

OpenPKSD

OpenPKSD

Hironobu SUZUKI

2001/Sep/26

PKSD

- Design and implement
 - Marc Horowitz in 1996
- Public key server daemon
 - Running under UNIX (Solaris, FreeBSD, Linux, etc.)
- Public keys pool server
 - Submitting public key
 - Retrieving someone's public keys
- PGP/GPG connectivity
 - hkp (Horowitz Key Protocol)

It is NOT

- PKSD is NOT the certificate authority
 - Only pool public keys
 - Never issue any certificate for someone's public key
 - “Philip Zimmermann’s philosophy”
 - Against the nightmare of “Big brother”, government or/and monopoly company

Why We Need PKSD

- Easy to get someone's public keys
- Easy to distribute my public key(s)
- Easy to submit my digital-signed public key
- Better than web page (maybe)
- Better than e-mail (maybe)
- You show only your digital fingerprint

Today

- Horowitz's pksd-0.9.4 + patch level 2
 - It have been working fine!!
 - OpenPGP sub-packet can't be handled
- Synchronize each pksd sites
 - 10 or more sync sites
 - 1,400,000 public keys entries
 - 3,000/day or more transactions between each sync sites

Tomorrow

- OpenPKSD sub-packet should be supported
- Server has 100,000,000 public keys!!
 - Current version of PKSD can't handle 1 billion keys
- Many transactions
 - Like as yahoo, e-bay and/or other huge scale sites

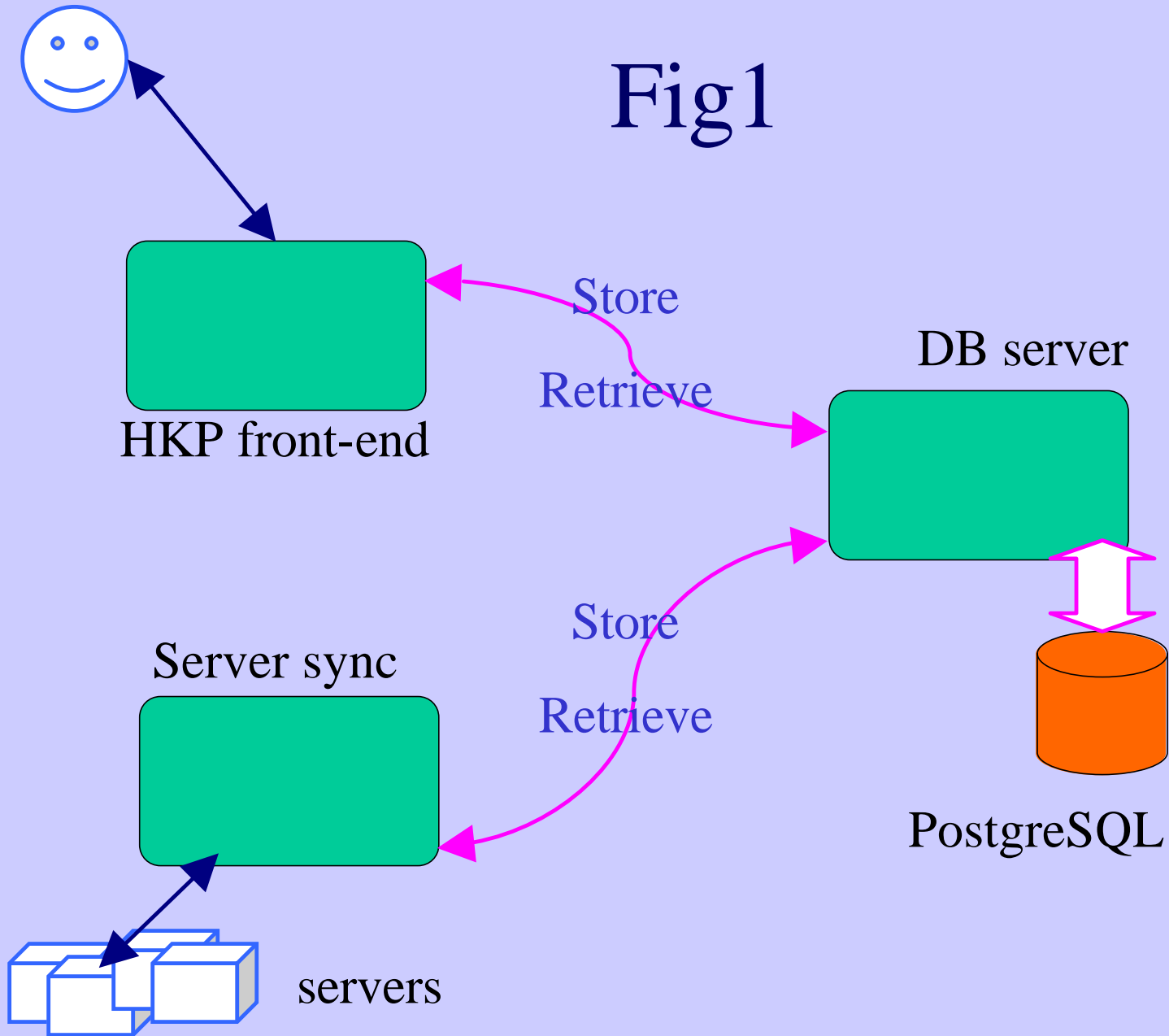
What we should do

- Support OpenPKSD sub-packet
 - RFC2440
- Strong DB
 - Not only high-performance but also flexible extension capability and easy operation
- Clustered servers
 - Scalability with cheaper H/Ws
- Sophisticate key-sync mechanism
 - Like as ihave/sendme, mirror DB or something good

OpenPKSD structure

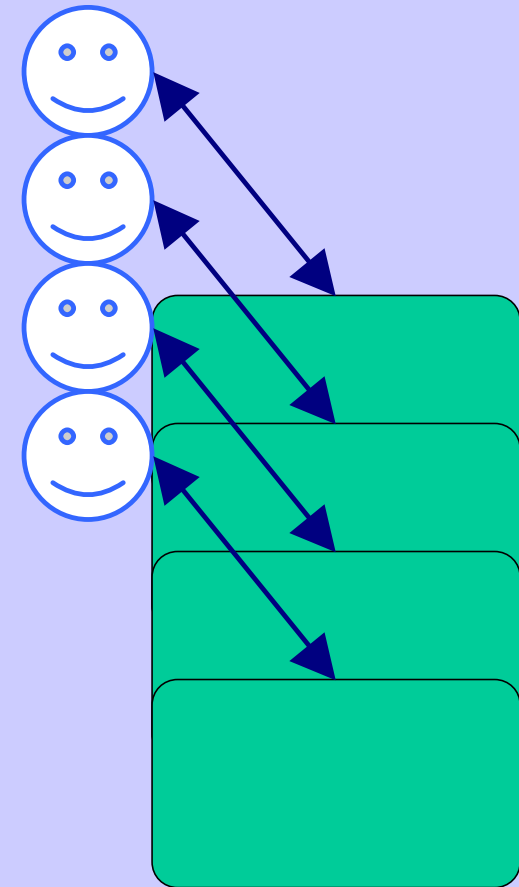
- Hkp front-end agent daemon
 - Connection between PGP/GPG and OpenPKSD
 - Extract and/or reconstruct key contents
- DB server daemon
 - Restore/retrieve contents
- Server-sync daemon
 - Synchronize keys between other sync servers

Fig1



HTTPD cluster

- HKP is based HTML format
- Httpd cluster is usual thing
- Apply same techniques



HKP front-end
cluster

PostgreSQL cluster

- Database cluster is usual idea
- 10,000,000 keys can be handled by one postgresQL server (really?)
- 100,000,000 keys!! Use DB cluster

Site sync, today

- Site A accept Alice's key
- Site A send Alice's key to other sync sites
 - By e-mail
- Very easy, but not perfect
 - I estimate there are 1 ~ 3 % different keys between each sites

Real site sync

- Site A has Alice's key
- Site B hasn't Alice's key yet
- Site B should ask Alice's key for Site A
- But how do they know B site hasn't Alice's key?
 - Do you know a good algorithm for it?