Certified variables & assets

June 10th, 2021

Roman & Hans

Agenda

- Replica certificates
- Certified variables
- Certificate validation

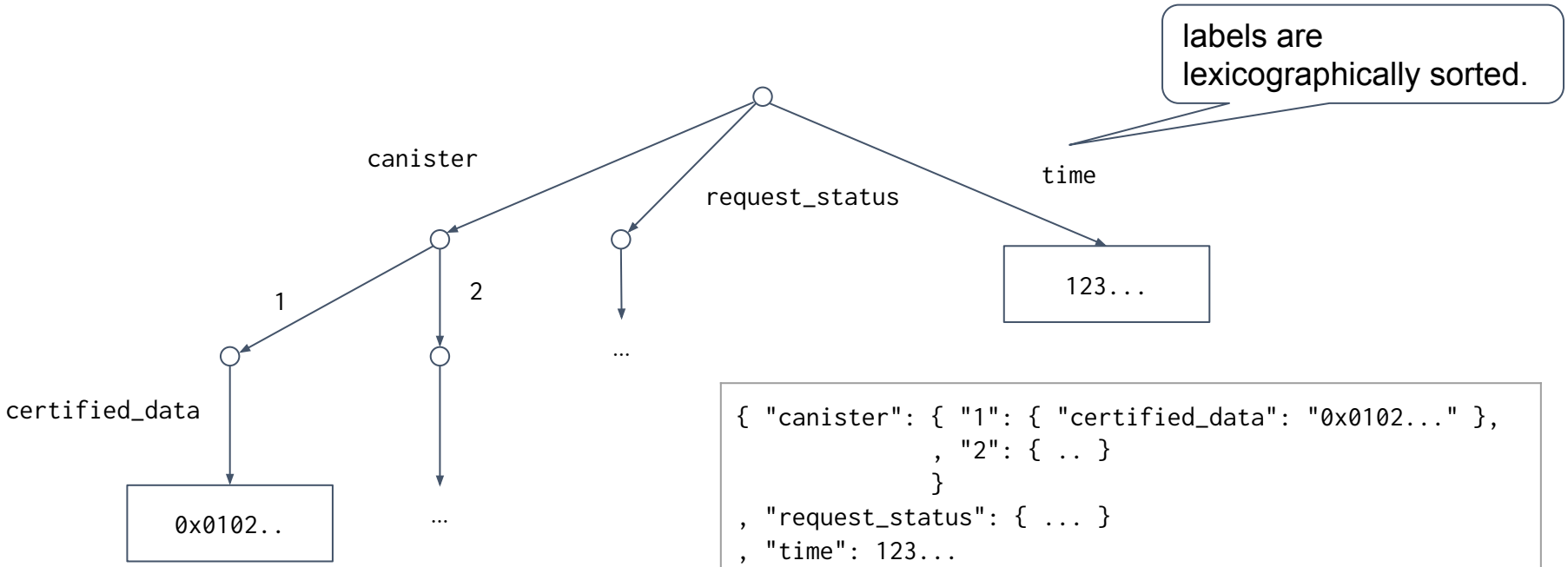
Replica certificates

- Data model
- Certification
- Proof structure

Data model

- Certification is based on an implicit contract between a service and a client. The client needs to know how the service represents the data.
- Replica uses labeled trees as universal data representation format.

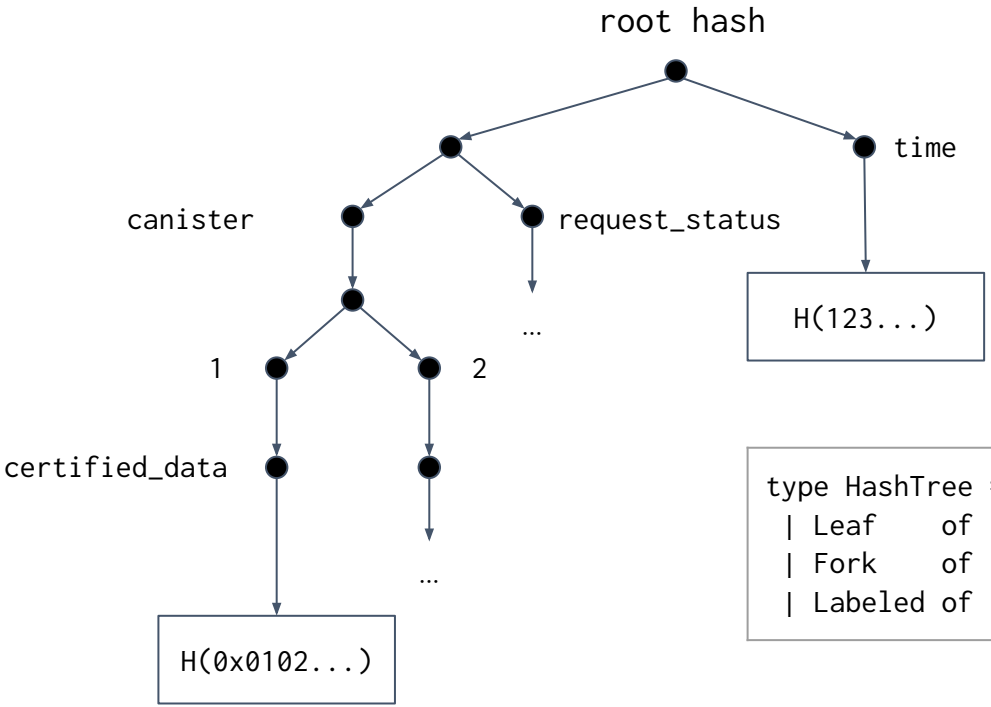
Data model: labeled trees




```
type Tree =
  | Leaf of bytes
  | Fork of (Label × Tree) list
```

```
{ "canister": { "1": { "certified_data": "0x0102..." },
                , "2": { .. }
                }
, "request_status": { ... }
, "time": 123...
}
```

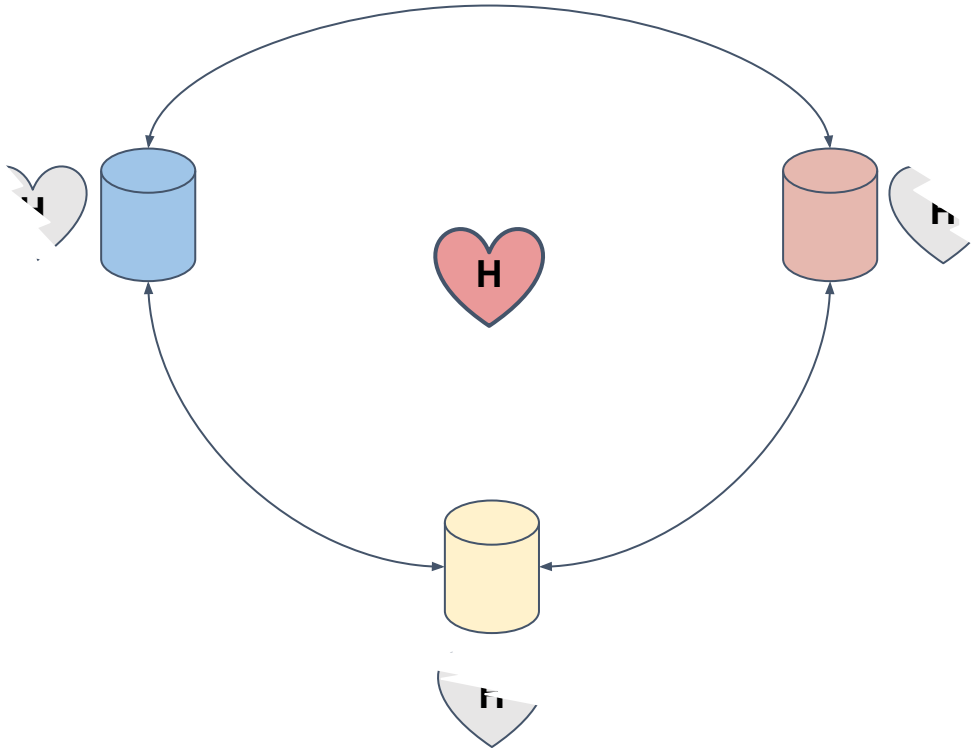
Data model: hash trees



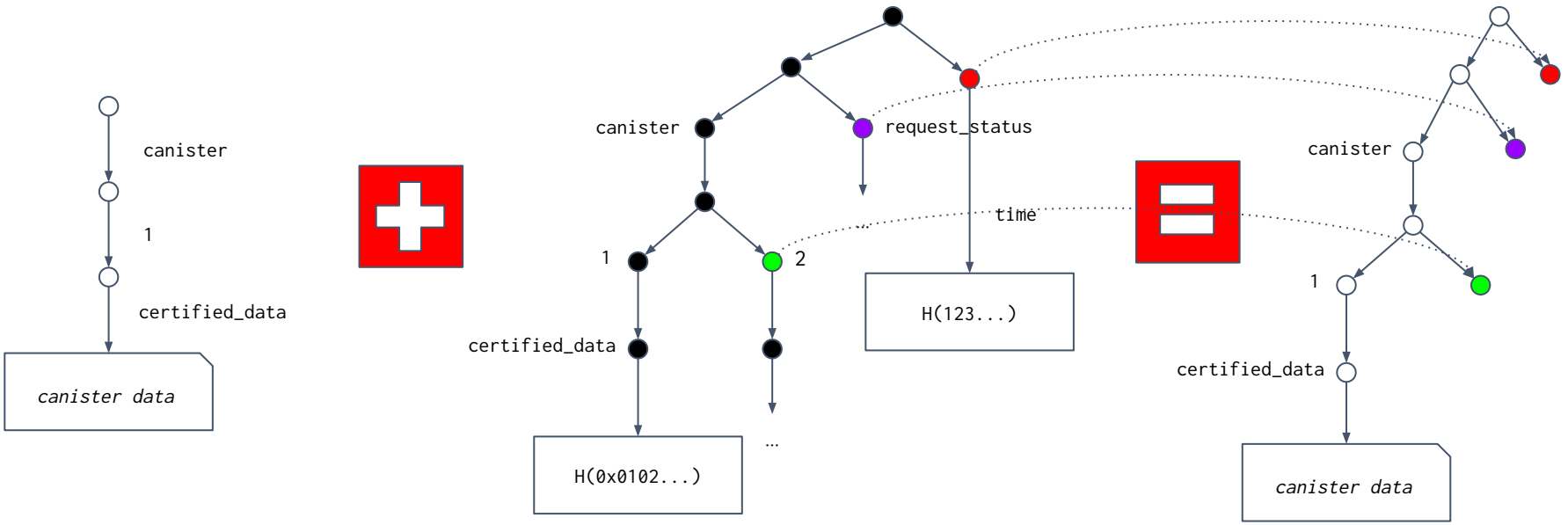

```
type HashTree =
 | Leaf    of  Hash
 | Fork    of (Hash × HashTree × HashTree)
 | Labeled of (Hash × Label    × HashTree)
```

Certification

Replicas sign the root hash they computed and exchange their shares. Eventually they reach consensus and obtain a subnet signature on the hash.

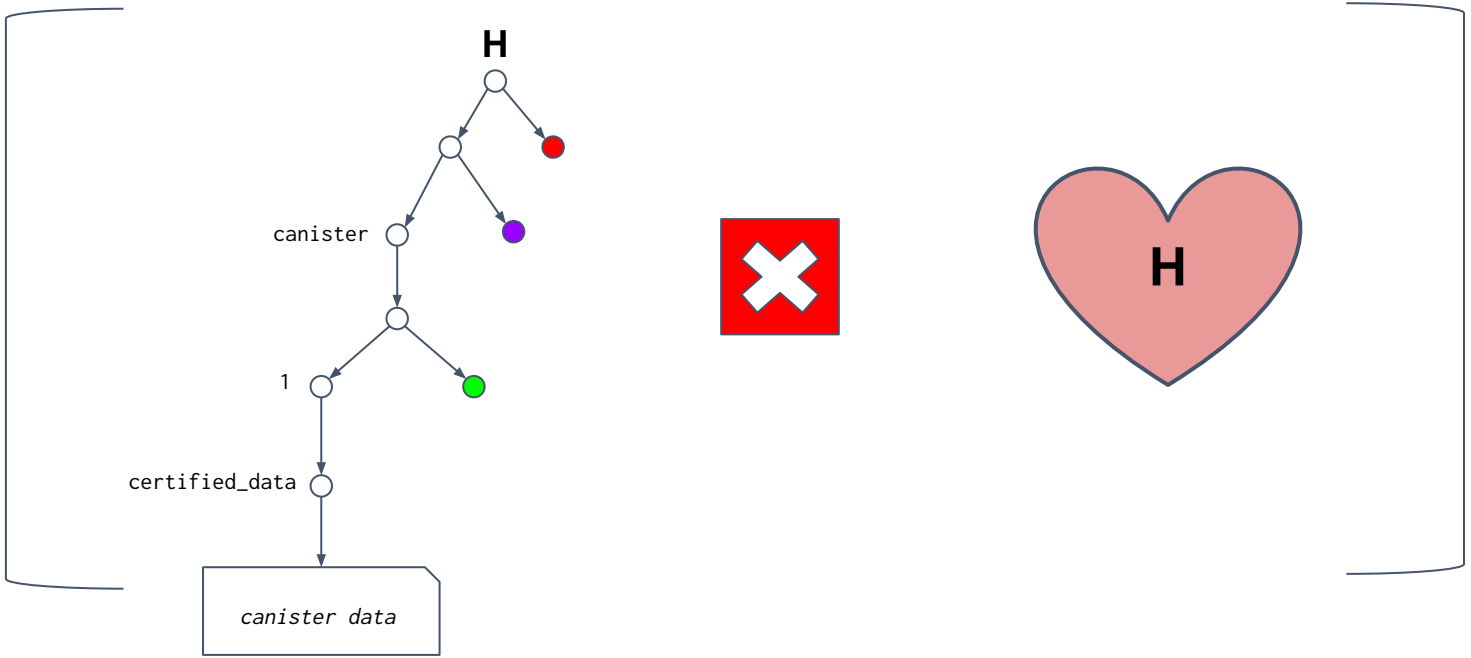


Proof structure




```
type Proof =
  | Leaf   of bytes  | Fork    of (Proof × Proof)    | Empty
  | Pruned of Hash   | Labeled of (Label × Proof)
```

Replica certificate



Certified variables

- Using certification in a canister
- Pros and Cons of certified variables
- Example: certified assets canister

Rust canister API

- set_certified_data : bytes -> ()

  This function should be called from init/post_upgrade/update every time the certified structure changes.

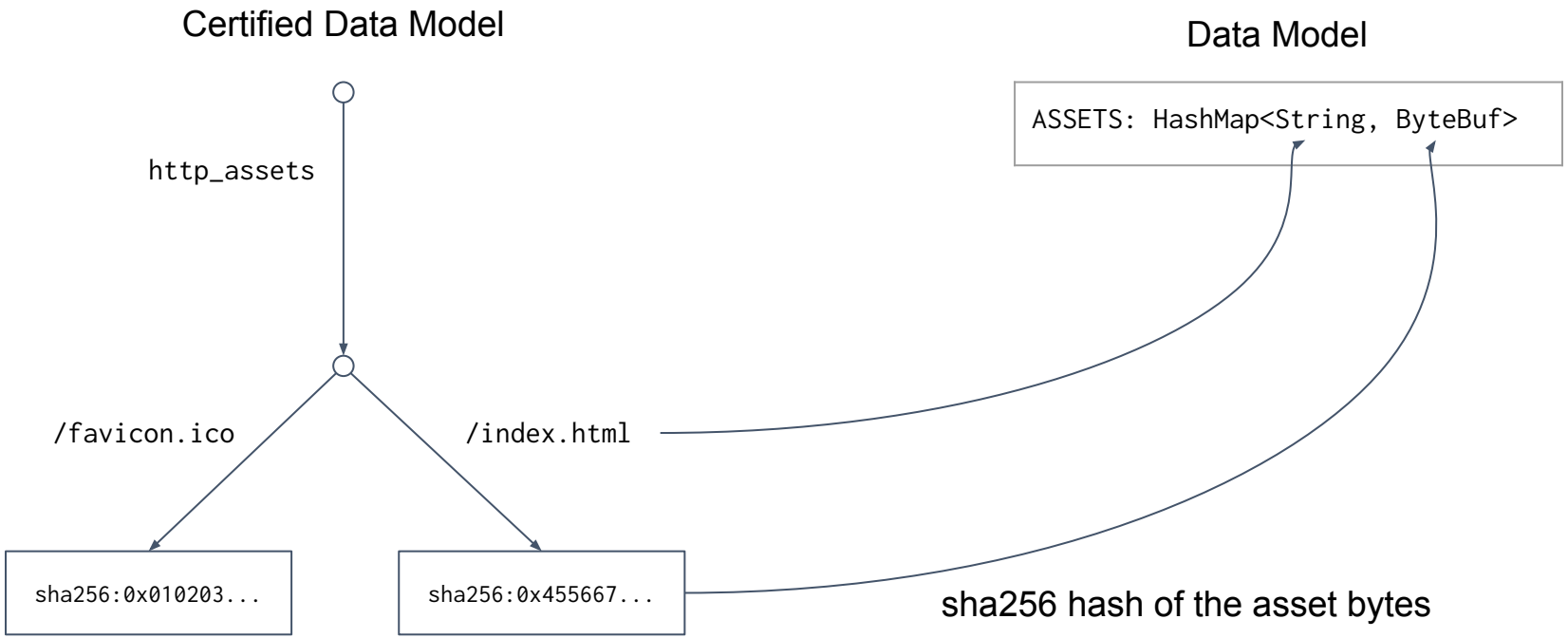
- data_certificate : () -> ?bytes

  This function can be called from a query call to obtain a certificate for the data set by the function above.

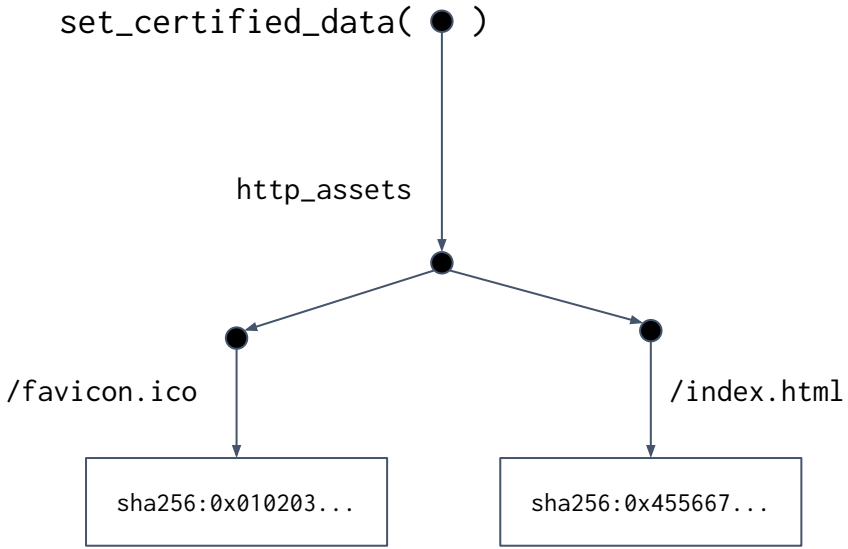
Minimal example: certified counter

```
thread_local! { static COUNTER: Cell<u64> = Cell::new(0); }
struct CertifiedCounter {
  certificate: ByteBuf,
  value: u64,
}

#[update]
fn increment() -> u64 { COUNTER.with(|c| {
  let value = c.update(|v| v + 1);
  set_certified_data(value.to_le_bytes().to_vec());
  value
}}

#[query]
fn read() -> CertifiedCounter { COUNTER.with(|c| {
    CertifiedCounter { certificate: data_certificate().into(), value: c.get() }
}}
```

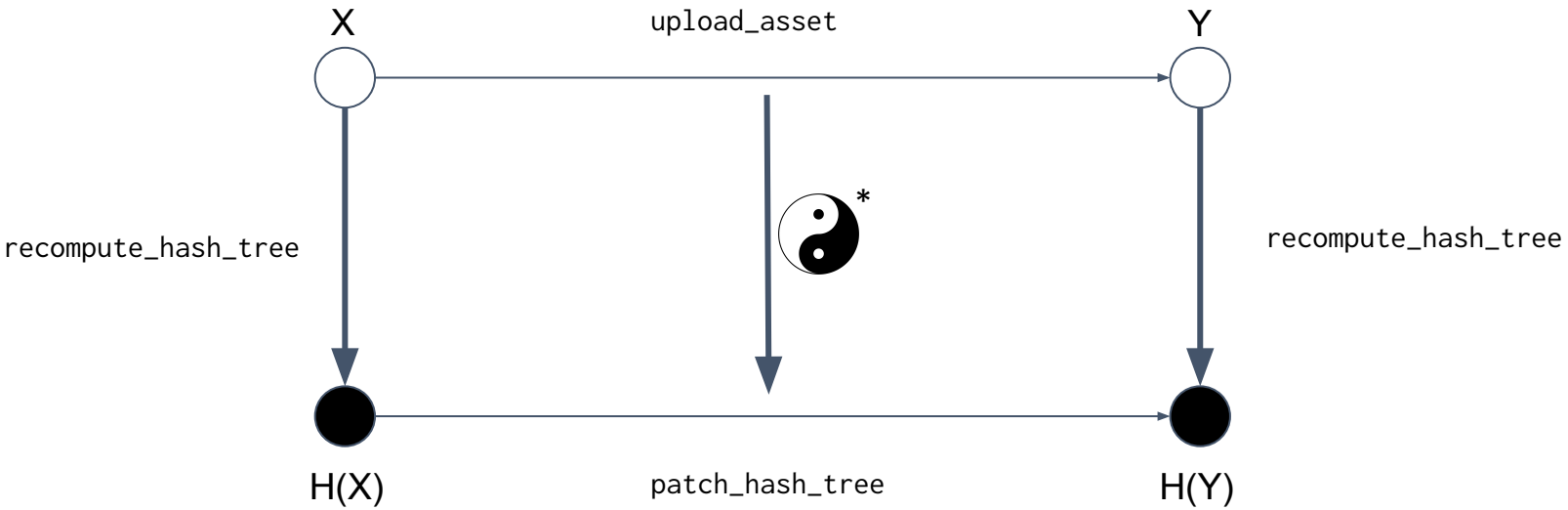
Example: certified assets canister



Example: certified assets canister



Obligatory CT diagram



* the ~~magic~~ science of [ic-certified-map]

Pros

- Very data & latency efficient: requires only a single query to any subnet replica.

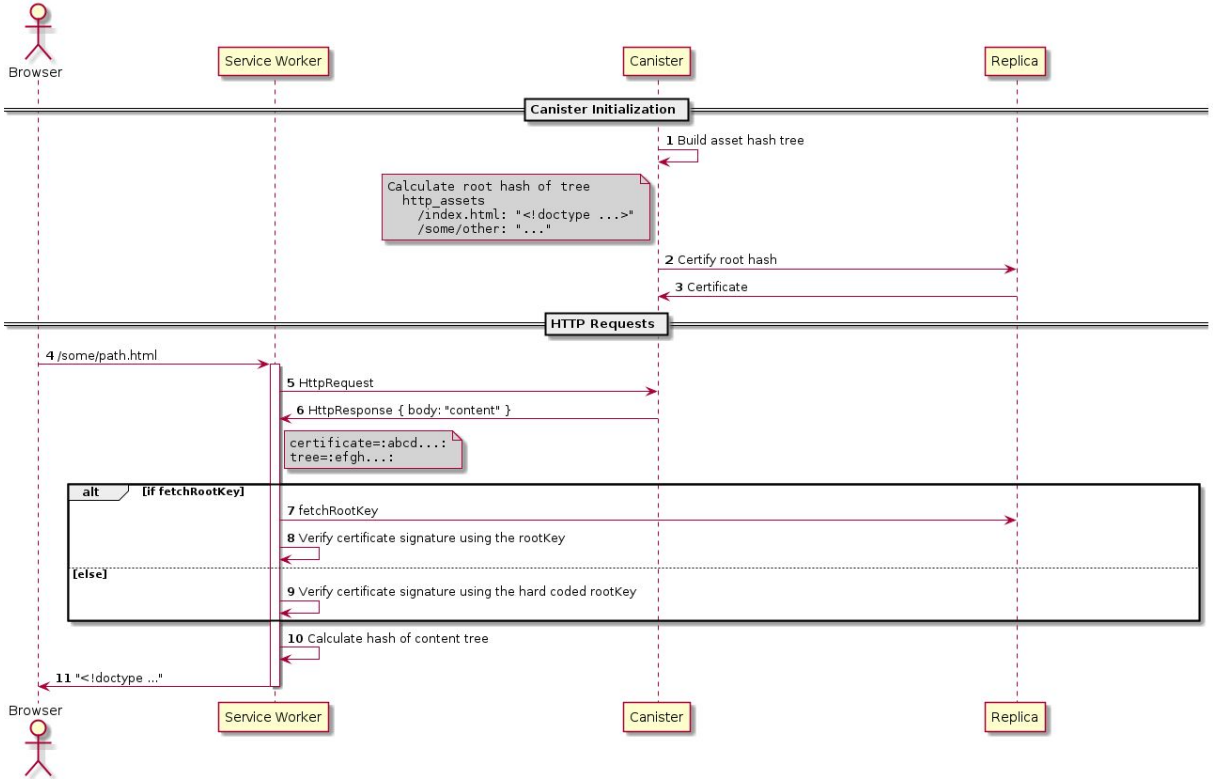
Cons

- Requires custom client code to verify responses.
- Can only be used for pre-computed queries. E.g., doing complex search filtering is **hard**.
- Requires writing complex code in the canister.
- Might be hard to add post-factum.

Certificate validation

- Certifying service worker

Sequence


Browser
Service Worker
Canister
Replica
Canister Initialization
1 Build asset hash tree
Calculate root hash of tree
http_assets
/index.html: "<!doctype ...>"
/some/other: "..."
2 Certify root hash
3 Certificate
HTTP Requests
4 /some/path.html
5 HttpRequest
6 HttpResponse { body: "content" }
certificate=:abcd...:
tree=:efgh...:
alt
[if fetchRootKey]
7 fetchRootKey
8 Verify certificate signature using the rootKey
[else]
9 Verify certificate signature using the hard coded rootKey
10 Calculate hash of content tree
11 "<!doctype ..."
Browser
Service Worker
Canister
Replica

Real World Example

```
curl -v https://h5aet-waaaa-aaaab-qaamq-cai.raw.ic0.app/index.html
```

ic-certificate:

certificate=:2dn3o2R0cmVlgwGDAYMBgwJIY2FuaXN0ZXKDAYMBgwGCBFggVO3kr4k8PoXhhbdIAjyy+Cs3LDBppKcWWOYKyc8pyLmDAYMBggRYIFooON6bZuXXuPk2SSNSDUG5j/l0q1
THMjVc8EepzpQIgwGDAYMCSgAAAAAAMAAZAQGDAYMBgwJOY2VydGlmaWVkX2RhdGGCA1ggPOAA4amBBMT2tGVabZxQglj96hG0RdDviKJSFes1of6CBFgg5v2+8RBKfktGILoTd6USC+e3L
rqBnEant7cvfYimU/+CBFggcl7xzo7+r66dN/NdP5l/D5FEQaEWiYCYcOSb+umDvQyCBFggUagRifP09v6KIDfpjZb/0c6rMkTLLhIGh2Wkj1JtdciCBFggrotk2XHArmb8zC6c3DrBYGkL
iRfH2r/ETWcDRrBkwsCCBFgg+hhvBWFKnZs06dh45WndmuvNxYe4Puvn0vgUwUWUnFmCBFggyO6FcG/5xyWh1+X1ps3RDIuVIkctPikUZ+fuK8EM/YiCBFggW9xw1UmN7+aVEqNhHUPoq1s
Kac0BpOBSrkwgdO3ARpaCBFgg8K1NwCiEu0jnwbwCD/Ixt7Gt5WScdROzaQi7V3/Kl46CBFggVfgeOlPvMLAK9Gkr8VJXNlJe1p7eIeleuiNAFMy2TFODAYIEWCDEi1UT6xPndn773moeEz
rxK0NghCCz/i+oXBFgzrEBNoMCRHRpbWWCA0nUu8j6suO+wxZpc2lnbmF0dXJlWDCzRhoF8CZ4IGqH+xBnedBYl7fQ39SxLHrSEYCKsJ2sKgN0yIGINUh627ESw2Z4r/FqZGVsZWdhdGlvb
qJpc3VibmV0X2lkWB3DuW6kfgTIl3y3A/Far1qqr3PVlse+GaQ6cY2LAmtjZXJ0aWZpY2F0ZVkB2NnZ96JkdHJlZYMBggRYICXG82N6hZGEU21kauXHWe89C3rs18CFgCRT1K+bD/gfgwGD
AkZzdWJuZXSDAYIEWCB6ER+Aqjk2VmBS0bhRaOdRA/voxQHTD+Q4gY2a4sorFoMBgwGCBFggMlN6Y2hvfLE7b9smcWTBsi2nYB3Sm5wZcZy/jkRgK5GDAYMCWB3DuW6kfgTIl3y3A/Far1q
qr3PVlse+GaQ6cY2LAoMCSnB1YmxpY19rZXmCA1iFMIGCMB0GDSsGAQQBgtx8BQMBAgEGDCsGAQQBgtx8BQMCAQNhAI5nIKjxQp5SXDWAIaBcJxT4TYKDVONTNb6fqwir75fapVUoQscz75
gYhDiujVTHDgj9rVzSkbRyIiHsdVEuvTNWeKT3rIUidQVoQP1j3A2G9bT/81OSrRUowXrqdJJuCoIEWCDaC0dWOvOuDjt6RC0cZQK3Yyk9awvzSnTLuWICPs4bBIIEWCAg88BQjI6azuZet
TYraEncf1rfGmjrCtkdLKcECn2NO4MCRHRpbWWCA0ngm/u87NWowxZpc2lnbmF0dXJlWDCNY3ZcE70CIoZHaP397ITSSmVe+kNlSJ4t6ZdckYfRhktRNuHaNK0K1BFrjIFJOPc=:,
tree=:2dn3gwJLaHR0cF9hc3NldHODAYMBggRYIBhAvfRio7C3hUEXTSlqCS/sa6qCmtn6tn7PXgEWBCOigwJLL2luZGV4Lmh0bWyCA1ggHeLPBq1UcefbY+hCvzPZf/nLXa0Ocx3kFz16L
tMZySaCBFggTS4W0tdkYuF5hxF6DzzK4xsOGZRwKAEgLq107bApNr4=:

Real World Example

```
curl -v https://h5aet-waaaa-aaaab-qaamq-cai.raw.ic0.app/index.html
```

ic-certificate:

certificate=:2dn3o2R0cmVlgwGDAYMBgwJIY2FuaXN0ZXKDAYMBgwGCBFggVO3kr4k8PoXhhbdIAjyy+Cs3LDBppKcWWOYKyc8pyLmDAYMBggRYIFooON6bZuXXuPk2SSNSDUG5j/l0q1
THMjVc8EepzpQIgwGDAYMCSgAAAAAAMAAZAQGDAYMBgwJOY2VydGlmaWVkX2RhdGGCA1ggPOAA4amBBMT2tGVabZxQglj96hG0RdDviKJSFes1of6CBFgg5v2+8RBKfktGILoTd6USC+e3L
rqBnEant7cvfYimU/+CBFggcl7xzo7+r66dN/NdP5l/D5FEQaEWiYCYcOSb+umDvQyCBFggUagRifP09v6KIDfpjZb/0c6rMkTLLhIGh2Wkj1JtdciCBFggrotk2XHArmb8zC6c3DrBYGkL
iRfH2r/ETWcDRrBkwsCCBFgg+hhvBWFKnZs06dh45WndmuvNxYe4Puvn0vgUwUWUnFmCBFggyO6FcG/5xyWh1+X1ps3RDIuVIkctPikUZ+fuK8EM/YiCBFggW9xw1UmN7+aVEqNhHUPoq1s
Kac0BpOBSrkwgdO3ARpaCBFgg8K1NwCiEu0jnwbwCD/Ixt7Gt5WScdROzaQi7V3/Kl46CBFggVfgeOlPvMLAK9Gkr8VJXNlJe1p7eIeleuiNAFMy2TFODAYIEWCDEi1UT6xPndn773moeEz
rxK0NghCCz/i+oXBFgzrEBNoMCRHRpbWWCA0nUu8j6suO+wxZpc2lnbmF0dXJlWDCzRhoF8CZ4IGqH+xBnedBYl7fQ39SxLHrSEYCKsJ2sKgN0yIGINUh627ESw2Z4r/FqZGVsZWdhdGlvb
qJpc3VibmV0X2lkWB3DuW6kfgTIl3y3A/Far1qqr3PVlse+GaQ6cY2LAmtjZXJ0aWZpY2F0ZVkB2NnZ96JkdHJlZYMBggRYICXG82N6hZGEU21kauXHWe89C3rs18CFgCRT1K+bD/gfgwGD
AkZzdWJuZXSDAYIEWCB6ER+Aqjk2VmBS0bhRaOdRA/voxQHTD+Q4gY2a4sorFoMBgwGCBFggMlN6Y2hvfLE7b9smcWTBsi2nYB3Sm5wZcZy/jkRgK5GDAYMCWB3DuW6kfgTIl3y3A/Far1q
qr3PVlse+GaQ6cY2LAoMCSnB1YmxpY19rZXmCA1iFMIGCMB0GDSsGAQQBgtx8BQMBAgEGDCsGAQQBgtx8BQMCAQNhAI5nIKjxQp5SXDWAIaBcJxT4TYKDVONTNb6fqwir75fapVUoQscz75
gYhDiujVTHDgj9rVzSkbRyIiHsdVEuvTNWeKT3rIUidQVoQP1j3A2G9bT/81OSrRUowXrqdJJuCoIEWCDaC0dWOvOuDjt6RC0cZQK3Yyk9awvzSnTLuWICPs4bBIIEWCAg88BQjI6azuZet
TYraEncf1rfGmjrCtkdLKcECn2NO4MCRHRpbWWCA0ngm/u87NWowxZpc2lnbmF0dXJlWDCNY3ZcE70CIoZHaP397ITSSmVe+kNlSJ4t6ZdckYfRhktRNuHaNK0K1BFrjIFJOPc=:,

tree=:2dn3gwJLaHR0cF9hc3NldHODAYMBggRYIBhAvfRio7C3hUEXTSlqCS/sa6qCmtn6tn7PXgEWBCOigwJLL2luZGV4Lmh0bWyCA1ggHeLPBq1UcefbY+hCvzPZf/nLXa0Ocx3kFz16L
tMZySaCBFggTS4W0tdkYuF5hxF6DzzK4xsOGZRwKAEgLq107bApNr4=:

Tree

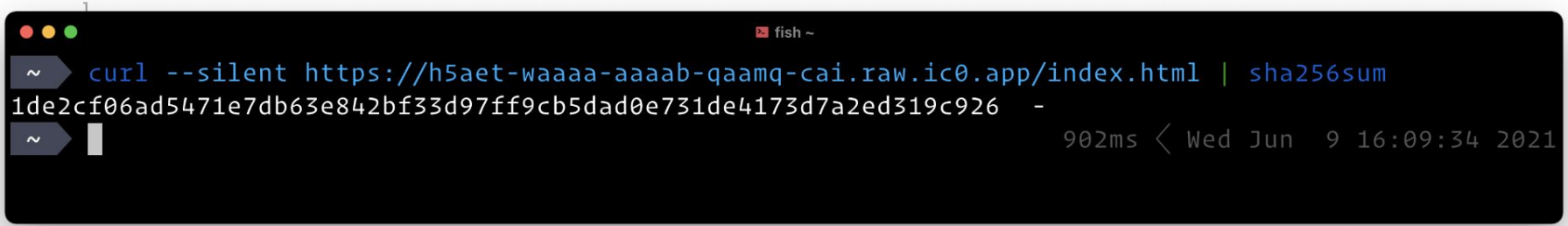
```
55799([
  2 /* labeled */, h'687474705F617373657473' /* "http_assets" */, [
    1 /* fork */, [
      1 /* fork */, [
        4 /* pruned */, h'1840BDF462A3B0B78541174D296A092FEC6BAA829AD9FAB67ECF5E01160423A2'],
        [2 /* labeled */, h'2F696E6465782E68746D6C' /* "/index.html" */, [
          3 /* leaf */, h'1DE2CF06AD5471E7DB63E842BF33D97FF9CB5DAD0E731DE4173D7A2ED319C926'
        ]
      ]
    ], [
      4 /* pruned */, h'4D2E16D2D76462E17987117A0F3CCAE31B0E1994702801202EAD74EDB02936BE'
    ]
  ]
])
```

Tree

```
55799([
  2 /* labeled */, h'687474705F617373657473' /* "http_assets" */, [
    1 /* fork */, [
      1 /* fork */, [
        4 /* pruned */, h'1840BDF462A3B0B78541174D296A092FEC6BAA829AD9FAB67ECF5E01160423A2'],
        [2 /* labeled */, h'2F696E6465782E68746D6C' /* "/index.html" */, [
          3 /* leaf */, h'1DE2CF06AD5471E7DB63E842BF33D97FF9CB5DAD0E731DE4173D7A2ED319C926'
        ]
      ]
    ], [
      4 /* pruned */, h'4D2E16D2D76462E17987117A0F3CCAE31B0E1994702801202EAD74EDB02936BE'
    ]
  ]
])
```

Tree

```
55799([
  2 /* labeled */, h'687474705F617373657473' /* "http_assets" */, [
    1 /* fork */, [
      1 /* fork */, [
        4 /* pruned */, h'1840BDF462A3B0B78541174D296A092FEC6BAA829AD9FAB67ECF5E01160423A2'],
        [2 /* labeled */, h'2F696E6465782E68746D6C' /* "/index.html" */, [
          3 /* leaf */, h'1DE2CF06AD5471E7DB63E842BF33D97FF9CB5DAD0E731DE4173D7A2ED319C926'
        ]
```



Tree

```
55799([
  2 /* labeled */, h'687474705F617373657473' /* "http_assets" */, [
    1 /* fork */, [
      1 /* fork */, [
        4 /* pruned */, h'1840BDF462A3B0B78541174D296A092FEC6BAA829AD9FAB67ECF5E01160423A2'],
        [2 /* labeled */, h'2F696E6465782E68746D6C' /* "/index.html" */, [
          3 /* leaf */, h'1DE2CF06AD5471E7DB63E842BF33D97FF9CB5DAD0E731DE4173D7A2ED319C926'
        ]
      ]
    ], [
      4 /* pruned */, h'4D2E16D2D76462E17987117A0F3CCAE31B0E1994702801202EAD74EDB02936BE'
    ]
  ]
])
```

H(H(H("http_assets") | H(H(0x184...3a2 | H(H("/index.html") | **0x1de...926**))) | H(0x4d2...8be))

0x3CE000E1A98104C4F6B4655A6D9C508258FDEA11B445D0EF88A25215EB35A1FE

Http Response

```
curl -v https://h5aet-waaaa-aaaab-qaamq-cai.raw.ic0.app
```

ic-certificate:

certificate=:2dn3o2R0cmVlgwGDAYMBgwJIY2FuaXN0ZXKDAYMBgwGCBFggVO3kr4k8PoXhhbdIAjyy+Cs3LDBppKcWWOYKyc8pyLmDAYMBggRYIFooON6bZuXXuPk2SSNSDUG5j/l0q1THMjVc8EepzpQIgwGDAYMCSgAAAAAAMAAZAQGDAYMBgwJOY2VydGlmaWVkX2RhdGGCA1ggPOAA4amBBMT2tGVabZxQglj96hG0RdDviKJSFes1of6CBFgg5v2+8RBKfktGILoTd6USC+e3LrqBnEant7cvfYimU/+CBFggcl7xzo7+r66dN/NdP5l/D5FEQaEWiYCYcOSb+umDvQyCBFggUagRifP09v6KIDfpjZb/0c6rMkTLLhIGh2Wkj1JtdciCBFggrotk2XHArmb8zC6c3DrBYGkLiRfH2r/ETWcDRrBkwsCCBFgg+hhvBWFKnZs06dh45WndmuvNxYe4Puvn0vgUwUWUnFmCBFggyO6FcG/5xyWh1+X1ps3RDIuVIkctPikUZ+fuK8EM/YiCBFggW9xw1UmN7+aVEqNhHUPoq1sKac0BpOBSrkwgdO3ARpaCBFgg8K1NwCiEu0jnwbwCD/Ixt7Gt5WScdROzaQi7V3/Kl46CBFggVfgeOlPvMLAK9Gkr8VJXNlJe1p7eIeleuiNAFMy2TFODAYIEWCDEi1UT6xPndn773moeEzrxK0NghCCz/i+oXBFgzrEBNoMCRHRpbWWCA0nUu8j6suO+wxZpc2lnbmF0dXJlWDCzRhoF8CZ4IGqH+xBnedBYl7fQ39SxLHrSEYCKsJ2sKgN0yIGINUh627ESw2Z4r/FqZGVsZWdhdGlvbqJpc3VibmV0X2lkWB3DuW6kfgTIl3y3A/Far1qqr3PVlse+GaQ6cY2LAmtjZXJ0aWZpY2F0ZVkB2NnZ96JkdHJlZYMBggRYICXG82N6hZGEU21kauXHWe89C3rs18CFgCRT1K+bD/gfgwGDAkZzdWJuZXSDAYIEWCB6ER+Aqjk2VmBS0bhRaOdRA/voxQHTD+Q4gY2a4sorFoMBgwGCBFggMlN6Y2hvfLE7b9smcWTBsi2nYB3Sm5wZcZy/jkRgK5GDAYMCWB3DuW6kfgTIl3y3A/Far1qqr3PVlse+GaQ6cY2LAoMCSnB1YmxpY19rZXmCA1iFMIGCMB0GDSsGAQQBgtx8BQMBAgEGDCsGAQQBgtx8BQMCAQNhAI5nIKjxQp5SXDWAIaBcJxT4TYKDVONTNb6fqwir75fapVUoQscz75gYhDiujVTHDgj9rVzSkbRyIiHsdVEuvTNWeKT3rIUidQVoQP1j3A2G9bT/81OSrRUowXrqdJJuCoIEWCDaC0dWOvOuDjt6RC0cZQK3Yyk9awvzSnTLuWICPs4bBIIEWCAg88BQjI6azuZetTYraEncf1rfGmjrCtkdLKcECn2NO4MCRHRpbWWCA0ngm/u87NWowxZpc2lnbmF0dXJlWDCNY3ZcE70CIoZHaP397ITSSmVe+kNlSJ4t6ZdckYfRhktRNuHaNK0K1BFrjIFJOPc=:,

tree=:2dn3gwJLaHR0cF9hc3NldHODAYMBggRYIBhAvfRio7C3hUEXTSlqCS/sa6qCmtn6tn7PXgEWBCOigwJLL2luZGV4Lmh0bWyCA1ggHeLPBq1UcefbY+hCvzPZf/nLXa0Ocx3kFz16LtMZySaCBFggTS4W0tdkYuF5hxF6DzzK4xsOGZRwKAEgLq107bApNr4=:

Certificate

```
({
  "tree": [ ... ],
  "signature": h'8514603AB488965DD3EF5B80B8D44FD08801B08B7AA29A8BCD9DE616C95B6CDDD967C92C01551C33EB2E9760238B581C',
  "delegation": {
    "subnet_id": h'C3B96EA47E04C8977CB703F15AAF5AAAAF73D596C7BE19A43A718D8B02',
    "certificate": /* CBOR encoded certificate */
  }
})
```

Certificate Tree

```
[1, [1, [1, /* forks */
  [2, h'63616E6973746572' /* "canister" */,
    [1, [1, [1, [4, /* pruned */ h'...'], [1, [1, [4, /* pruned */ h'...'],
                [1, [1, [2, h'0000000003000190101' /* Canister ID */,
                            [1, [1, [2, h'6365727469666965645F64617461' /* "certified_data" */,
                                    [3, h'3CE000E1A98104C4F6B4655A6D9C508258FDEA11B445D0EF88A25215EB35A1FE']],
  [4, h'E6FDBEF1104A7E4B4620BA1377A5120BE7B72EBA819C46A7B7B72F7D88A653FF']],
  [4, h'725EF1CE8EFEAFAE9D37F35D3F997F0F914441A11689809870E49BFAE983BD0C']]],
  [4, h'51A81189F3F4F6FE8A2037E98D96FFD1CEAB3244CB2E12068765A48F526D75C8']],
  [4, h'AE8B64D971C0AE66FCCC2E9CDC3AC160690B8917C7DABFC44D670346B064C2C0']]],
  [4, h'FA186F05614A9D9B34E9D878E569DD9AEBCDC587B83EEBE7D2F814C145949C59']]],
  [4, h'C8EE85706FF9C725A1D7E5F5A6CDD10C8B9522472D3E291467E7EE2BC10CFD88']],
  [4, h'5BDC70D5498DEFE69512A3611D43E8AB5B0A69CD01A4E052AE4C2074EDC04696']]],
  [4, h'F0AD4DC02884BB48E7C1BC020FF231B7B1ADE5649C7513B36908BB577FCA978E']],
  [4, h'55F81E3A53EF30B00AF4692BF1525736525ED69EDE21E95EBA234014CCB64C53']],
  [1, [4, h'C48B5513EB13E7767EFBDE6A1E133AF12B43608420B3FE2FA85C1160CEB10136'],
  [2, h'74696D65' /* "time" */, [3, h'D4BBC8FAB2E3BEC316' /* Wednesday, June 9, 2021 5:46:49.890 PM GMT */ ]]]]
```

Validation Done!

Recap

In canister code: create an asset hash tree which contains the content, call `set_certified_data(root_hash)`

On client:

```
let (content, certificate, hash_tree) = get_certified_asset(...);
if sha256(content) != hash_tree.lookup(["http_assets", "/index.html"]) {
  return false;
}

let witness = certificate.lookup(["canister", canister_id, "certified_data"]);
if witness != compute_root_hash(hash_tree) {
  return false;
}

return true;
```

Libraries

Rust CDK has a `**ic-certified-map**` crate to help with managing assets and hash trees. `**ic-types**` also has a compatible client side HashTree (which will converge with the ic-certified-map one).

In JavaScript (client side), the `**Agent JS**` has a `Certificate` and `HashTree` classes to manage serialization, deserialization and validation of both certificates and calculate the root hash. This is what the service worker uses.

On Motoko, there is a library to make and manage a HashTree and getting the root hash. Not sure about the state of assets tree.

The asset canister already supports certification.

The wallet and Identity Service both use the same static assets strategy and so are initialized on installation, never changed.

Resources

- Interface Spec: Certification
- Certified assets canister
- Certifying service worker