Third Party Data Digital Signature Requirements
September 2021

Why does Figure want a digital signature on third party data?

- When Figure Lending funds a loan, we board the loan onto Provenance (https://provenance.io/). Provenance is a blockchain ecosystem, which enables the buying and selling loans, including the audit/validation of the loan authenticity.
- Figure will include third party data used to originate a loan in the loan packet provided to Provenance along with data specific to the underwriting decision, loan documents/disclosures and any other critical documentation.
- In order to ensure accuracy of data, Figure will only provide third party data that includes a digital signature from the originating party. Key data points, such as credit or property data used in the underwriting decision, will be individually validated and verified back to the digitally signed sourced data.
- The purpose of validating the data source is to allow users of Provenance, including investors, loan buyers, auditors, to verify that the data used in the decision of a loan is in fact the data provided at the point and time the originator requested the data.

What does Figure consider a digital signature?

- Figure considers a digital signature an electronic representation or confirmation that the data provided from the third party came directly from the source which can be independently verified.
- A digital signature should only be a couple extra lines of output attached to the existing responses third parties already provide that include the digital signature.

What are the vendor requirements?

- Customize existing responses to include a digital signature.
- Create and maintain a public and private key pair to cryptographically sign the data which will be used to validate the digital signature.

How to implement a digital signature?

- Step 1: Set up a public and a private key pair to be used for signing. The key type should be EC using the P-256 elliptic curve (asymmetric).
- Step 2: Send us the public key portion of the key pair and any chain of trust needed. Note, we will need this anytime the private key is updated.

- Step 3: Canonicalize your response data. Then, using the SHA256 with ECDSA algorithm, sign your canonicalized data with your private key. It is important that the format you currently send your data (e.g JSON, XML) be canonical prior to signing.
- Step 4: Send us the data as normal but include the signed data results in the response header.
- Note, as an alternative to developing in-house, Vault offers “crypto as a service.”

For example, here’s how a message would be signed and converted to a transportable signature that can be sent in the response header:

```
Signature ecdsaSign = Signature.getInstance("SHA256withECDSA");

ecdsaSign.initSign(privateKey);

ecdsaSign.update(responseData.getBytes("UTF-8"));

byte[] signature = ecdsaSign.sign();

String transportableSignature = Base64.encode(signature);
```

Provenance will then verify the signature in the response header:

```
String signature = Base64.decode(transportableSignature);

Signature ecdsaVerify = Signature.getInstance("SHA256withECDSA");

ecdsaVerify.initVerify(publicKey);

ecdsaVerify.update(responseData.getBytes("UTF-8"));

boolean result = ecdsaVerify.verify(signature);
```