OpenPGP.conf:
An update on the sks-keyservers.net services

Kristian Fiskerstrand
94CB AFDD 3034 5109 5618
35AA 0B7F 8B60 E3ED FAE3

9 September 2016
Table of Contents

1 Quick intro to keyservers and related protocols

2 sks-keyservers.net

3 Keyserver network and OpenPGP community
Quick intro to keyservers, protocols etc I

▷ Synchronizing Key Server (SKS) is the most used keyserver today.

▷ One notable feature of SKS is the implementation of the fast reconciliation algorithms that were part of Yaron Minsky’s PhD thesis.

▷ SKS replaced PKS that synchronized using email and did not include the updated features of OpenPGP presented in RFC4880.

▷ SKS is written in objective caml, and the main development team is Yaron, John Clizbe and myself. Contributions are welcome.
Quick intro to keyservers, protocols etc II

- SKS supports elliptic curves defined in RFC6637 ("Elliptic Curve Cryptography (ECC) in OpenPGP") since version 1.1.5 (released 5th May 2014)
  - ECDH, ECDSA
  - Curves: nistp521, nistp384, nistp256, brainpoolP256r1, brainpoolP384r1, brainpoolP512r1, secp256k1

- Support for EdDSA / Ed25519 and Curve25519 is included in version 1.1.6 (released 7th August 2016)
  - EdDSA
  - Curves/Schemes: Ed25519 (EdDSA), Curve25519 (ECDH)
Quick intro to keyservers, protocols etc III

- SKS is single-threaded
- Recommendations for setting up SKS is listed on [https://bitbucket.org/skskeyserver/sks-keyserver/wiki/Peering](https://bitbucket.org/skskeyserver/sks-keyserver/wiki/Peering)
- These recommendations includes instructions on setting up a reverse proxy
- In particular this avoids DoS due to slow connections
- Additionally some server administers that experience high loads use a load-balanced setup (10)
Quick intro to keyservers, protocols etc IV

- The keyservers in the pool is accessed based on the Horowitz Key Protocol (HKP).
- HKP is a layer on top of HTTP defining how to access the keyserver. It was never formally accepted as a standard but the basis is found in [http://tools.ietf.org/html/draft-shaw-openpgp-hkp-00](http://tools.ietf.org/html/draft-shaw-openpgp-hkp-00).
- Development since the initial draft is based on community consensus, mainly between GnuPG and SKS as reference implementations.
Quick intro to keyservers, protocols etc V

▷ Some HKP examples http://pool.sks-keyservers.net:11371

Stats request:
/pks/lookup?op=stats

Verbose index:
/pks/lookup?op=vindex
&search=kf@sumptuouscapital.com

Get:
/pks/lookup?op=get
&search=0x0b7f8b60e3edfae3
&options=mr – Machine readable output
Quick intro to keyservers, protocols etc VI

▷ Alternative keyservers still exist, including Hockeypuck (written in Go) and that has preliminary support for SKS synchronization.

▷ Other keyservers also exist that use different interfaces, e.g. PGP Corp’s/Symantec’s keyservers are using LDAP which includes the Symantec Encryption Server (former PGP Universal) used in some corporate settings.
| 1 | Quick intro to keyservers and related protocols |
| 2 | sks-keyservers.net |
| 3 | Keyserver network and OpenPGP community |
Introduction I

▷ **https://sks-keyservers.net** provides a convenient way for end users of OpenPGP to retrieve and update keys from synchronised and responsive HKP keyservers

▷ The project was started in 2006[1]

▷ The servers are mainly based on SKS (p. 3).

▷ Several pools of keyservers are available

  - **pool.** Main pool (on port 11371)
  - **hkps.pool.** TLS enabled pool (HKPS) (more on p. 15)
  - **{NA,EU,OC}.pool.** Geographical pools (more on p. 14)
  - **p80.pool.** Firewall friendly, servers responding on port 80.
## Introduction II

<table>
<thead>
<tr>
<th>subset.pool.</th>
<th>Supports latest functionalities. Currently SKS 1.1.6 only; ensuring support for e.g. Elliptic Curve keys c.f. RFC6637 as well as Ed25519/Curve25519.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ha.pool.</td>
<td>High availability. Only including servers reported as operating as clustered setups.</td>
</tr>
<tr>
<td>Tor/Onion</td>
<td>Pool to allow access as a Tor hidden service (more on p. 18)</td>
</tr>
</tbody>
</table>
Introduction III

▷ These pools are functioning as a DNS Round Robin
▷ Currently ~ 100 servers included in the main pool and another ~ 30 detected but rejected (inclusion criteria on p. 13)
▷ The pools are accessible using both IPv4 and IPv6 (with some special pools limited to a single stack, e.g. ipv\{4,6\}.pool.sks-keyservers.net)
▷ GnuPG defaults to keys.gnupg.net which is a DNS CNAME to pool.sks-keyservers.net
Inclusion criteria

▷ Servers responds to HKP queries on port 11371
▷ The server’s number of available keys can’t be less than than median number of keys − Δ where Δ is calculated in a two-stage process. First excluding outliers (diff exceeding ±0.5σ) then Δ = \text{Max}(0.5\sigma, 300) of the recalculated σ
▷ Upgraded to a compatibility level of SKS 1.1.5 or higher.
▷ Server name reported is a fully qualified domain name
▷ HTTP/1.1 POST does not result in HTTP status 417
▷ Reverse Proxy (providing via header or based on server header)
▷ Server is not in the global exclude list
Geographical pools

(1) During an update run, determine the response time from multiple client measuring stations (grouped by pool) as the mean of N attempts and the bandwidth capacity for each individual server.

(2) Calculate the mean ($\mu$) and standard deviation ($\sigma$) of the aforementioned parameters across the servers in the pool.

(3) Exclude outliers to the results based on $N(\sigma)$ rejection criteria.

(4) Calculate new $\mu$ and $\sigma$ for the remainder of the servers.

(5) Calculate the SRV weights for the individual servers.

(6) Sort the servers by weights and pick top 10 for each individual pool.

SRV weights are defined as:

$$
\text{weight} = \alpha + \min\left(\beta_R \left(\frac{1}{(R - (\mu_R - 2\sigma_R))^y}\right), \phi\right) \\
+ \beta_B \left(\frac{B}{\sum_{j=1}^{m} B_j}\right) \\
+ \beta_P \cdot D_P \cdot \rho
$$

where:

- $\alpha$ is a constant to provide a base weight
- $\beta_i$ is the loading (weight) of the respective factor in determining the SRV weight
- $D_P$ is a dummy variable that is 1 if a reverse proxy exists and 0 otherwise
- $\rho$ is the additional weight given for a reverse proxy enabled server
- $y$ is a constant (even number) that defines penalization for deviation
- $\phi$ is the ceiling of weight to be added for $R$

Further information is available at [2]
HKPS Pool I

▶ HKPS is regular HKP over TLS (port 443)
▶ In order to achieve such a setup across multiple independent servers, all participants in the pool needs a certificate signed by the sks-keyservers.net CA (SCA)
▶ The certificates contain the CN of the individual servers and include the pool hostnames as subjectAltNames.
▶ The SCA issued certificate needs to be presented whenever the hkps SNI is requested, but several operators also maintain their own certificates from the global PKIX system for their individual hostname.
HKPS Pool II

▷ The public SCA cert is available at https://sks-keyservers.net/ca/sks-keyservers.netCA.pem and is signed by my OpenPGP key in a detached signature with .pem.asc suffix.

▷ The Certificate Revocation List (CRL) is published at https://sks-keyservers.net/ca/crl.pem

▷ the SCA cert is also included in GnuPG:
http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=dirmngr/sks-keyservers.netCA.pem;hb=refs/heads/master
HKPS Pool III

▷ More info on https://sks-keyservers.net/overview-of-pools.php#pool_hkps

▷ The GnuPG 2.1 branch doesn’t require any additional configuration starting with 2.1.11

▷ Using the HKPS pool for 1.4 and 2.0 branches of GnuPG requires the following configuration (requires a full curl implementation, i.e. not curl-shim):

```
~/gnupg/gpg.conf for GnuPG 1.4 and 2.0 branches

keyserver hkps://hkps.pool.sks-keyservers.net
keyserver-options ca-cert-file=/path/to/CA/sks-keyservers.netCA.pem
```
Tor/Onion pool

- An experimental Tor hidden service is running as `hkp://jirk5u4osbsr34t5.onion`
- The pool currently consists of ~15 servers
- Tor support is included since GnuPG 2.1.10
- Since it still uses hkp, the tor service needs to respond to port 11371
Table of Contents

1 Quick intro to keyservers and related protocols

2 sks-keyservers.net

3 Keyserver network and OpenPGP community
Number of keys

▷ I started recording statistics on the number of keys in 2011.

▷ Since then there are about 1.5 million new keys available.

▷ There has been a noticeable change to the growth rate at the time of the Snowden revelations.

▷ A more detailed breakdown of the keys (key length, algorithms etc) is available at http://blog.sumptuouscapital.com/2014/01/openpgp-key-statistics/
Network analysis

- With a growing number of servers, keys and other updates the time to replicate the data across the network increases.
- In order to optimize reconciliation times, bottlenecks in the network have to be detected and fixed.
- This is mostly a manual operation aided by tools for network analysis.
The keyserver community is generally friendly and is active in providing support and responding to new requests

Wiki at https://bitbucket.org/skskeyserver/sks-keyserver/wiki/Home

Mailing list at sks-devel@nongnu.org, both for SKS questions and general keyserver operational procedures
Duplicate keyids in focus

▷ Since the evil32 dataset was released there has been a bit of discussion on the use of short keyids in the web output of keyservers
▷ The OpenPGP security model is founded on object based security where the content of the OpenPGP keyblock is self-protected
▷ Any OpenPGP key needs to be verified out of band before being used
▷ Verification happens using full fingerprint
▷ keyids provide a convenient quick identifier
▷ Duplicate keyids (32 or 64 bit) in existence don’t affect security when using OpenPGP or the keyservers at all (if anything it increases security of people are sufficiently confused to properly verify keys)
Questions

Any questions?
References
