# Verbreitung von OpenPGP-Keys mit WKD, WKS und DANE

s3lph

s3lph@s3lph.me

@s3lph@chaos.social

@s3lph:kabelsalat.ch

CoSin 2024

29.06.2024

# Über mich

- s3lph (he/him)
- Linux Systems Engineer
- Macht `$dinge` im CCC Basel
  - Ansible
  - Python
  - E-Mail
- www.s3lph.me

# Einführung

- OpenPGP: Dezentrale PKI
- Herausforderung: Verbreitung von Keys
- Bisherige Ansätze:
  - Key an E-Mails anhängen
  - Autocrypt-Header in E-Mails
  - Zentrale Keyserver, z.B. `keys.openpgp.org` → HTTP Keyserver Protocol (HKP)

# HTTP Keyserver Protocol

- IETF Draft: draft-gallagher-openpgp-hkp-04
- Suche nach Keys
  - `GET /pks/lookup?op=index&search=s3lph@s3lph.me`
- Key herunterladen
  - `GET /pks/lookup?op=get&search=s3lph@s3lph.me`
  - `GET /pks/lookup?op=get&search=0xE17129ADADF4A2A4`
- Key hochladen
  - `POST /pks/add`
  - ASCII-armored Key im Body

# Probleme mit zentralen Keyservern

- Verfügbarkeitsprobleme
- Keine Authorisierung zum Hochladen von Keys
- i.d.R. append-only
- Bekannte Angriffe gegen Keyserver
  - Key Poisoning

# DANE

- *DNS-based Authentication of Name Entities*
  - Voraussetzung: DNSSEC
- RFC7929
- DNS Record Type: `OPENPGPKEY` / `TYPE61`
- Wire Format: Public Key im Binärencoding
- Zone File Format:
  - `OPENPGPKEY` → Base64
  - `TYPE61` → Length + Hexdump

# DANE

- `3d5fbcde113df14facdc43b7b55390bff549d12e8f377b209f8d2134._openpgpkey.s3lph.me.`
  - NSEC3 verwenden!
- Hash: $\mathrm{substr}(0, 56, \mathrm{hexdump}(\mathrm{sha256}(\mathit{localpart})))$
  - Die letzten 8 Zeichen werden nicht verwendet
- Nur 1 Key pro Record (kein Keyring!)
- Aber: Mehrere Records möglich

# DANE: Tools

- `sq network dane fetch -o - s3lph@s3lph.me`
  - `DANE: s3lph@s3lph.me: Cert not found` → sq kann noch kein ed25519
- `gpg -v --auto-key-locate=dane,nodefault --locate-keys s3lph@s3lph.me`
  - `automatically retrieved 's3lph@s3lph.me' via DANE`

# WKD

## Web Key Directory

**The Web Key Directory (WKD) is GnuPG's standard system for key discovery**. That is, it returns a public key for a supplied mail address. It is a distributed system in the same way email is distributed.

# WKD

- IETF Draft: draft-koch-openpgp-webkey-service
- HTTPS
- 2 verschiedene URL-Schemas:
  - **Advanced Method** unterstützt mehere Domains
  - **Direct Method** unterstützt nur eine Domain

# Advanced Method

- `https://openpgpkey.example.org/.well-known/openpgpkey/example.org/hu/apr3aj3jqcf89yd69qd8pkjp3pzawxhx?l=example`
- Hash: $\mathrm{z\_base32}\,(\mathrm{sha1}\,(\mathrm{ascii\_lowercase}\,(\mathit{localpart})))$
  - Z = Base32-Alphabet von Phil Zimmermann (PGP-Entwickler)
- URL-Parameter `l` nachträglich hinzugefügt
  - Case Sensitivity
  - Catch-All, `+suffix`
  - ...
- Directory Listing sollte deaktiviert sein
- Die Subdomain openpgpkey kann auch ein CNAME/Redirect sein

# Direct Method

- `https://example.org/.well-known/openpgpkey/hu/apr3aj3jqcf89yd69qd8pkjp3pzawxhx?l=example`
- Direct Method wird nur verwendet, wenn Advanced-Subdomain nicht existiert

# Exkurs: WKDaaS

- `openpgpkey.example.org. IN CNAME wkd.keys.openpgp.org.`
- Keys einer Domain vom `keys.openpgp.org` Keyserver via WKD verfügbar
- Stellt sich on-the-fly ein *Let's Encrypt*-Cert aus
- ⇒ WKD/DANE nicht zwingend vertrauenswürdiger als HKP

# Tools

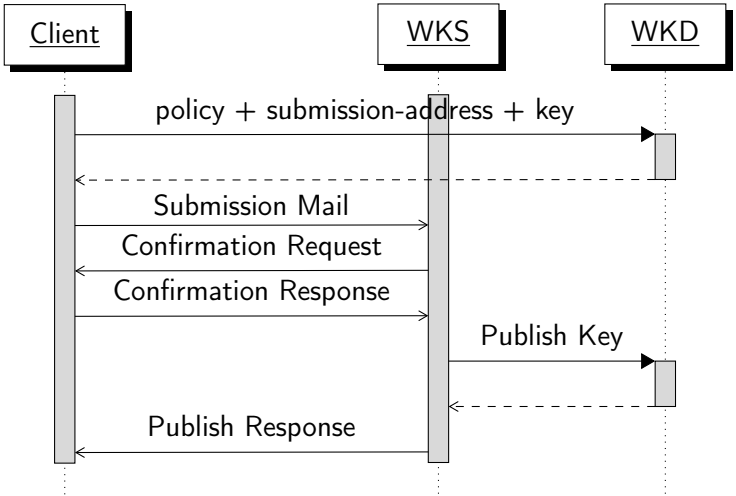
```
> sq network wkd fetch s3lph@s3lph.me
> sq network wkd url s3lph@s3lph.me
> sq network wkd direct-url s3lph@s3lph.me

> gpg -v --auto-key-locate=wkd,nodefault --locate-keys s3lph@s3lph.me
```

# WKS

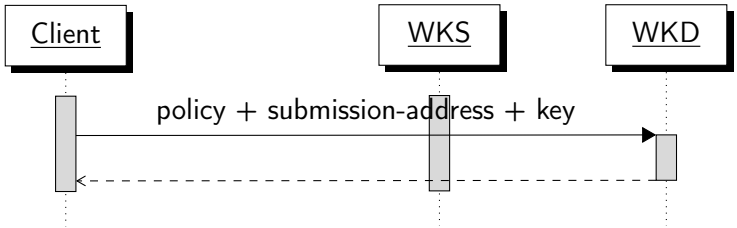
- Web Key Service
- IETF Draft: draft-koch-openpgp-webkey-service
- Protokoll, um Keys in eine WKD zu bekommen
- Über PGP/MIME-signierte/-verschlüsselte E-Mails

- Ziel: Soll von MUAs implementiert werden.

# Ablauf


Client
WKS
WKD
policy + submission-address + key
Submission Mail
Confirmation Request
Confirmation Response
Publish Key
Publish Response

# Ablauf




- 2 neue Endpunkte in der WKD:
  - `/.well-known/openpgpkey/example.org/submission-address`
  - `/.well-known/openpgpkey/example.org/policy`

# Policy & Submission Address

- `submission-address`: Adresse, an die der Key gesendet werden muss:

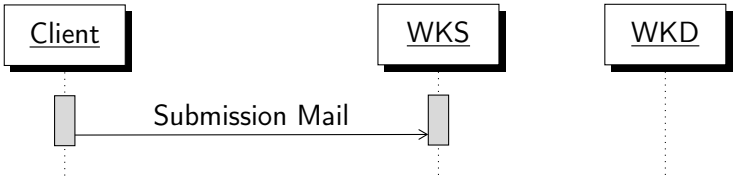
  `wks@example.org`

  - Key der Submission Address muss in der WKD liegen

- `policy`: Angaben über Verhalten des WKS:

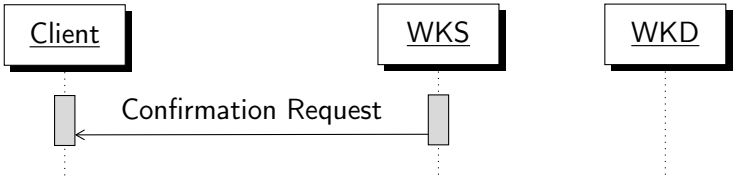
  ```
  submission-address: wks@example.org
  mailbox-only
  ```

# Ablauf




- MIME Multipart E-Mail
- ASCII-Armored Pubkey als ein MIME-Part

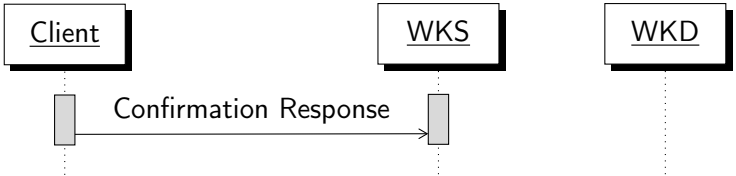
# Ablauf




- Verschlüsselte, unsignierte PGP/MIME Antwort mit Challenge:

```
type: confirmation-request
sender: wks@example.org
address: user@example.org
fingerprint: 5621C9CD7FFF8A1E514B1190566A47CD
nonce: Stv8R6rN1d1fCTzCnui2HdWcHOlzok3HBGfT9PpIm9U
```

# Ablauf




- Verschlüsselte + signierte PGP/MIME Antwort mit Response:

```
type: confirmation-response
sender: wks@example.org
address: user@example.org
nonce: Stv8R6rN1d1fCTzCnui2HdWcHOlzok3HBGfT9PpIm9U
```

# Clients

- Gemäss GnuPG Wiki:
  - KMail
  - ~~Thunderbird + Enigmail~~
- `gpg-wks-client`
  - Benötigt ein zusätzliches CLI-Mailprogramm (sendmail, mutt, ...)

# Server

- `gpg-wks-server`

# Server

- `gpg-wks-server`
  - Anbindung an Mailserver
  - Anbindung an Webserver
  - GnuPG Keyring mit Submission Key
  - Einrichtung: `https://wiki.gnupg.org/WKS`

# Probleme Mit gpg-wks-{client,server}

- Sehr strikte und unübliche Message-Formate werden strikt enforced
- Schwer zu erzeugen insb. wenn nicht vom MUA implementiert
- `gpg-wks-client` benötigt ein "oldschool" Mailsetup

# Probleme Mit gpg-wks-{client,server}

## Robustness Principle (“Postel’s Law”)

Be conservative in what you send, be liberal in what you accept.

# EasyWKS

- Eskaliertes Wochenend-Projekt
- Implementation von WKS-Client und -Server in Python
- `https://git.kabelsalat.ch/s3lph/easywks`

# Ziele

- Drop-in Replacement für gpg-wks-server
  - Identisches on-disk Layout
- "Modernere" Integration mit Mailinfrastruktur
  - LMTP+SMTP anstatt stdin+sendmail
- "Notfalls auch von Hand bedienbar"
  - "Ist da irgendwo eine Confirmation Response?" → Publish
  - "Ist da irgendwo eine Public Key?" → Confirmation Request
- Einfach bedienbarer Standalone-Client
  - IMAP/POP3+SMTP, Autodiscovery
  - Key aus dem GnuPG Keyring

# Ziele

- Drop-in Replacement für gpg-wks-server
  - Identisches on-disk Layout
- "Modernere" Integration mit Mailinfrastruktur
  - LMTP+SMTP anstatt stdin+sendmail
- "Notfalls auch von Hand bedienbar"
  - "Ist da irgendwo eine Confirmation Response?" → Publish
  - "Ist da irgendwo eine Public Key?" → Confirmation Request
- Einfach bedienbarer Standalone-Client
  - IMAP/POP3+SMTP, Autodiscovery
  - Key aus dem GnuPG Keyring

# Ziele

- Drop-in Replacement für gpg-wks-server
  - Identisches on-disk Layout
- "Modernere" Integration mit Mailinfrastruktur
  - LMTP+SMTP anstatt stdin+sendmail
- "Notfalls auch von Hand bedienbar"
  - "Ist da irgendwo eine Confirmation Response?" → Publish
  - "Ist da irgendwo eine Public Key?" → Confirmation Request
- Einfach bedienbarer Standalone-Client
  - IMAP/POP3+SMTP, Autodiscovery
  - Key aus dem GnuPG Keyring

# Ziele

- Drop-in Replacement für gpg-wks-server
  - Identisches on-disk Layout
- "Modernere" Integration mit Mailinfrastruktur
  - LMTP+SMTP anstatt stdin+sendmail
- "Notfalls auch von Hand bedienbar"
  - "Ist da irgendwo eine Confirmation Response?" → Publish
  - "Ist da irgendwo eine Public Key?" → Confirmation Request
- Einfach bedienbarer Standalone-Client
  - IMAP/POP3+SMTP, Autodiscovery
  - Key aus dem GnuPG Keyring

# Demo Time!

```
sq network wkd url demo@s3lph.me
sq network wkd fetch -o- demo@s3lph.me
client.py
sq network wkd fetch -o- demo@s3lph.me
```

# Roadmap

- DANE OPENPGPKEY ✓
  - Zone Transfer zu DNSSEC-signierendem Nameserver
- Web-Upload von Keys?
- Sequoia-PGP anstatt PGPy?
- Rewrite in Rust?

# Roadmap

- DANE OPENPGPKEY ✓
  - Zone Transfer zu DNSSEC-signierendem Nameserver
- Web-Upload von Keys?
- Sequoia-PGP anstatt PGPy?
- Rewrite in Rust?

# Roadmap

- DANE OPENPGPKEY ✓
  - Zone Transfer zu DNSSEC-signierendem Nameserver
- Web-Upload von Keys?
- Sequoia-PGP anstatt PGPy?
- Rewrite in Rust?

# Roadmap

- DANE OPENPGPKEY ✓
  - Zone Transfer zu DNSSEC-signierendem Nameserver
- Web-Upload von Keys?
- Sequoia-PGP anstatt PGPy?
- Rewrite in Rust?

Fragen?

s3lph@s3lph.me

@s3lph@chaos.social

@s3lph:kabelsalat.ch