# SSH agent - `draft-miller-secsh-agent`

Damien Miller
djm@openssh.com
https://bsky.app/profile/damienmiller.bsky.social

# What does it do?

Two primary uses in SSH itself. Several other uses beyond these.

- Emulation of a “user HSM” - private keys can be unwrapped once, stored safely by the agent and used with low-friction thereafter.
- Agent forwarding - access to a SSH agent can be forwarded over SSH connections, allowing use of keys to follow the user around as they access remote systems

The latter especially represents a security/usability tradeoff.

# Protocol overview

Evolution of legacy SSHv1 agent protocol

Very simple client-initiated synchronous request/response protocol

- Reuses SSH wire encoding and familiar [len, type, payload] framing

Main verbs: *add key*, *add smartcard key*, *delete key*, *delete all keys*, *list keys*, *sign*

Several extensibility mechanisms: named extension requests, key constraints, new key types, signature flags. Most support `name@domain.org` names

*En passant* defines wire encodings for private keys

# Client requests

Request from client:

```
uint32              length
byte                type
byte[length - 1]    contents
```

Example (length implied):

```
byte                SSH_AGENTC_SIGN_REQUEST (13)
string              public key blob
string              data
uint32              flags
```

# Server responses

Server response uses the same wire format:

```
uint32              length
byte                type
byte[length - 1]    contents
```

Example:

```
byte                SSH_AGENT_SIGN_RESPONSE (14)
string              signature
```

Empty response payloads are common, e.g. SSH_AGENT_SUCCESS

# Message types

Client requests:

```
SSH_AGENTC_REQUEST_IDENTITIES
SSH_AGENTC_SIGN_REQUEST
SSH_AGENTC_ADD_IDENTITY
SSH_AGENTC_REMOVE_IDENTITY
SSH_AGENTC_REMOVE_ALL_IDENTITIES
SSH_AGENTC_ADD_SMARTCARD_KEY
SSH_AGENTC_REMOVE_SMARTCARD_KEY
SSH_AGENTC_LOCK
SSH_AGENTC_UNLOCK
SSH_AGENTC_ADD_ID_CONSTRAINED
SSH_AGENTC_ADD_SMARTCARD_KEY_CONSTRAINED
SSH_AGENTC_EXTENSION
```

Server Responses:

```
SSH_AGENT_FAILURE
SSH_AGENT_SUCCESS
SSH_AGENT_IDENTITIES_ANSWER
SSH_AGENT_SIGN_RESPONSE
SSH_AGENT_EXTENSION_FAILURE
SSH_AGENT_EXTENSION_RESPONSE
```

# Extension requests

Extension request message

```
byte                SSH_AGENTC_EXTENSION
byte[length - 1]    contents
```

Also used for discovery:

```
byte                SSH_AGENTC_EXTENSION (27)
string              “query”
```

Allows “*name@domain.com*” style extensions with graceful degradation.

# Key constraints

Keys may be added with “constraints”

```
byte                constraint1_type
byte[]              constraint1_data
...
```

Defined constraints include auto-deletion after interval, confirm on each use and an extension mechanism to support named constraints, i.e. “constraint@domain.com”

All constraints are defined as “critical” - an agent that cannot parse or does not support a constraint MUST refuse the request.

# Interface with SSH protocol

SSH agent forwarding can be requested by the client on a per-channel basis.
- Currently: **`SSH_MSG_CHANNEL_REQUEST`** ***"auth-agent-req@openssh.com"***
- Draft specified IANA codepoint: "***agent-req***"

SSH server will expose a listening endpoint
- E.g. a Unix domain socket, communicated via $SSH_AUTH_SOCK env
- Listener state machine is identical to other SSH channel listeners

Connections to listener create new channels
- Current name: ***"auth-agent@openssh.com"***, proposed: ***"auth-agent"***
- Usual SSH data channel state machine and lifecycle thereafter.

# Extensions in the real world

OpenSSH has a number of extensions, all AFAIK backwards-compatible

- Additional key types, e.g. “*sk-ecdsa-sha2-nistp256@openssh.com*”
- Restricted keys:
  - *“session-bind@openssh.com”* extension
  - *“restrict-destination-v00@openssh.com”* key constraint
  - Requires additional client and server changes, details at https://www.openssh.com/agent-restrict.html

Others have extended the SSH agent protocol for additional operations, non-SSH key types.

Message IDs 240-255 reserved for organisation-local use.

# Wide Implementation support

OpenSSH in 2000

PuTTY in 2001

Dropbear and Paramiko since 2005

libssh and Apache Mina since 2009

Golang x/crypto since 2014

AsyncSSH since 2015

Disclaimer: dates for everything here but OpenSSH is based on my reading of the projects' git histories; I could be wrong

# Draft history

This draft has been kicking around since 2016

Several rounds of feedback from several SSH implementers

Progress blocked on IANA SSH registries previously requiring Standards Action

- Fixed by RFC 9519 in 2024/01

Draft is strictly documenting existing practice, except:

- IANA names, e.g. “`agent-req`” vs “`auth-agent-req@openssh.com`”
- Extension query mechanism

# Thanks