Another Nail in the Coffin: OpenPGP Key Servers Gone Rogue

Tobias Mueller  mueller@informatik.uni-hamburg.de
Marius Stübs  stuebs@informatik.uni-hamburg.de
Outline

Introduction
- Use Cases for Key Servers
- SKS – Synchronising Key Servers
- OpenPGP basic structure + RSA basics
- **Why we need revocation**

Problems
- Missing Integrity Check
- Equivocation
- **The current concept and it’s implementation are broken**

Solution
- Certificate Transparency Log
- PGP Revocation Bytes
- **How to design a Revocation Service**
Intro

- Why do we need to exchange keys?
- What do I do when my keys got stolen?
  - Generate a Revocation Certificate beforehand
- How do PGP packets work?

Source: xkcd.com
Sending and encrypted Email

- Alice needs a key for sending an encrypted Email to Bob
Promise: Availability without prior contact

bob@example.com

creates public key 0xFC42A3B

uploads key

Key Server

stores public key

Database lookup

Alice

Wants to contact bob@example.com

Key request

Encrypted email
Key Servers’ promise: Availability of Cryptographic Keys

- Receiving the actual key material by UID or fingerprint
- Mitigate Deletion of Keys on one Key Server by Re-Distribution
- Update certificate, e.g. new sub-keys, new UIDs, new certifications
- Check for revocations

You have to check for integrity yourself
What if my **private key** gets compromised?

- When an attacker could successfully obtain the private key of Bob, Bob needs to **revoke** his key.
  Notice - Bob should generate a **revocation certificate for his key in advance**
Revocation – The happy path
Revocation – The evil path
Revocation – The evil path, Equivocation
Validation

<table>
<thead>
<tr>
<th>RSA Public Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>e = 100010</td>
</tr>
<tr>
<td>n = 1010101011101011010110101010</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UID = <a href="mailto:bob@example.com">bob@example.com</a></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Signature = 0xFE032CD108AEE563</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Key is revoked</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Signature = 0xD108AEE563FEA32C</th>
</tr>
</thead>
</table>

We call them **Revocation Bytes**

- don’t contain personally identifiable information (PPI)
Invalid packets get discarded

RSA Public Key
\[ e = 100010 \]
\[ n = \]
\[ 1010101011101011011010101010 \]
\[ 1010101011010010000001010011 \]
\[ 110101110111101111101110010 \]

UID = bob@example.com

Signature = 0xFE932CD108AEE563

Attacker says, key is revoked

Signature = 0xFFFFFFFFFFFFFFFFFFF

Invalid Revokation Bytes

\[ \text{\textcopyright}^{\text{\textregistered}} \text{korkr\textsuperscript{\textregistered}ss@cafép.e\textsuperscript{\textregistered}} \]
..., so what does the key server check then?
Anatomy of an OpenPGP Certificate

<table>
<thead>
<tr>
<th>Frame</th>
<th>Version</th>
<th>Sigtype</th>
<th>PK Algorithm</th>
<th>Hash Algorithm</th>
<th>Hashed Area Length</th>
<th>Hashed Area</th>
<th>Unhashed Area Length</th>
<th>Unhashed Area</th>
<th>Hash Prefix 1</th>
<th>Hash Prefix 2</th>
<th>MPI Length</th>
<th>EdDSA Signature R</th>
<th>EdDSA Signature S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>88 78</td>
<td>04</td>
<td>20</td>
<td>16</td>
<td>00 20</td>
<td>16 21 04 cd 6f 2d 93 e8</td>
<td>b4 a0 07 47 a2 88 70 05</td>
<td>00 0a</td>
<td>09 10 89 b4 a0 07</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0008</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td>87 0d 61 17 6a d5 c7 89</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td>02 5b dd 82 a9 02 1d 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0030</td>
<td>47 a2 88 70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0034</td>
<td>06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0035</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0036</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0038</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0040</td>
<td>eb 95 9a 50 d2 c4 20 ce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0050</td>
<td>0e 5a 4b 7a 2f 27 73 2a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0058</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>005a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0060</td>
<td>1e 2b 65 f4 24 20 11 3d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0070</td>
<td>fd 98 16 f7 98 e1 33 48</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Enigmail showing a revoked certificate (but it has not been revoked!)

<table>
<thead>
<tr>
<th>Account / User ID</th>
<th>Cre...</th>
<th>Key ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice <a href="mailto:alice@q2.re">alice@q2.re</a></td>
<td>2018-1...</td>
<td>40197D620EC...</td>
</tr>
</tbody>
</table>

[Image of a computer screen showing a dialog box for downloading OpenPGP keys, with an entry selected for import.]
SKS – Synchronising Keyserver

- Main implementation of HKP enabled daemon
- Set-reconciliation with near optimal communication
- Loosely connected set of servers in the pool
SKS – Synchronising Keyserver – inherent problems

- **Cannot delete keys**
  - GDPR concerns
  - Poison keys

- **Equivocation**
  - Strip packets

---

the pool of SKS keyservers: as anyone can upload anybody's key, and it does not allow to delete keys, it's IMHO by not compatible with GDPR.

---

Subject: Re: [Sks-devel] Unusual traffic for key 0x69D2EAD9 and 0xB33B4659
Date: Mon, 04 Feb 2019 10:49:30 +0100
Mailer: Roundcube Webmail/1.2.3

Hi,

Don't get me wrong, but within three days I've got 450G traffic which can be assigned to sks by 99.9%. Estimated to 30 days this means 4.5T (which is in good agreement of your 2+T/Key for these two poison keys).
Single threaded Ocaml

- Arcane architecture for high performance pools
- Maintenance
- Pool size

No integrity validation

- No semantic checks, i.e. identity
- No cryptographic checks, e.g. Invalid signatures
- Only rudimentary validation against the OpenPGP spec

And how do we tell sks to use more than ONE instance for the SAME database?!

You don't. SKS just isn't built that way. To get concurrency, you need to run multiple separate instances of SKS and configure them to gossip between each other. Then you can put a load balancing reverse proxy in front to simulate a multi-threaded server. Kristian and a few others have been operating this way for a while now.

Any chance that sks will be fixed some day?

Short answer, no. SKS is effectively running as end-of-life software at this point. It gets emergency bugfixes but little else. The improvements you are talking about would require an enormous refactoring of the codebase, likely requiring migration to a different database engine. Given that there are fundamental design flaws (poison keys) that aren't getting fixed, performance issues just aren't on the radar. Sorry.

1. Future updates for the key will be denied, including legitimate ones by key holder (FreePBX team). 2. DoS is still possible just by accessing/-fetching the key. To fix that, you’ll have to remove the DoS packets (large user packets with random gibberish, not valid per OpenPGP packet spec, does not validate cryptographically) or the whole key. 3. Anyone can create another poison key at any time and there's no way to fix that without breaking compat, it's a fundamental flaw :-(
The final breath of SKS
(*2002 - †2019)

1. Future updates for the key will be denied, including legitimate ones by key holder (FreePBX team). 2. DoS is still possible just by accessing/fetching the key. To fix that, you'll have to remove the DoS packets (large user packets with random gibberish, not valid per OpenPGP packet spec, does not validate cryptographically) or the whole key. 3. Anyone can create another poison key at any time and there's no way to fix that without breaking compat, it's a fundamental flaw :-(

I have not been running a keyservers myself since the first poison-key incident. Given the ongoing poison-key problems I am unlikely now to ever run one again, unless the entire codebase is overhauled. I have neither the time, energy nor skillset to perform this task, and it is becoming clear that nobody else does either.

This is not anyone's fault. It is just an unfortunate reality that we have to accept. SKS is end-of-life.
What can we do?
Requirements for a Revocation Service

- **Integrity**
  - A key server must not be able to alter data it has received.

- **Non-Repudiation**
  - The key server must be held accountable for serving manipulated data.

- **Privacy**
  - The key server must not learn who a client wishes to communicate with.
  - The uploaded data must not contain Personally Identifiable Data (PII).

- **Offline-Capability**
  - The store and forward architecture of the email ecosystem does not require the recipients to be online at the same time.

- **Timeliness**
  - Once a packet has been uploaded to a key server, it must serve it to clients requesting it.

- **Equivocation-resistance**
  - A key server must provide the same answer regardless of the identity of the client.
Let’s encrypt?

- A public service
- Fully automated
- Objective: Hands-out certificates for your domain, e.g. bobswebsite.de
- Runs as a script on your server
- You have to provide proof of access
  - e.g. by putting downloadable files somewhere
- Then you get a certificate for you domain
- And all major browsers accept it

- It is possible to **verify** automatically **before** publishing!

- Why not try this for emails?
What do we need?

Let's revoke!

CT-Log
Certificate Authority

Let's revoke
Revocation Service

Proofs ownership

Bob
E-mail address owner

Publishes revocation certificate
Towards an Equivocation resistant key revocation scheme

- Publicly Verifiable Append-only Data Structure
- Expensive to operate
CT-log-based Revocation Protocols

- WebPKI does not have an effective revocation scheme
- WebPKI uses Certificate Transparency to uncover wrongly issued certificates
- Google, Cloudflare, and others operate CT logs
Certificate Transparency - Merkle Tree
Certificate Transparency – Audit Proof

Figure 5

Audit proof for this certificate
Certificate Transparency - Infrastructure

Diagram showing the interaction between Certificate Authority, Log Server, Monitor, Auditor, example.com, and Client (browser) with arrows indicating flows and colors for different components.
Revocation Service

- Naively, clients store their revocation as specially encoded hostname
  - E.g. for foo@bar.com as revocationbytes.foo.at.bar.com
  - Then request a certificate for the hostname

- Clients can then inspect the log for those hostnames

- CT log is huge (~600MB)

- Optimise:
  - Provide well-known suffix, i.e. rvc-svc.org
Revocation Publishing Protocol

Bob

revokes
key

Revocation
Service

Generates TLS certificate for
revocationbytes.
keyid.rvc-svc.org

Certificate
Authority

Add to
CT Log

signed
certificate

revocationbytes
Revocation Querying Protocol

- Alice
- CT monitor

Request for *.keyid.rvc-svc.org

revocationbytes.keyid.rvc-svc.org

Verify!
Evaluation

- + No PII
- + Same security as CT
- o DNS label length
- o CT logs cease to exist
- - searching the log leaks recipient
Thanks!

- Running SKS incurs legal and operational risks
- Users can be equivocated on or served malicious data
- Blockchain-based revocation scheme for OpenPGP certificates

Contact

- Tobias Mueller <mueller@informatik.uni-hamburg.de>
  - 0564 46F0 7732 1A69 1C67 14EA 8A01 4674 C937 42FD

- Marius Stübs <stuebs@informatik.uni-hamburg.de>
  - 5887 4C3C CB21 3B1B F1A4 4A46 47D5 016C 42FF 7C2C