Asset certification

A user interacting with the Internet Computer needs to be able to confirm that the responses they receive are actually coming from the Internet Computer and have not been tampered with. Traditionally, on the Internet, this problem is solved using public-key cryptography. The server running the service has a secret key and uses that to sign all its responses. A user can then verify the signature on the response using the server's public key.

Just like a web server in Web2 maintains a public-key/secret-key pair, the Internet Computer blockchain as a whole maintains a public-key/secret-key pair. Additionally, each individual subnet in the Internet Computer also maintains its own public-key/secret-key pair. When a new subnet is formed, the NNS issues a certificate for the subnet which contains a signature of the subnet's public key with the Internet Computer's public key. When the subnet responds to a user's message, the response contains a certificate chain, which includes a signature on the response by the subnet's public key and the certificate issued by the NNS to the subnet. The user can verify the certificate chain using the Internet Computer's public key similar to verifying a certificate chain in Web2.

Each blockchain node shares only a piece of its subnet secret key. As a result, each node is incapable of signing a message by itself. But if at least 2/3rd of the nodes of a subnet agree on a message, they together can combine their secret key pieces to sign the message. The signed message can be verified easily using the subnet's public key. If the verification succeeds, it means that at least 2/3rd of the blockchain nodes running the canister agreed to deliver that message. The technology used by the Internet Computer to generate and maintain the secret key shares, and sign messages using the secret key shares is called **chain-key technology**.

The Internet Computer supports two types of messages: Query calls and Update calls. Query calls are similar to HTTP `GET` requests and do not modify the state of the Internet Computer. The query calls do not go through the consensus protocol. The user can make a query call to any blockchain node in the subnet, and only that (possibly malicious) blockchain node answers the query. As generating a certificate requires consensus from at least 2/3rd of the nodes of the subnet, the Internet Computer doesn't issue a certificate when responding to query calls.

For efficiency reasons, the canisters deliver web pages to the client via query calls. However, as the client needs to verify the received content, the Internet Computer introduces the notion of **Certified Variables**. In a nutshell, a canister can a-priori choose to create a certificate for a piece of data and store it in the replicated state. Any user can later access the data along with its certificate via query calls.

We can use the notion of the certified variables to certify all the assets (HTML, CSS, Javascript files, images, videos, etc.) of an app a-priori. There are 2 ways of performing the asset certification. 1) The canister developer can explicitly write code to manage and certify all the assets. 2) The canister developer can create an "asset canister", by creating a canister with type set to "asset" and specifying the folder containing all the assets. The asset canister is a regular canister, except that the boilerplate code for managing and certifying all the assets is taken care of for us.

When a canister issues a response along with its certificate, a **HTTP Gateway** can be used to verify the certificate before passing on the response to the client.

For more information on certification, check **Certified Variables**.

Asset Certification Wiki Article

Rust Canister Development Security Best Practices