NativeSPMultipleCredentials

There are a few typical reasons for supplying the SP with multiple keypairs, or "credentials", to use for authentication and encryption:

- separating signing/TLS and encryption keys
- key "rollover"
- federation or partner-dictated requirements to use particular certificates

These scenarios are discussed in more detail below, with examples explaining how to configure the SP in each case. In each scenario, the common factor is the use of a "Chaining" `<CredentialResolver>` to "wrap" two or more individual resolvers (usually of type "File") and enable each credential to be loaded and used. The varying part pertains to how credentials are labeled so that the selection process works as intended in each case.

Note also that the use of encryption in this context is specific to XML Encryption, and for the present, limited to use of SAML 2.0.

Separate Signing and Encryption Keys

This is the simplest (but also not all that common) case, and is handled by labeling the individual resolvers in the chain with a `use` property, typically of "signing" or "encryption".

```
<CredentialResolver type="Chaining">
     <CredentialResolver type="File" key="signing.key" certificate="signing.crt" use="signing"/>
     <CredentialResolver type="File" key="decrypt.key" certificate="decrypt.crt" use="encryption"/>
</CredentialResolver>
```

For this use case, no additional labeling or configuration should be necessary to get the right key selected.

Key Rollover

Typically with rolling over keys or certificates you have two separate concerns: migrating your signing/authentication keys and your encryption keys. On the authentication side, the solution is to publish the new key/certificate in your metadata and wait for that to propagate before switching your SP from the old credential to the new one. In that scenario, you need not ever configure both within the SP itself (i.e., it's different from this topic's goal).

On the other hand, with encryption you have the opposite problem. Once you publish a new key in your metadata, your peers may start using it but not all of them will switch at the same time. So in this case, you **MUST** configure both the old and new credentials into the SP so that either key can be available for decryption.

It's likely the case that both of these scenarios apply; often, a single credential is used for both signing and encryption and the migration is from one credential for both to another for both.

Therefore, combining these issues, if you want to perform rollover of a single keypair for both signing and encryption, you can follow these steps:

1. Add a second private key to your SP configuration explicitly as a decryption key (`use="encryption"`).
2. Publish the corresponding public key in your metadata and wait for propagation.
3. Switch the encryption usage constraint in the SP configuration from the new key to the old key.
4. Remove the old public key from the metadata and wait for propagation.
5. Remove the old private key from your SP configuration.

As an example, if you start here:

```
<CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
```

Then the configuration after step 1 might be:

```
<CredentialResolver type="Chaining">
     <CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem" use="encryption"/>
     <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
</CredentialResolver>
```

After step 3:

```
<CredentialResolver type="Chaining">
     <CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem"/>
     <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem" use="encryption"/>
</CredentialResolver>
```

And after step 5:

```
<CredentialResolver type="File" key="new-key.pem" certificate="new-cert.pem"/>
```


The above examples are not meant to be taken literally. If your files go by different names, live in non-default locations, then you will obviously need to adjust. Also take care as to whether one or both of your private keys has been encrypted on disk. You may need to supply a `password` attribute in your elements to load the key. In all cases **CHECK YOUR LOGS** any time you are manipulating keys. You **MUST** ensure that all keys are loading correctly and that no errors are being logged during normal use.

The above process is suitable for cases in which the metadata's `<md:KeyDescriptor>` elements do not carry a `use` XML attribute and there is no opportunity to introduce such a `use` attribute into metadata. Other approaches may be more suitable for non-Shibboleth implementations.

Ultimately, the chosen process depends on how much control you have over your metadata. In many scenarios, the metadata is controlled by a 3rd party (such as a federation) so the SP operator's choices are often limited.

Multiple Certificate Scenarios

The most complex (and ill-advised) case is if you're trying to selectively apply different keys or certificates when interacting with different relying parties. The primary advice on this is "don't". Unfortunately, this isn't always possible, as some federations insist on imposing PKI-based constraints on their members. To accomodate this, the SP includes a mechanism for attaching "names" to credentials and then referencing the credentials in a `<RelyingParty>` element.

Assuming a scenario in which you want to use one credential by default, but a second credential when dealing with a particular relying party, you will typically do the following:

1. Configure both credentials together in a chain.
2. Add one or more `<RelyingParty>` elements in the appropriate spot with a `keyName` property that matches the "CN" from the desired credential's certificate subject (or that matches a subjectAltName).

Example using certificate subject as keyName

```
<ApplicationDefaults ...>
    ...
    <Errors .../>
    <RelyingParty Name="https://idp.example.org/idp/shibboleth" keyName="trusted.example.org"/>
    ...
    <CredentialResolver type="Chaining">
        <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
        <CredentialResolver type="File" key="trusted-key.pem" certificate="trusted-cert.pem"/>
    </CredentialResolver>
</ApplicationDefaults>
```

If you find that each candidate credential shares essentially the same certificate subject information, then you can use a locally-chosen name in your `<RelyingParty>` element and add the same value to a `keyName` attribute or `<Name>` element in the `<CredentialResolver>`.

Example using locally chosen keyName

```
<ApplicationDefaults ...>
    ...
    <Errors .../>
    <RelyingParty Name="https://idp.example.org/idp/shibboleth" keyName="Special"/>
    ...
    <CredentialResolver type="Chaining">
        <CredentialResolver type="File" key="sp-key.pem" certificate="sp-cert.pem"/>
        <CredentialResolver type="File" key="trusted-key.pem" certificate="trusted-cert.pem" keyName="Special"/>
    </CredentialResolver>
</ApplicationDefaults>
```