Policy considerations

6.1 Master key

As a form of a local Web of Trust, a Master key could be implemented as a trust anchor.

This Master key will only ever be used to sign all other keys related to the team. It should therefore be uploaded to the key servers to allow the world to verify the signatures, but not published anywhere else, to avoid someone accidentally using it to send an encrypted message to the team (they should use the Team key for that).

As performance is not an issue, the largest possible key length should be selected when creating the Master key. A long expiration time can be selected.

During creation on a clean system, make sure to create a revocation certificates as well, and to follow all steps related to storage and backup as explained in chapter 3.2 to keep the key and related data safe.

6.2 Team key

As team members may leave the team, the Team key should not be valid for very long to reduce the chance of incidents.

A common Team key is used for 1 year, created with a validity of 13 months allowing some time overlap for a smooth transition period.

Revocation certificates should always be created and stored in a secured and controlled location such as a locked cabinet or a safe.

If a Master key has been created, it should be used to sign the Team key. Additionally, the policy could state that all team members (who have a personal key) sign the Team key as well.

The team key should be widely published, such as sent to the key servers and published on the team website.

6.3 Personal keys

As personal keys are only used by a single individual, the expiration can be longer than the Team key.

The key's identity should contain the official e-mail address in your domain. The policy should state if it is permissible to add additional identities, with e-mail addresses outside your domain, to official personal keys, or that separate keys must be used for each identity.

If a Master key has been created, it should be used to sign all personal keys. Additionally, the policy could state that all team members sign each other's keys as well.

Revocation certificates should always be created. How and where they are stored is a policy choice. For example, this can be up to the person whose key it is, or they can all be stored together in a secured and controlled location such as a locked cabinet or a safe.

Personal keys are ideally suitable for key signing and participating in the Web of Trust, so you are encouraged to consider establishing this in policy (see chapter 5.5).

6.4 Designated Revoker

A special feature of PGP/GnuPG is to assign a specific key as Designated Revoker for another key. This Designated Revoker is allowed to be used to create a revocation certificate for that key, in addition to the key itself.

This is a good failsafe facility, allowing a possibility to enforce revocation of a key even if the key holder no longer has access to their key, or is unwilling to revoke it.

The revocation certificate generation using a Designated Revoker key uses a different command from a normal revocation certificate generation.

Please refer to the documentation. In GnuPG on the command line `--edit-key` / `addrevoker` adds the Designated Revoker and `--desig-revoke` will generate the revocation certificate.

Although Designated Revokers are part of the PGP/GnuPG standard, not all products have built-in support for it. In those cases, you could export the key pair, add this option using a different product (such as GnuPG itself), and then re-import the updated key pair into your key ring.

If a Master key has been set up, this could be set as Designated Revoker for all keys related to the team, such as the Team key, all personal keys, and possibly application keys.

Be extra careful with the key that has been assigned to be the Designated Revoker, as a malicious actor can revoke all your keys with it.

6.5 Signing and encrypting

The policy should clearly state when messages are to be signed, when encrypted, when both signed and encrypted. Commonly seen options are:

- Always sign all outgoing e-mail.
- Always encrypt outgoing messages that are a reply to an encrypted message (whether considered sensitive or not).
- Always encrypt outgoing messages if they contain sensitive information.

This is of course only needed for such messages that are allowed to go over e-mail to begin with, which depends on the classification of the data (see chapter 5.2).

7 Threats and pitfalls

Key updates

As PGP/GnuPG does not have a centralized authority, key updates such as revocations, identity changes, or extended expiration dates, are more or less invisible. Applications that use keys will only see whether a key has expired, as that information is in the key, but they generally do not check with the key servers if there is any update to the key. As such, everyone who has a copy of your Public key will be unaware of updates unless they are manually informed (or if they happen to check).

Key selection

When contacting a person or organization for the first time, you need to find out which key to use. This will be easy if their current key is listed on their website. If there is no such hint, this may be a challenge if there are multiple keys listed for that person or organization on the key servers. In that case, it is usually safest to ask first, by sending a short inconspicuous e-mail, with your own key attached.

Fake keys

Anyone can create PGP/GnuPG keys, and anyone can upload keys to the key servers.

Be aware that this may include any adversary who wants to impersonate you with a fake key. It is good practice to regularly check the key servers if there are fake keys on them with your identity.

Vulnerable keys

Especially keys with a very long (or no) expiration date may become vulnerable due to technological progress, when certain algorithms and key lengths are no longer safe to use. Keep track of technological progress and replace vulnerable keys as soon as possible (revoke the key and create a new key).

Encryption mistakes

Encryption is typically used to protect sensitive information in storage, and especially in transit. Special care must be taken (test it first) to avoid leaking information when encryption is used incorrectly. Specifically:

- Unless the Subject header is encrypted as well, information may be gained from it.
- Unless the entire message plus attachments is encrypted as 1 block of data (PGP/MIME), be careful that attachment filenames themselves may be sensitive as well.
- A common mistake is to encrypt the e-mail text message only, but not the attachments.

Company mail servers

Company mail servers may be configured to alter e-mails when sending and/or receiving, thus breaking digital signatures along the way. This should be tested first. Using inline signatures (see chapter 1.6.2.2) may mitigate this.

- Incoming messages: text may be added before it enters your mailbox, saying it has been “scanned for malware”, or to warn that the message was “received from an external source, be careful with any web links in it”.
- Outgoing messages: a standard disclaimer may be added at the bottom of all e-mails.

Appendix A: Implementations

The following is a selection of software products. More can be found at https://www.openpgp.org/software/

Product	Windows	Linux	Mac OS	iOS	Android	Notes
Key management						
GnuPG	✓	✓				
gpg4usb	✓	✓				
Gpg4win	✓					Contains Kleopatra, and plugins
GPGTools			✓			Contains a plugin for Apple Mail
Kleopatra	✓	✓				Contains a plugin for KMail
OpenKeychain					✓	Contains a plugin for K-9 Mail
PGPro				✓		
Browser and e-mail client plugins						
Autocrypt	✓					For Thunderbird
Enigmail	✓	✓	✓			For SeaMonkey, Epyrus, Interlink Mail & News and Postbox. Previously also for Thunderbird (see the note below)
FlowCrypt				✓	✓	Browser extension for Firefox, Chrome and Brave (for Gmail)
gpg4o	✓					For Outlook
Gpg4win	✓					For Outlook (GpgOL) and Windows Explorer (GpgEX)
GPGTools			✓			For Apple Mail
Kleopatra	✓	✓				For KMail
Mailvelope	✓					Browser extension for Firefox, Chrome, and Edge (for various webmail providers)
OpenKeychain					✓	For K-9 Mail
p≡p	✓			✓	✓	For Outlook and Thunderbird
Seahorse		✓				For Evolution
Stand-alone e-mail clients						
Claws Mail	✓	✓				
eM Client	✓		✓			
FairEmail					✓	
Mutt		✓	✓			
Thunderbird	✓	✓	✓			See the note below

Webmail solutions with built-in support	
Hushmail	Limited OpenPGP support
Mailfence	
Proton Mail	

Table 2: PGP/GnuPG implementations

Thunderbird supports accounts hosted at all webmail providers, as well as Office 365. Versions prior to 78 (released July 2020) relied on a local GnuPG installation and the Enigmail plugin. Since version 78, Thunderbird has had native OpenPGP support built-in, as well as native support for X.509 S/MIME certificates. Keys stored in the GnuPG key rings have to be manually transferred into the new environment.

Appendix B: Further information

GnuPG references

- GnuPG documentation
  https://www.gnupg.org/documentation/manuals/gnupg.pdf
- GnuPG FAQ
  https://gnupg.org/faq/gnupg-faq.html

History of PGP/GnuPG

- Phil Zimmermann
  http://www.philzimmermann.com/
- A Short History of the GNU Privacy Guard
  https://lists.gnupg.org/pipermail/gnupg-announce/2007q4/000268.html
- What's the matter with PGP?
  https://blog.cryptographyengineering.com/2014/08/13/whats-matter-with-pgp/

Attacks on Key Servers

- SKS Keyserver Network Under Attack (June 2019)
  https://gist.github.com/rjhansen/67ab921ffb4084c865b3618d6955275f
- SKS Keyserver Network Attack: Consequences (June 2019)
  https://gist.github.com/rjhansen/f716c3ff4a7068b50f2d8896e54e4b7e
- OpenPGP Certificate Flooding (June 2019)
  https://dkg.fifthhorseman.net/blog/openpgp-certificate-flooding.html
- OpenPGP Certificate Flooding (July 2019)
  https://lwn.net/Articles/792366/
- OpenPGP Flooding Attack Mitigations (July 2019)
  https://anarc.at/blog/2019-07-30-pgp-flooding-attacks/