A Review of Completely Public, PGP Public Key Servers

Pruthvirajsinh R. Chauhan
M.E. Scholar
Computer Engineering (WiMC)
Gujarat Technological University, Ahmedabad

Samuel Johnson
Scientist Engineer
Computer Centre,
Physical Research Laboratory, Ahmedabad

Gardas Naresh Kumar
Co-ordinator
Centre for Development of Advanced Computing, Pune

Abstract

Pretty Good Privacy (PGP) is current de-facto security standard of internet community to ensure privacy and authenticity of their electronic communication by use of encryption and digital signature. PGP uses Public Key Cryptography hence it needs two keys namely Public and Private Key. Public key needs to be stored in such a way that it becomes easily available to anyone in the world who wishes to share confidential information with the owner of the key. Public Key Servers (PKS) are setup specifically for this sole purpose, which is to store and distribute public keys to the world on behalf of the owner of the key. There are Public PKS on the internet which provide above mentioned services to any individual. By "Public" we mean that the PKS is Open Source, is completely decentralized and is not under control of any single organization. This paper describes working of publically available PKSs with practical and insightful pros and cons of each.

1. Introduction

Ever since the inception of key based cryptography, problem of key distribution has been at the center of attention. Asymmetric key cryptography nearly solves the problem by using public and private keys. Private Key of a user is always private and is not needed to be shared with others for encryption to work. The world needs to know only public key of the user. The whole process works as shown in figure 1.

But here the problem of providing easy access to public keys to the whole world is still present. As a solution Public Key Server (PKS) are deployed where users upload their public keys.

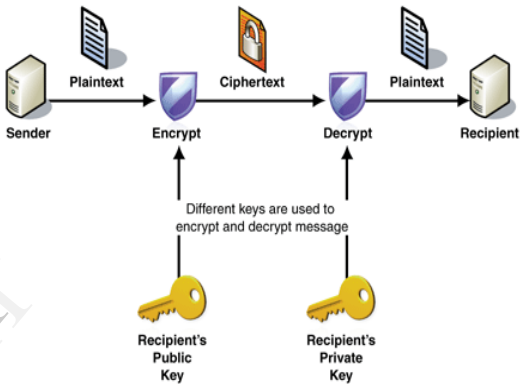

Figure 1. Public Key Cryptography

PKS stores Public keys and make them easily available so that anyone can encrypt the message using receiver's public key after fetching it from PKS. The whole process is shown in the figure 2.

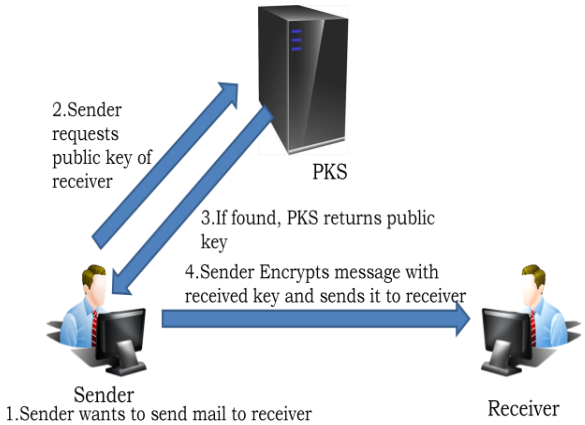

Figure 2. Working of PKS

Hence a public key server (PKS) is simply a computer which provides services to store and retrieve public keys to the users over network. Any user with a valid public key can upload his key to the server. From there onwards anybody can get its public key from the Key Server.

1.1. Pretty Good Privacy

Pretty Good Privacy is a brain child of Philip Zimmermann. He released it in 1991 to counter privacy intrusion attempts made by government organizations. PGP makes use of symmetric and asymmetric key cryptography to provide confidentiality and integrity of electronic communication media like e-mail and other data files.

Due to patent issues with certain encryption algorithm a new standard was created which was open and available to use for all with no patent issues called OpenPGP.

OpenPGP

OpenPGP is based on PGP and provides following types of services to the users.

- digital signatures of documents
- encryption and decryption of data ranging from text files to whole hard disk partitions
- compression of data

Since OpenPGP is based on PGP, OpenPGP and PGP keys are compatible with each other.

GPG

Currently open source implementation of OpenPGP called GNU PGP (GPG) is widely used by community.

Throughout available literature PGP and GPG are used interchangeably, this paper also uses these terms to signify the same thing.

2. GPG modes of Operation

Following operations can be carried out by PGP.

2.1. Confidentiality via Encryption

OpenPGP can use both symmetric and asymmetric encryption to keep the data secret.

2.1.1. Public Key Encryption

As the name suggests Asymmetric algorithm is used in process, but it is not used directly to encrypt the data per se. Data is still encrypted using Symmetric Encryption algorithm but the symmetric key is encrypted itself by Private key of the sender. A different random symmetric key called "Session Key" is generated per message and is sent along message in encrypted form. Session key is discarded once message is delivered. Whole sequence is shown in Figure 3.

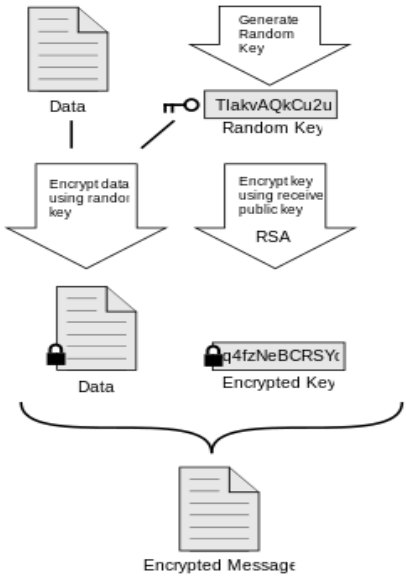

Figure 3. GPG Public Key Encryption

On the receiver's side first the session key is decrypted and then the message is decrypted using session key as shown in figure 4.

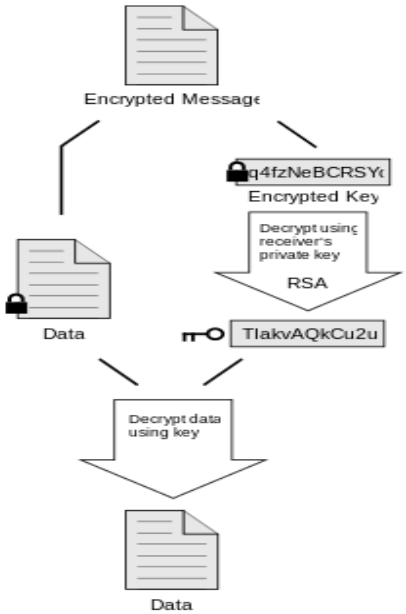

Figure 4. GPG Public Key Decryption

2.1.2. Symmetric Key Encryption

In symmetric key operation also the key of the user is not used directly to encrypt the data. In most implementations the secret key of the user is used as a seed to generate a new session key per message.

After a session key is generated the message is encrypted using it. After that the session key is encrypted using the shared secret key and it is sent along with encrypted message.

On receiver side the receiver first decrypts the session key with shared secret key and then uses decrypted session key to decrypt the message itself.

2.2. Authentication via Digital signature

The PGP Digital Signature is generated as following. First a hash value or message digest of the data is generated. The hash value is then encrypted using private key of the sender.
The sequence is as shown in figure 5:

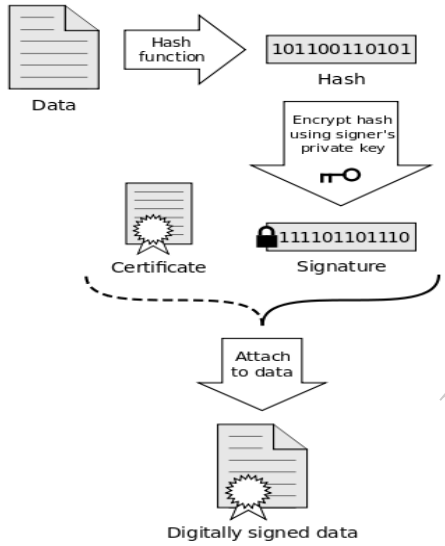

Figure 5. GPG Digital signature

The flow on receiver's side is shown in figure 6.

On receiver side the receiver first decrypts the hash value using sender's public key and then generates the hash of the received message at its side. If both decrypted hash and locally generated hash match each other than message is intact and it is assured that message is not tempered with.

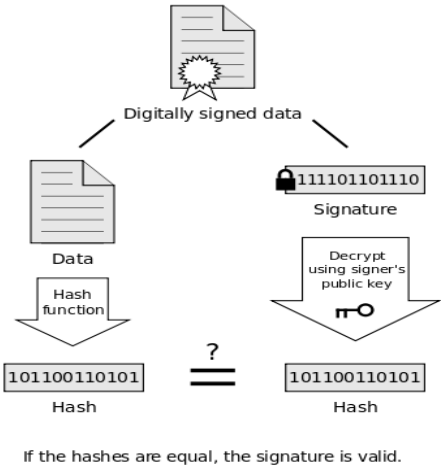

Figure 6. GPG Digital signature Verification

GPG doesn't restrict us to use only Encryption or only Digital signature. We can use both at the same time. First a digital signature is generated and then the whole message with its digital signature is encrypted using either symmetric or asymmetric encryption.

2.3 Supported Algorithms

GnuPG currently supports following algorithms:

Block ciphers (symmetric encryption algorithms):

- IDEA
- CAST5
- Camellia
- Triple DES
- AES
- Blowfish
- Twofish.

Asymmetric-key ciphers:

- ElGamal
- RSA

Cryptographic hashes:

- RIPEMD-160, MD5
- SHA-1
- SHA-2
- Tiger

Digital signatures:

- DSA
- RSA

3. HTTP Keyserver Protocol

In March 2003 D.Shaw proposed an IETF draft to formalize a protocol for transfer of GPG Public key using established Internet protocol HTTP called “HTTP Key Server Protocol”-HKP. Since then all PKS use this protocol to transfer GPG Public keys over internet.

3.1 Requesting Data from PKS

Keyserver requests are done via a HTTP GET URL that encodes the request data within it.

Following are some of the example request to PKS using HKP protocol.

Search for all keys containing the string "pruthviraj":

http://pool.sks-keyservers.net:11371/pks/lookup?op=vindex&search=gtuprldummy

Get key 0x737435ED (32-bit key ID):

http://pool.sks-keyservers.net:11371/pks/lookup?op=get&search=0x737435ED

The HKP URL consists of “op” and “search” variables. The “op” defines what the operations the URL is requesting from the server.The search variable dictates what terms or Ids should be searched within the database of server to list the keys.

The "**get**" operation requests keys from the PKS.The response to a successful "get" request is a HTTP document containing a GPG Keyring and ASCII armoured representation of it.

The response may be returned if the keys are matched in following format.

<Initial Line Break>
-----BEGIN PGP PUBLIC KEY BLOCK-----
ASCII ARMORED KEY DATA
....
....
....
-----END PGP PUBLIC KEY BLOCK-----
<Final Line Break>

If no keys match the request, the keyserver should return an appropriate HTTP error code such as 404 ("Not Found").

3.2 Submitting Keys to PKS

Keys are submitted to PKs using HTTP Post method. For example:

http://pool.sks-keyservers.net:11371/pks/add

The body of the POST message contains a "keytext" variable which is set to an ASCII armoured GPG key ring. The ASCII armoured key ring should also be “urlencoded” to make it compatible with HTTP’s POST method standard. More than one key can be submitted at once using a single POST request.

4. Existing publically available Public Key Servers

Following are some of the popular publically available PKS:

SKS

- GPG based PKS
- Largest key server on internet with 3.4 million keys
- 78 servers in its pool which are spread across the globe
- Open Source
- Developed using OCaml and uses Berkely DB as its backend
- HKP Compatible
- WoT based key authenticity verification
- Very Robust and quick reconciliation

MIT PGP PKS

- Amongst the very first to implement PGP based PKS
- Less Secure than SKS
- Original server now obsolete
- Currently runs a version of SKS and syncs with SKS pool
- HKP Compatible
- WoT based key authenticity verification

Hockeypuck,

- GPG based PKS
- Open Source
- Developed using Go! language and uses PostgreSQL as its backend
- HKP Compatible
- Can sync with SKS pool
- WoT based key authenticity verification

5. Authenticity of the Key

A key uploaded to a server is of use only if it is genuinely uploaded by the owner. Hence it is key server's responsibility to make sure that the key and the identity of the owner are closely tied together. In GPG the Identity of the owner is its email address. The PKS must provide a mechanism through which the users can verify the authenticity of the key. To verify whether the email ID embedded in the uploaded key is really the owner of the email address or someone else has uploaded a fake key of that email ID should be the main goal of any PKS.

For example if a key has an email ID prc@example.com then users of the PKS should be able to verify that the key was indeed uploaded by owner of email ID and not by any other malicious person.

Today all the available PKS mentioned in section 4 make use of concept called "Web of Trust" to verify authenticity of the key. Next section explains working of Web of trust in GPG Universe.

6. Web of Trust

Web of Trust is used in PGP to check authenticity of the public key. WoT can be represented as a directed graph where vertices are GPG Public Keys, and the directed edges represent digital signature.

Usually one can decide whom to trust as an introducer of new keys, to a lesser or stronger degree. If the resulting web of keys and trust relationships allows one to establish a link between oneself and the target identity, then communication can take place.

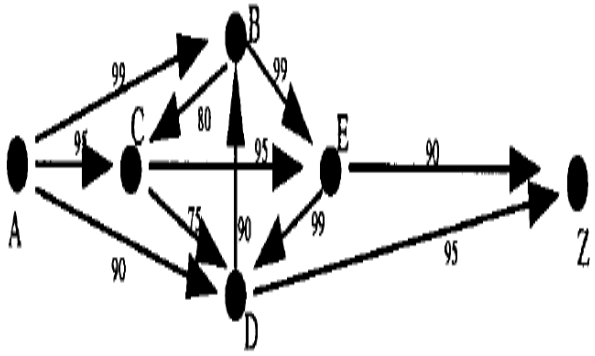

Figure 7. A sample Web of Trust

Above figure gives an illustration of such a web of trust. Point A represents the user Adam as starting point, and Z (Zohan) is the target point. This translates as user A holding the keys of B, C, D, E, and Z, having signed keys B, C, and D himself, and wanting to communicate with Z. The values in the figure represent trust invested in links, represented as percentile values, or in other words, a value bound to a signature, assigned by the signer, and indicating how high a probability he assigns to the signee really being whom he claims to be.

This promptly gives raise to some concerns. For example, for such a model to be valid, the way trust is determined must be standardized, as to make clear what a probability value actually means. Probability values could be assigned to a close acquaintance being whom she claims to be, or to a stranger whose passport and drivers, license have been inspected being whom he claims to be. Additionally, one must not only examine the value of trust bound to individual links, but also the trust one must invest in intermediate nodes to introduce third ones.

7. Critical Review

As we mentioned earlier that term "public" in Public PKS means "no central authority to control the operations of the PKS."

Most popular server is SKS which is totally distributed and is replicated across several pools of servers across the globe.

Geographically dispersed data replication is not new concept but wherever it is used all the replicating copies reside in different servers of the same owner. Here there are different owner of each servers which voluntarily maintain servers in public interest. Here the

servers are not under any single organization but are owned individually.

PGP was created in first place to counter government authorities' unethical snooping of private electronic communication. Hence distributed ownership of servers holding PGP keys is obvious requirement for the PGP to work without being affected by any influence.

For this "Web Of Trust" is used where the authenticity of the keys is dependent on who has signed the key and how many has signed the key. Here no single entity can control the validation of the key.

The trust factor of the key depends on the Social network of the key owner. If the owner has very good social network including people with highly trusted keys than he can easily get his key signed by more people. If someone with very few connections than it will be very difficult for him to get his key trusted even if his key is legitimate.

Also if one widely trusted person has signed the key than it doesn't mean that all the others signees of the key are also legitimate. WoT is challenged by many such real world social engineering challenges.

If an organization implements a PKS which uses WoT, than the organization's employees has to keep getting their keys signed by other employees. This process becomes quite cumbersome if we have to repeat it each time we want to change our keys. Keys needs to be changed or updated for several reasons one of the most popular being private key getting compromised.

As every organization is also a small society, this social dependence on each other may result in internal clash and may malign the status quo of the organization.

8. Conclusion and Future work

Public Key servers are integral part of the PGP Public Key Infrastructure. All available PKS were developed with only one goal in mind which was to counter authorities' snooping abilities. To achieve this they used "Web of Trust" mechanism, which needs quite large social networks with enough tech savvy friends. This setup may be useful for Information privacy activists and hackers but is not suitable for day to day use by others.

All "public" PKS cannot be fully trusted to store our key because concept Web of Trust is too full of flaws to establish authenticity of the key. If we cannot surely verify that weather the key stored on PKS is legitimate or not then such PKS is of little use.

Still existing Public PKS provide a good foundational framework on to which we can build a PKS which is more reliable and can also be used by average day to day computer user. Public PKS are also not suitable to be used directly in a private organization because of its social structure requirements.

Hence future works to enhance PKS includes development of a new robust key authentication mechanism which is free of social engineering challenges posed by WoT. The PKS should be flexible enough so that it can be easily deployed and managed by private organization. The future mechanism must be able to support HKP for it to be compatible with all current PGP implementations.

References

[1] **Jason Hogg, Don Smith, Fred Chong.** *Web Service Security - Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0.* s.l. : Microsoft, 2005.

[2] **P. Zimmermann, W. Stallings,D. Atkins.** *IETF RFC-1991,PGP Message Exchange Formats.* s.l. : IETF, August,1996.

[3] **L. Donnerhacke, H. Finney,R. Thayer,J. Callas.** *IETF RFC 2440 - OpenPGP Message Format.* s.l. : IETF, November 1998.

[4] **Koch, Werner.** *The GNU Privacy Guard Manual.* Boston : the Free Software Foundation, August 2013.

[5] Pretty Good Privacy. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/Pretty_Good_Privacy.

[6] Digital signature. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/Digital_signature.

[7] GNU Privacy Guard. *Wikipedia.* [Online] http://en.wikipedia.org/wiki/GNU_Privacy_Guard.

[8] **Shaw, D.** *IETF Draft - The OpenPGP HTTP Keyserver Protocol (HKP).* s.l. : IETF, March, 2003.

[9] **Fiskerstrand, Kristian.** *SKS Keyservers.* [Online] 2013. http://sks-keyservers.net/.

[10] sks-keyserver - Bitbucket. [Online] https://bitbucket.org/skskeyserver/sks-keyserver/overview.

[11] MIT PGP Public Key Server. [Online] http://pgp.mit.edu/.

[12] **Marshall, Casey.** Hockeypuck. [Online] 2013. http://hockeypuck.github.io/.
[13] **marshall, Casey.** hockeypuck in Launchpad. [Online] 2013. https://launchpad.net/hockeypuck.
[14] **Penning, Henk P.** analysis of the strong set in the PGP web of trust. [Online] [Cited: 08 November 2013.] http://pgp.cs.uu.nl/plot/.
[15] *Walking the Web of Trust.***Caronni, Germano.** Gaithersburg : IEEE, June,2000.ISBN - 0-7695-0798-0.
[16] *Trolling the Web of Trust.* **Lee, Micah.** Amsterdam : OHM, July,2013.
[17] Mining pgp keyservers. *Cryptome.* [Online] 5 July 2013.http://cryptome.org/2013/07/mining-pgp-keyservers.htm.