Migrating to SHA-2

> **Deprecated**
>
> Note that this page has been deprecated. The information it contains is no longer current.

Migrating to SHA-2

As of January 1, 2014, NIST disallows the use of the SHA-1 digest algorithm in conjunction with digital signatures (see: NIST SP 800-57 Part 1, Revision 3, July 2012, Tables 3 and 4). This was a major driver behind the Phase 1 Recommendations of the Metadata Distribution Working Group, which gave rise to the Phase 1 Implementation Plan, a major effort to phase out the use of SHA-1 in InCommon metadata that began on December 18, 2013 and ended on June 30, 2014.

> As of June 30, 2014, *all metadata aggregates distributed by InCommon are signed using the SHA-256 digest algorithm*. Because of this, we know that many of the SPs in the InCommon Federation support SHA-256.

Like metadata, SAML messages may be digitally signed for authenticity and integrity. Any SAML entity (IdP or SP) may sign a message but in practice it is the IdP that wields a signing key since SAML assertions issued by the IdP MUST be signed according to the SAML spec. The SP uses the IdP's public signing key in metadata to verify the signature on the assertion just-in-time, that is, when the assertion is presented to the SP by the browser user.

It is believed that most IdPs in the InCommon Federation are signing assertions using the SHA-1 digest algorithm. This is because most IdP deployments use the Shibboleth IdP software, which is known to support SHA-1 only, at least out of the box.

To enable SHA-2 support in the Shibboleth IdP, you can use a 3rd-party extension to enable SHA-2 capability for Shibboleth IdP versions 2.3 and later. Note that the resulting Shibboleth configuration is all or none, either SHA-1 or SHA-256.

> **Recommendations for Shibboleth IdPs**
>
> Sites running the Shibboleth IdP software should deploy a test IdP with SHA-2 support as described in the next section and then migrate to that test environment as appropriate.

In comparison, SimpleSAMLphp has good support for signing assertions using the SHA-256 digest algorithm. It can be configured to sign assertions on a per-SP basis, that is, a simpleSAMLphp IdP can sign assertions using the SHA-1 digest algorithm for some SPs and the SHA-256 digest algorithm for others.

> **Recommendations for simpleSAMLphp IdPs**
>
> Sites running the simpleSAMLphp IdP software can and should migrate to SHA-256 as soon as possible. A perfectly reasonable strategy would be to configure the use of SHA-256 by default but to fall back to SHA-1 for those few remaining SPs (perhaps external to InCommon) unable to handle SHA-256.

A Migration Strategy for Shibboleth IdPs

This section describes how to migrate a Shibboleth IdP from SHA-1 to SHA-256.

1. Evaluate the use of back-channel protocols on your production IdP with an eye towards eliminating unused protocols and endpoints. Phase out seldom-used protocols if possible. An optimally configured IdP will support SAML2 on the front channel only.
2. **Deploy a test IdP**. Configure this test IdP to be nearly identical to your production IdP (same entityID, same metadata sources, same attribute release policy, etc.).

   > Your test IdP should have *the same entityID* as your production IdP so that the two are indistinguishable by relying parties (such that the two really are **one logical IdP**). Consequently, a single entity descriptor in metadata is sufficient to describe both IdPs. Any SP that consumes that metadata will interoperate with either your test IdP or your production IdP.

   There are at least two deployment options:

   a. *Deploy the test IdP on the same host*. In this case, the endpoint locations of the test IdP will have the same hostname but a different path. This is perhaps the simplest option since then the production IdP and the test IdP can easily share the same signing key. (In this scenario, the test IdP is really an extension of the production IdP environment.)
   b. *Deploy the test IdP on a different host*. In this case, the endpoint locations will have a different hostname but the same path as the production IdP. One option is to copy the production signing key onto the new host (without exposing that key of course). Another option is to use a new signing key (which should be no less secure than the production signing key). The certificate corresponding to this new signing key may be added to the IdP's entity descriptor in metadata so that there are two certificates in metadata, one for the production IdP and one for the test IdP.
3. Exercise the test IdP. There are at least two test scenarios depending on how the test IdP is deployed:
   a. Using IdP-initiated SSO on the test IdP, systematically push SAML2 assertions to endpoints at select partner SPs.
   b. If the test IdP is deployed on a different host, map the IdP domain name (in metadata) to the IP address of the test IdP using /etc/hosts on a client machine. Using SP-initiated SSO, systematically test select partner SPs using the client machine.

4. Install a 3rd-party extension on the test IdP only. The extension allows you to change the signature/digest algorithm that the IdP uses to sign assertions. Configure the test IdP to sign assertions using the SHA-256 digest algorithm.
5. Repeat step 3. This round of testing may uncover SPs that are not compatible with the SHA-256 digest algorithm.

If you encounter a partner SP that is not compatible with SHA-256, with no hope of upgrading, you can work around the incompatibility by deploying an IdP Proxy. Initially, the SP component of the IdP Proxy is integrated with your test IdP and both are configured to sign assertions using the SHA-256 digest algorithm. The IdP component of the IdP Proxy is integrated with the partner SP and both are configured to sign assertions using the SHA-1 digest algorithm.

Note: If the partner SP does not support message-level encryption, the IdP Proxy can compensate for that shortcoming as well.