OpenSSH vs OpenSSL Key Formats

This article is (probably too much of) an overview of the subject matter, but take heart: it will lead you down the right path, or so we hope. :)

My goal here is to provide a space to disambiguate and provide some vocabulary that will increase your understanding and make your googling easier.

The files that we're talking about are the ones that look like this:

- SSH
  - `id_rsa`
  - `id_ecdsa`
  - `*.pub`
- OpenSSL (has lots of different names for the same thing)
  - `priv.key`
  - `pubkey.pem`
  - `key.der`
  - `example.com.crt`
  - `cert.pem`

If you're looking specifically for info on SSH Public Keys, zoom ahead to this:

- The SSH Public Key Format (https://coolaj86.com/articles/the-ssh-public-key-format/)

Private Keys (Both)

Update: OpenSSH has now added it's own "proprietary" key format, which is described in the next section. This section is about the standard key formats, which do work for OpenSSH.

Both `ssh-keygen` (OpenSSH) and `openssl` (OpenSSL, duh) *can* generate private keys in standard DER/ASN.1 (x.509) formats.

Typically (as in every case as far as I'm aware), it's one of the following:

- PKCS#1 (for RSA only, supported in OpenSSH and OpenSSL)
  - RSA Only
  - OpenSSH & OpenSSL
- SEC1
  - EC (ECDSA) Only
  - OpenSSH & OpenSSL
- PKCS#8 (for RSA, EC(DSA), and others, supported in OpenSSL... not new standard for either)
  - RSA, EC (ECDSA), and others
  - OpenSSL (not sure about OpenSSH)

That's true for WebCrypto (and node crypto) as well - except that WebCrypto also supports JWK.

There are also various libraries like Rasha.js (http://git.coolaj86.com/coolaj86/rasha.js) (RSA tools for JavaScript) and Eckles.js (http://git.coolaj86.com/coolaj86/eckles.js) (ECDSA tools for JavaScript), both of which I worte, that support JWK as well.

Anyway, the PEM files look like this for both:

```
-----BEGIN PRIVATE KEY-----
MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgiYydo27aNGO9DBUW
eGEPD8oNi1LZDqfxPmQlieLBjVShRANCAAQhPVJYvGxpw+ITlnXqOSikCfz/7zms
yODIKiSueMN+3pj9icDgDnTJl7sKcWyp4Nymc9u5s/pyliJVyd680hjK
-----END PRIVATE KEY-----
```

For formats that don't embed the key type in the actual data you'll also see headers like `-----BEGIN RSA PRIVATE KEY-----` and `-----BEGIN EC PRIVATE KEY-----` (and the corresponding footers).

ECDSA keys are often referred to simply as `EC` (it's one of those "PIN number" / "DVD video" type things where the "DSA" descriptior is redundant much of the time).

OpenSSH Private Keys

Traditionally OpenSSH supports PKCS#1 for RSA and SEC1 for EC, which have `RSA PRIVATE KEY` and `EC PRIVATE KEY`, respectively, in their PEM type string.

Now it its own "proprietary" (open source, but non-standard) format for storing private keys (`id_rsa` , `id_ecdsa`), which compliment the RFC-standardized ssh public key format.

Although still PEM-encoded, you can tell when a key is in the custom OpenSSH format by the `OPENSSH PRIVATE KEY` indicator.

The new key files looks like this:

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAaAAAABNlY2RzYS
1zaGEyLW5pc3RwMjU2AAAACG5pc3RwMjU2AAAAQQR9WZPeBSvixkhjQOh9yCXXlEx5CN9M
yh94CJJ1rigf8693gc90HmahIR5oMGHwlqMoS7kKrRw+4KpxqsF7LGvxAAAAqJZtgRuWbY
EbAAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBH1Zk94FK+LGSGNA
6H3IJdeUTHkI30zKH3gIknWuKB/zr3eBz3QeZqEhHmgwYfCWoyhLuQqtHD7gqnGqwXssa/
EAAAAgBzKpRmMyXZ4jnSt3ARz0ul6R79AXAr5gQqDAmoFeEKwAAAAOYWpAYm93aWUubG9j
YWwBAg==
-----END OPENSSH PRIVATE KEY-----
```

Despite looking like it they don't actually contain DER-encoded x.509/ASN.1 keys and they're *not* OpenSSL compatible.

Note that they begin with `b3BlbnNzaC1rZXktdjE` which, when base64-decoded, reads `openssh-key-v1` .

If you'd like to learn the specifics of the format, take a look at this:

- The OpenSSH Private Key Format (https://coolaj86.com/articles/the-openssh-private-key-format/)

I wasn't able to find any documentation on the format whatsoever, so I think the above documentation I made from reading the source and reverse engineering valid keys is the best the web has to offer at present.

Public Keys (Both)

The one thing that you should know about public keys is that, in many cases they can be derived from the private parts of the private key (but not the other way around, obviously) and the private key typically contains the public parts embedded into it.

Thus a "private" key is actually a full *key pair*.

This is nice because it keeps code complexity down for applications that don't implement crypto themselves, but use libraries that just need the right parts. For example, my VanillaJS libs that convert between keypair formats don't need to depend on Big Int (https://coolaj86.com/articles/big-int-encoding/) libraries, so they remain small and manageable.

SSH naming conventions

SSH doesn't use extensions for its private keys, but they're always PEM (as shown above).

By default they're named either `id_rsa` or `id_ecdsa`, depending on the suite of the cryptography used (RSA or EC).

There are some other suffixes for outdated crypto standards (and perhaps newer ones if this article is really old by the time you read it), but we won't go into those here.

Public keys end in `.pub` and they're their own special format.

SSH Public keys (`.pub`)

SSH Public keys have their own special format. In short, they look like this:

```
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC
E9Uli8bGnD4hOWdeo5KKQJ/P/vOazI4MgqJK54w37emP2JwOAOdMmXuwpxbKng3KZz27mz+nKW
IlXJ3rzSGMo= P-256@localhost
```

If you'd like to learn more about that (`id_rsa.pub` , `id_ecdsa.pub` , etc), this should both whet your whistle and quench your thirst:

- The SSH Public Key Format (https://coolaj86.com/articles/the-ssh-public-key-format/)

And you may also enjoy SSH Fingerprints Explained (https://coolaj86.com/articles/ssh-pubilc-key-fingerprints/).

The OpenSSL smorgasbord

Oh man... people just name OpenSSL keys anything.

The conventions are plentiful and kinda inconsistent. However, they're mostly used for either HTTPS or application-level cryptography and a couple of common themes have emerged:

- `privkey.pem`
- `key.pem`
- `example.com.key`
- `priv.key`
- `key.der`

Since Let's Encrypt it's become more popular to name the private key `privkey.pem` , and I'm a big fan of that convention (and, as such, I've made it the default for Greenlock.js (https://git.coolaj86.com/coolaj86/greenlock-express.js)).

OpenSSL Public Keys

If you're actually using OpenSSL for SSL (now known as TLS), you don't really have the concept of a "public key" as such. It's not its own *thing* per say.

When you create a Certificate Signing Request (CSR), which lists the domains you intend to secure you must supply your private key the tool doing the signing.

It will then extract the public key and embed it in the CSR, which is signed, returned to you, and later verified by your web browser against your private key.

That file is usually named something like this:

- `csr.pem` (the intermediary before you get a cert)
- `cert.pem` (Let's Encrypt convention)
- `fullchain.pem` (when including authority chains)
- `example.com.crt`
- `example.com.pem`

(sidenote: if you're interested in how I reverse-engineered CSR to create small libraries to handle it instead of the typically *HUGE* ones, I talk a little bit in CSR, My Old Friend (https://coolaj86.com/articles/csr-my-old-friend/) and ASN.1 for Dummies (https://coolaj86.com/articles/asn1-for-dummies/), which is maybe too light on the direct subject but hopefully at least entertaining)

OpenSSL Public Keys Reprise

In the non-ssl cases where you're actually using raw public keys they look like this:

```
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEIT1SWLxsacPiE5Z16jkopAn8/+85
rMjgyCokrnjDft6Y/YnA4A50yZe7CnFsqeDcpnPbubP6cpYiVcnevNIYyg==
-----END PUBLIC KEY-----
```

Again I'll reference ASN.1 for Dummies (https://coolaj86.com/articles/asn1-for-dummies/) if you're interested to know what all that gobbledygook means.

I don't know what the most common conventions are for these public keys, since they're largely application specific but I like to call mine `pubkey.pem`, sometimes with something extra to designate the type, like `pubkey-ec-p256.pem`.

OpenSSL private keys are typically A file in `id_rsa` or `id_ecdsa` (without the `.pub`) is the private key.

SSH Private keys (`id_rsa`) are stored in one of the standard OpenSSL formats.

RSA vs EC / ECDSA

If the suject of the differences between RSA and EC piques your (and you found the format of this article and my wirting style to be palatable enough), I'll suggest something else with which to chase this all down:

- RSA vs ECDSA ()

By AJ ONeal

If you loved this and want more like it, sign up!

E-mail address | Get Weekly Code Dives