# Key Generation

## How to Generate a Secure Private Key

The security and privacy of your SAML deployment depends on the security of the private keys used for message-level signing and encryption, as well as the keys used to create secure back channels for transporting SAML messages over TLS. The corresponding public keys are bound to Signing and Encryption Keys, as discussed in the Key Usage topic. See the TLS Server Certificates topic regarding keys and certificates used for browser-facing TLS.

**Prepare to Generate a New Private Signing Key!**

Before generating a new private signing key for your IdP, read the IdP Key Handling topic.

A private key used for message-level signing and encryption is necessarily an online key, that is, it must be available to the SAML software at runtime. An online key may be encrypted, but the password or passphrase used to decrypt the key generally has to be available in an unencrypted file so that the SAML software can be restarted in unattended fashion. Therefore an online key is considerably more vulnerable than an offline key, and must be protected accordingly. In particular, a private key stored in the file system as an ordinary file should have strict permissions to prevent unauthorized copying.

Develop a strategy for securing a private key **before** you generate it. For instance, the following strategy is highly recommended:

1. Start with a secure system for your IdP or SP…and keep it that way!
2. Generate the private key directly on the secure system
3. Prevent the private key from ever leaving the secure system
4. Ensure ongoing access to the private key is strictly controlled

If you generate the private key on any other system, then *that* system must also be secure. Indeed, every system the private key comes in contact with must be secure—at least as secure as the target system—or the private key must be encrypted at rest. Moreover, the private key must be encrypted while in transit to the secure system. All in all, that is *much* more work (and error-prone), so the best advice is don't do it. Generate your private keys on the target system (IdP or SP) in the first place.

Here's another way to state these **basic security requirements**:

- *Until the private key is securely stored on the target system (IdP or SP), it needs to be encrypted, both at rest and in transit*.
- *Under no circumstances should an unencrypted private key come to rest on an insecure system or transit the network over an unprotected channel*.

It is easy to generate a private key and a corresponding long-lived, self-signed certificate with OpenSSL. On a linux system, type:

```
$ /usr/bin/openssl req -new -x509 -newkey rsa:2048 -keyout key.pem \
    -days 3650 -subj "/CN=hostname.example.org" -out cert.pem
```

The above command will store the private key in file key.pem and the corresponding public key certificate in file cert.pem. The latter is ultimately added to SAML metadata.

**Test your OpenSSL software installation**

OpenSSL is a subtly complicated tool having many versions with various capabilities (and bugs). It is recommended that *all OpenSSL commands be tested in advance* to ensure that the tool is functioning as expected.

When you issue the above OpenSSL command, you will be prompted to enter a pass phrase for the purpose of decrypting an encrypted private key. If you're generating the private key directly on the target system, it is not necessary to encrypt the private key (as discussed above). Simply press return when prompted to enter a pass phrase or use the `-nodes` option in the command above to issue an unencrypted private key straightaway.

If, however, you're generating the private key on any other host, you must encrypt the private key as stipulated earlier. Once the private key has been secured on the target system (IdP or SP), it may be decrypted *in situ* with the following OpenSSL command:

```
$ /usr/bin/openssl rsa -in key.pem -out key.pem
```

Simply press return when prompted to enter a new pass phrase.