Notary Trust Pinning Design

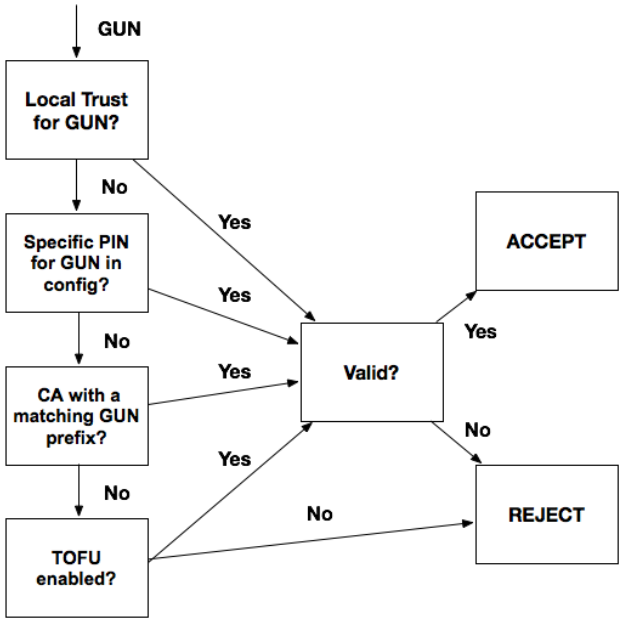
(living document)

Trust Pinning with Notary

By default notary operates in a TOFU (Trust On First Use) model: if a repository has never been seen before, notary will download the `root.json` file over HTTPS, and that will be the root of trust going forward. This behavior is configurable, and can be disabled by adding a `TOFU: false` under the `trust_pinning` section of `notary.json`.

There are two ways to bootstrap trust with notary outside of using TOFU: pin to a specific certificate/public key and/or pin to a specific set of CAs. All of those can be configured under the `trust_pinning` section of `notary.json`.

The order of priority of checking trust for a specific GUN in notary works as follows:



This means that notary operates in the following fashion:

- if a user already has prior trust locally for a GUN, that will be what notary will check for continuity.

    - **Validity check:** first we ensure that certificates exist for this GUN in an uncorrupted certificate store. We then verify the root against these certificates' keys. We then re-add the certificates, which will check expiry times, and then check that the root's data is signed by public keys from these certificates.

- If there is no local trust for a specific GUN, notary will access the configuration file and attempt to match the root certificate/root key inside of the `root.json` with one of the entries under `Certs` in `notary.json`.
    - **Validity check:** given we have a match under the `Certs` key for our GUN, we can validate the root against the key for this fingerprinted cert, add the cert to our certificate store which will also ensure it is not expired. We then check that the certificate's public key produced valid signatures for the given root we are trying to validate.

- If there is no specific pin in `Certs` for this GUN, notary will then check to see if there is a set of `CA` that have been configured as an owner of this space, by checking the prefix of every CA entry against the GUN being accessed. This entry will map to a certificate bundle of one or more root CA certificates
    - **Validity check:** given we have a prefix match under the `CA` key for our GUN, we load the specified filepath for the `CA` PEM to ensure it is the proper format and validate each expiry and key length in the cert bundle. We add the `CA` cert(s) to our certificate pool, and then only consider specified certificates that verify against our certificate pool (potentially with intermediate certificates that are included in the certificates we are validating against the CA certs) when adding to our certificate store. We then check that the added certificates' public keys produced valid signatures for the given root we are trying to validate.

- If all of the above fail, notary will finally check if `TOFU` is enabled or disabled, and reject or accept downloading this repository accordingly.
    - **Validity check:** if `TOFU` is enabled, notary will simply attempt to verify all new certs against themselves and add all of them to the certificate store. We then check that the added certificates' public keys produced valid signatures for the given root we are trying to validate.

Here's the structure of the `trust_pinning` section under `notary.json`:

```
trust_pinning: {
      ca: {
            <GUN prefix>:Path to CA bundle file
      },
      certs: {
            <GUN>: CertID/Public Key ID
      },
```

```
      disable_tofu: true/false
}
```

Operational Notes

System administrators interested in ensuring their production hosts only operate on trusted data are assumed to have the ability of pre-distributing a `config.json` file (and the necessary CA PEM files) to all the hosts that are currently operating the docker command-line.