

UCAN & WNFs

 What, Why, and Integration 

Decentralized Digital Identity

DIDs



Decentralized Digital Identity

DIDs

EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Decentralized Digital Identity

DIDs

- One or more public keys
- Truly “universal” user IDs
- Agnostic about backing
 - Self-attesting
 - Database
 - Blockchain
- For users, devices, and more
- Relates to verifiable credentials

EXAMPLE 2: Minimal self-managed DID Document

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "publicKey": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "owner": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "authentication": [{
    // this key can be used to authenticate as DID ...9938
    "type": "RsaSignatureAuthentication2018",
    "publicKey": "did:example:123456789abcdefghi#keys-1"
  }],
  "service": [{
    "type": "ExampleService",
    "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Decentralized Digital Identity

did:key & UCAN

Decentralized Digital Identity

did:key & UCAN

- “Just” a public key (e.g. RSA, EdDSA)

Decentralized Digital Identity

did:key & UCAN

- “Just” a public key (e.g. RSA, EdDSA)
- Self-certifying, extremely flexible

Decentralized Digital Identity

did:key & UCAN

- “Just” a public key (e.g. RSA, EdDSA)
- Self-certifying, extremely flexible
- Well suited to capabilities/authZ (vs identity/authN)

Decentralized Digital Identity

did:key & UCAN

- “Just” a public key (e.g. RSA, EdDSA)
- Self-certifying, extremely flexible
- Well suited to capabilities/authZ (vs identity/authN)
- Made practical with UCANs
 - did:key → authN
 - UCAN → authZ

Decentralized Digital Identity

Variety

Decentralized Digital Identity

Variety

- Microsoft ION, 3Box's Ceramic, Sovrin, did:key, and well over 400 others

Decentralized Digital Identity

Variety

- Microsoft ION, 3Box's Ceramic, Sovrin, did:key, and well over 400 others
- Can federate, but hasn't been done yet win the wild
 - Fission working towards interop with ION as first step

User Controlled, Local-First, Universal Auth & ID

UCAN



UCAN

Fission Use Case → Highly Flexible & Secure

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility
- Won't always have access to the “root” device

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility
- Won't always have access to the “root” device
- Must work offline

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility
- Won't always have access to the “root” device
- Must work offline
- Extensible semantics

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility
- Won't always have access to the “root” device
- Must work offline
- Extensible semantics
- Flexible granularity

UCAN

Fission Use Case → Highly Flexible & Secure

- Work directly in a browser without plugins
- Browser is hostile — compatible with WebCrypto non-exportable keys
- User controlled / user owned
- Pseudonymous, principle of least authority & least visibility
- Won't always have access to the “root” device
- Must work offline
- Extensible semantics
- Flexible granularity
- Revocable

UCAN

Object Capability Model (OCAP)

UCAN

Object Capability Model (OCAP)

- ACL is “reactive auth”

UCAN

Object Capability Model (OCAP)

- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

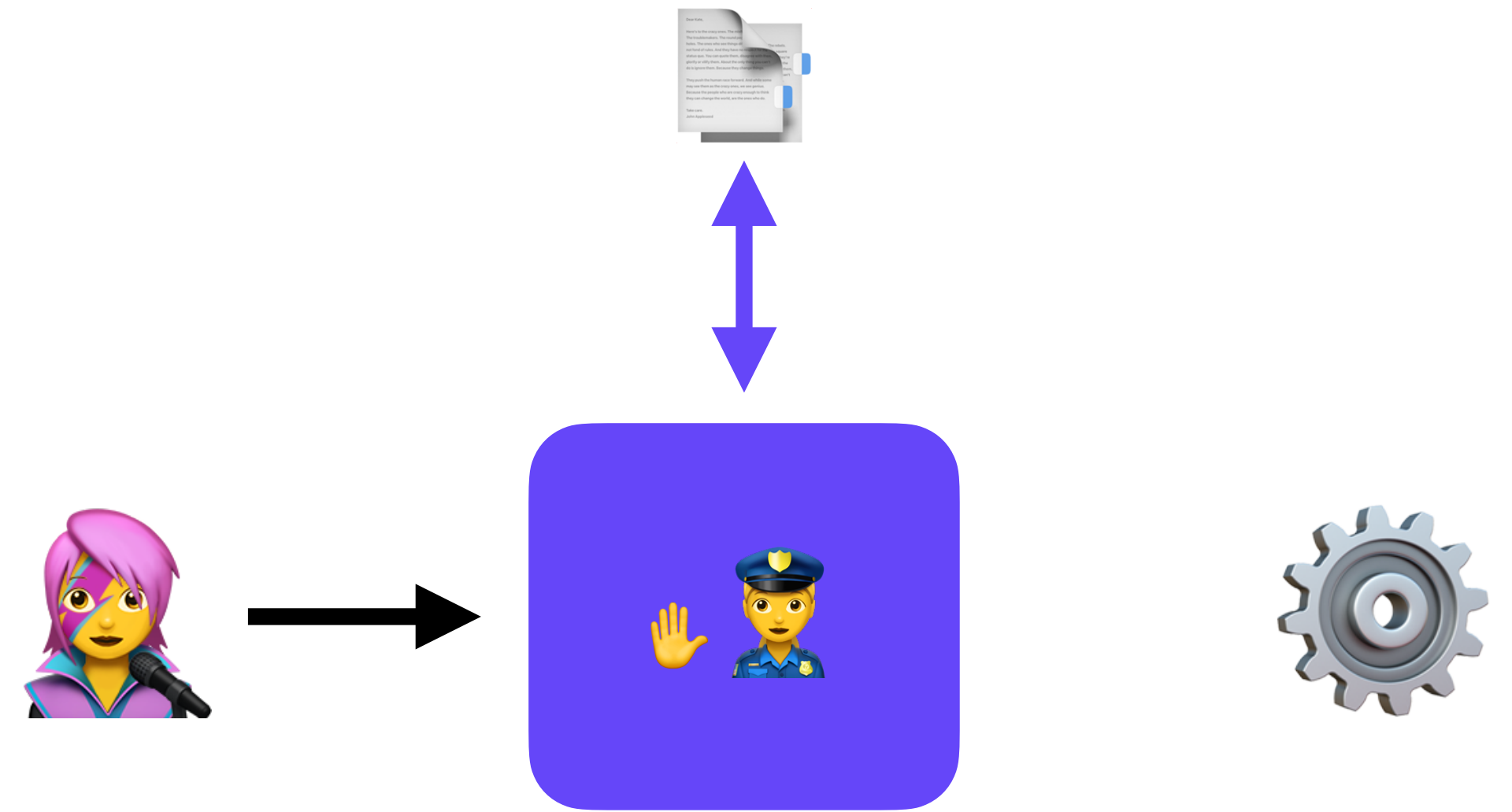
- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

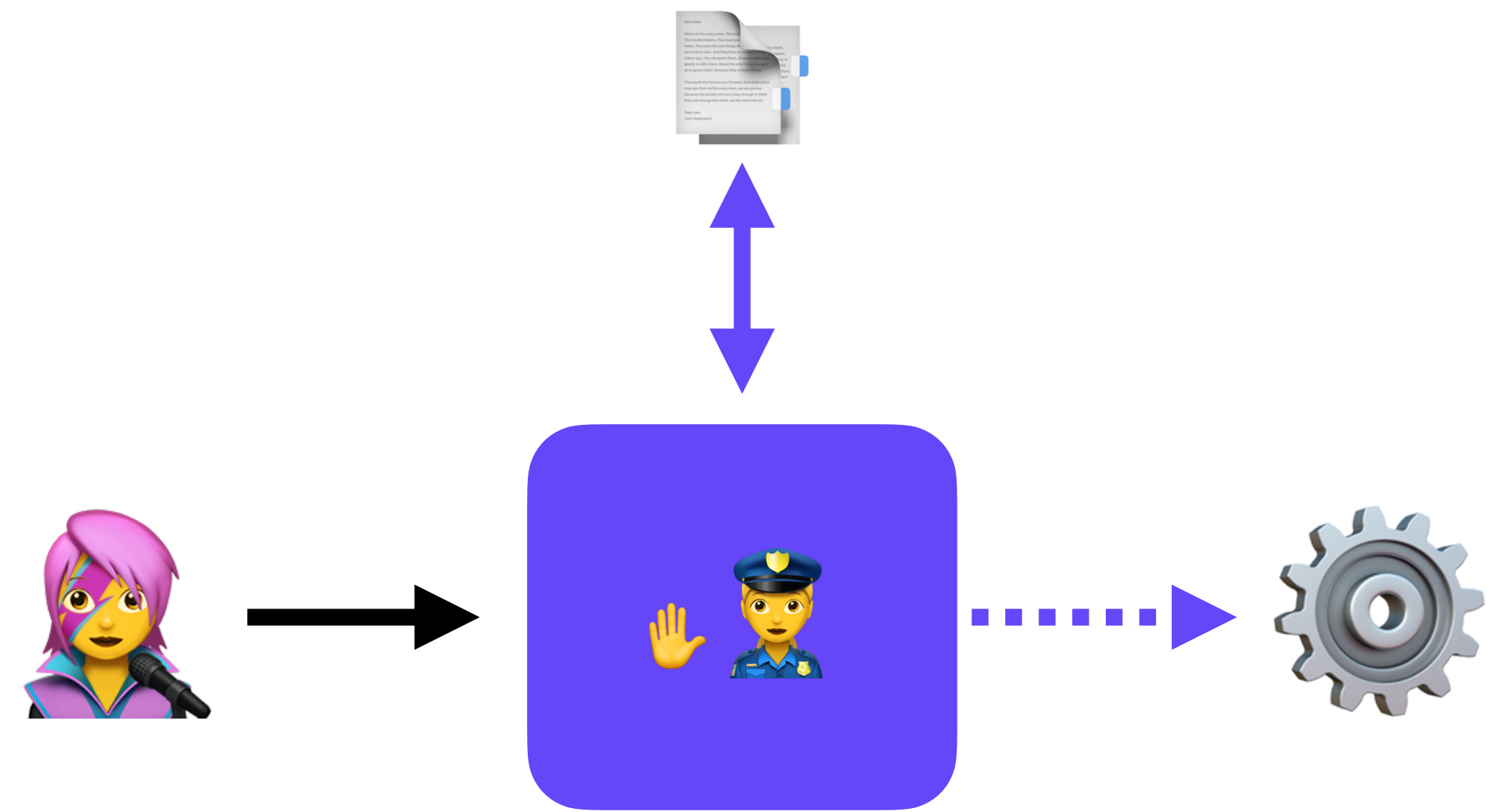
- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

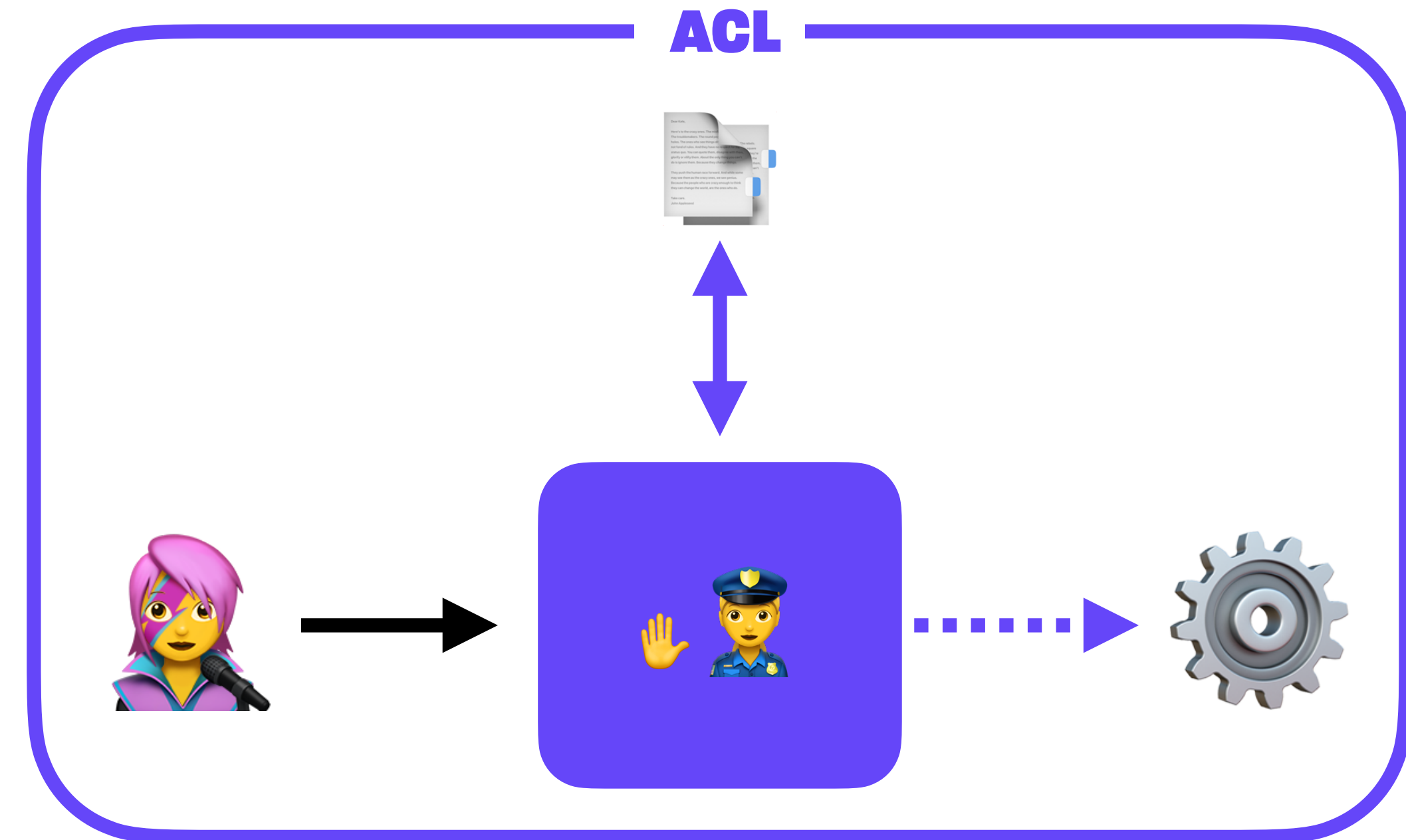
- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

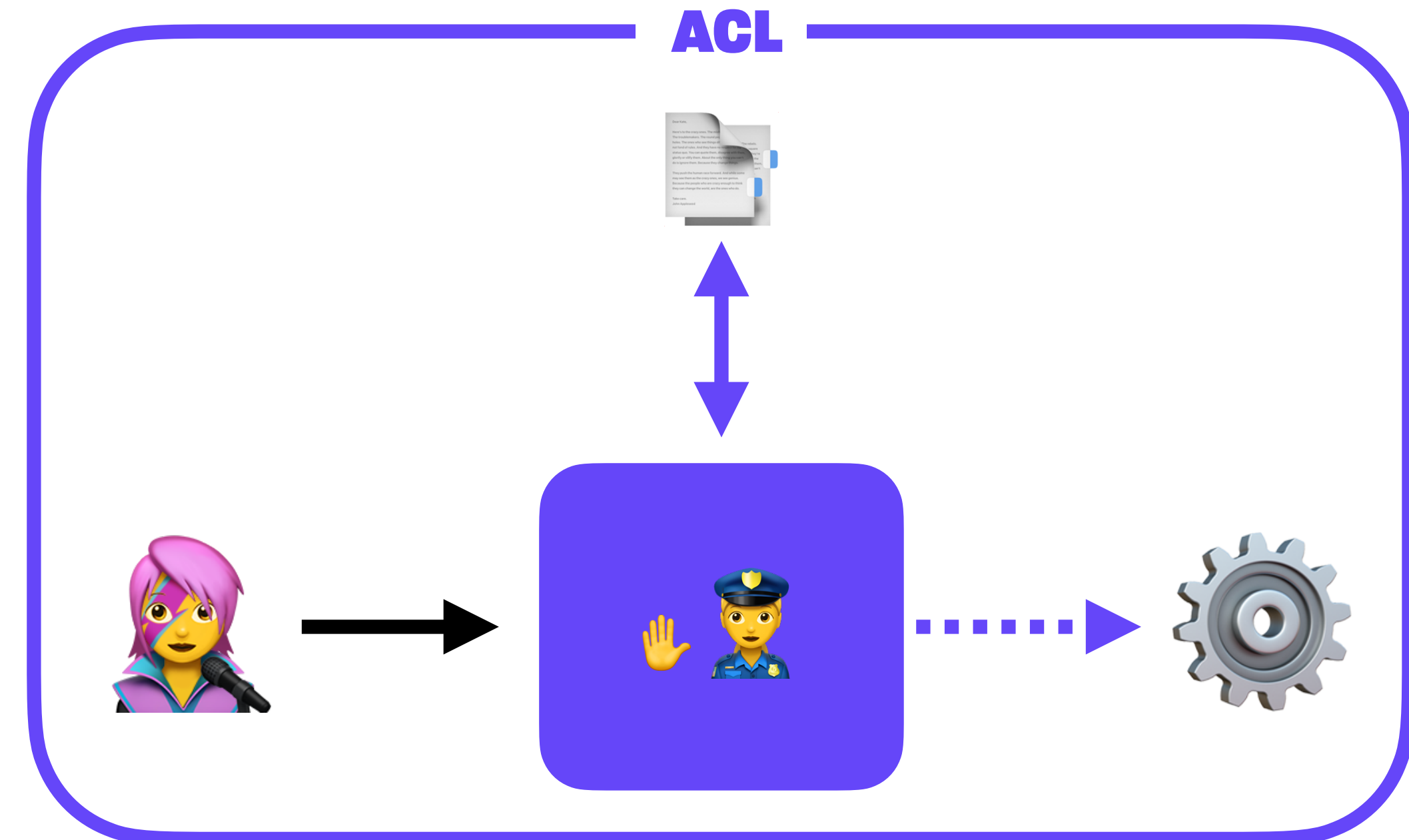
- ACL is “reactive auth”



UCAN

Object Capability Model (OCAP)

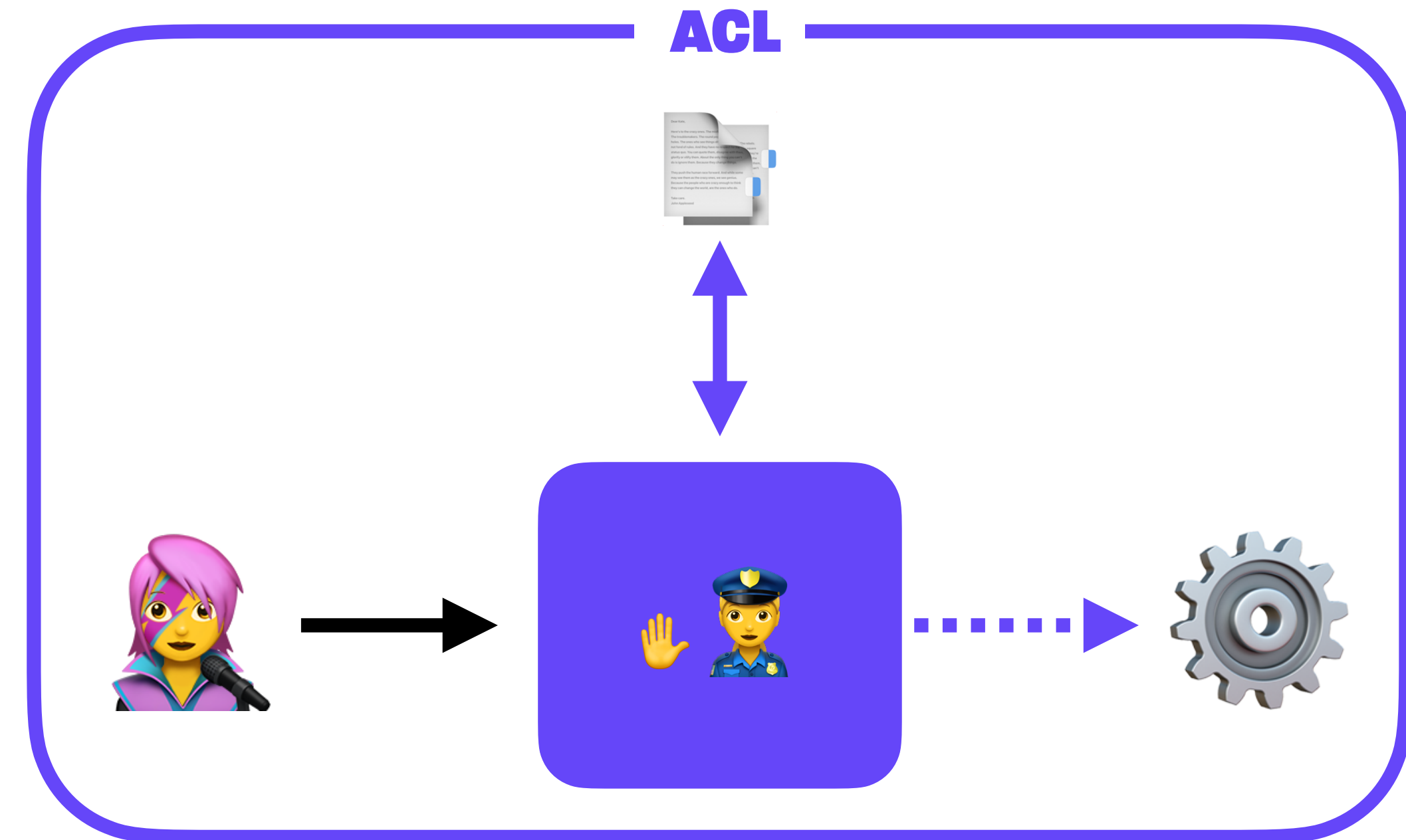
- ACL is “reactive auth”
- OCAP is “proactive auth”
 - Contains all the info about access
 - Any guarding done up front (e.g. time limiting)
 - Generally some reference, proof, or key
 - Anything directly created (parenthood)
 - Delegate subset of access to another (introduction)
- Long history (e.g. X.509, SDSI, SPKI, Macaroons)



UCAN

Object Capability Model (OCAP)

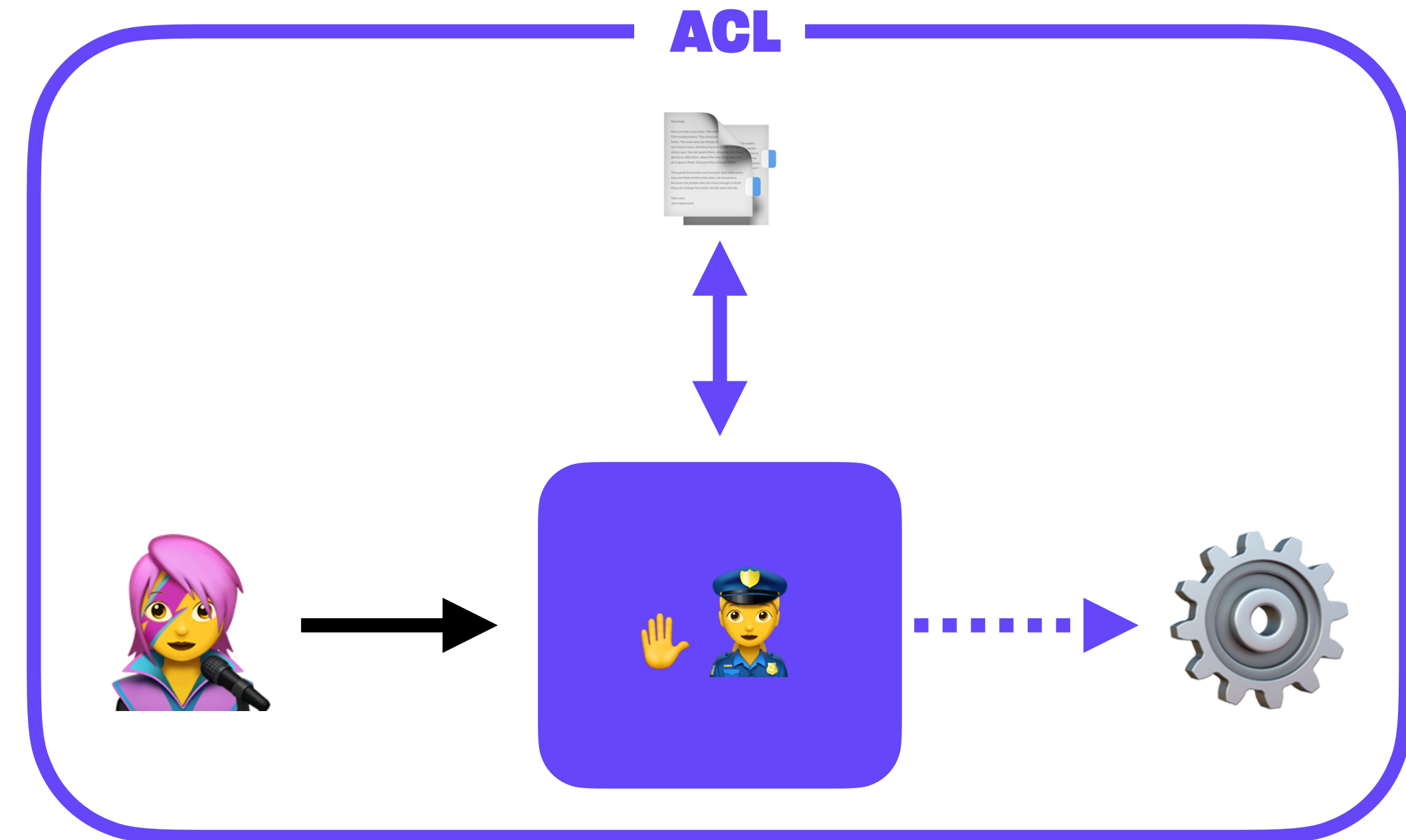
- ACL is “reactive auth”
- OCAP is “proactive auth”
 - Contains all the info about access
 - Any guarding done up front (e.g. time limiting)
 - Generally some reference, proof, or key
 - Anything directly created (parenthood)
 - Delegate subset of access to another (introduction)
- Long history (e.g. X.509, SDSI, SPKI, Macaroons)



UCAN

Object Capability Model (OCAP)

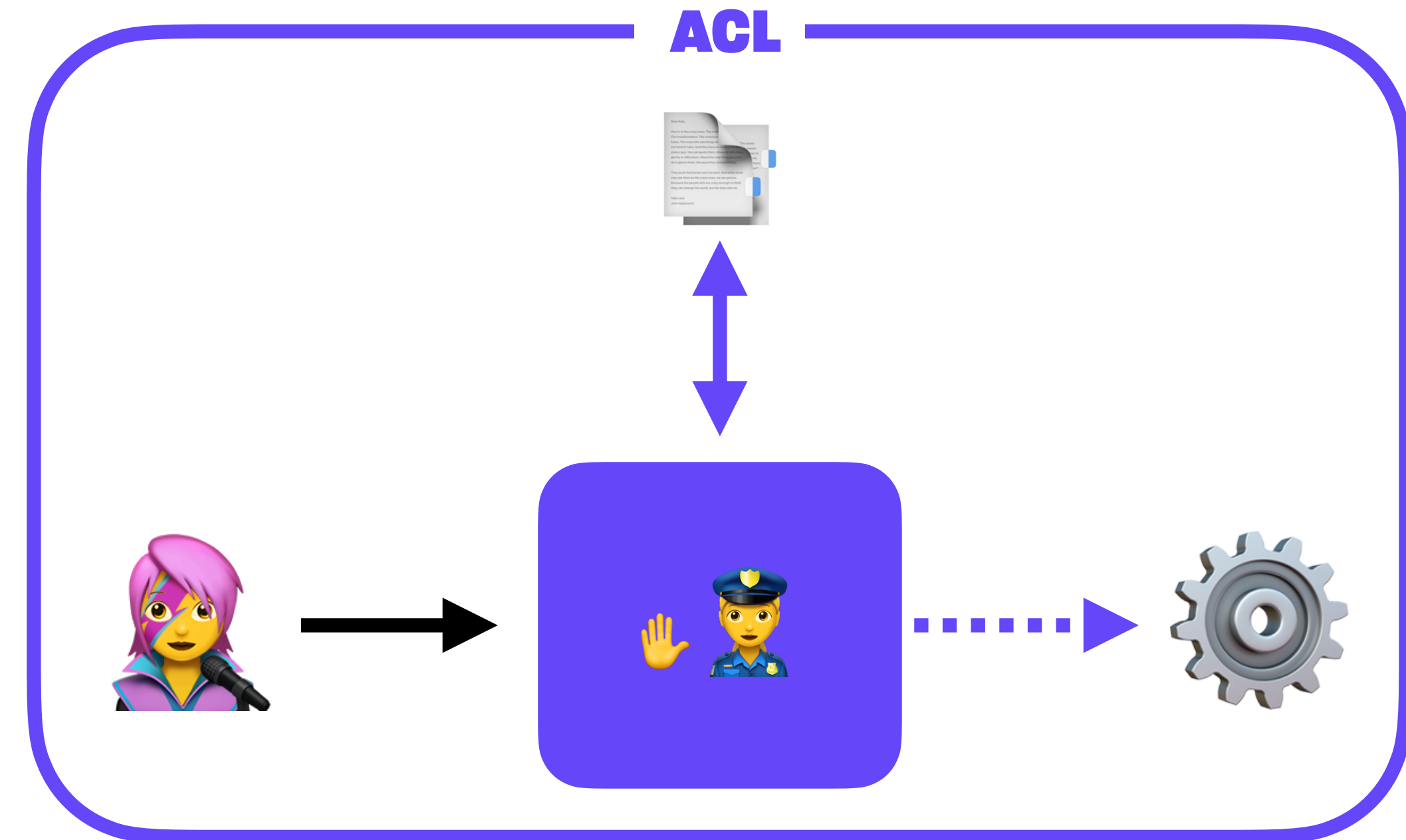
- ACL is “reactive auth”
- OCAP is “proactive auth”
 - Contains all the info about access
 - Any guarding done up front (e.g. time limiting)
 - Generally some reference, proof, or key
 - Anything directly created (parenthood)
 - Delegate subset of access to another (introduction)
- Long history (e.g. X.509, SDSI, SPKI, Macaroons)



UCAN

Object Capability Model (OCAP)

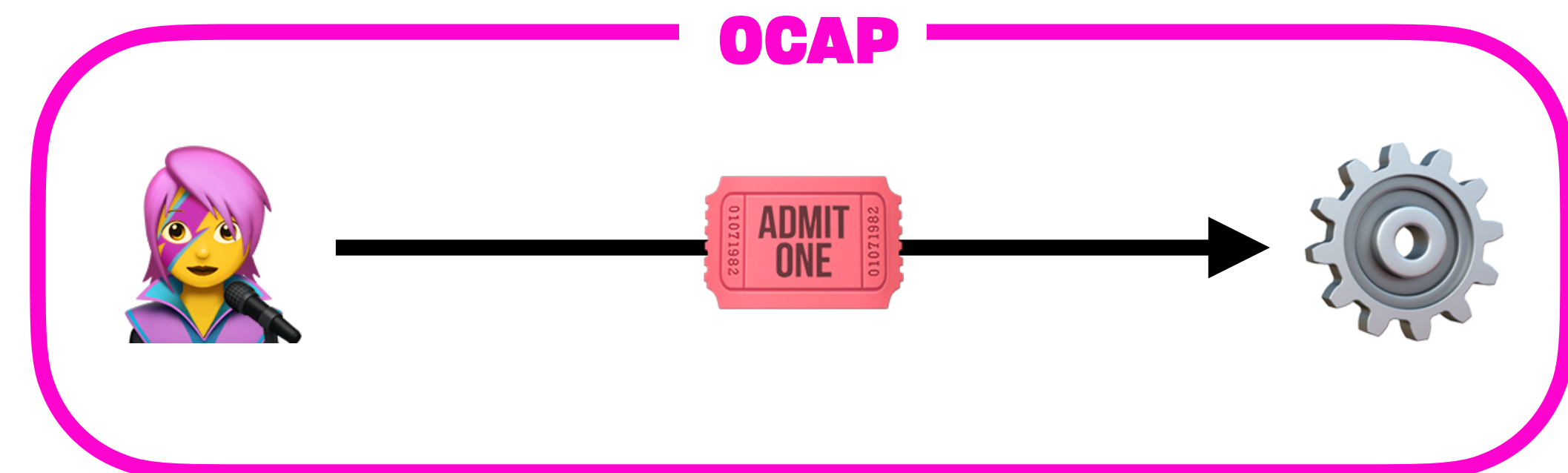
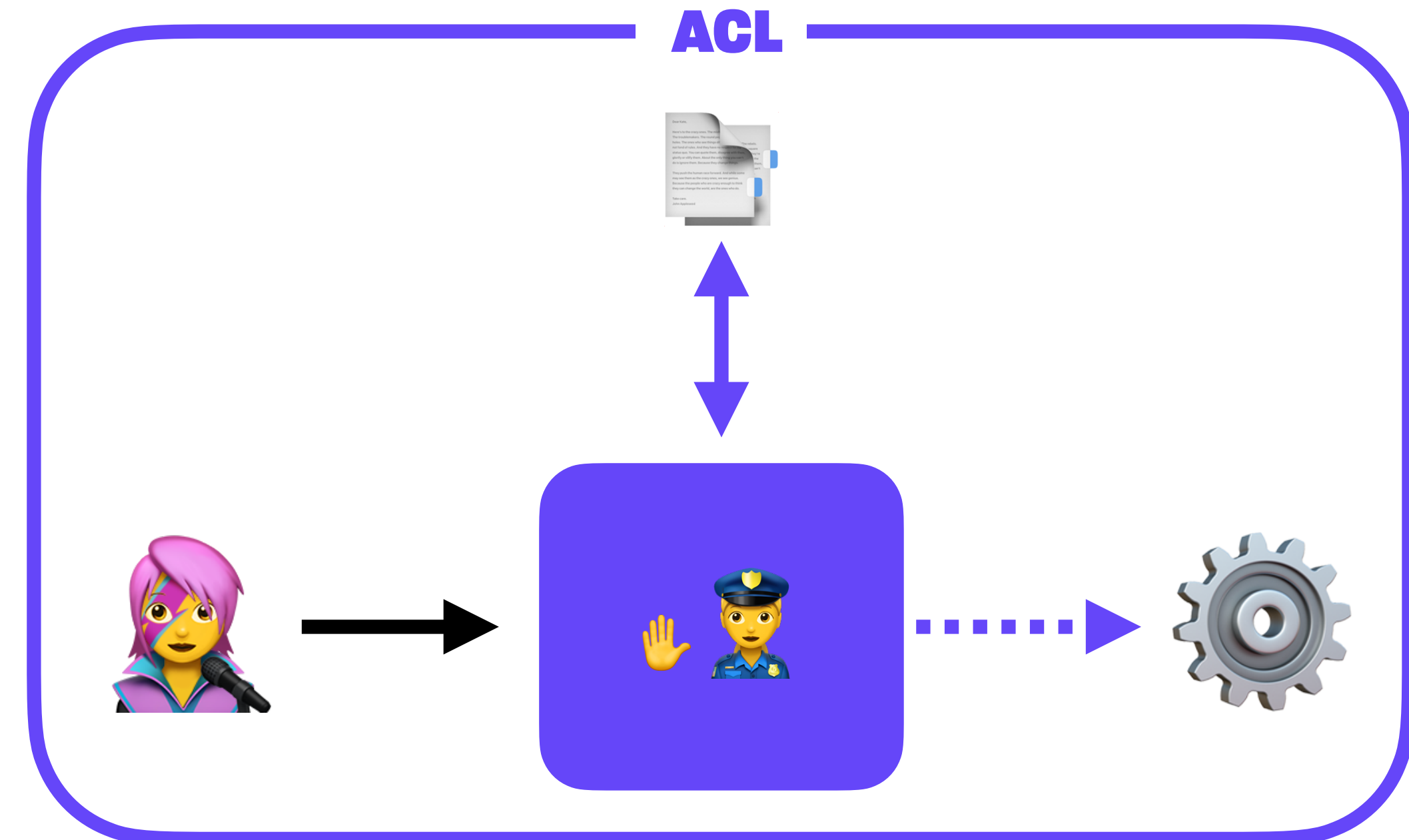
- ACL is “reactive auth”
- OCAP is “proactive auth”
 - Contains all the info about access
 - Any guarding done up front (e.g. time limiting)
 - Generally some reference, proof, or key
 - Anything directly created (parenthood)
 - Delegate subset of access to another (introduction)
- Long history (e.g. X.509, SDSI, SPKI, Macaroons)



UCAN

Object Capability Model (OCAP)

- ACL is “reactive auth”
- OCAP is “proactive auth”
 - Contains all the info about access
 - Any guarding done up front (e.g. time limiting)
 - Generally some reference, proof, or key
 - Anything directly created (parenthood)
 - Delegate subset of access to another (introduction)
 - Long history (e.g. X.509, SDSI, SPKI, Macaroons)



UCAN

Chained Attenuation

UCAN

Chained Attenuation



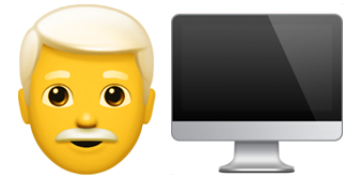
UCAN

Chained Attenuation



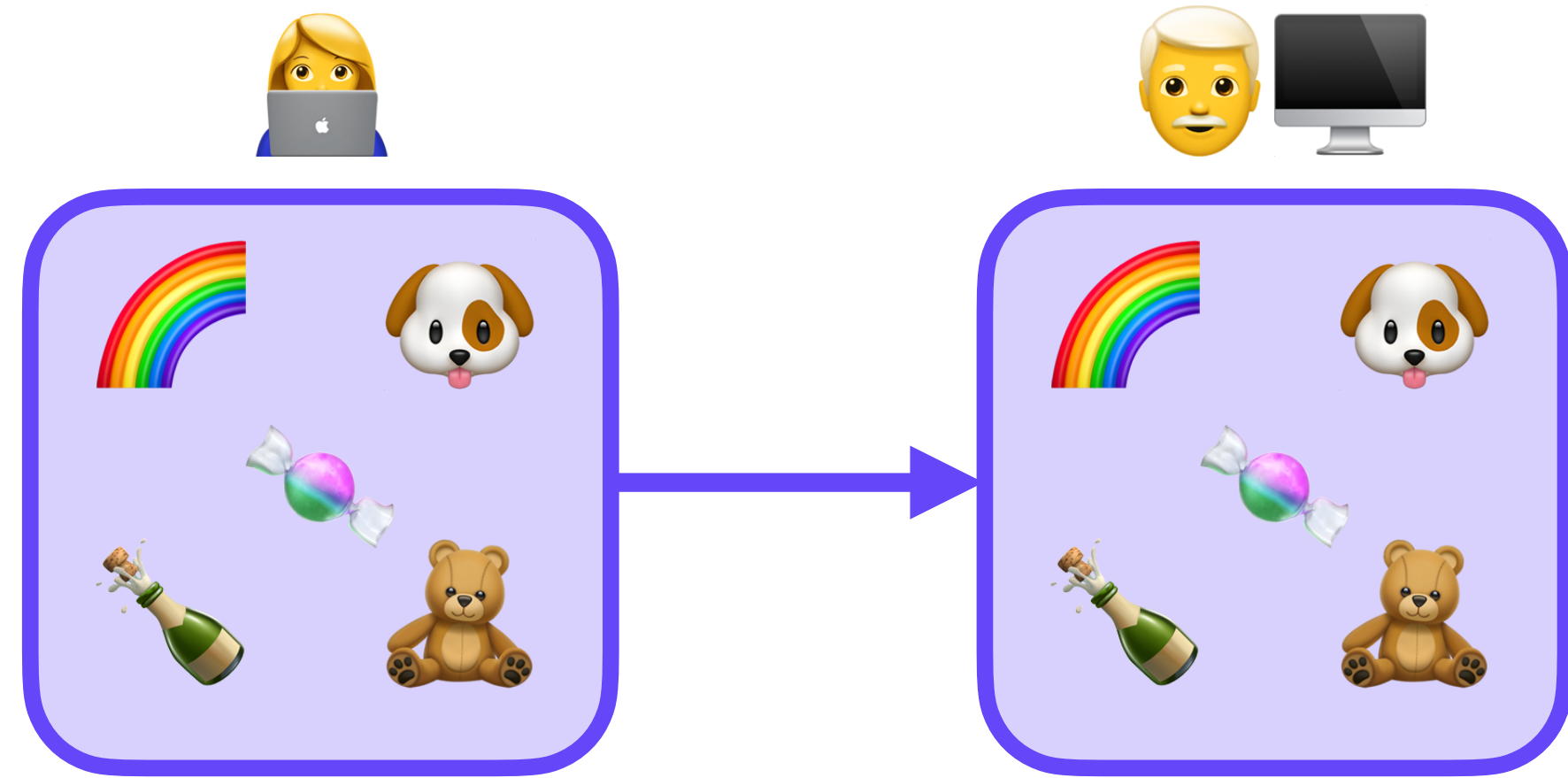
UCAN

Chained Attenuation



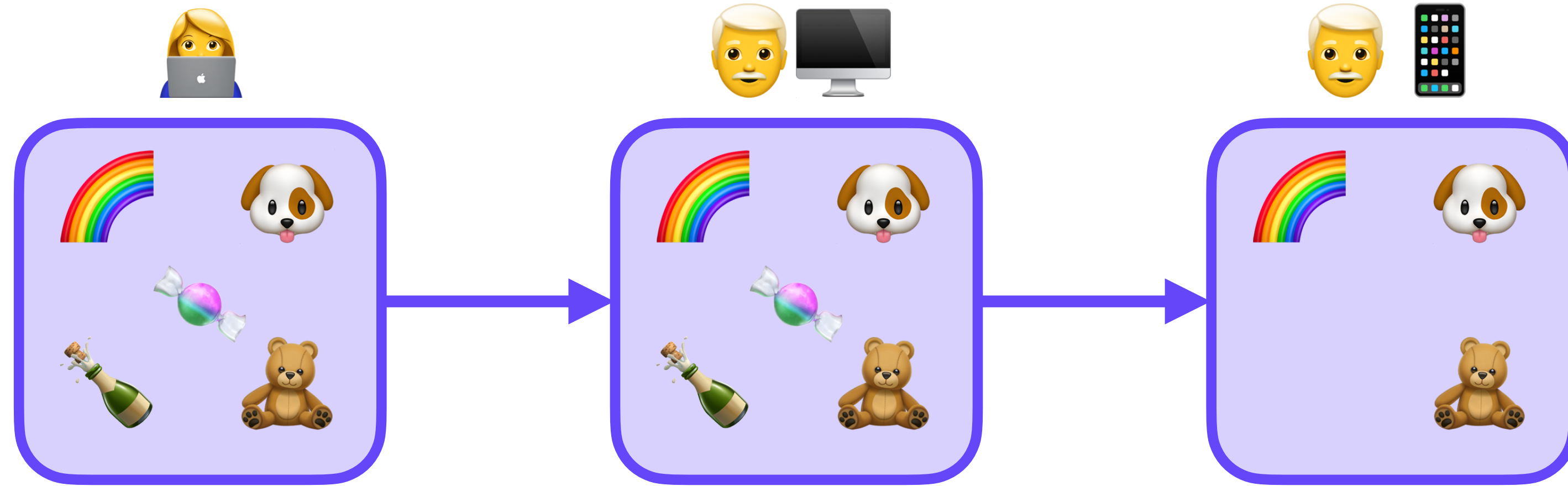
UCAN

Chained Attenuation



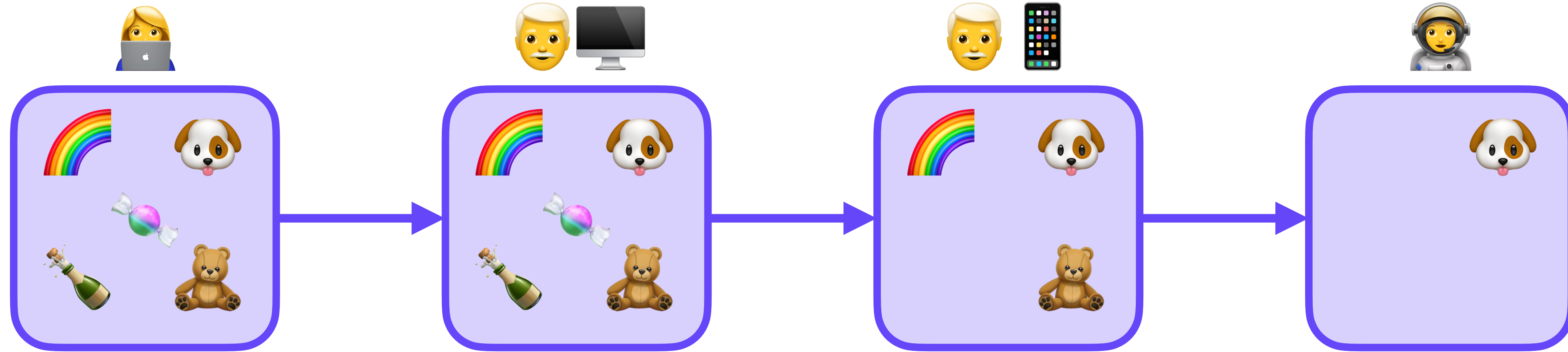
UCAN

Chained Attenuation



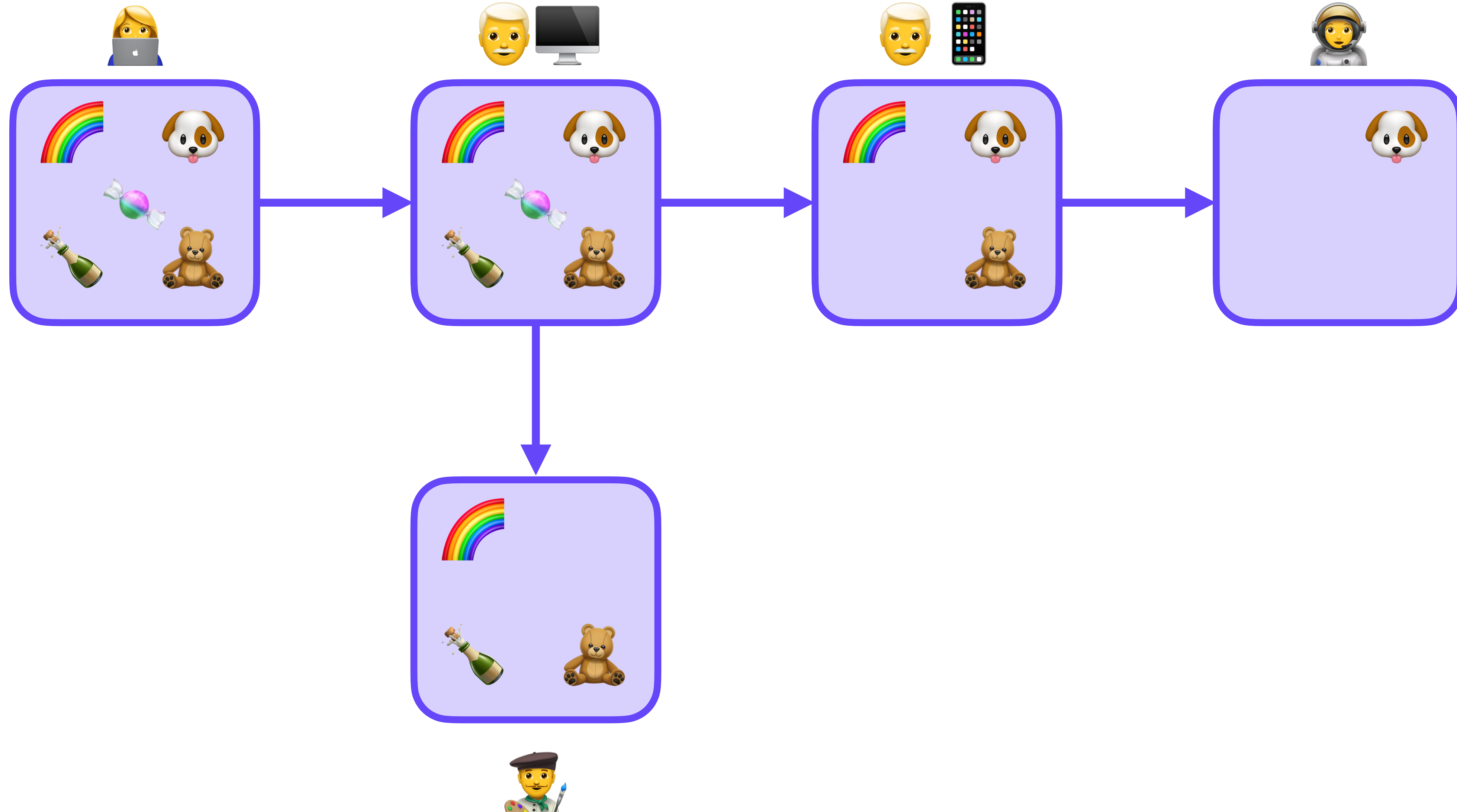
UCAN

Chained Attenuation



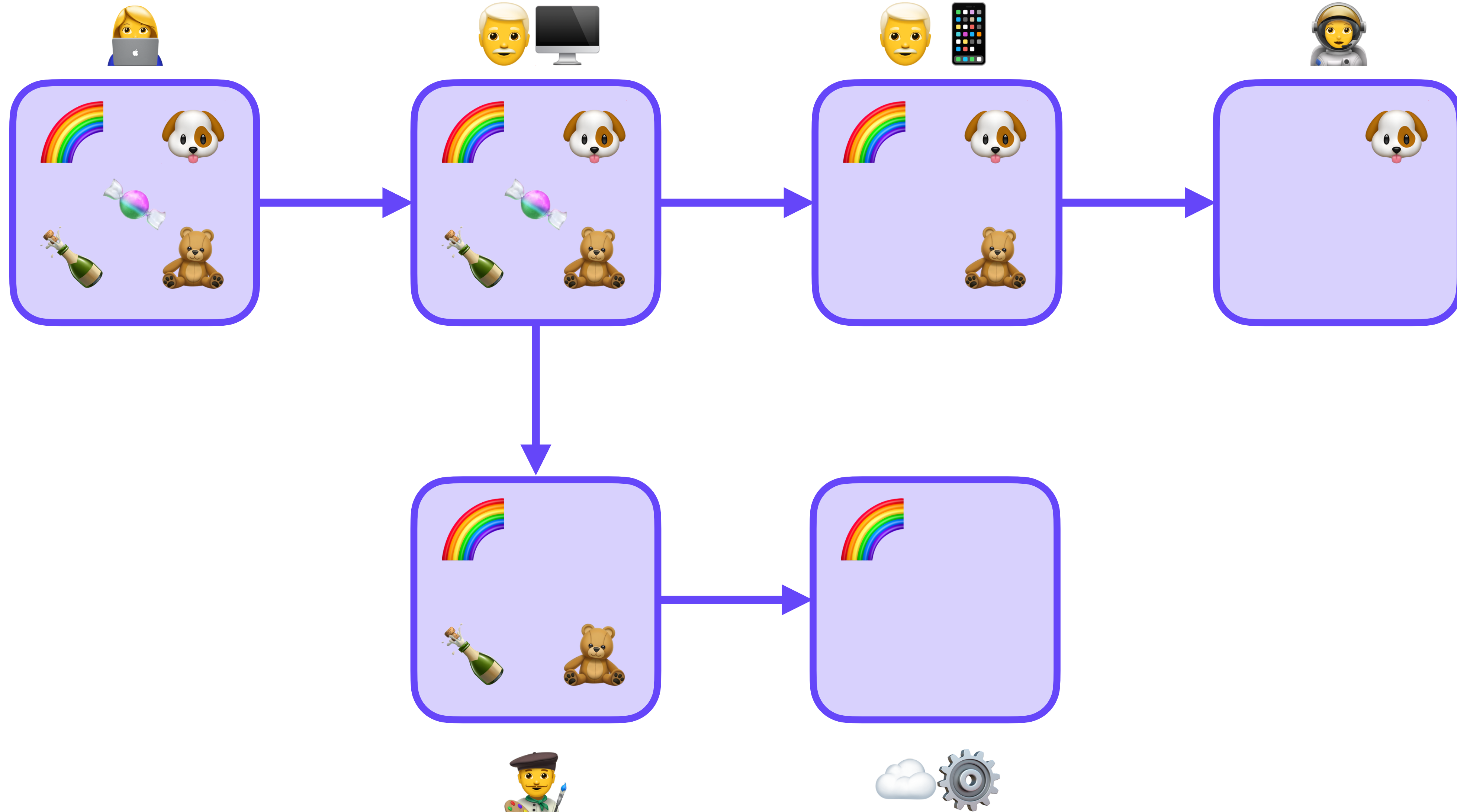
UCAN

Chained Attenuation



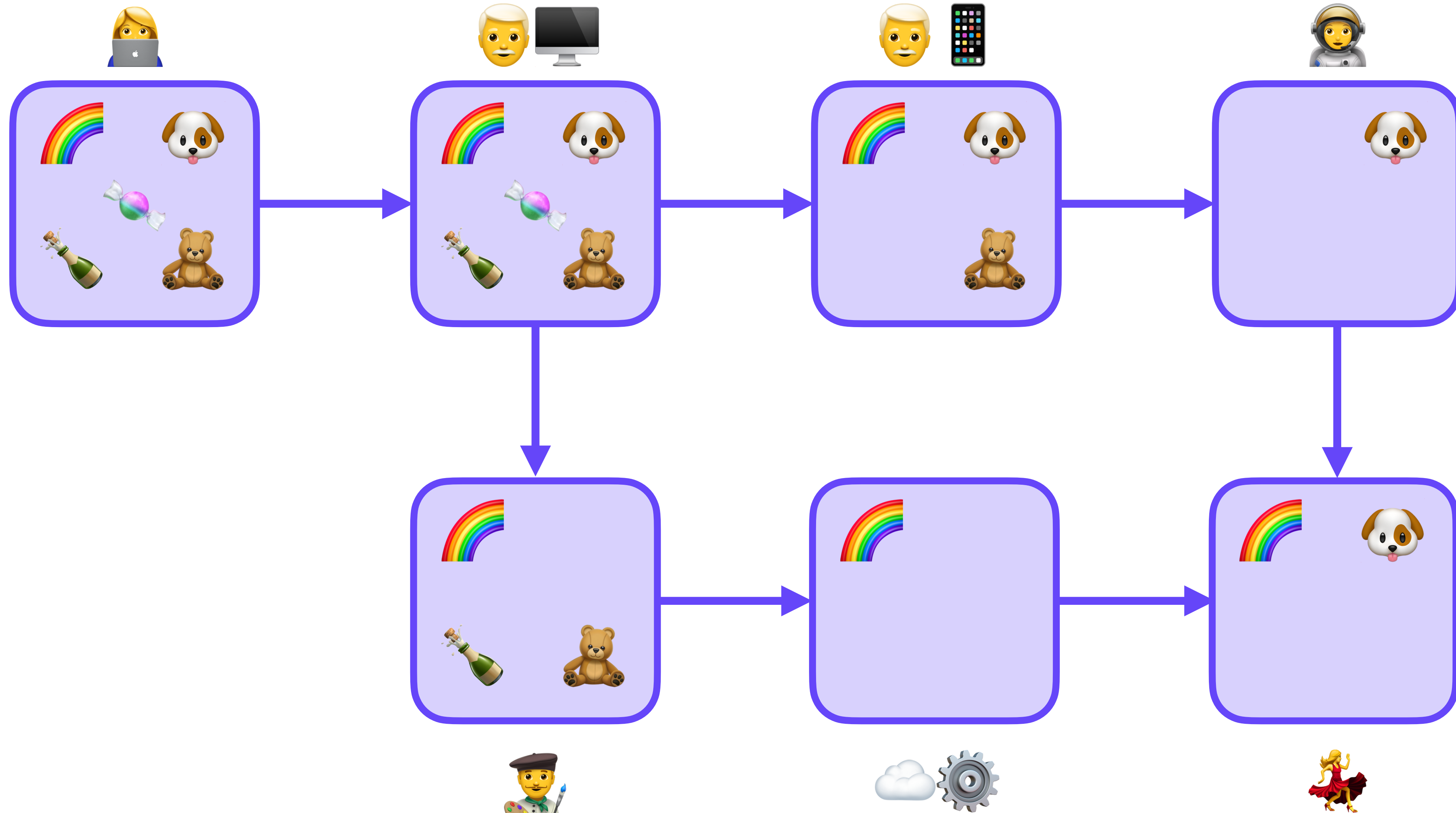
UCAN

Chained Attenuation



UCAN

Chained Attenuation



UCAN

Tradeoffs & Hybridization

UCAN

Tradeoffs & Hybridization

- **Pure ACL, reactive**
 - Centrally view who has access to what
 - Check on every request, bottleneck
 - At-will revocation
 - Access rules grow in complexity
 - More complex provisioning

UCAN

Tradeoffs & Hybridization

▪ **Pure ACL, reactive**

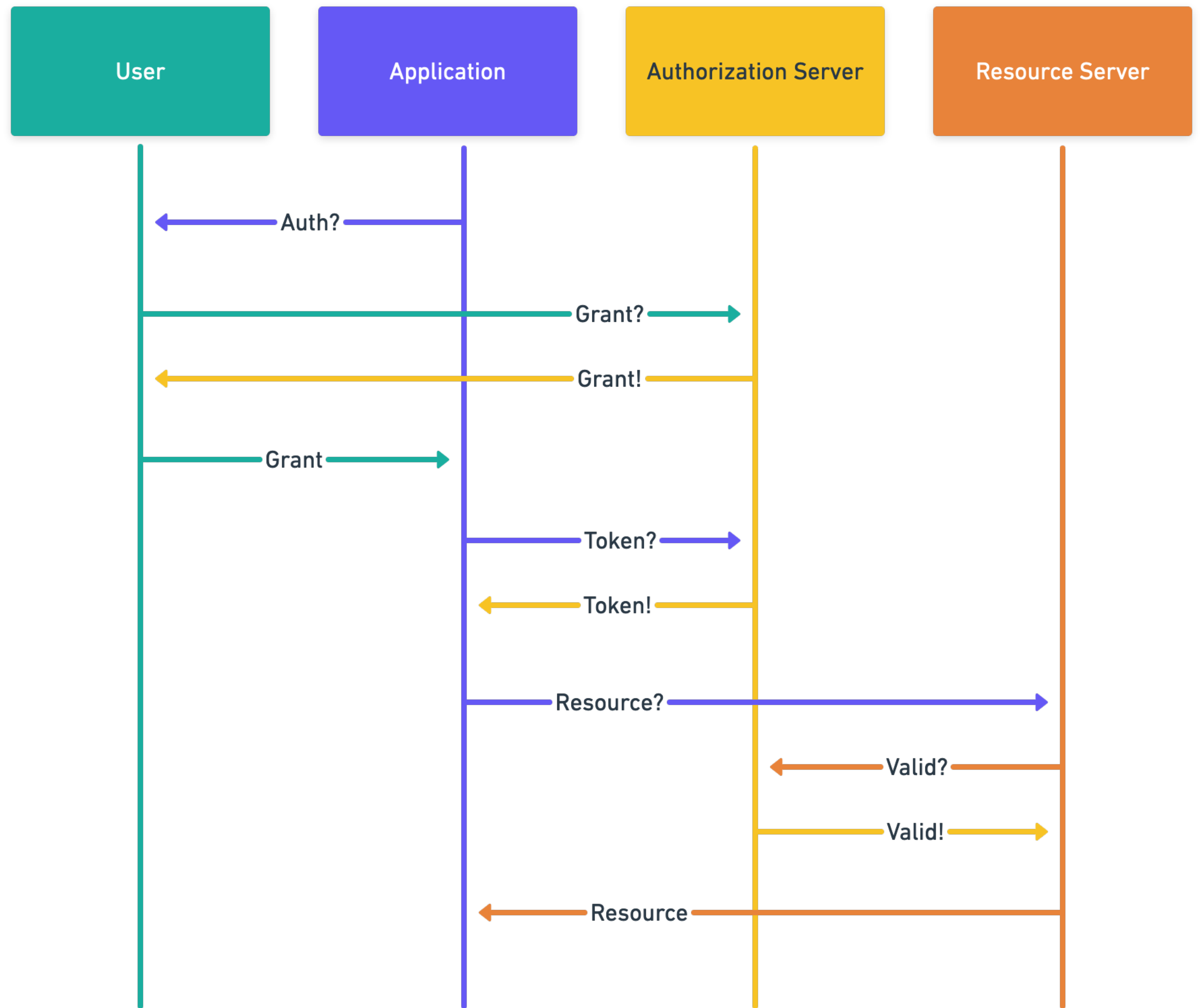
- Centrally view who has access to what
- Check on every request, bottleneck
- At-will revocation
- Access rules grow in complexity
- More complex provisioning

▪ **Pure OCAP, proactive**

- Works offline & everywhere
- User owned or provisioned
- No resource contention, infinite scale
- Easy interop (as we'll see)
- Principle of least authority
- Revocation more difficult
- Tracking possible but has tradeoffs

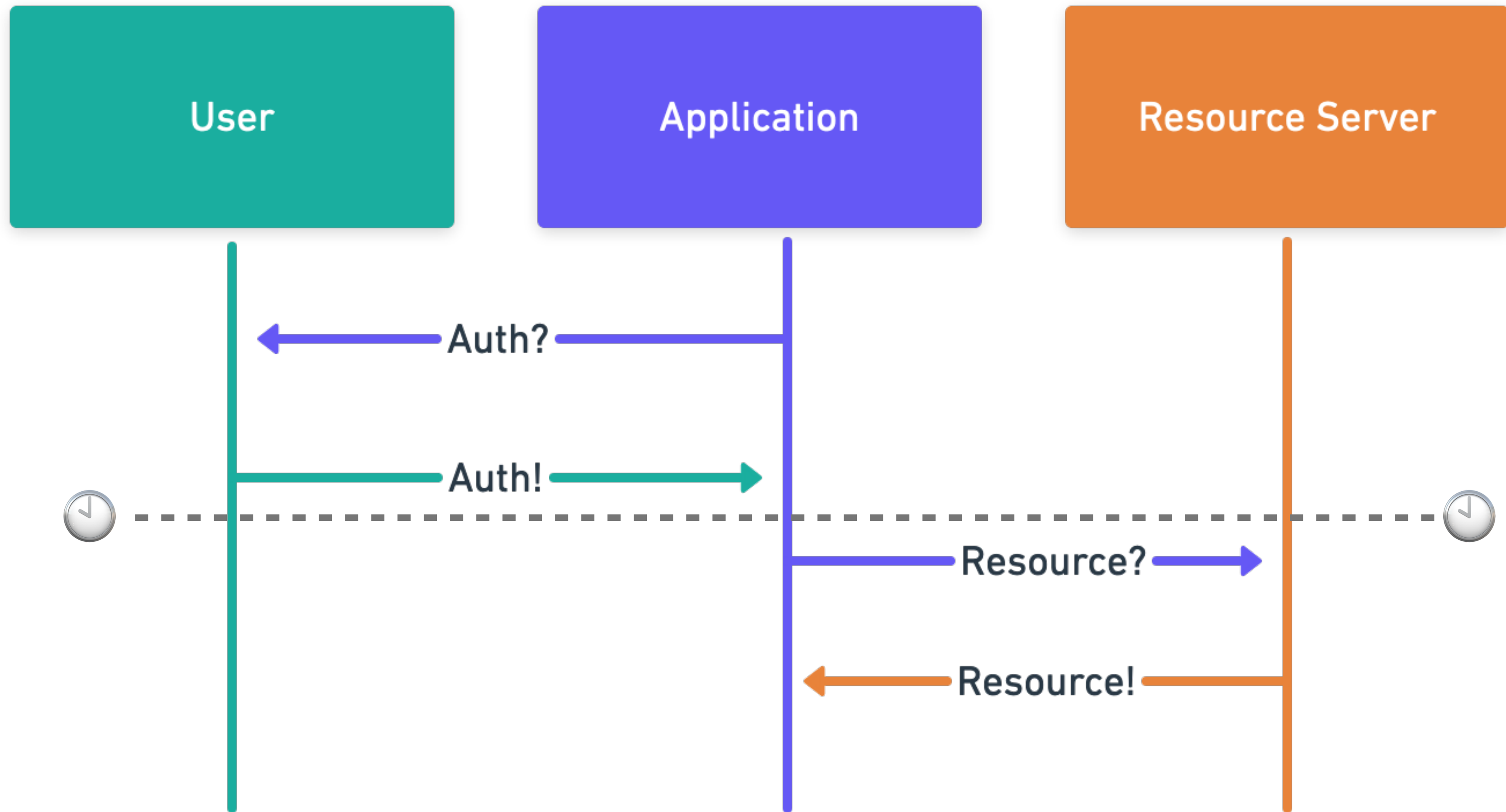
UCAN

OAuth Sequence



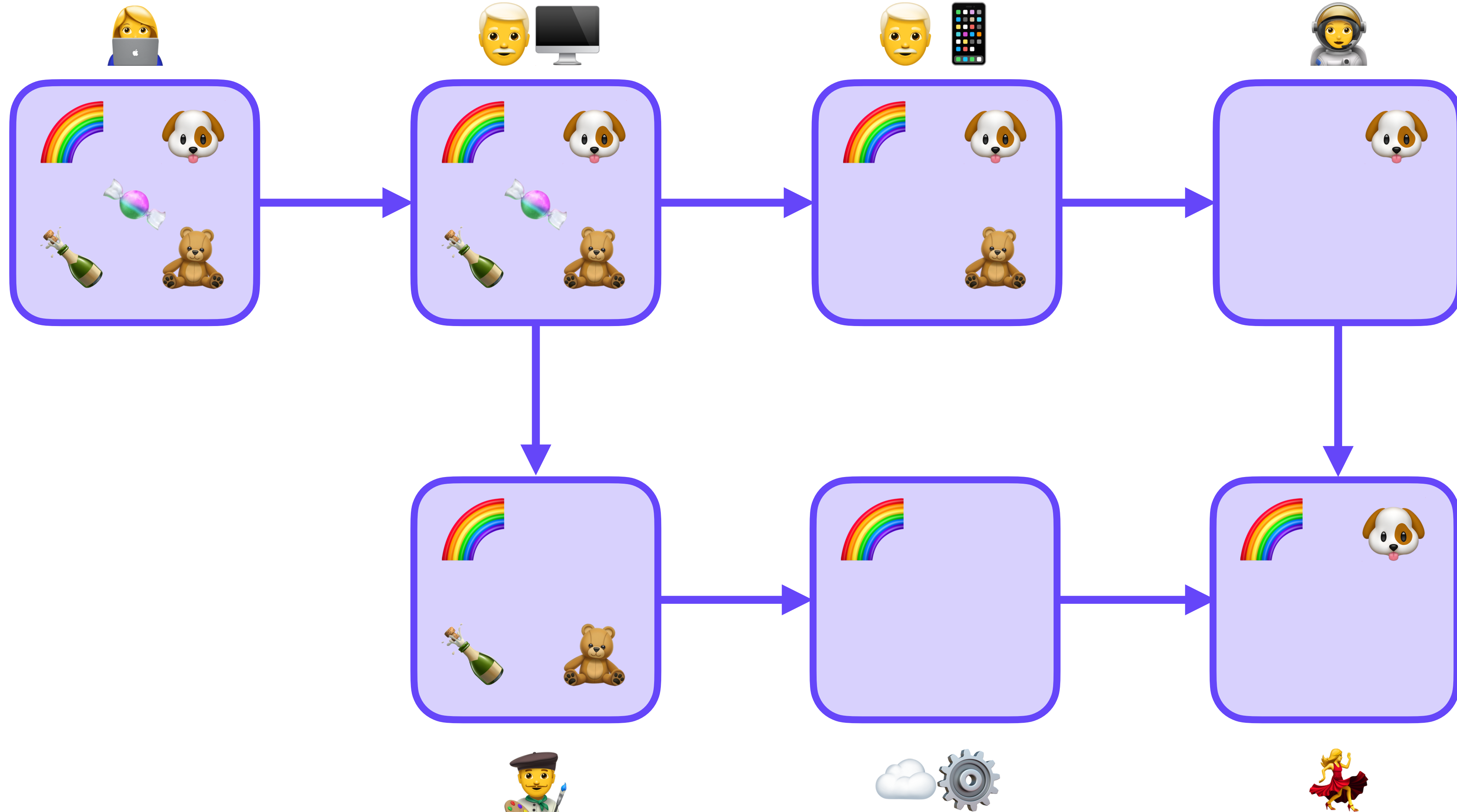
UCAN

UCAN Sequence



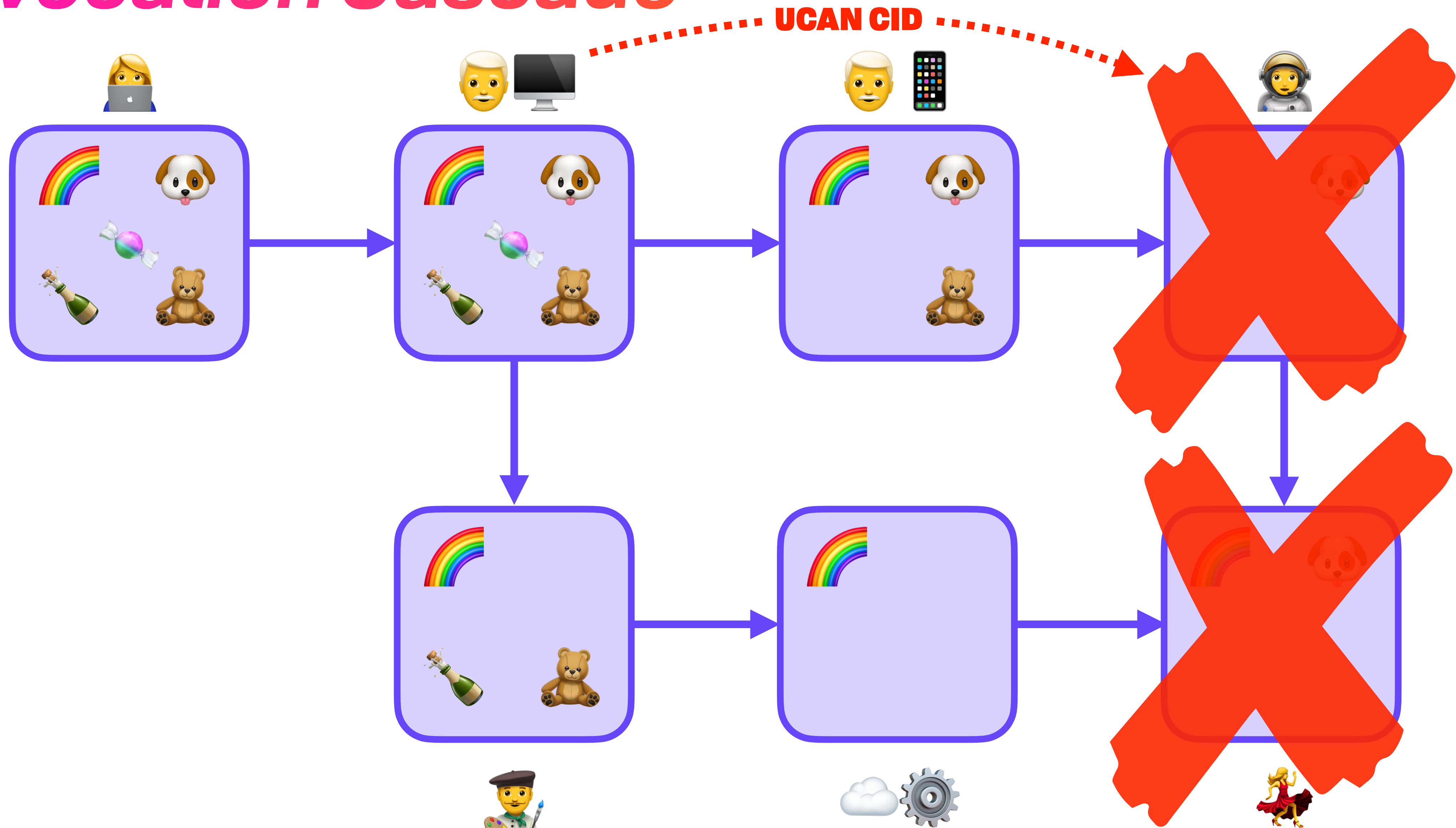
UCAN

Revocation Cascade



UCAN

Revocation Cascade



UCAN JWT

```
{
  "alg": "EdDSA",
  "typ": "JWT"
}
{
  "aud": "did:key:zStEZpzSMtTt9k2vszgvCwF4fLQQSyA15W5AQ4z3AR6Bx4eFJ5crJFbuGxKmbma4",
  "iss": "did:key:z5C4fuP2DDJChhMBCwAkpYUMuJZdNWWH5NeYjUyY8btYfzDh3aHwT5picHr9Ttjq",

  "nbf": 1611204719,
  "exp": 1611300000,

  "fct": [
    {
      "sha256": "B94D27B9934D3E08A52E52D7DA7DABFAC484EFE37A5380EE9088F7ACE2EFCDE9",
      "msg": "hello world"
    }
  ]

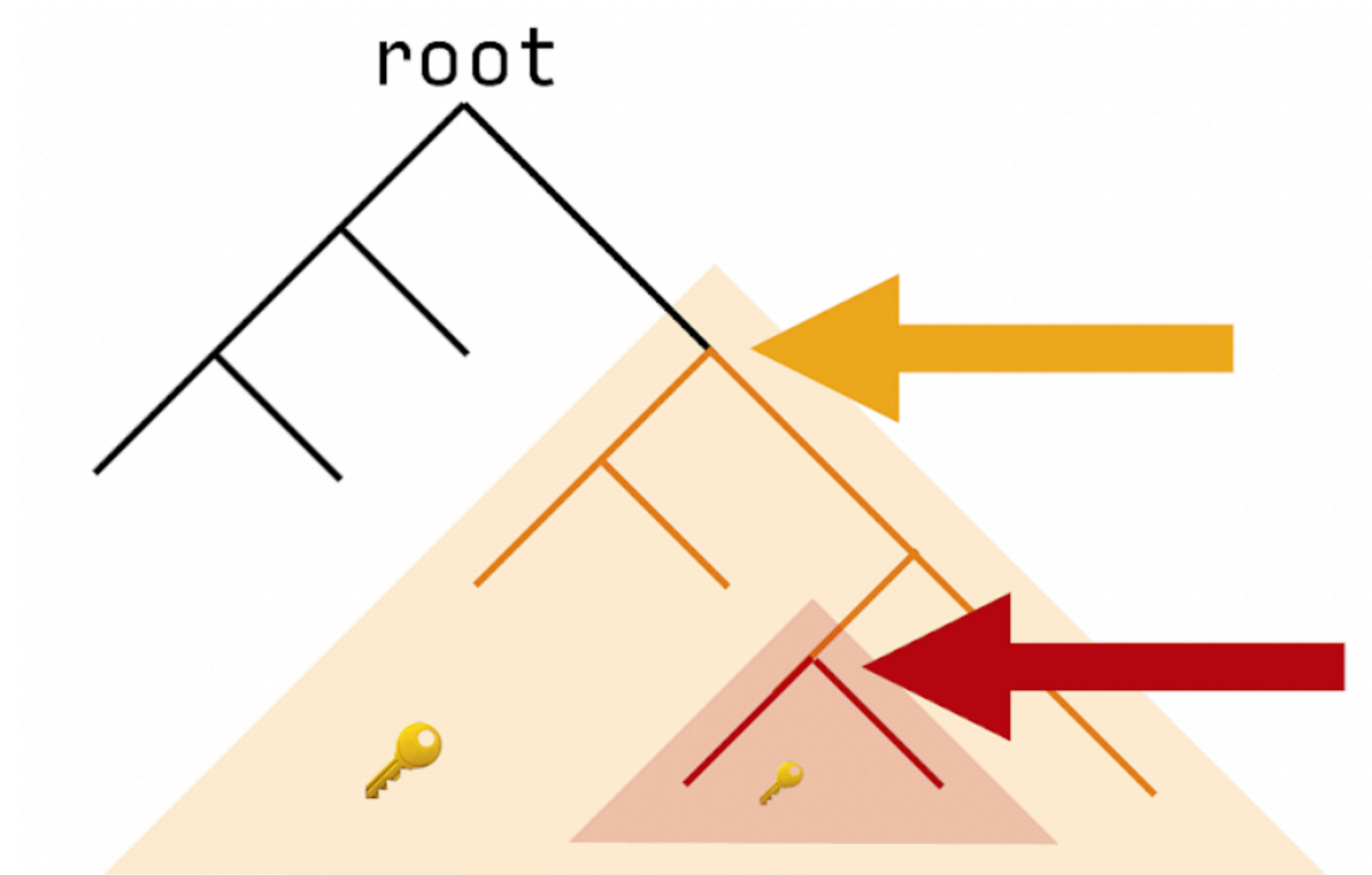
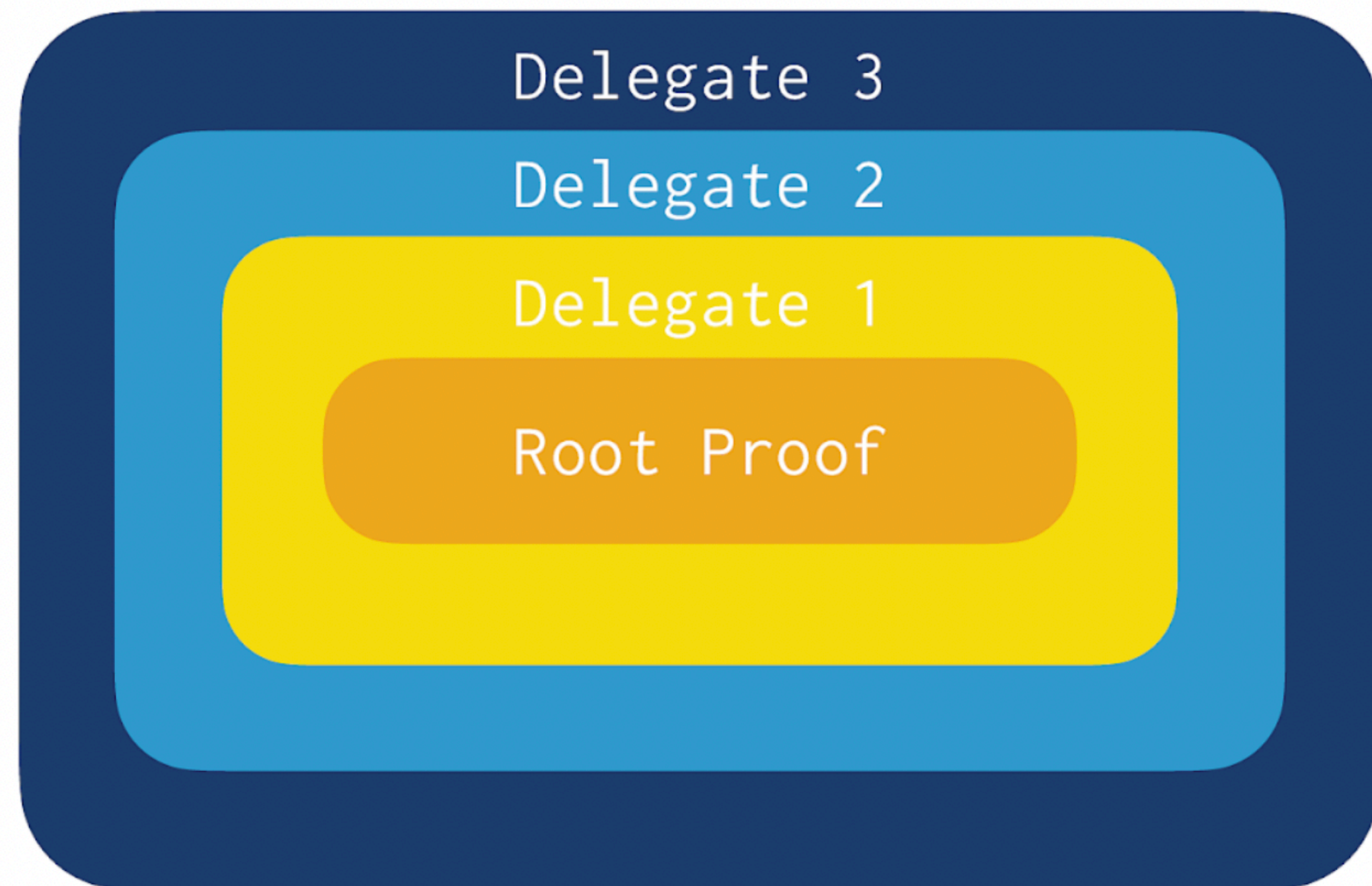
  "att": [
    {
      "wnfs": "boris.fission.name/public/photos/",
      "cap": "OVERWRITE"
    },
    {
      "email": "boris@fission.codes",
      "cap": "SEND"
    }
  ],

  "prf": [
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsInVhdDI6IjAuMS4wIn0.eyJhdWQiOiJkaWQ6a2V5bnpTdD
  ]
}
8XfAytaZS82wHcjoTyoghMyxXiWdR7Nn7A29DNSl0EiXLdwJ6xC6AfgZWF1b0sS_TuYI30G85AmiExREkrS6tD
```

UCAN

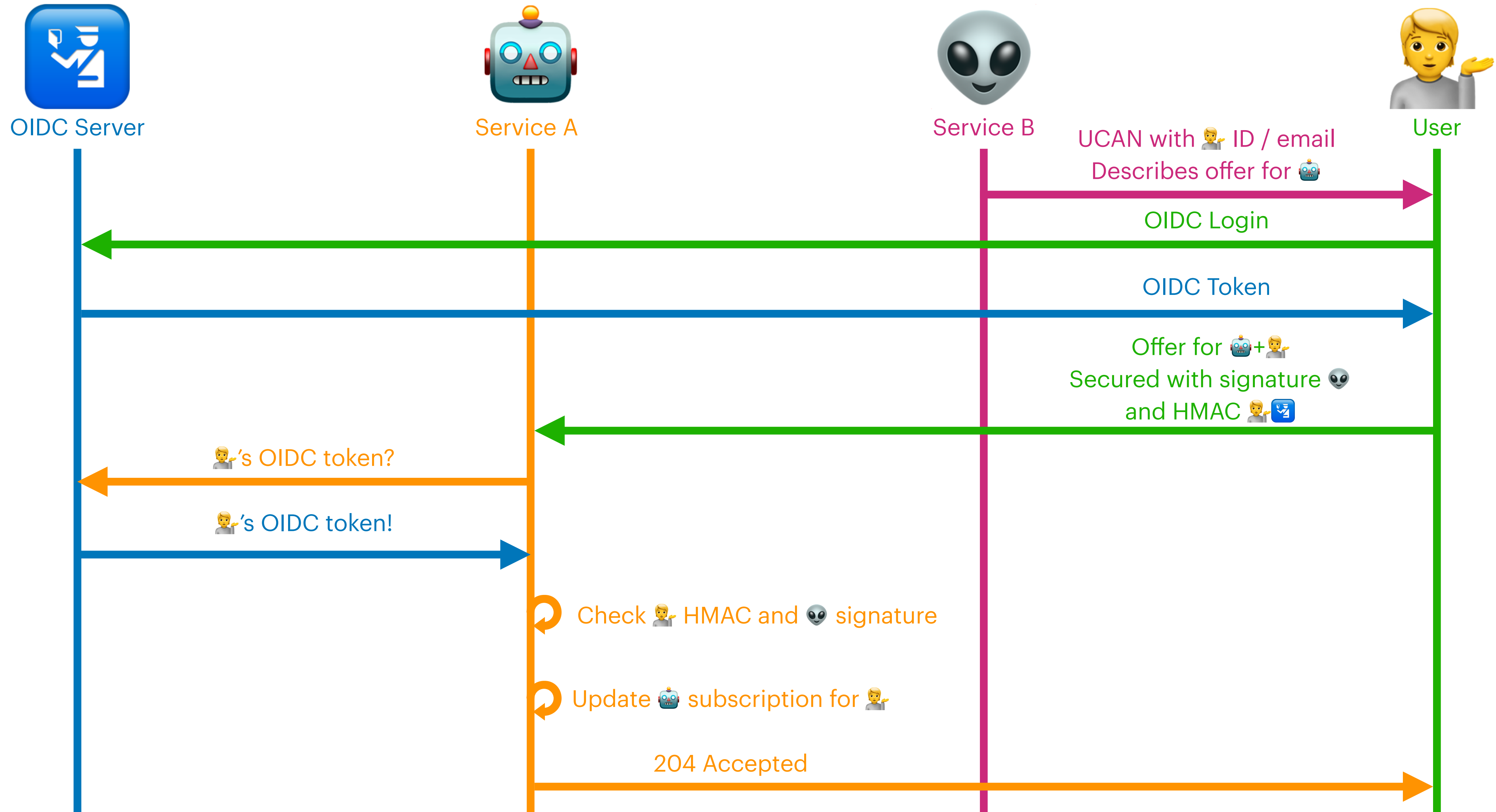
Auth Chaining

- OCAP, provable chains, revocable
- Non-exportable 2048-bit RSA (WebCrypto), Ed25519 & BLS everywhere else



UCAN

Trustless Interop



High Level Auth Topologies

OCAP FileCoin & Accounts



OCAP FileCoin & Accounts

Fully Managed (Similar to Today)

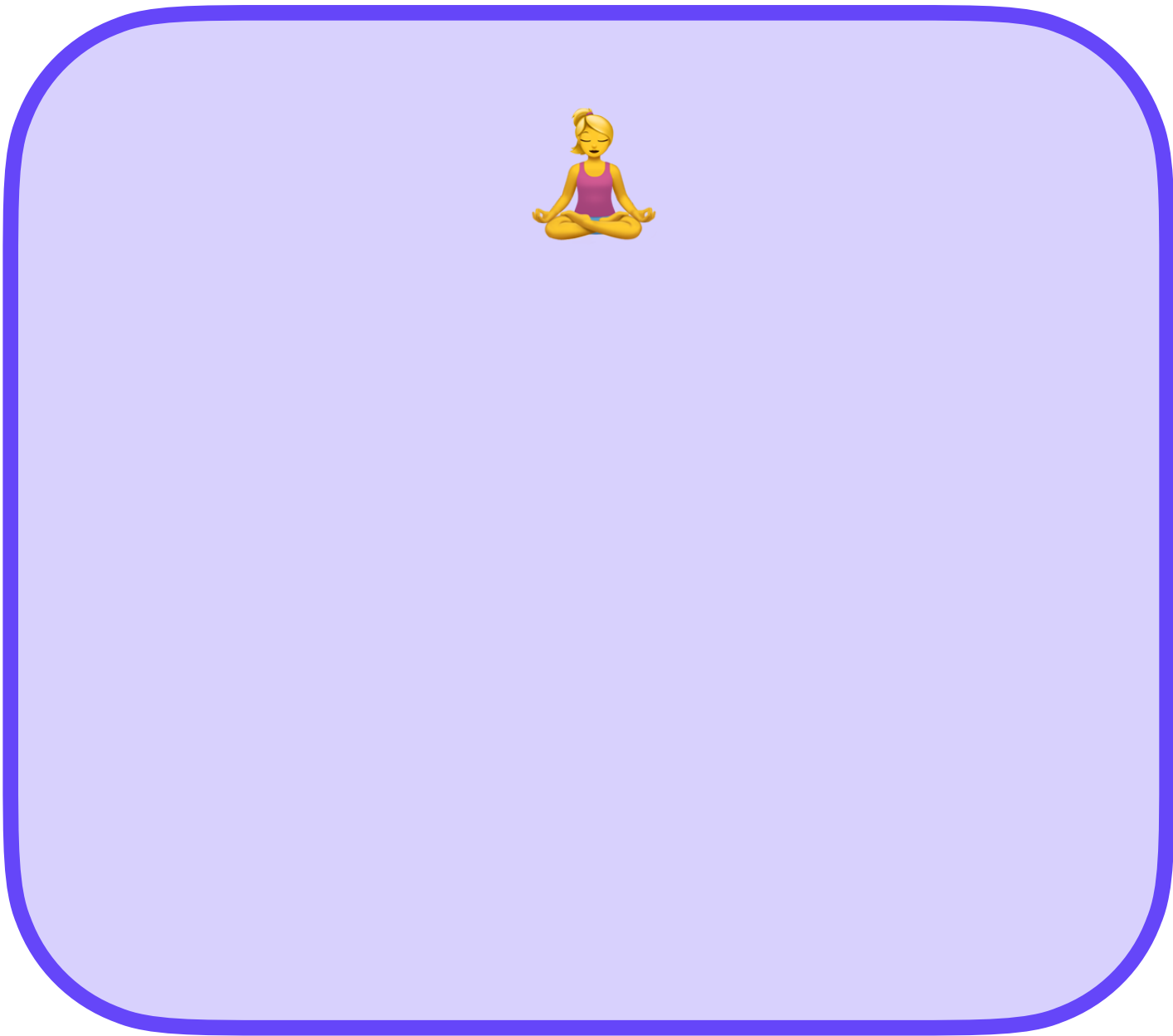
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



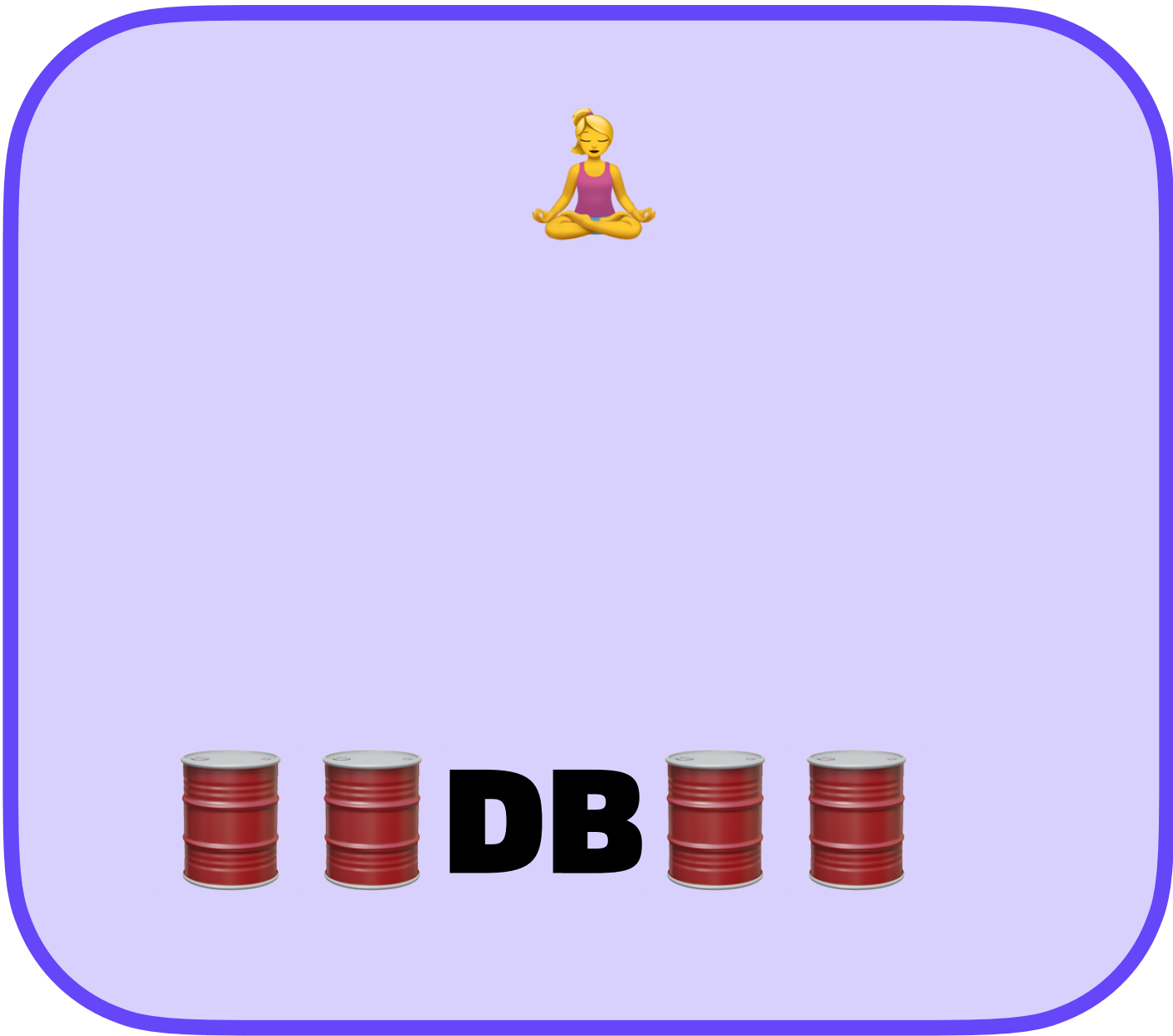
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



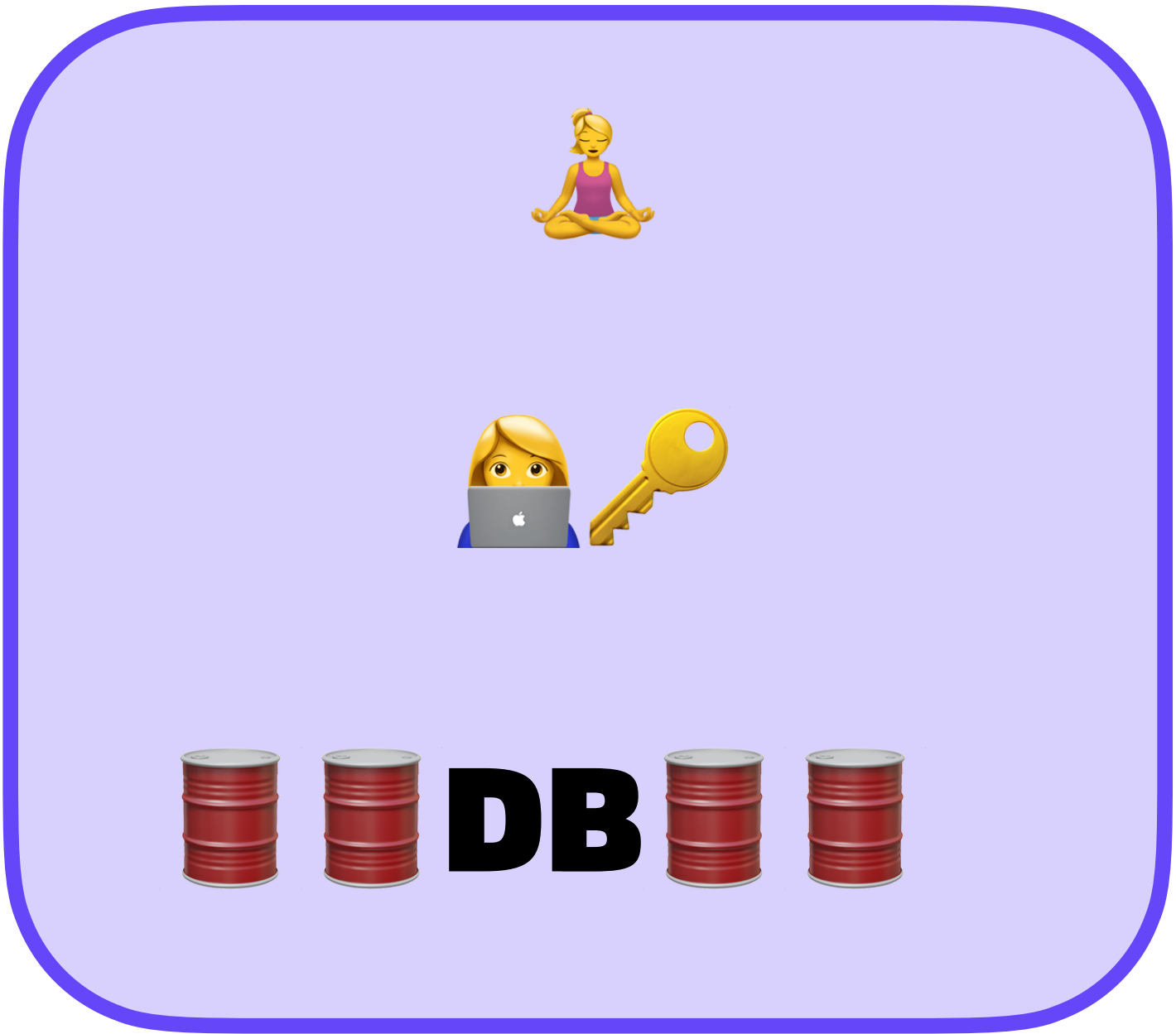
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



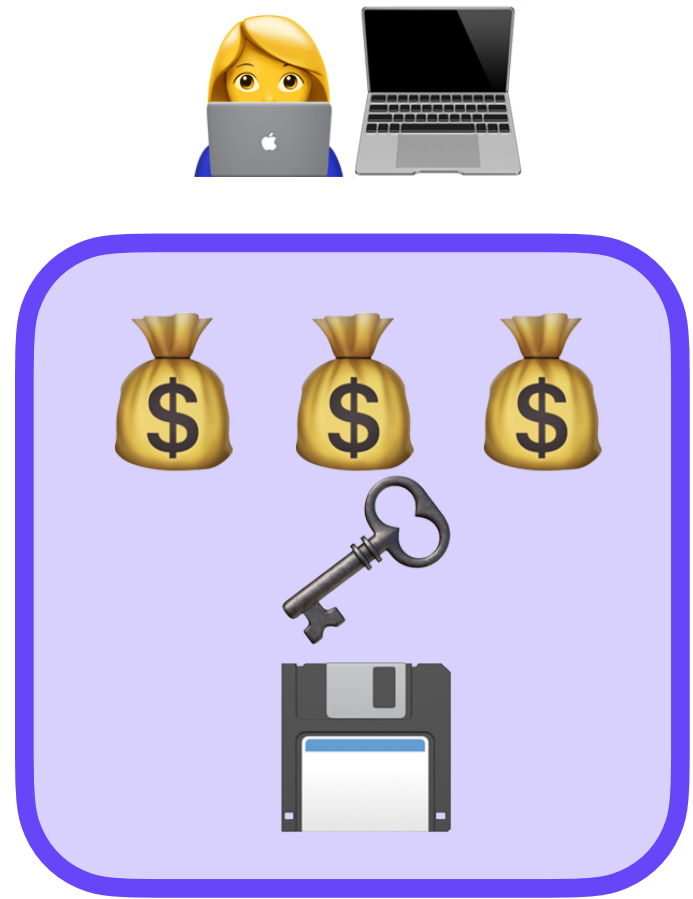
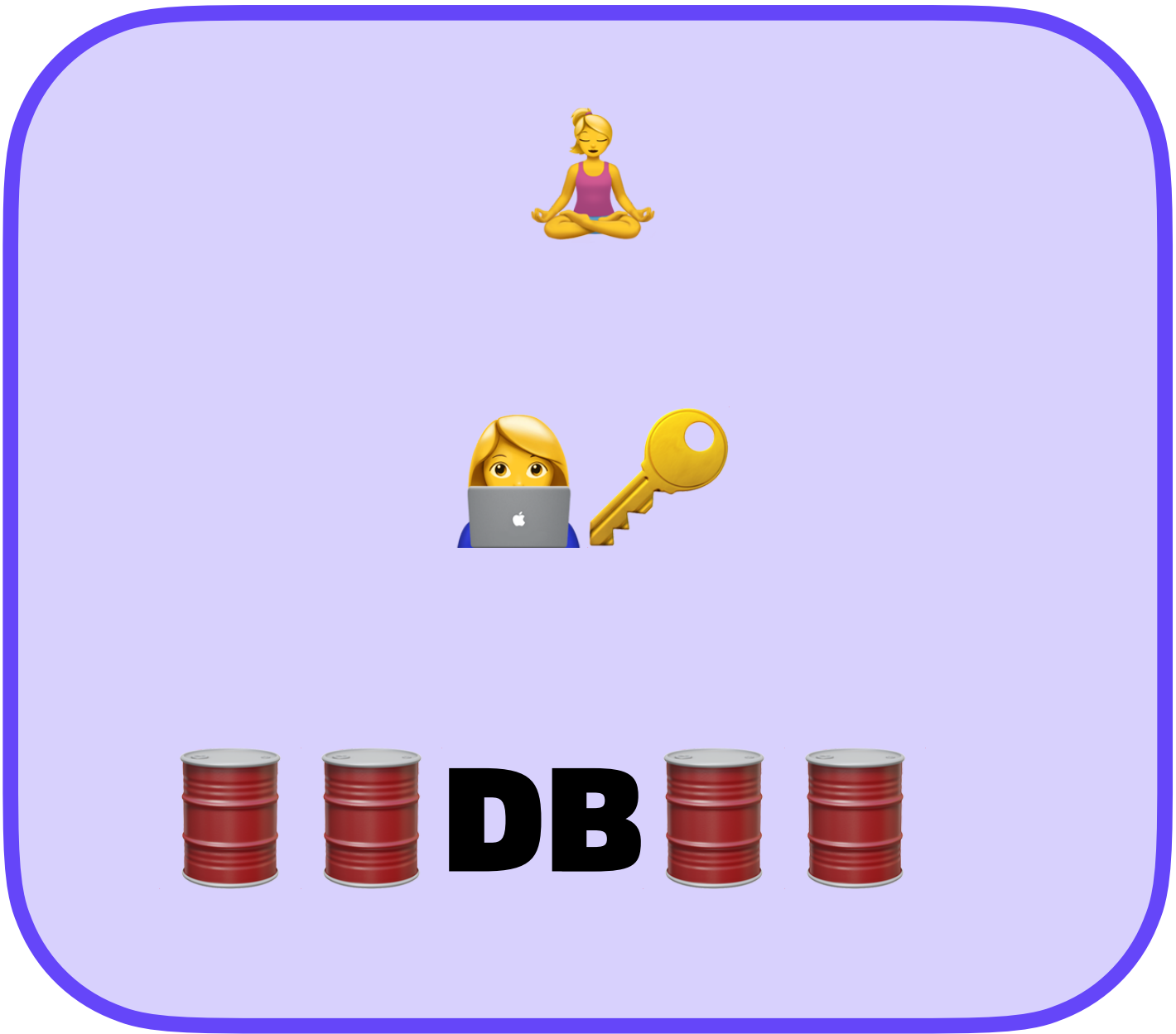
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



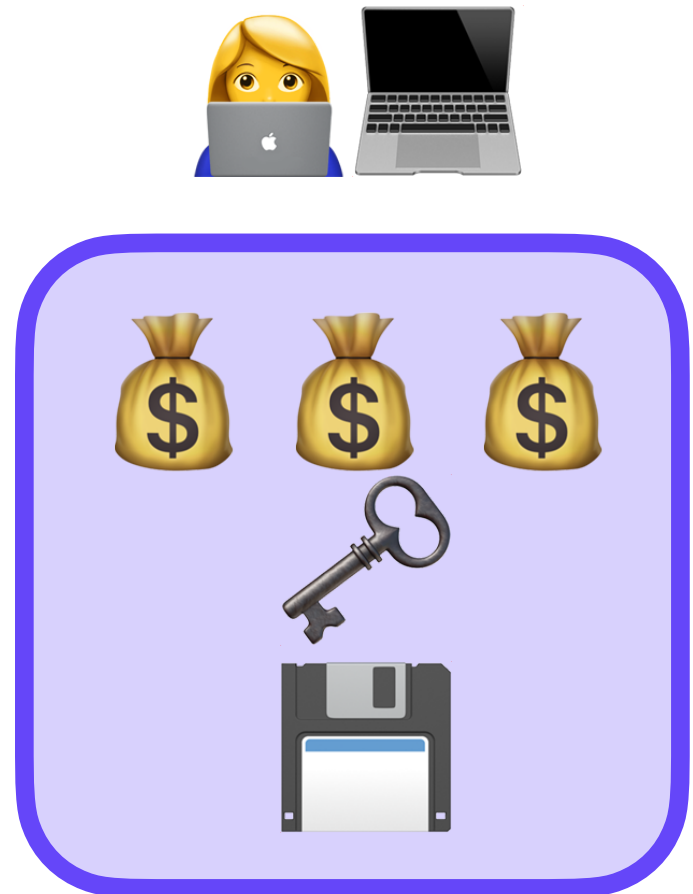
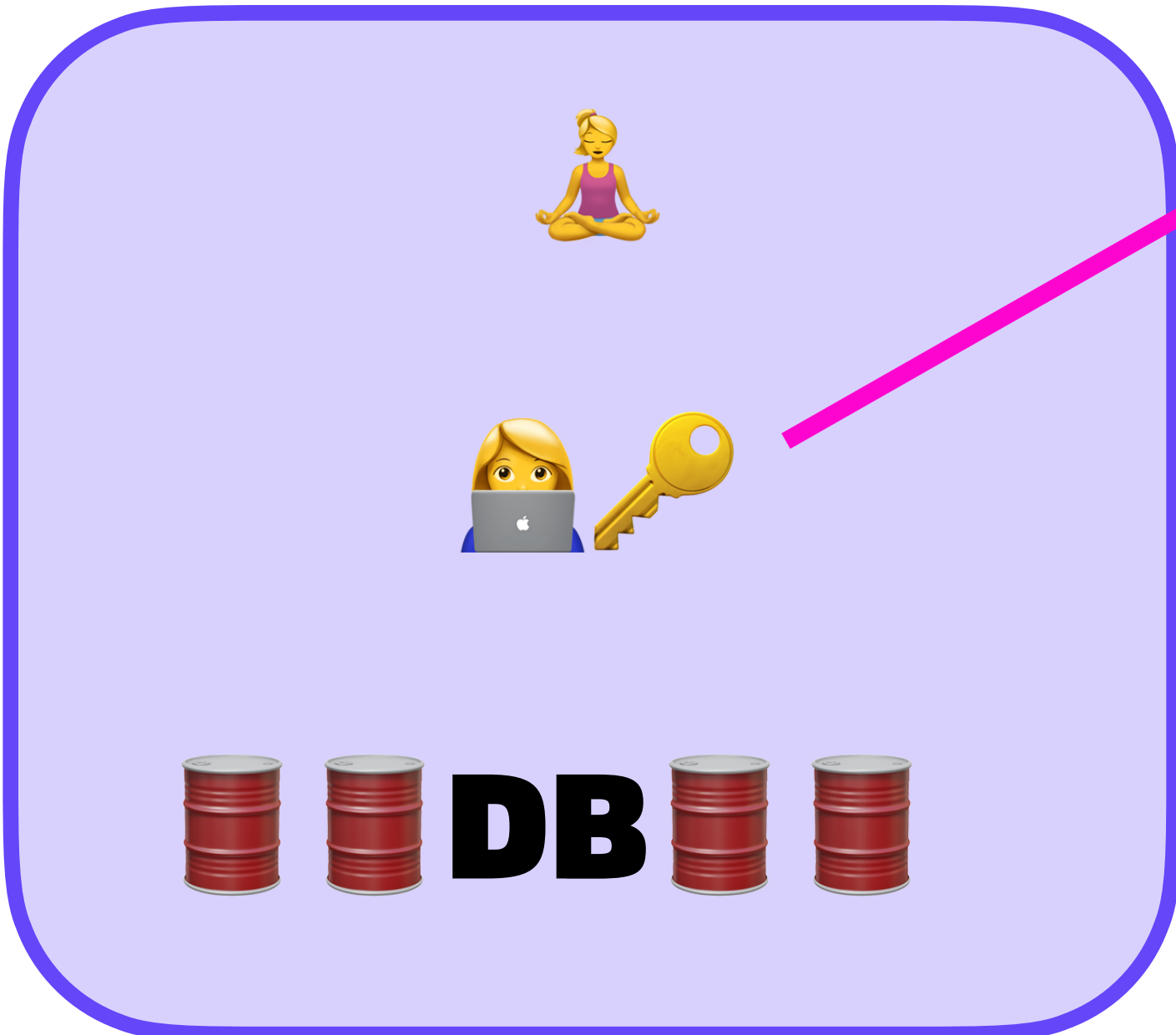
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



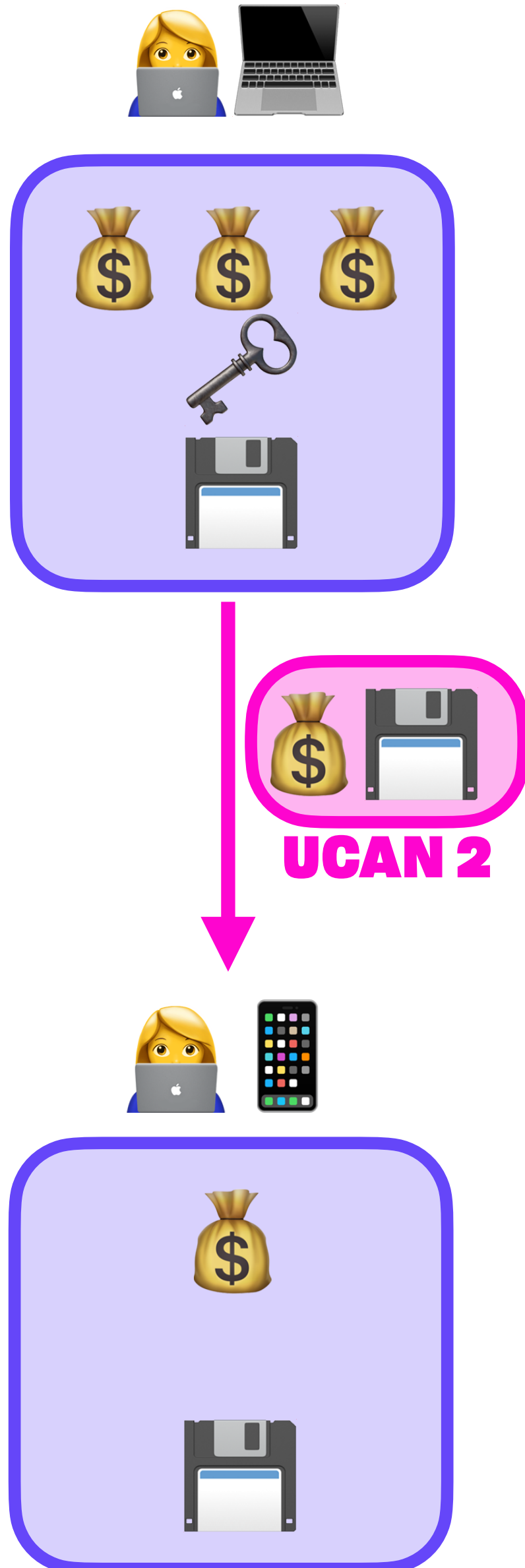
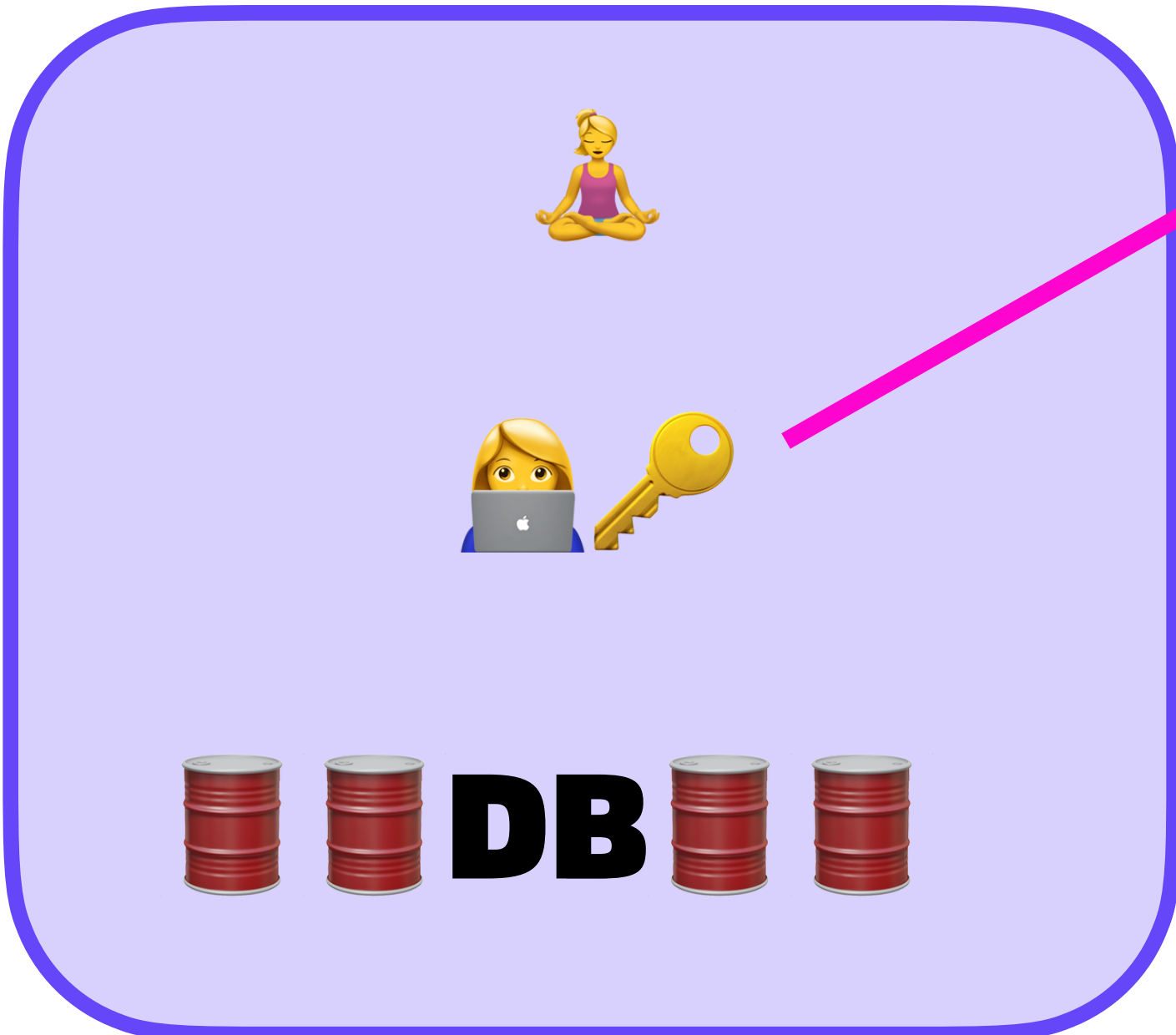
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



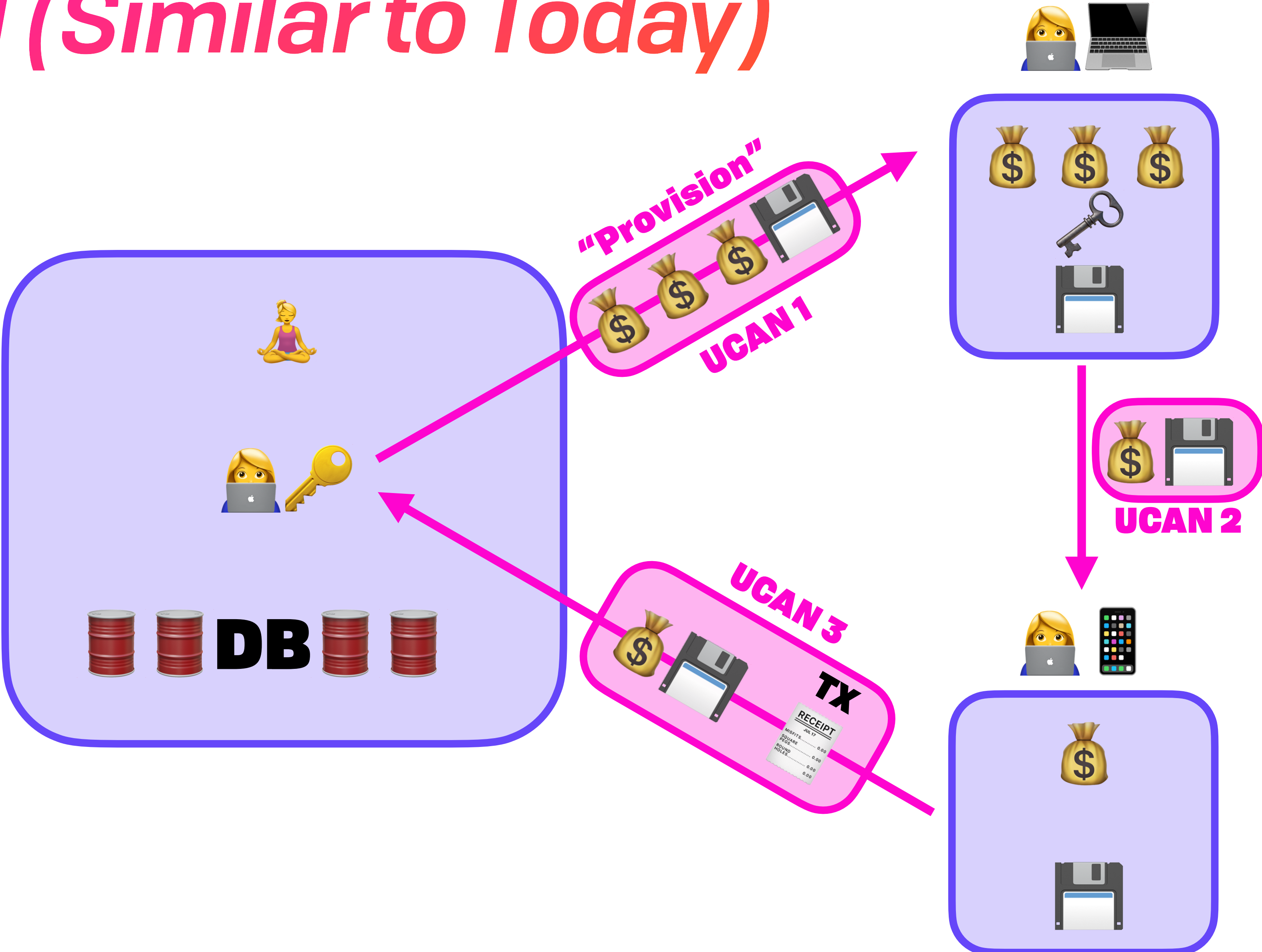
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



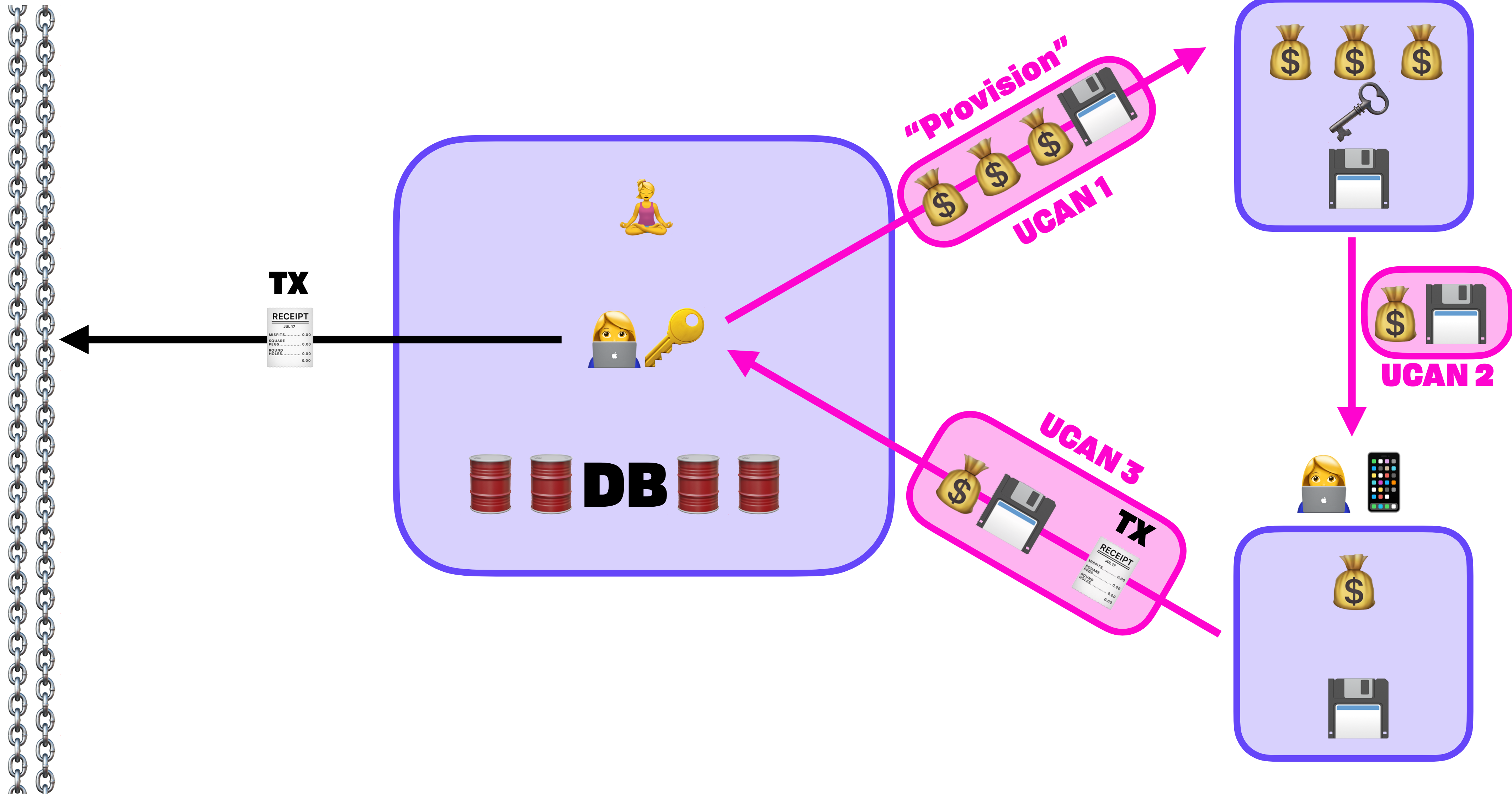
OCAP FileCoin & Accounts

Fully Managed (Similar to Today)



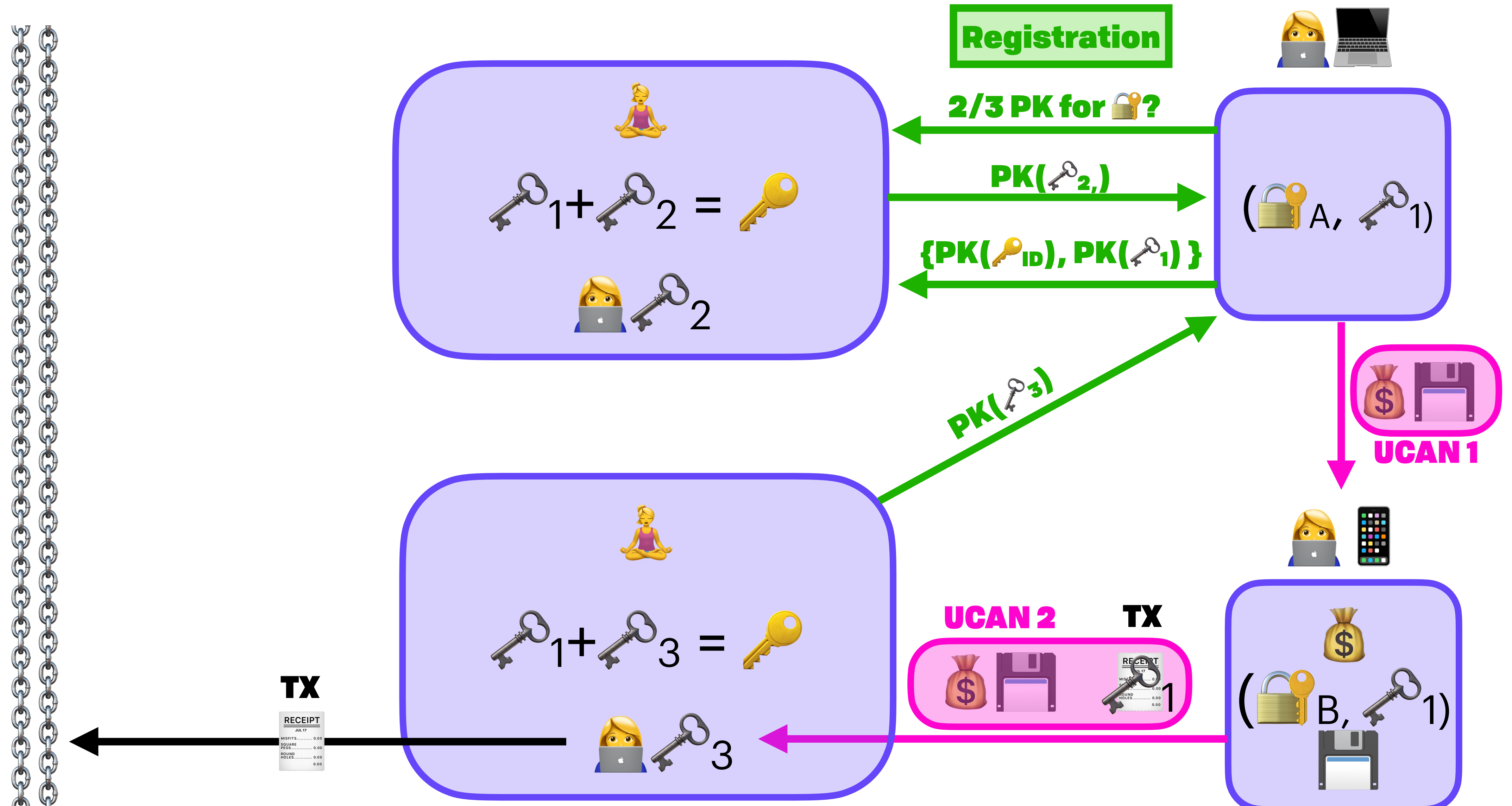
OACAP FileCoin & Accounts

Fully Managed (Similar to Today)



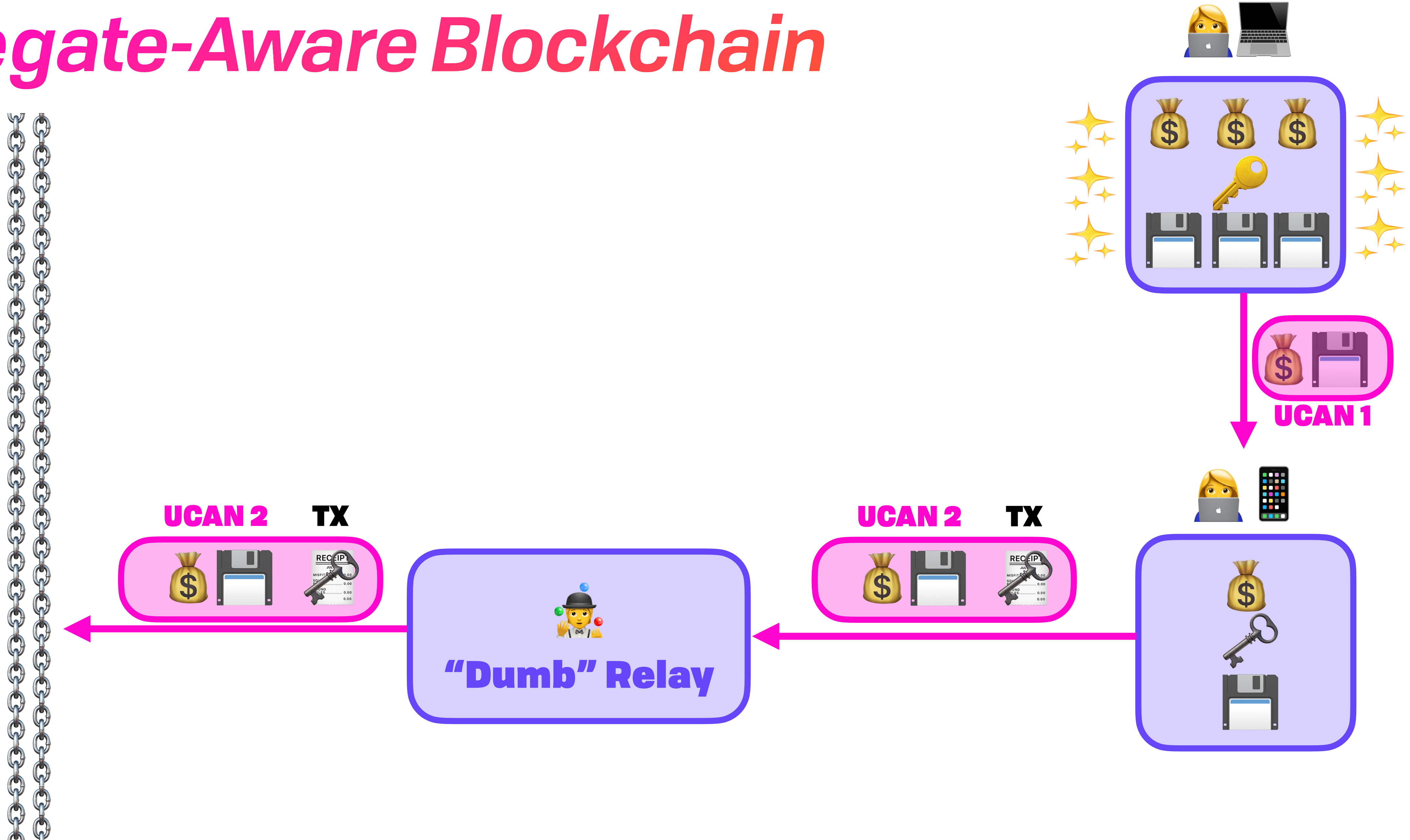
OCAP FileCoin & Accounts

BLS Cosigner (Self Sovereign)



OACAP FileCoin & Accounts

Delegate-Aware Blockchain

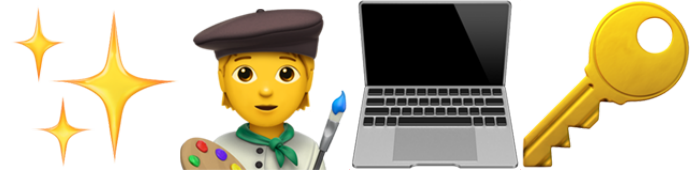


OCAP FileCoin & Accounts

Bonus: Payment Channel Interop

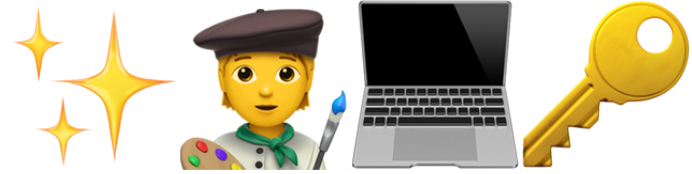
OCAP FileCoin & Accounts

Bonus: Payment Channel Interop

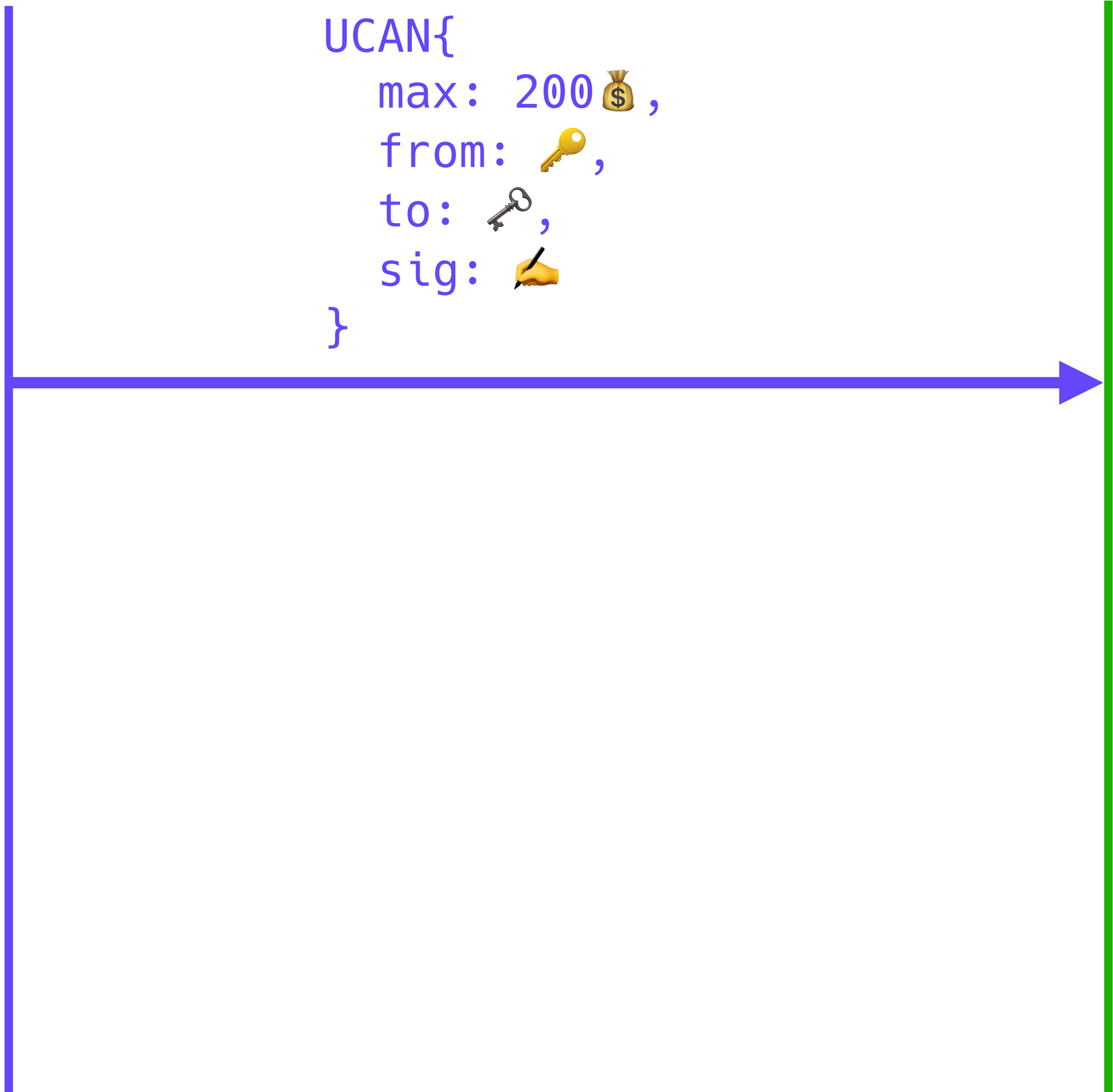


OCAP FileCoin & Accounts

Bonus: Payment Channel Interop

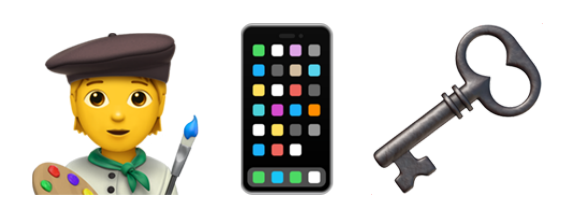


```
UCAN{  
  max: 200$,  
  from: key,  
  to: key,  
  sig: sig  
}
```



OCAP FileCoin & Accounts

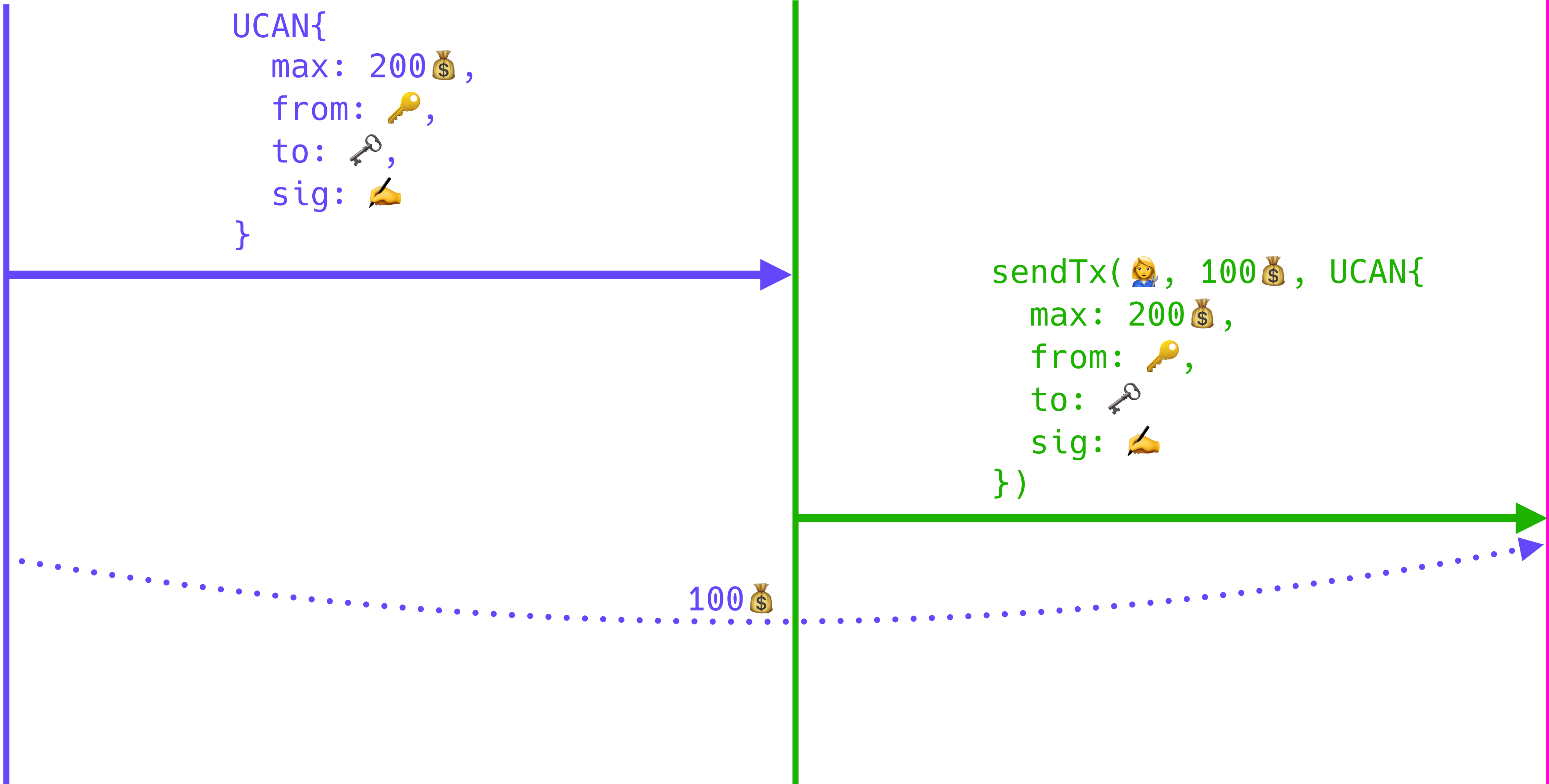
Bonus: Payment Channel Interop



```
UCAN{  
  max: 200 $ ,  
  from: key ,  
  to: key ,  
  sig: sig  
}
```

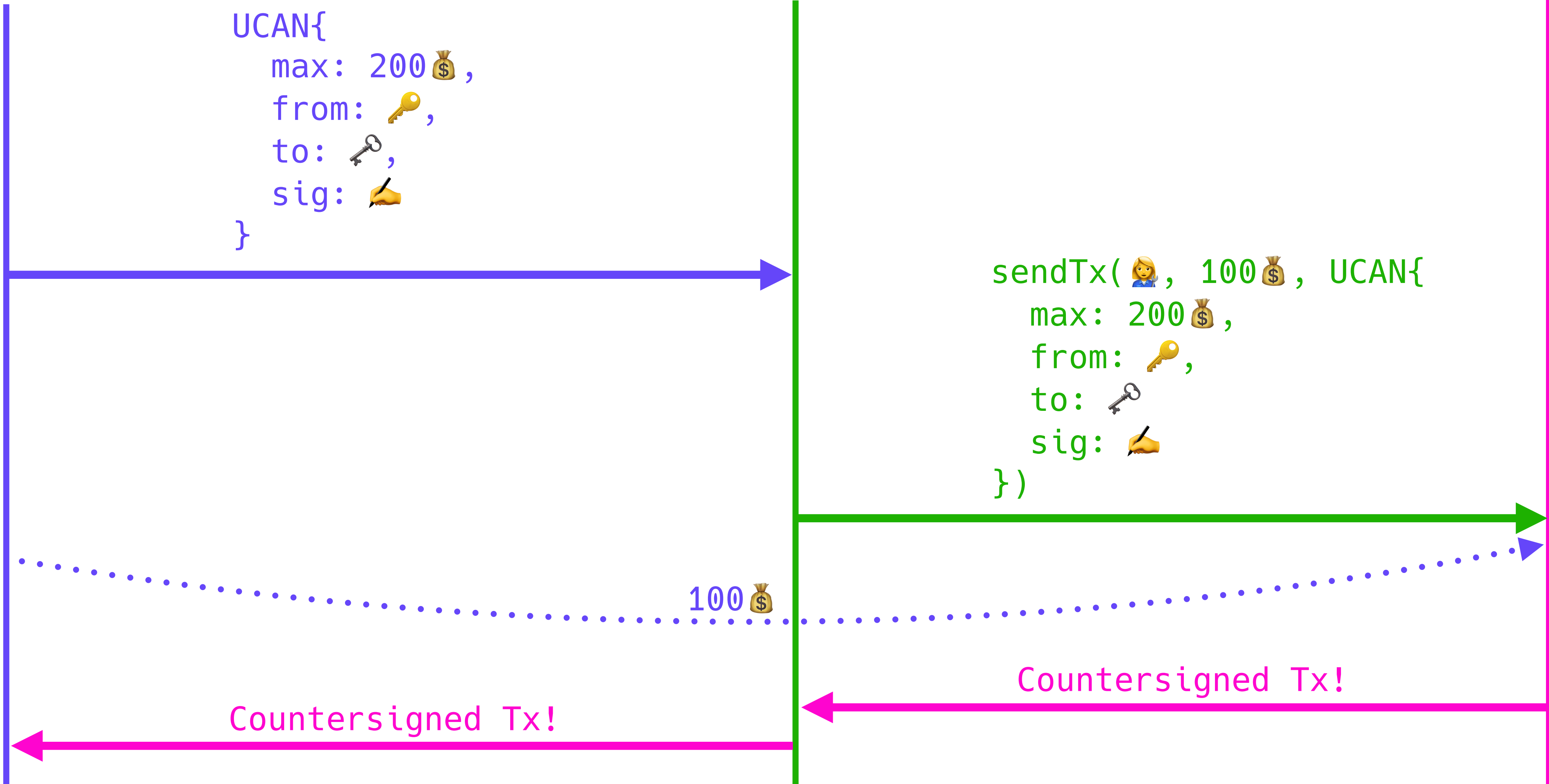
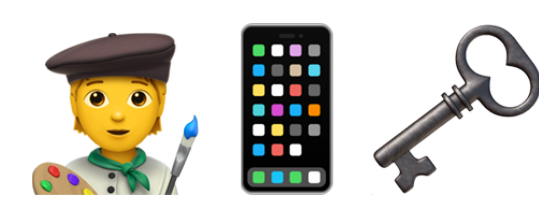
```
sendTx( person , 100 $ , UCAN{  
  max: 200 $ ,  
  from: key ,  
  to: key ,  
  sig: sig  
})
```

100 \$



OCAP FileCoin & Accounts

Bonus: Payment Channel Interop



```
UCAN{
  max: 200$,
  from: key,
  to: key,
  sig: hand
}
```

```
sendTx(P3, 100$, UCAN{
  max: 200$,
  from: key,
  to: key,
  sig: hand
})
```

100\$

Countersigned Tx!

Countersigned Tx!

User Controlled, Serverless, Universal Auth & ID

Read vs Write



Securing Data Access

WNFS Layout

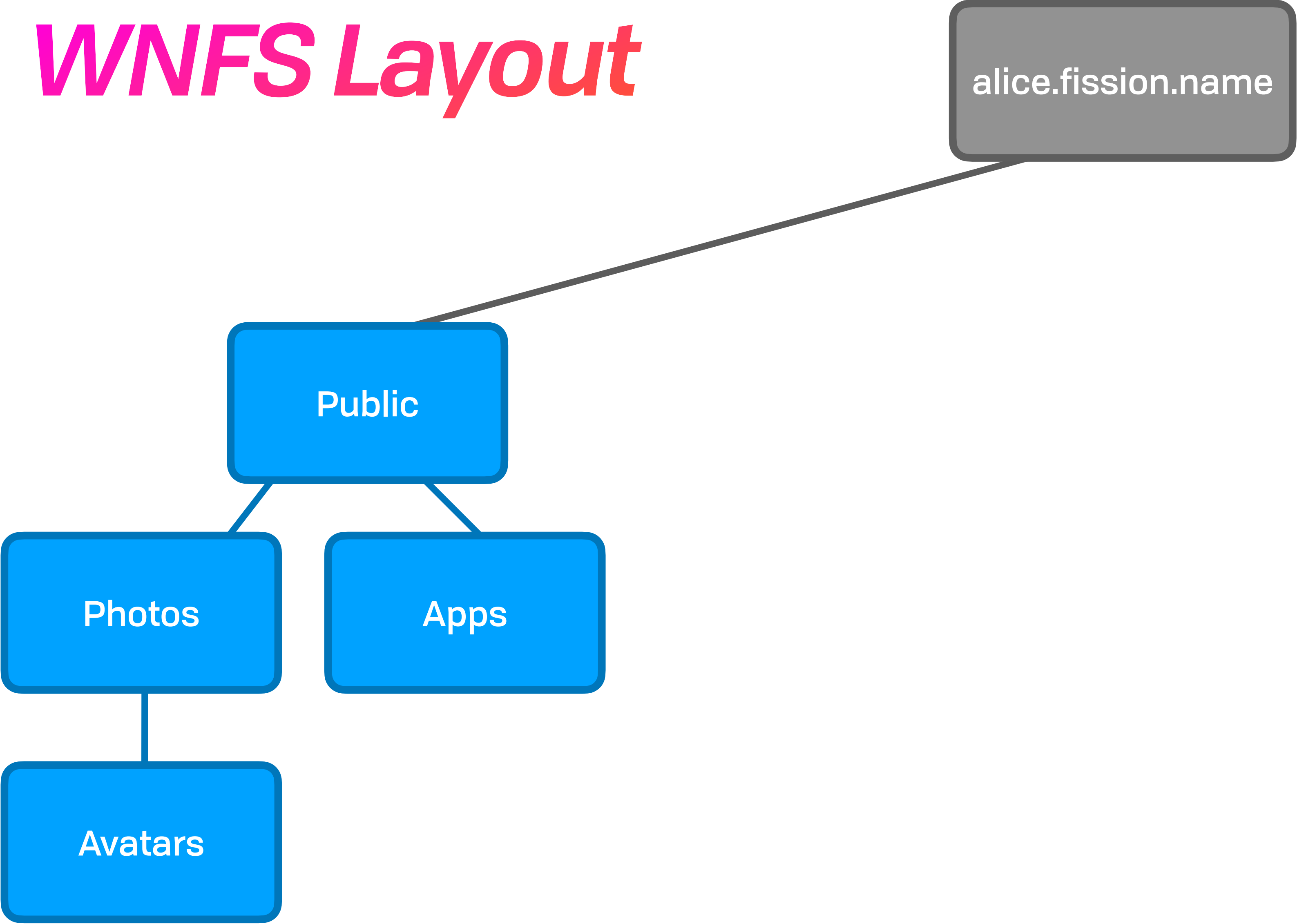
Securing Data Access

WNFS Layout

alice.fission.name

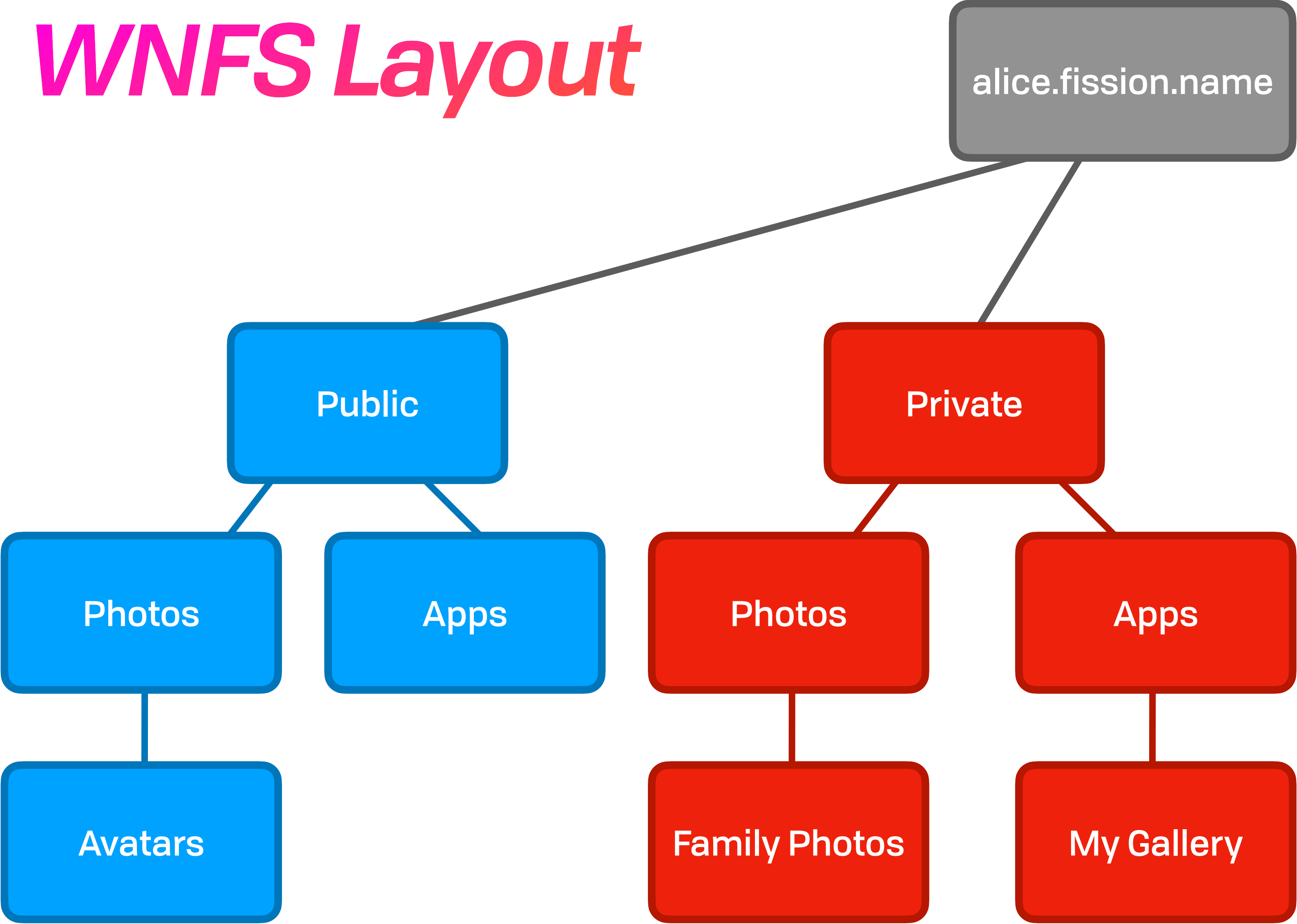
Securing Data Access

WNFS Layout



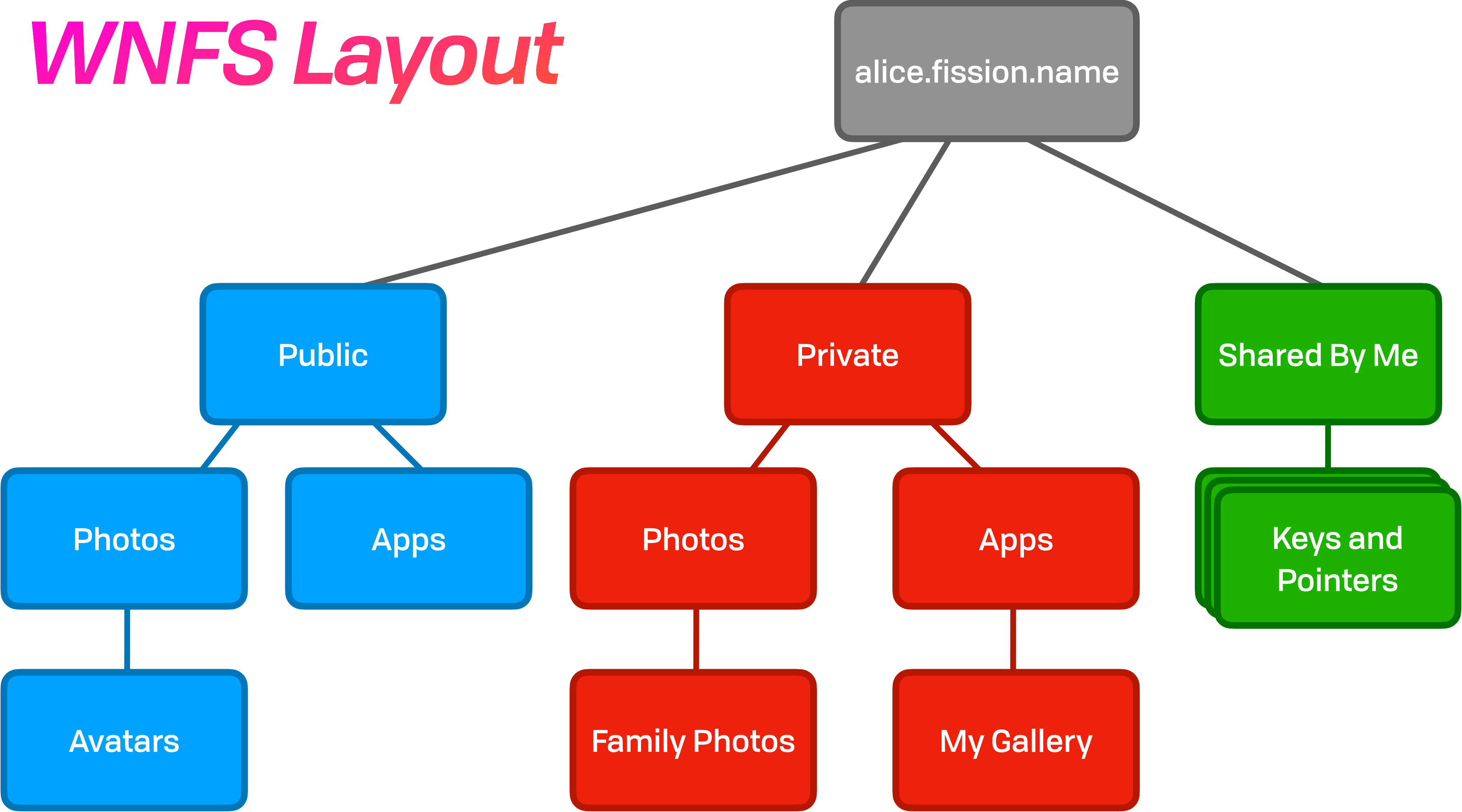
Securing Data Access

WNFS Layout



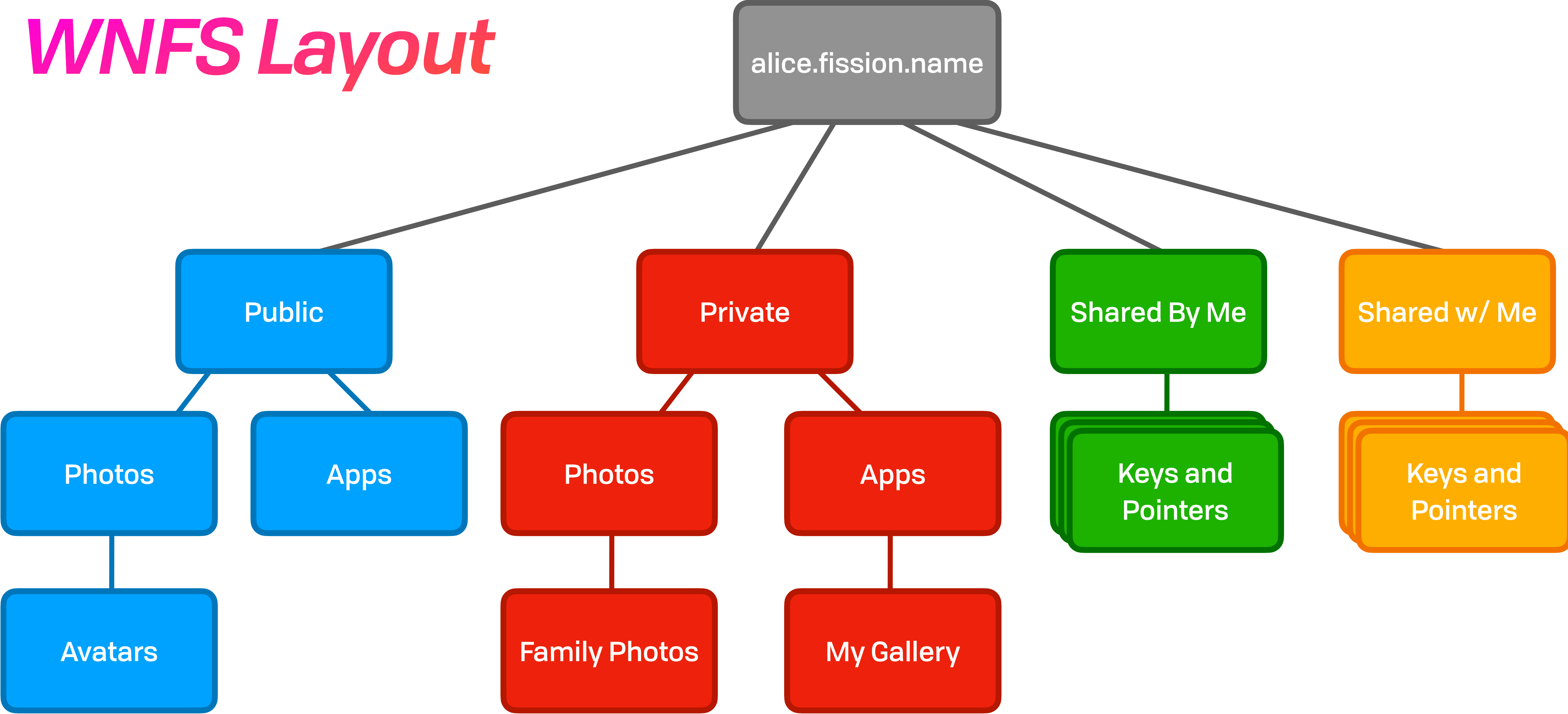
Securing Data Access

WNFS Layout



Securing Data Access

WNFS Layout

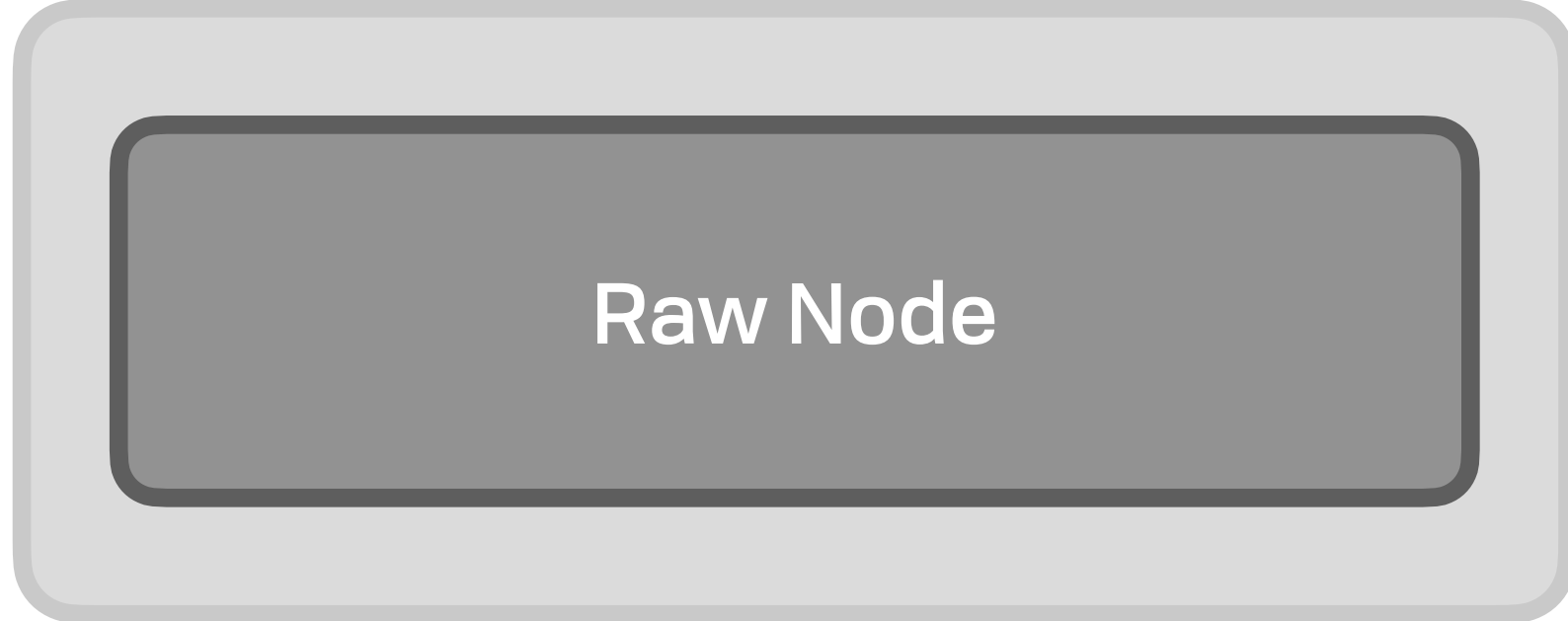


Securing Data Access

Virtual Nodes

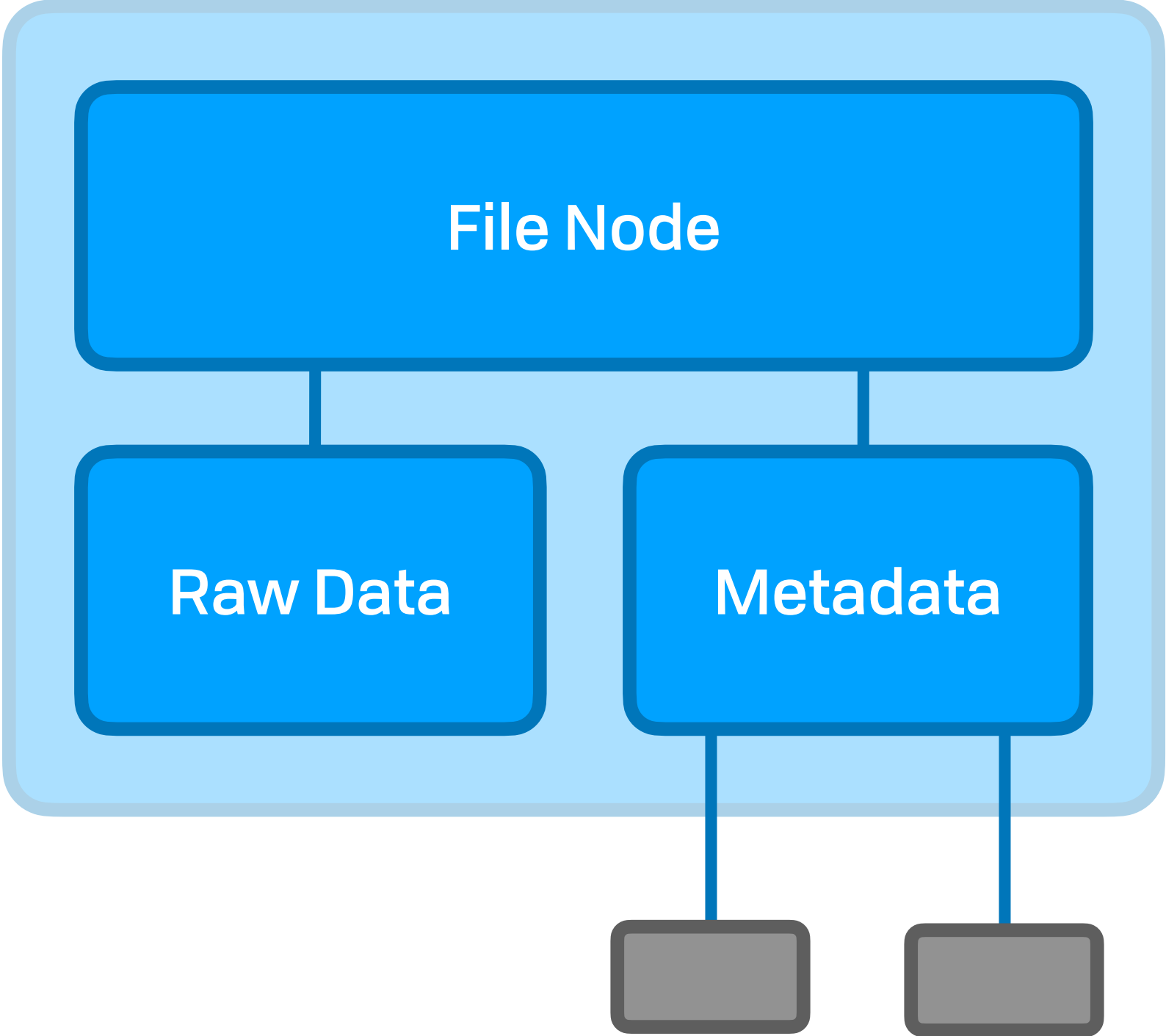
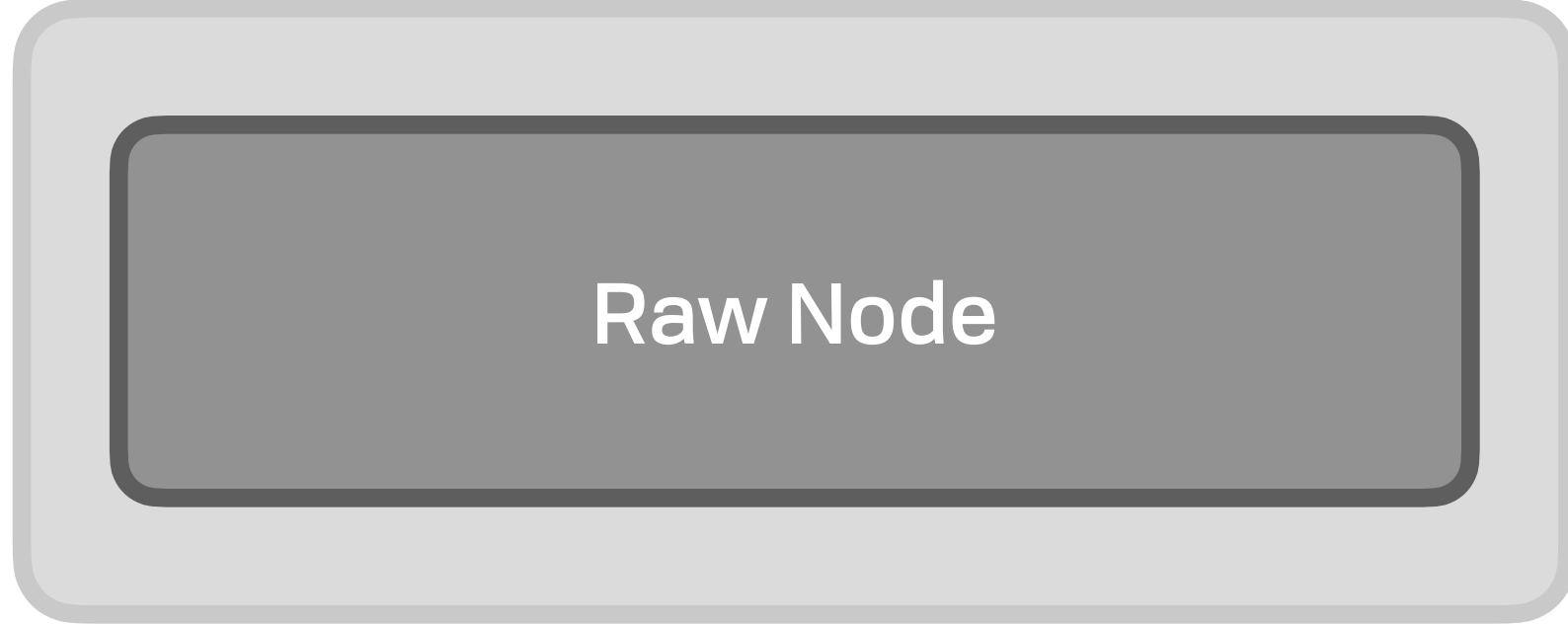
Securing Data Access

Virtual Nodes



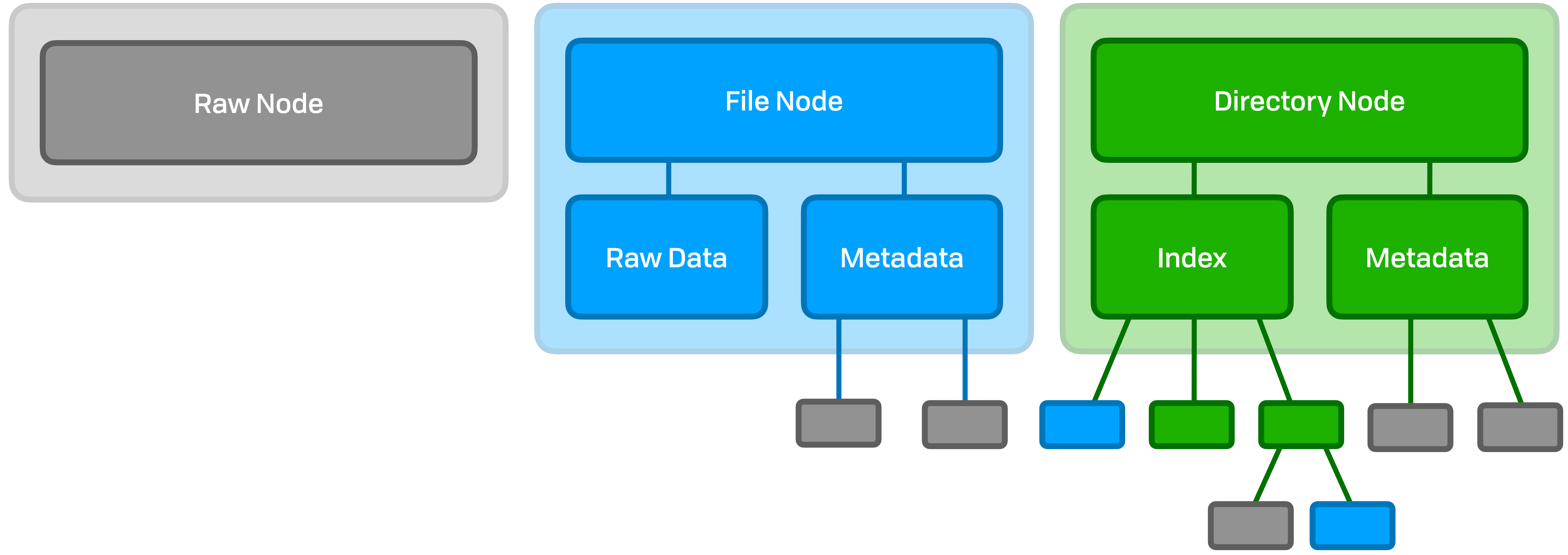
Securing Data Access

Virtual Nodes



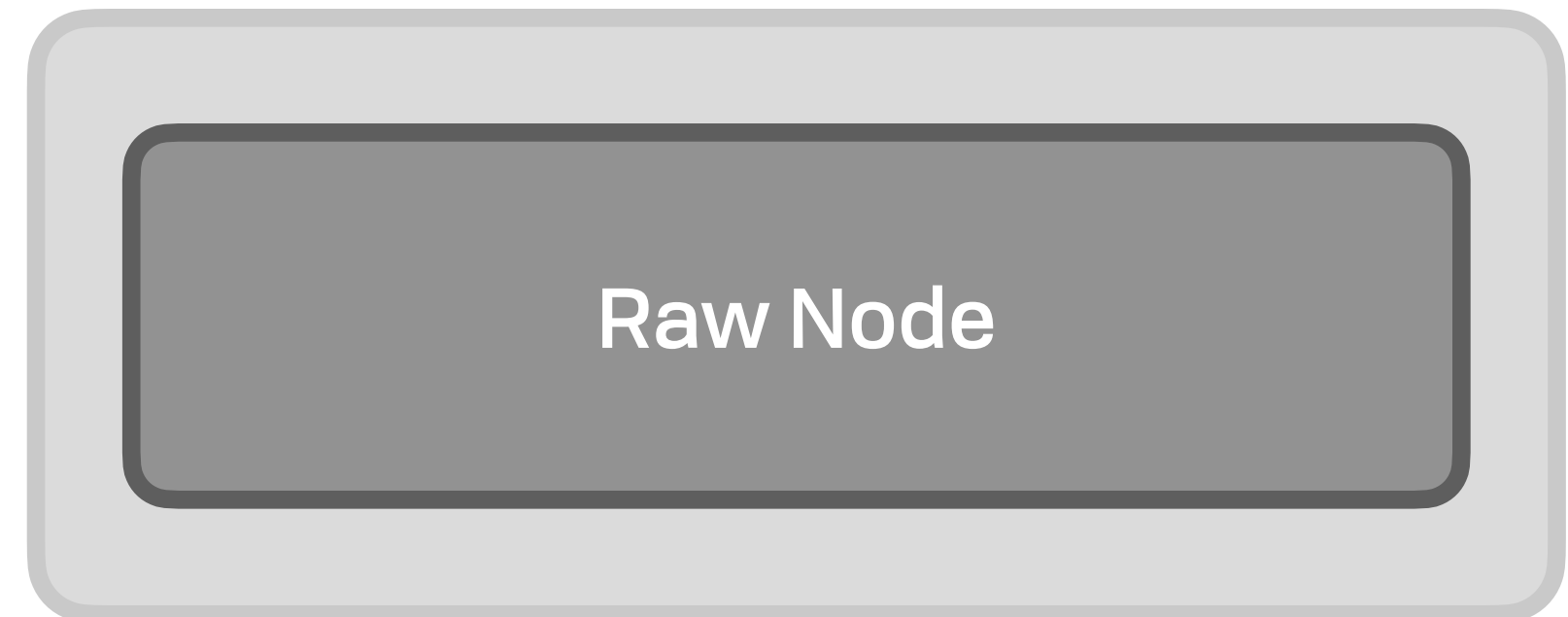
Securing Data Access

Virtual Nodes

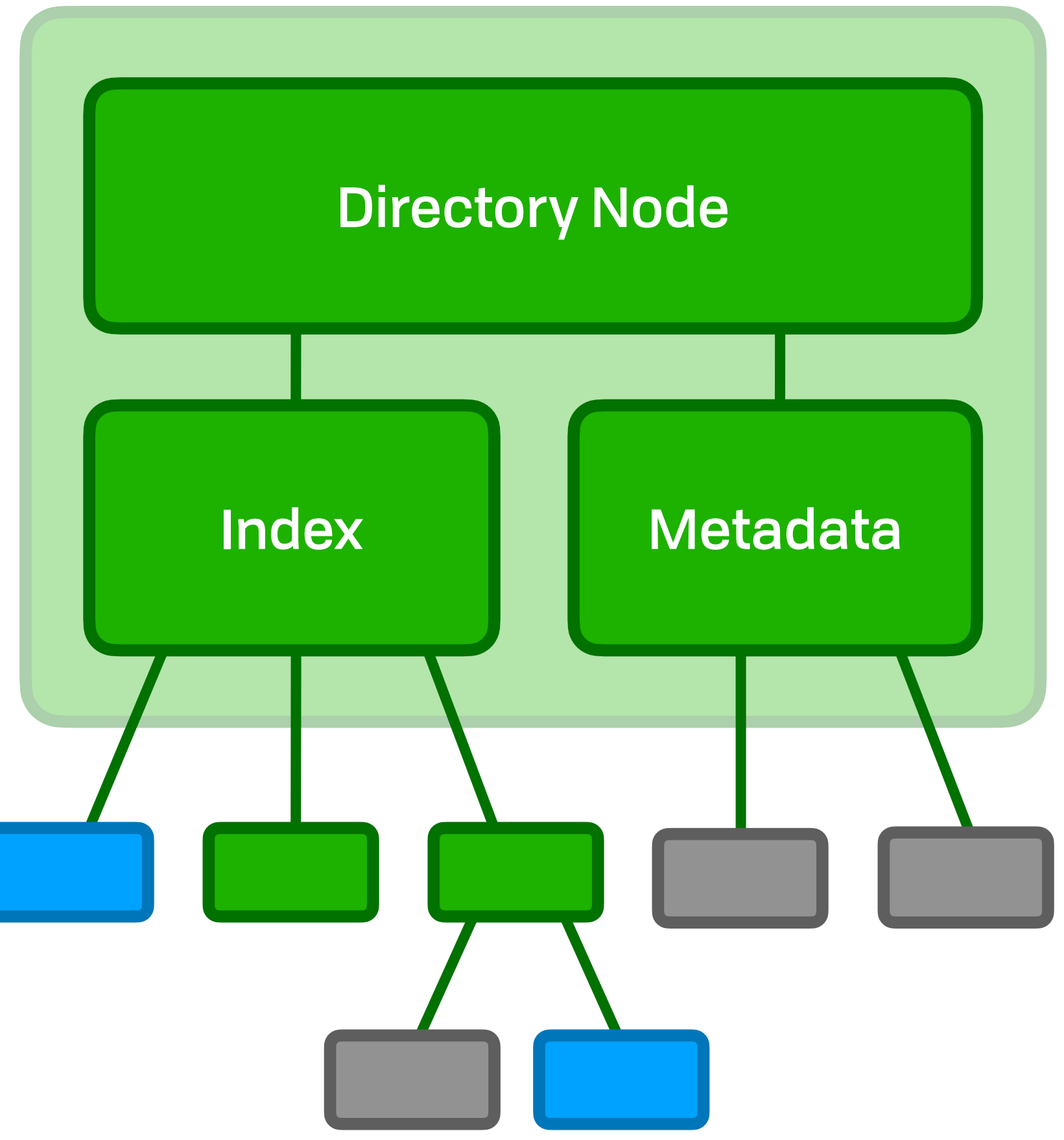
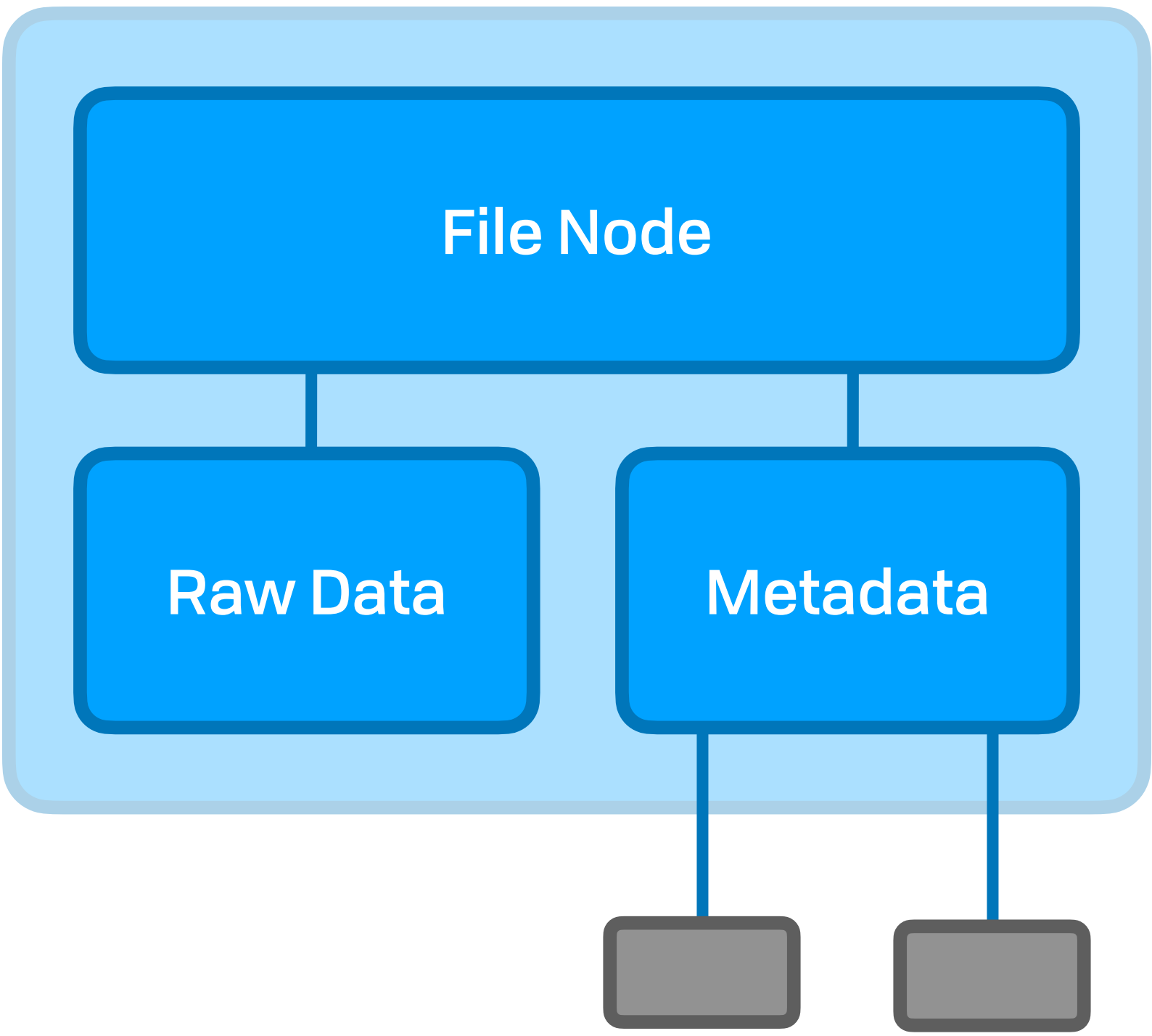


Securing Data Access

Virtual Nodes



- Virtual Node
 - Consistent interface
- Arbitrary metadata
 - Tags, creators, MIME, sources, &c



Securing Data Access

Hard & Soft Links

Securing Data Access

Hard & Soft Links

- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate

Securing Data Access

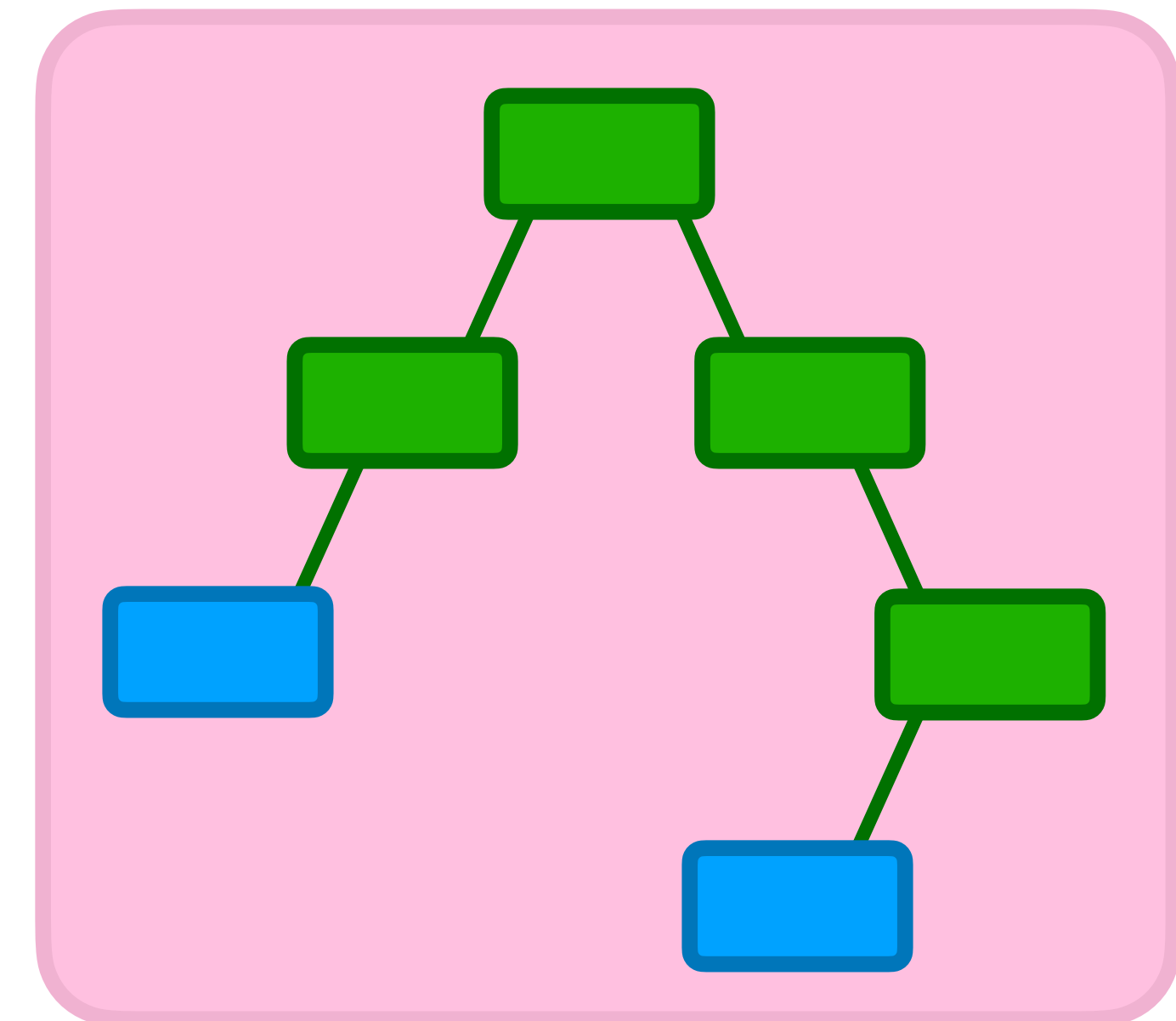
Hard & Soft Links

- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate
- Soft links
 - Like a symlink or web link
 - 2 pointers ~ latest
 - May break
 - Always some version available

Securing Data Access

Hard & Soft Links

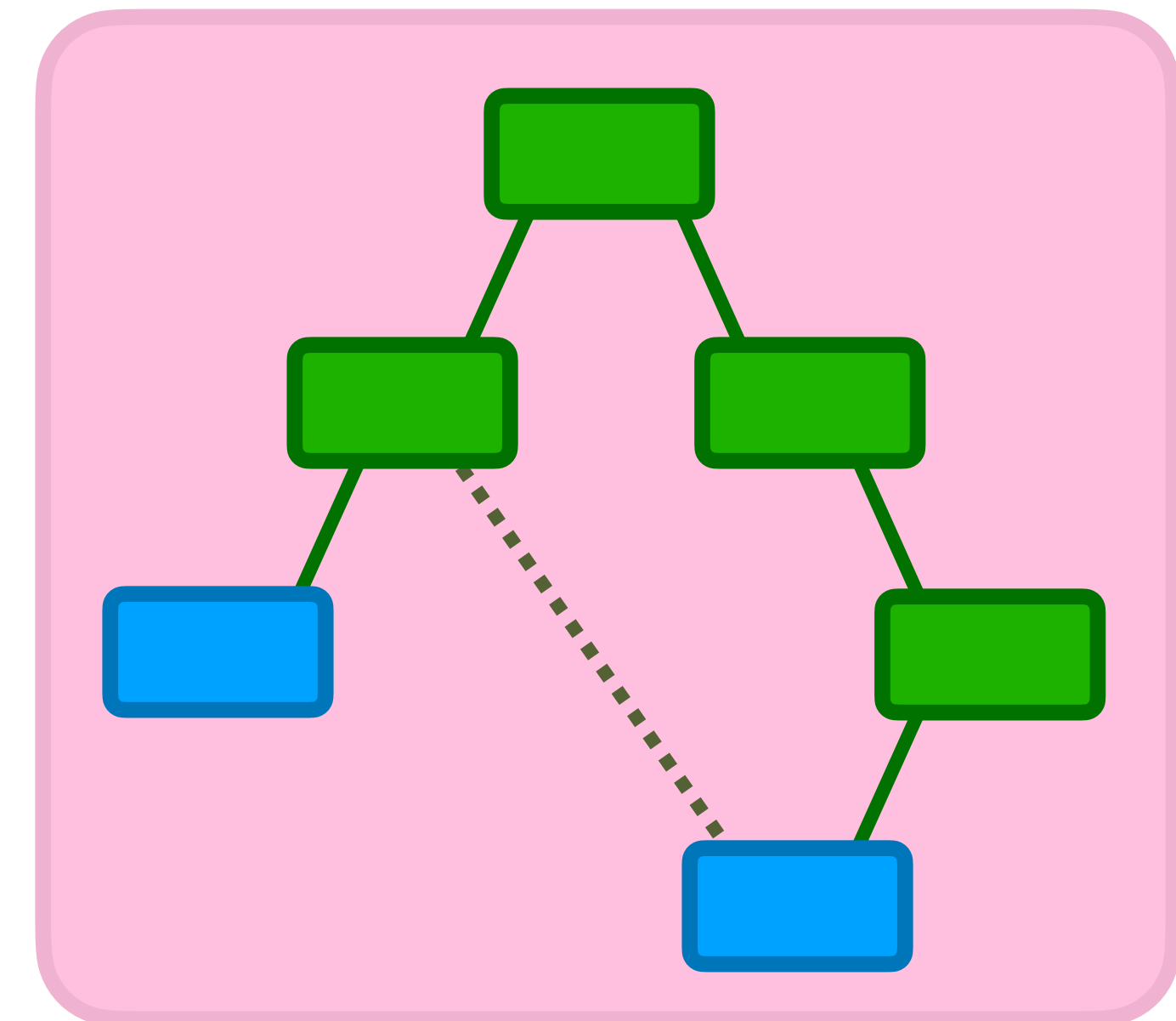
- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate
- Soft links
 - Like a symlink or web link
 - 2 pointers ~ latest
 - May break
 - Always some version available



Securing Data Access

Hard & Soft Links

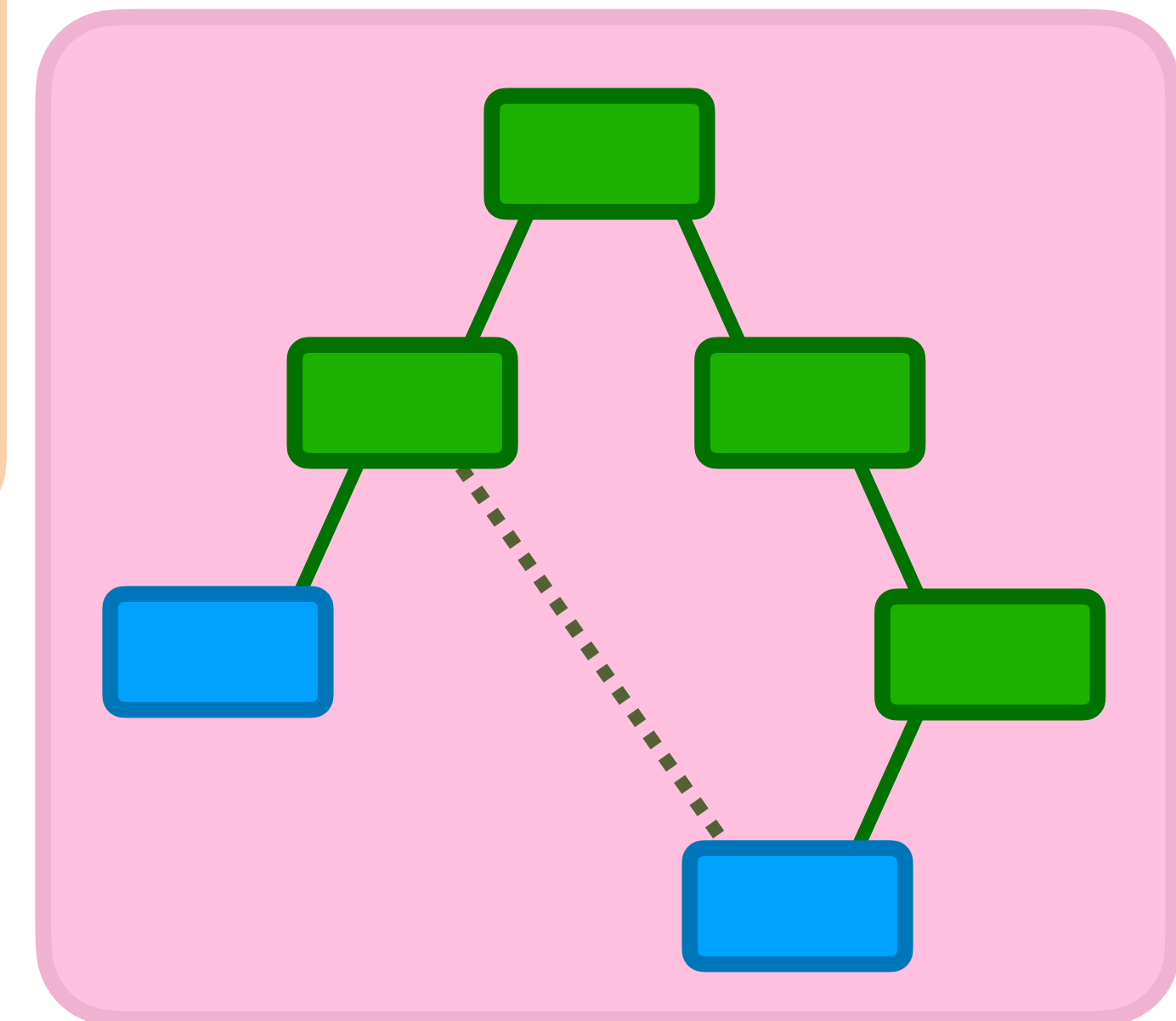
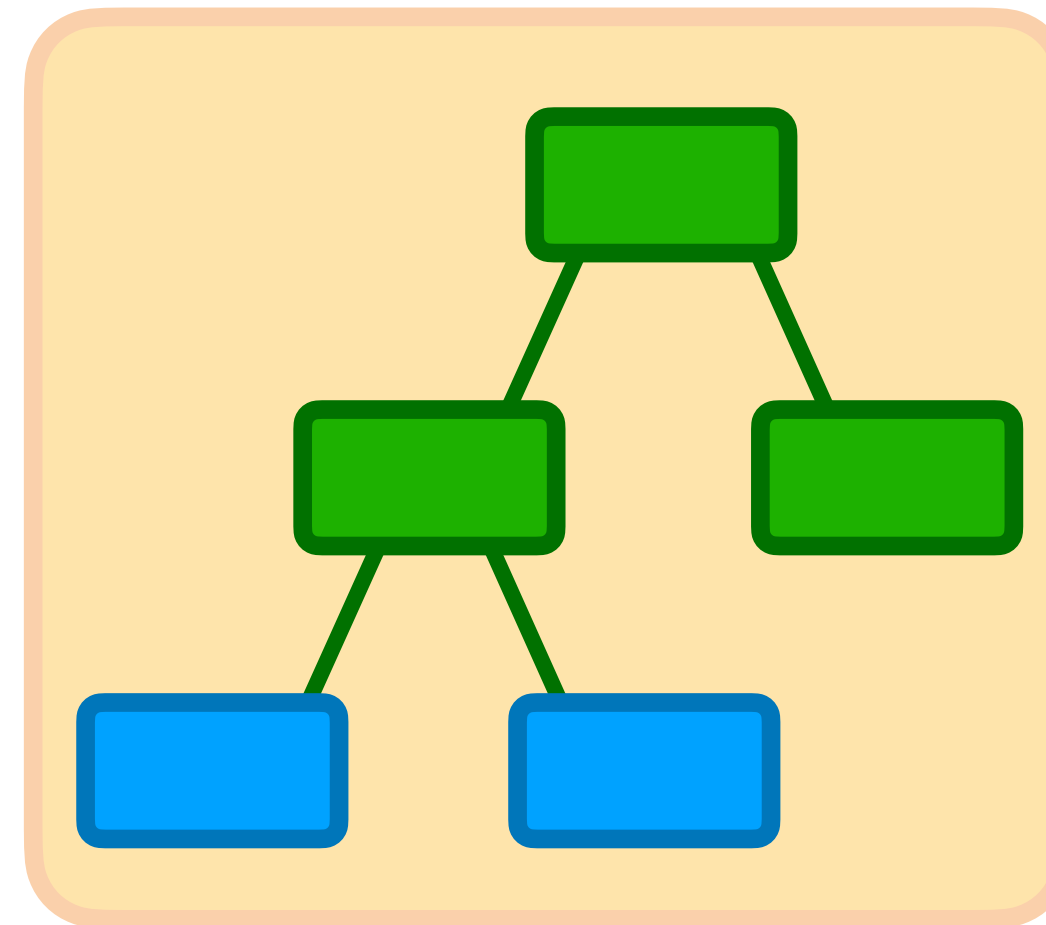
- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate
- Soft links
 - Like a symlink or web link
 - 2 pointers ~ latest
 - May break
 - Always some version available



Securing Data Access

Hard & Soft Links

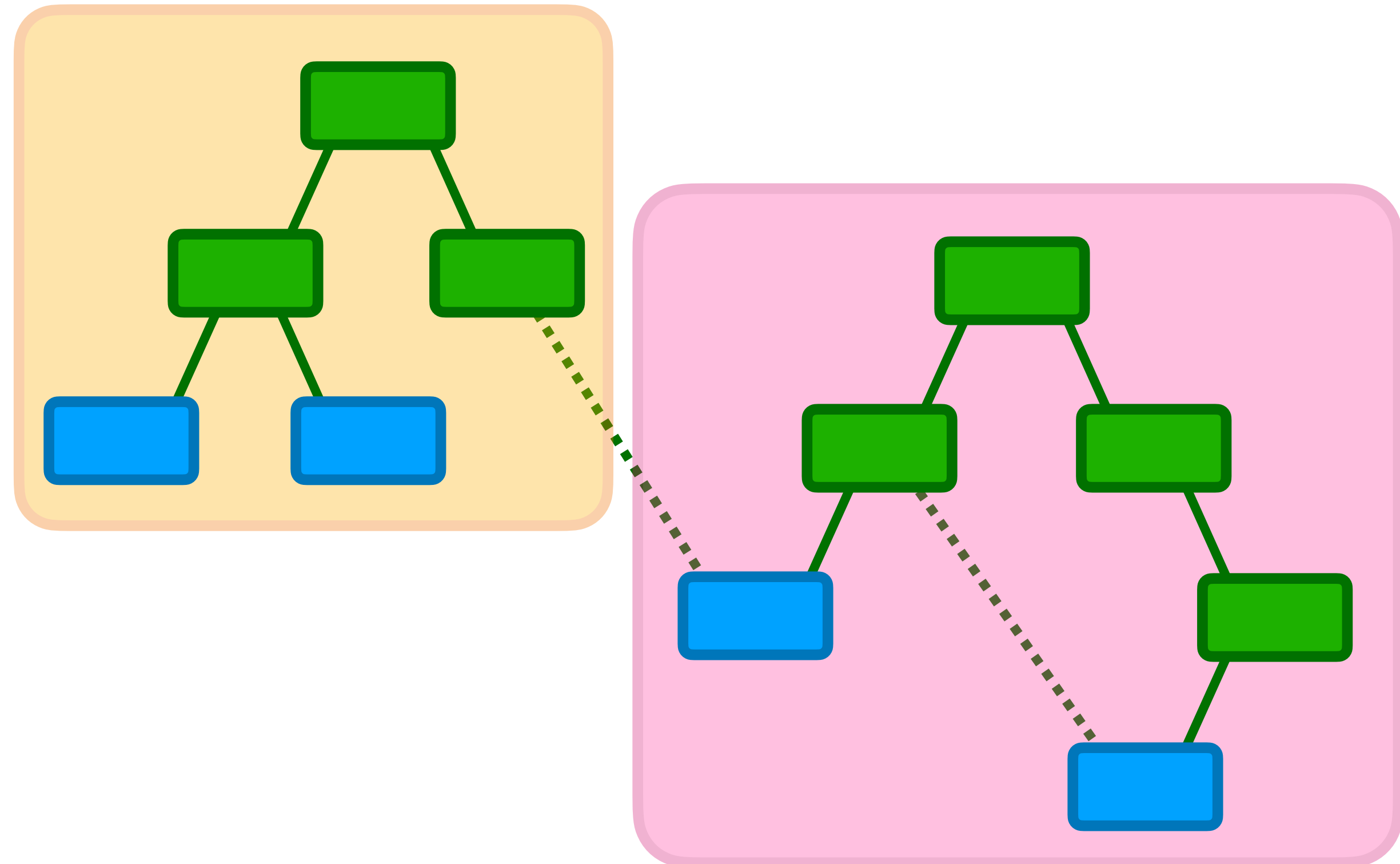
- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate
- Soft links
 - Like a symlink or web link
 - 2 pointers ~ latest
 - May break
 - Always some version available



Securing Data Access

Hard & Soft Links

- Hard links
 - New for the web!
 - Direct reference
 - 2 pointers ~ duplicate
- Soft links
 - Like a symlink or web link
 - 2 pointers ~ latest
 - May break
 - Always some version available

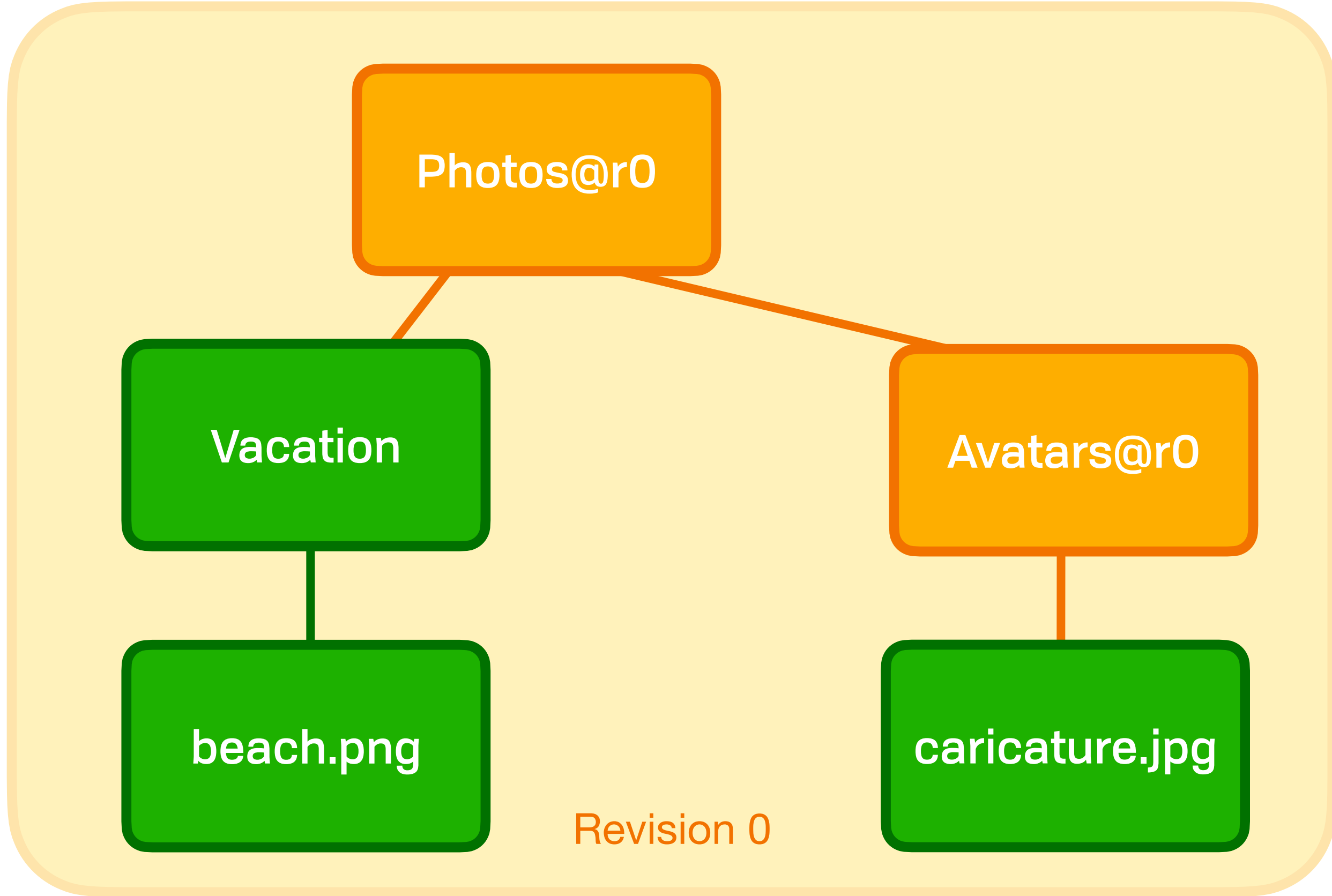


Securing Data Access

Persistent Versioning

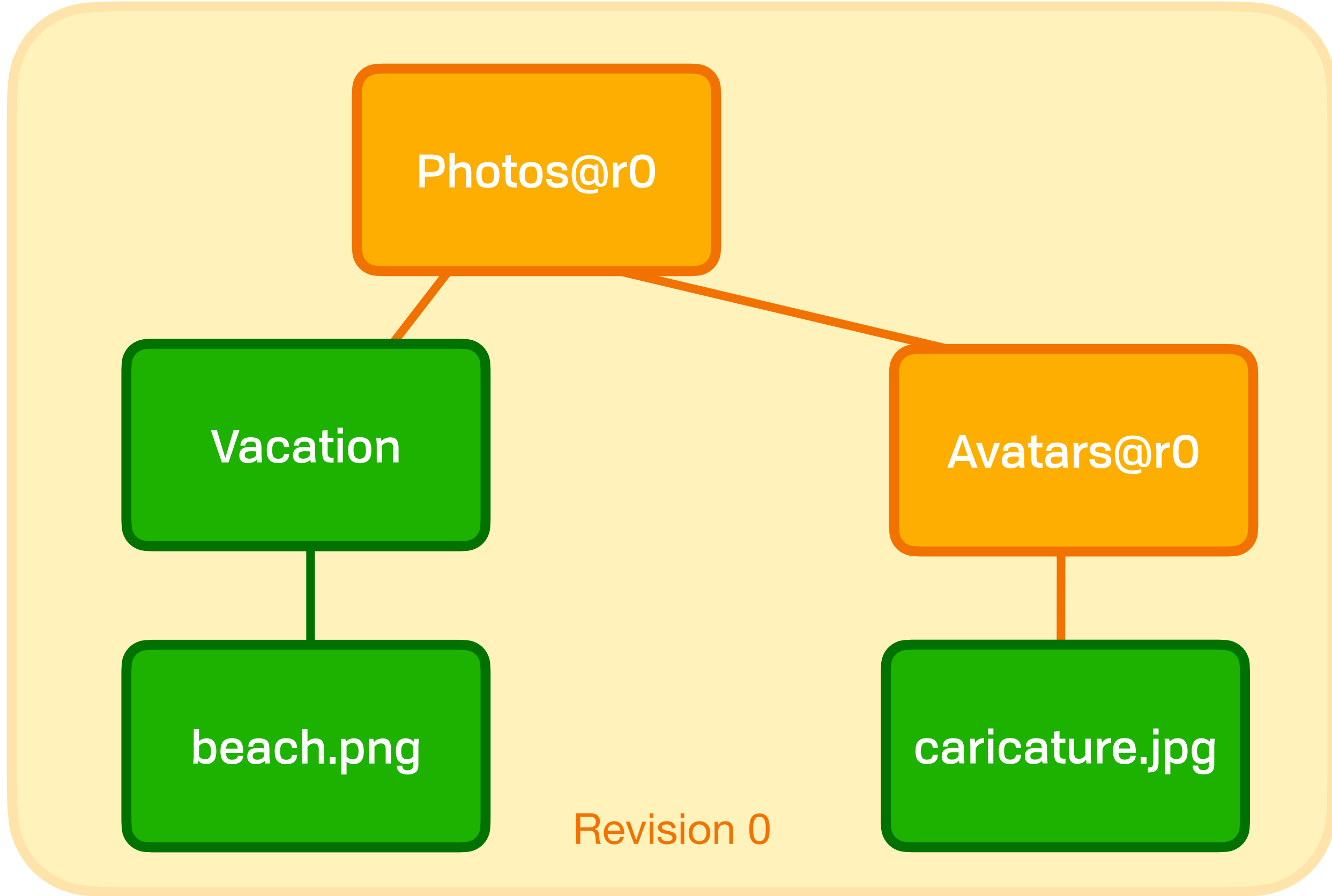
Securing Data Access

Persistent Versioning



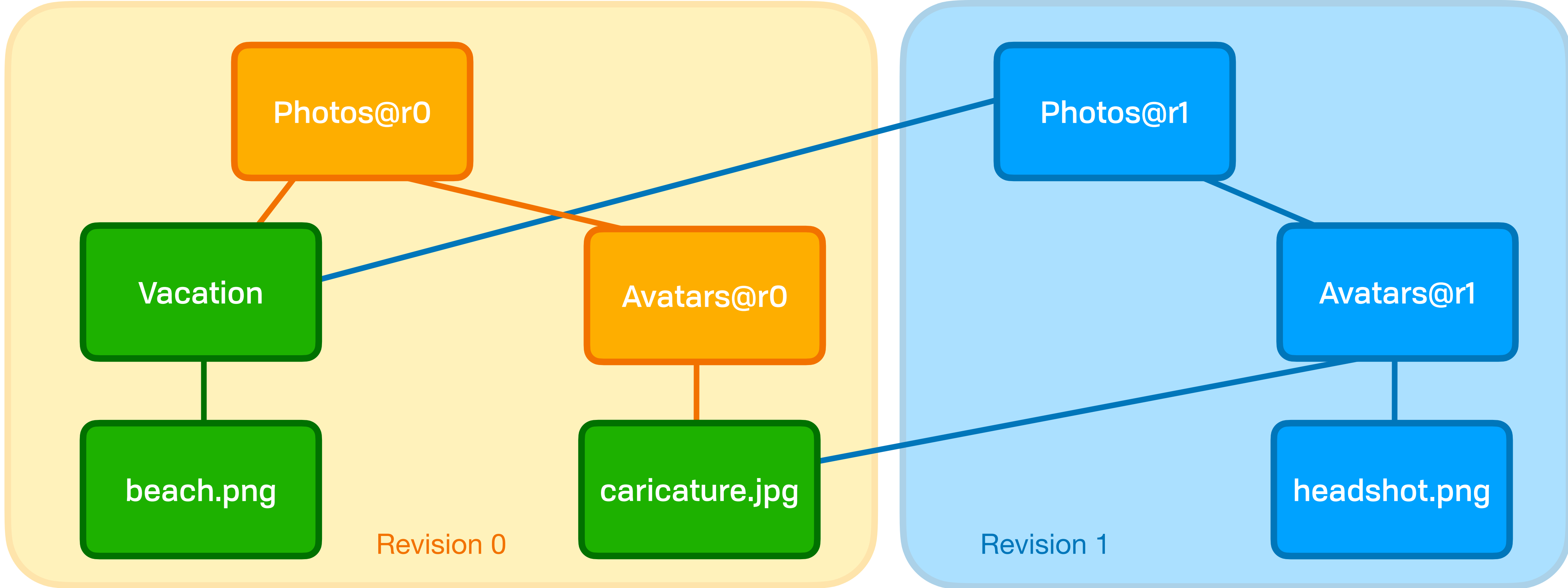
Securing Data Access

Persistent Versioning



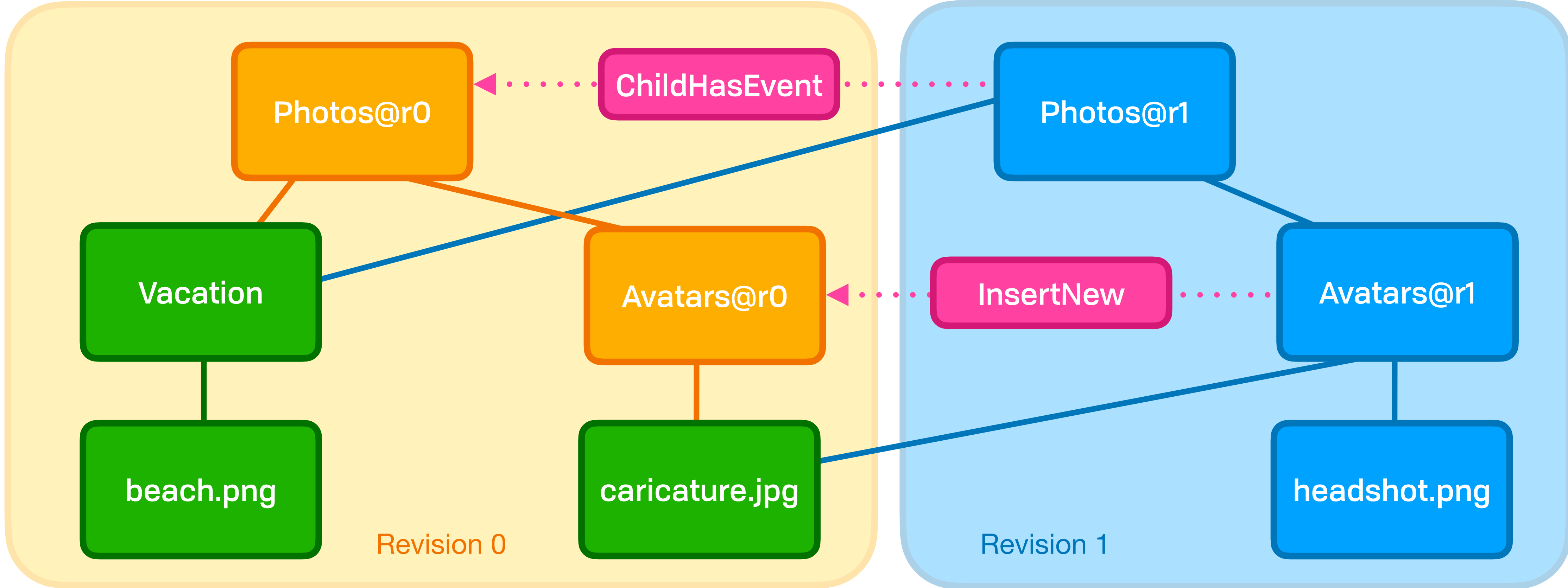
Securing Data Access

Persistent Versioning



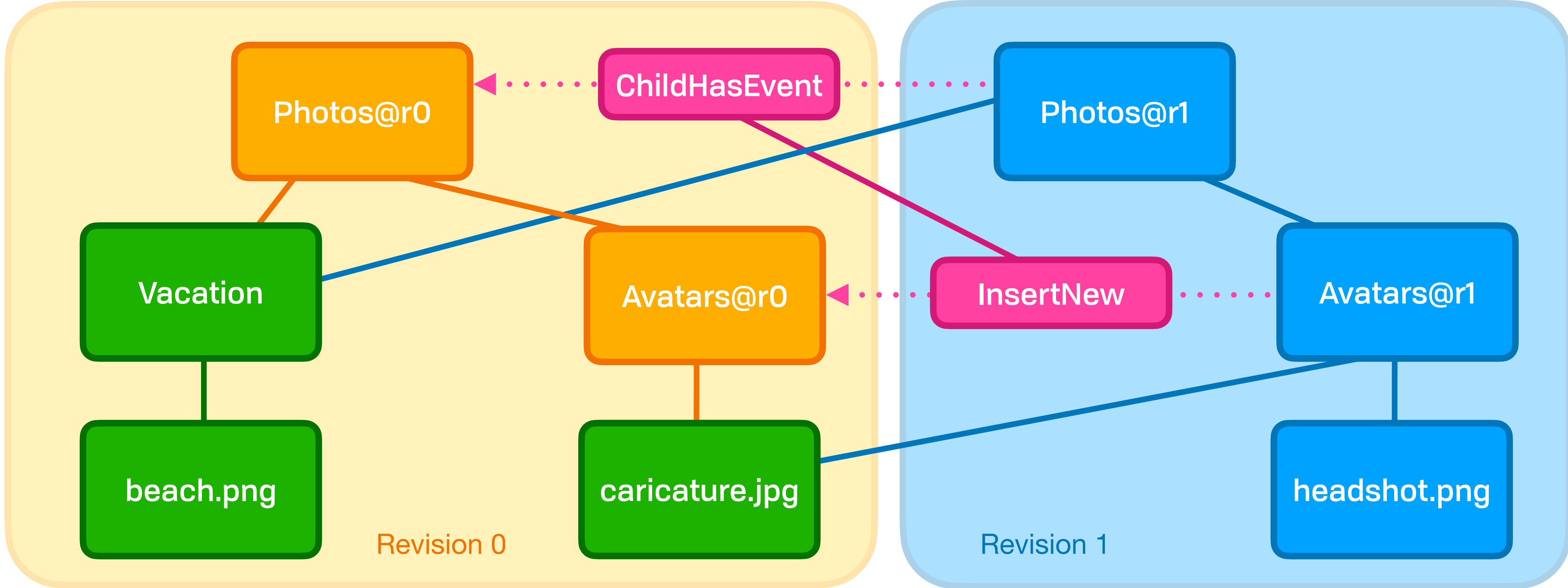
Securing Data Access

Persistent Versioning



Securing Data Access

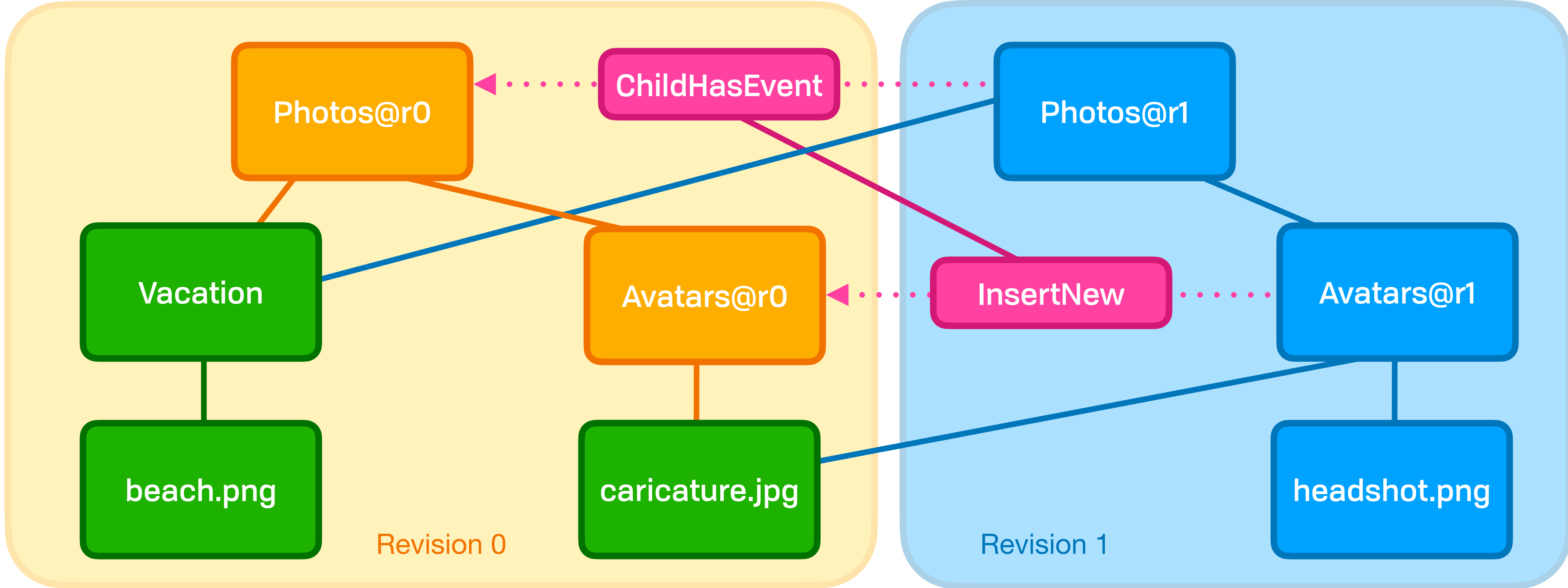
Persistent Versioning



Securing Data Access

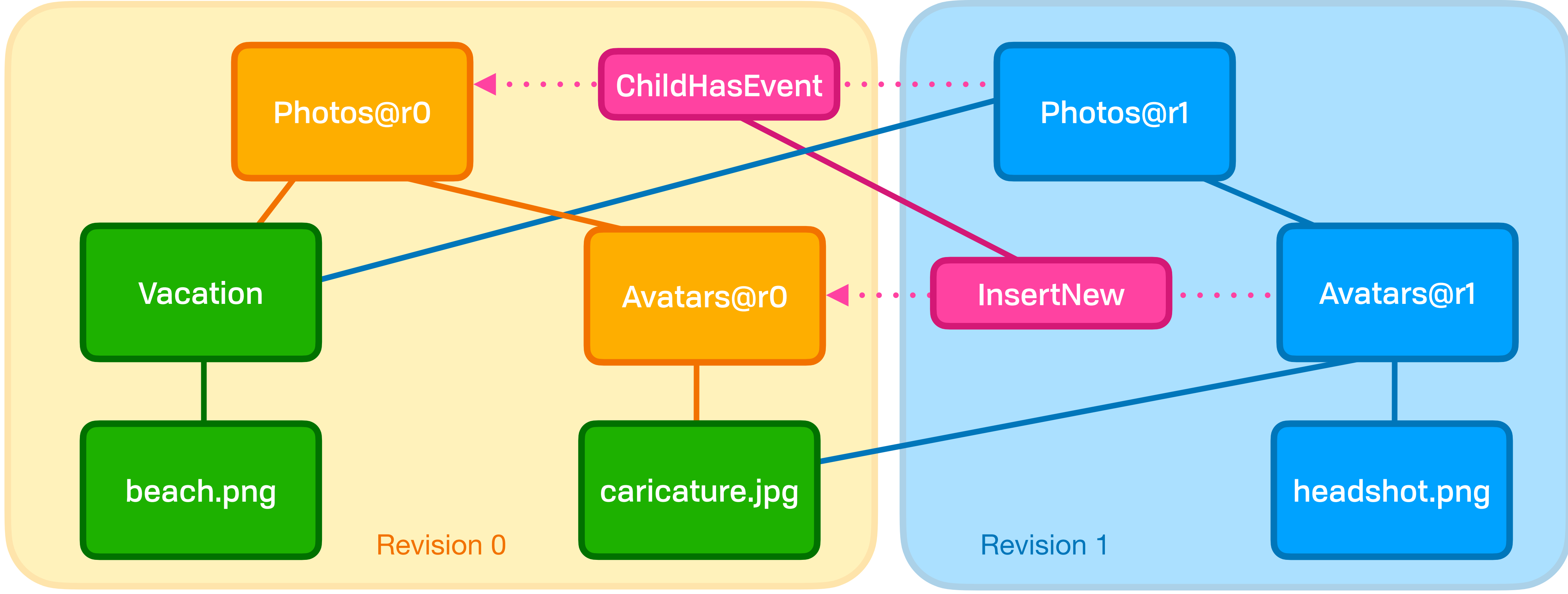
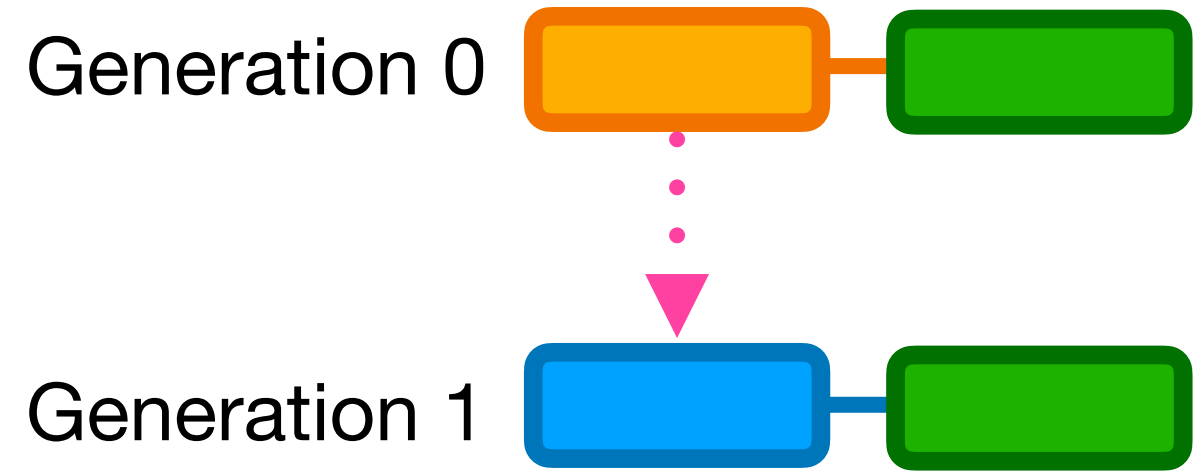
Persistent Versioning

Generation 0  



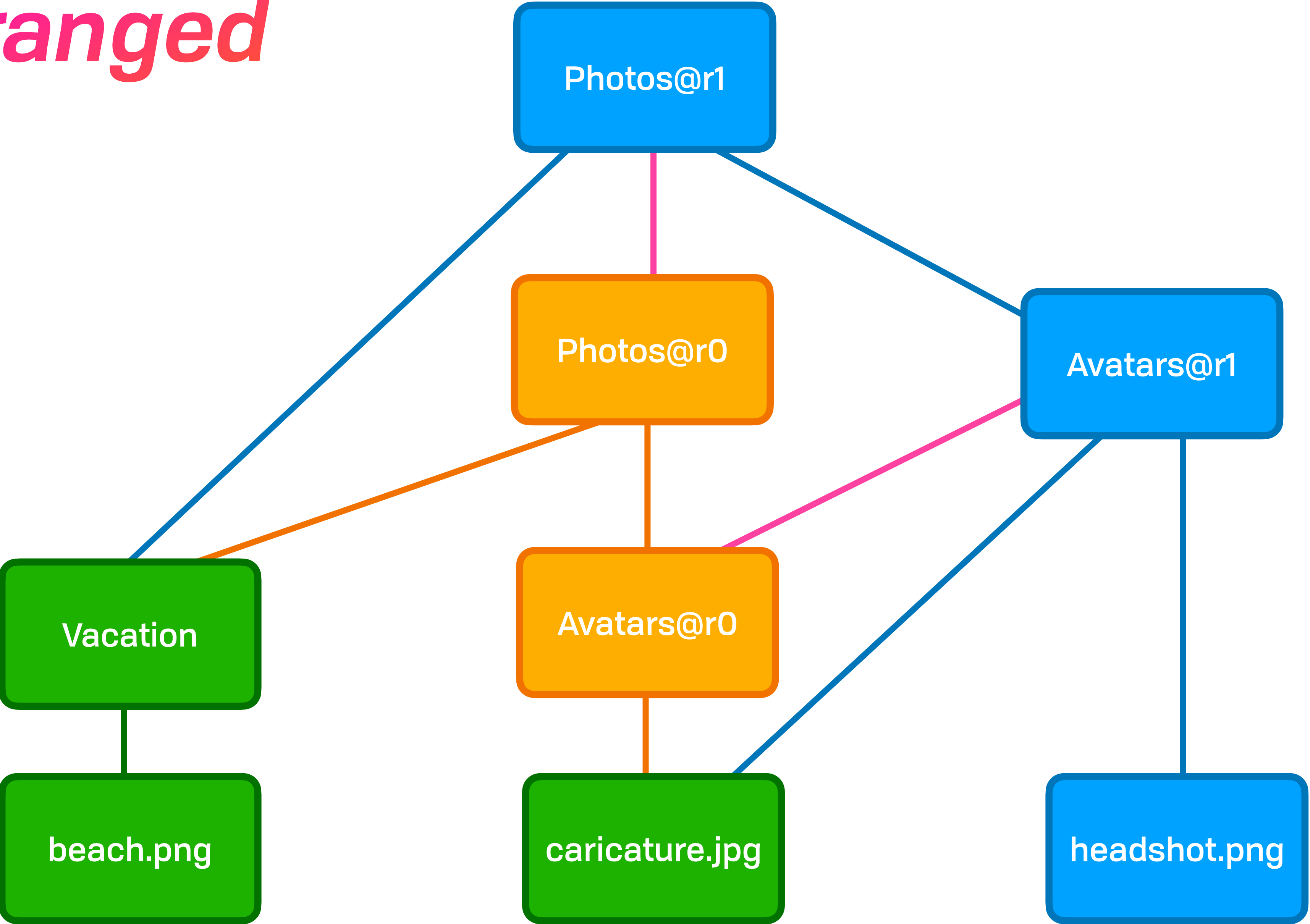
Securing Data Access

Persistent Versioning



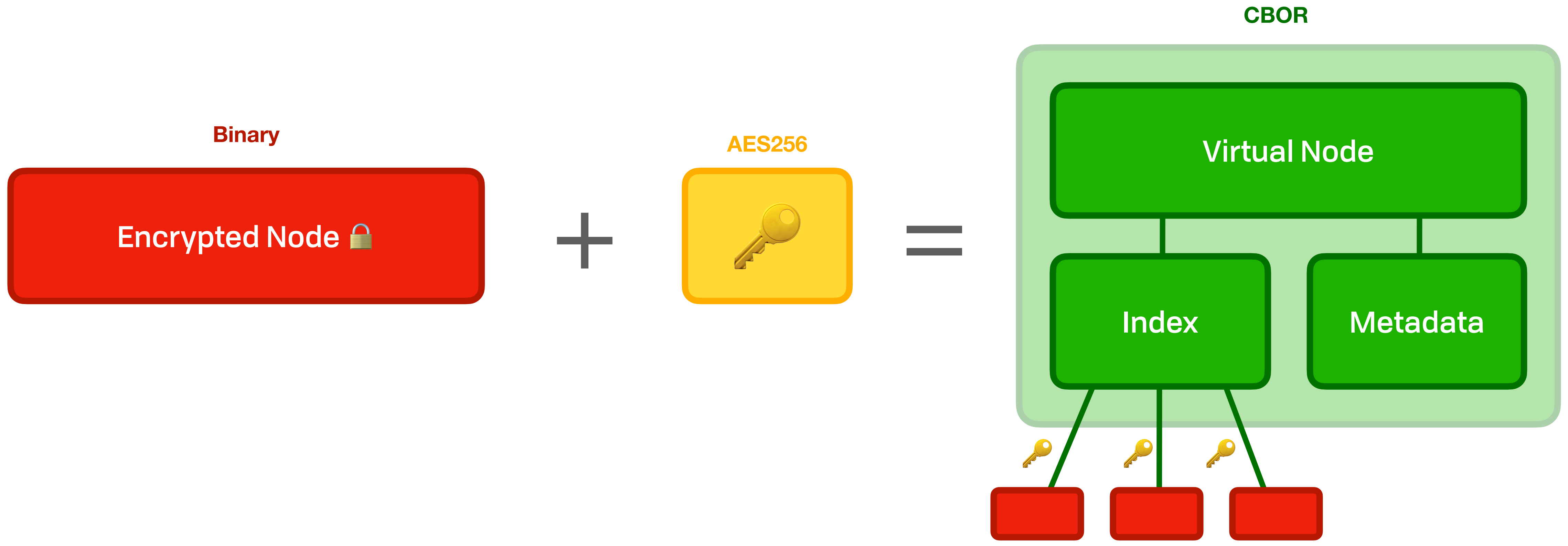
Securing Data Access

Rearranged



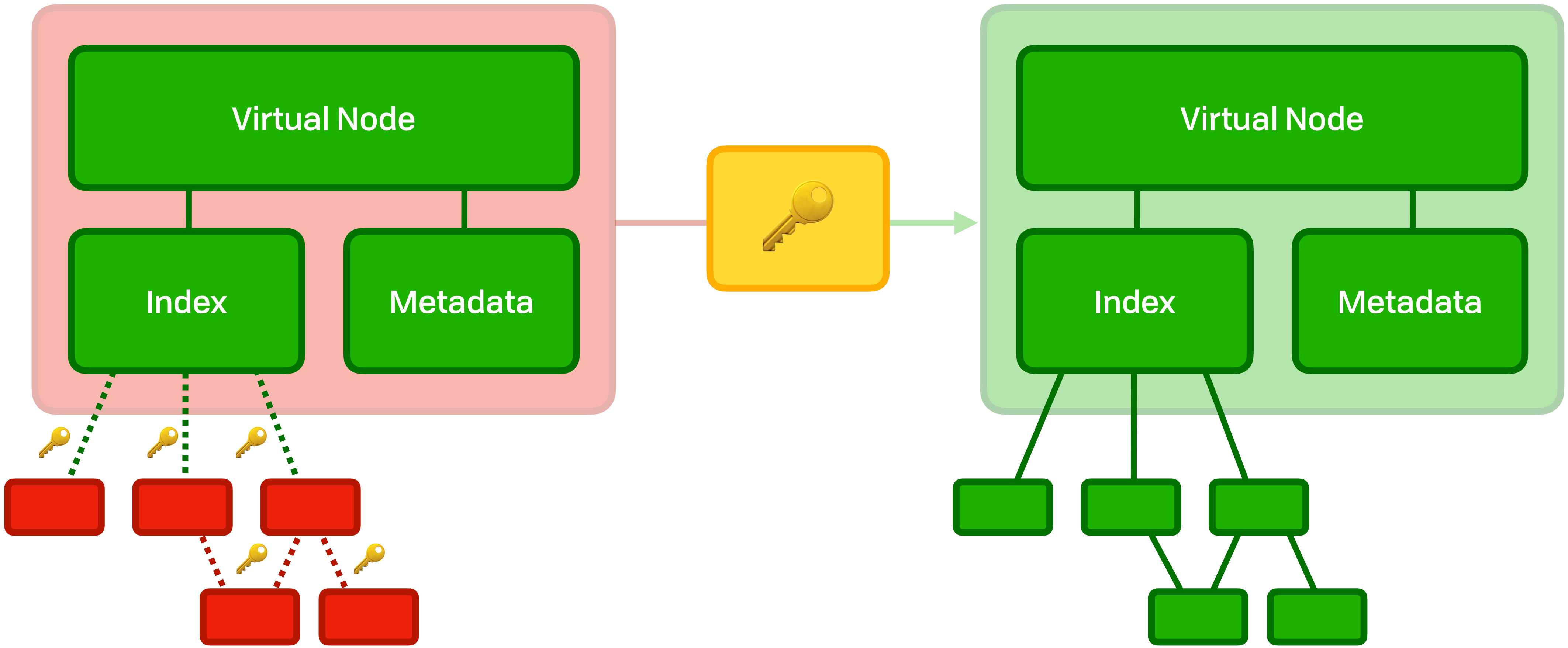
Securing Data Access

Private Nodes 🙈



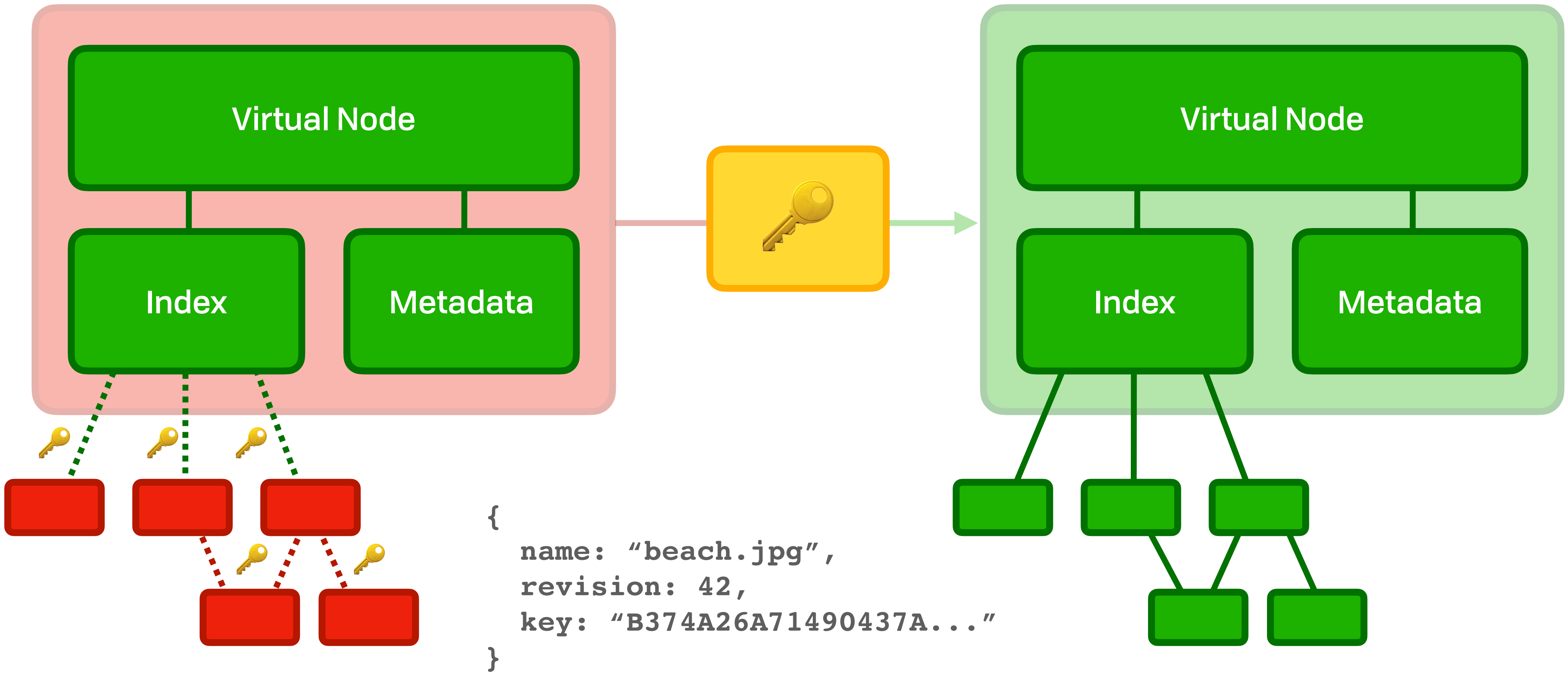
Securing Data Access

Cryptree 🎄



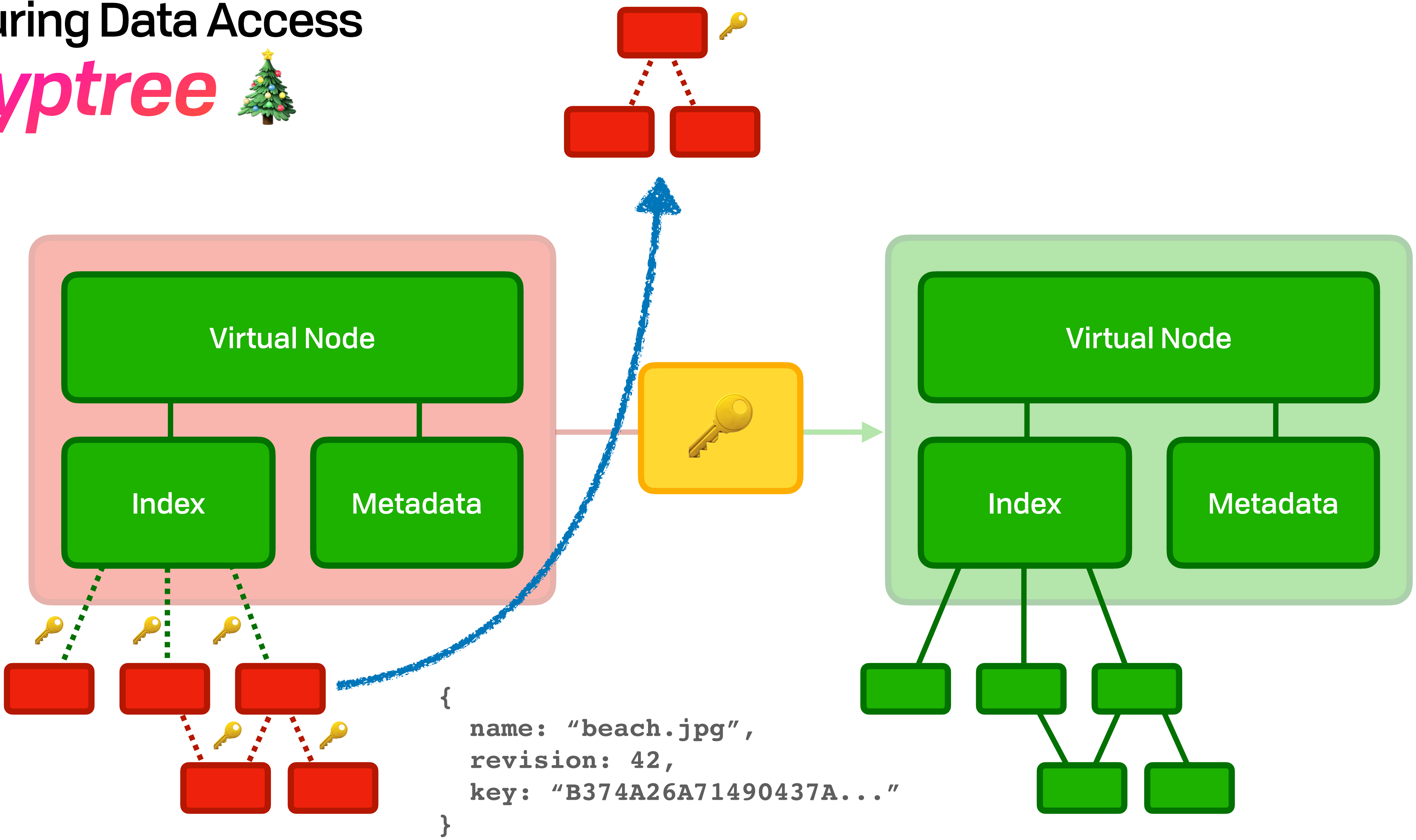
Securing Data Access

Cryptree 🎄



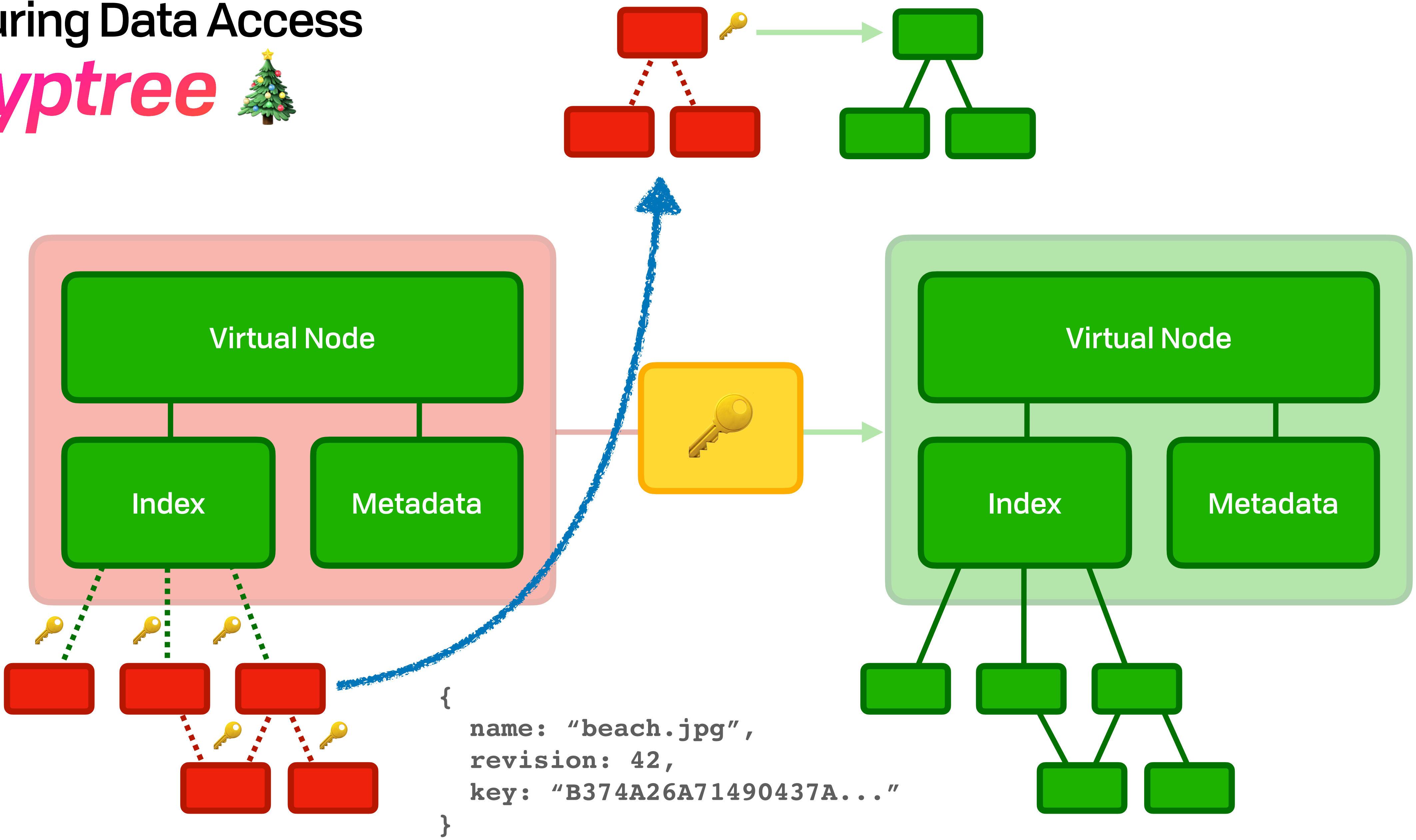
Securing Data Access

Cryptree 🎄



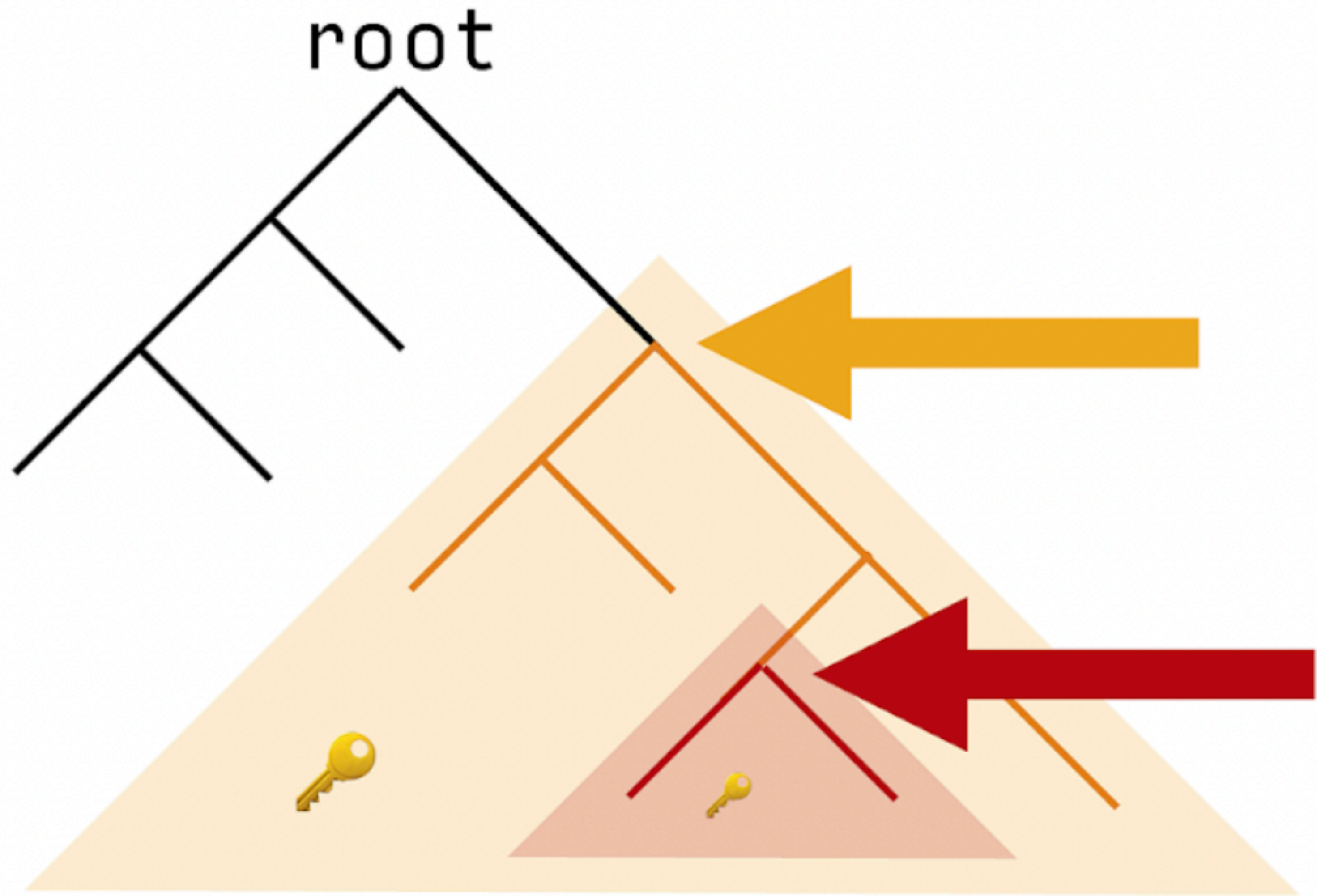
Securing Data Access

Crypttree 🎄



Securing Data Access

Subtree Read Access



Securing Data Access

Future Light Cone Restriction

Securing Data Access

Future Light Cone Restriction

- Ratchet keys for backwards secrecy
 - Spiral ratchet for quick fast forwards

Securing Data Access

Future Light Cone Restriction

- Ratchet keys for backwards secrecy
 - Spiral ratchet for quick fast forwards

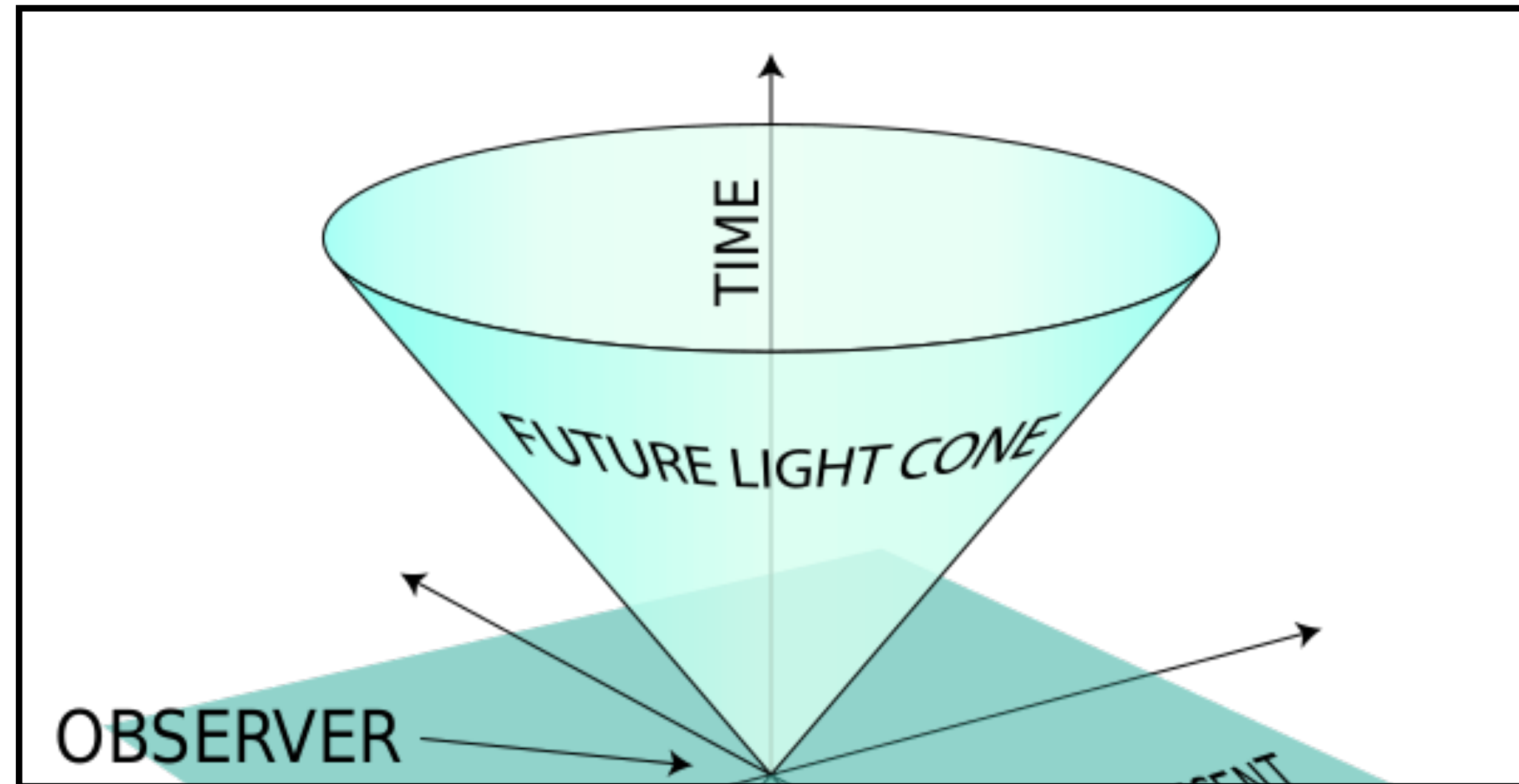
```
37-36-35-34-33-32-31
|
38 17-16-15-14-13 30
|
39 18 5-4-3 12 29
|
40 19 6 1-2 11 28
|
41 20 7-8-9-10 27
|
42 21-22-23-24-25-26
|
43-44-45-46-47-48-49...
```

Securing Data Access

Future Light Cone Restriction

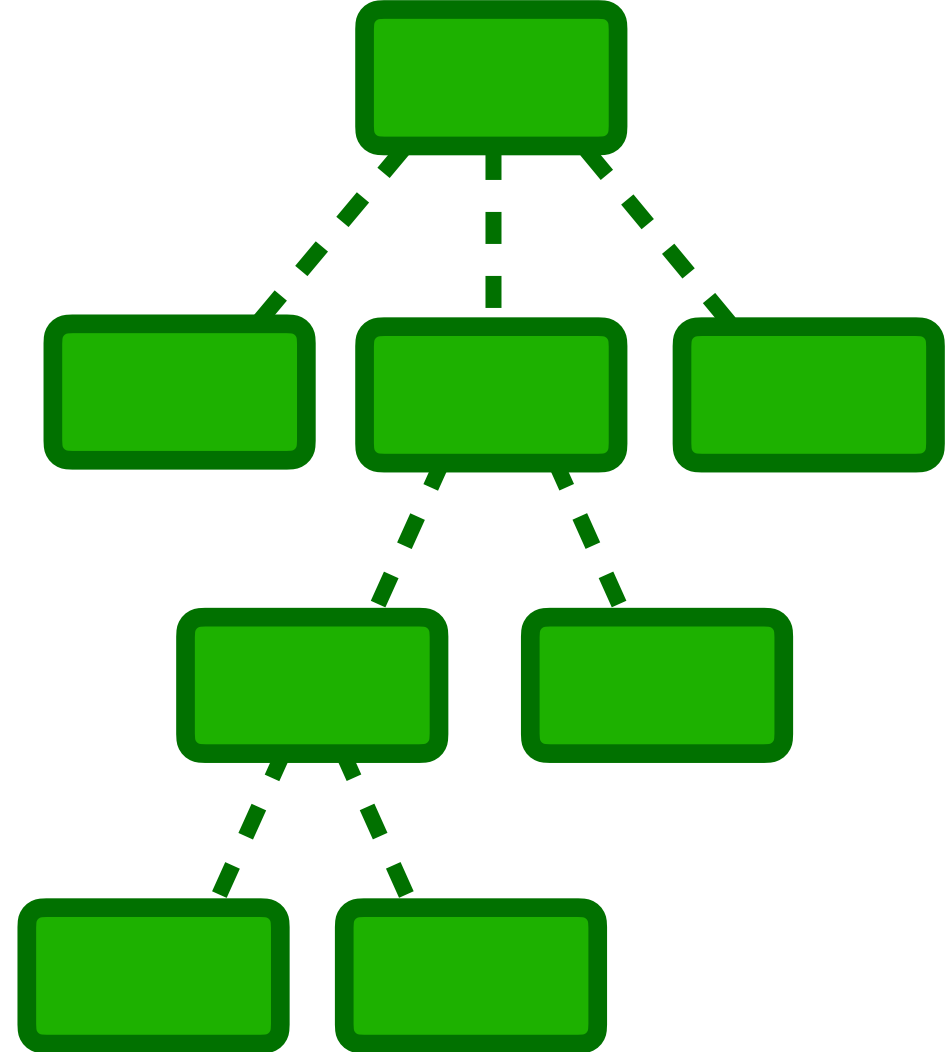
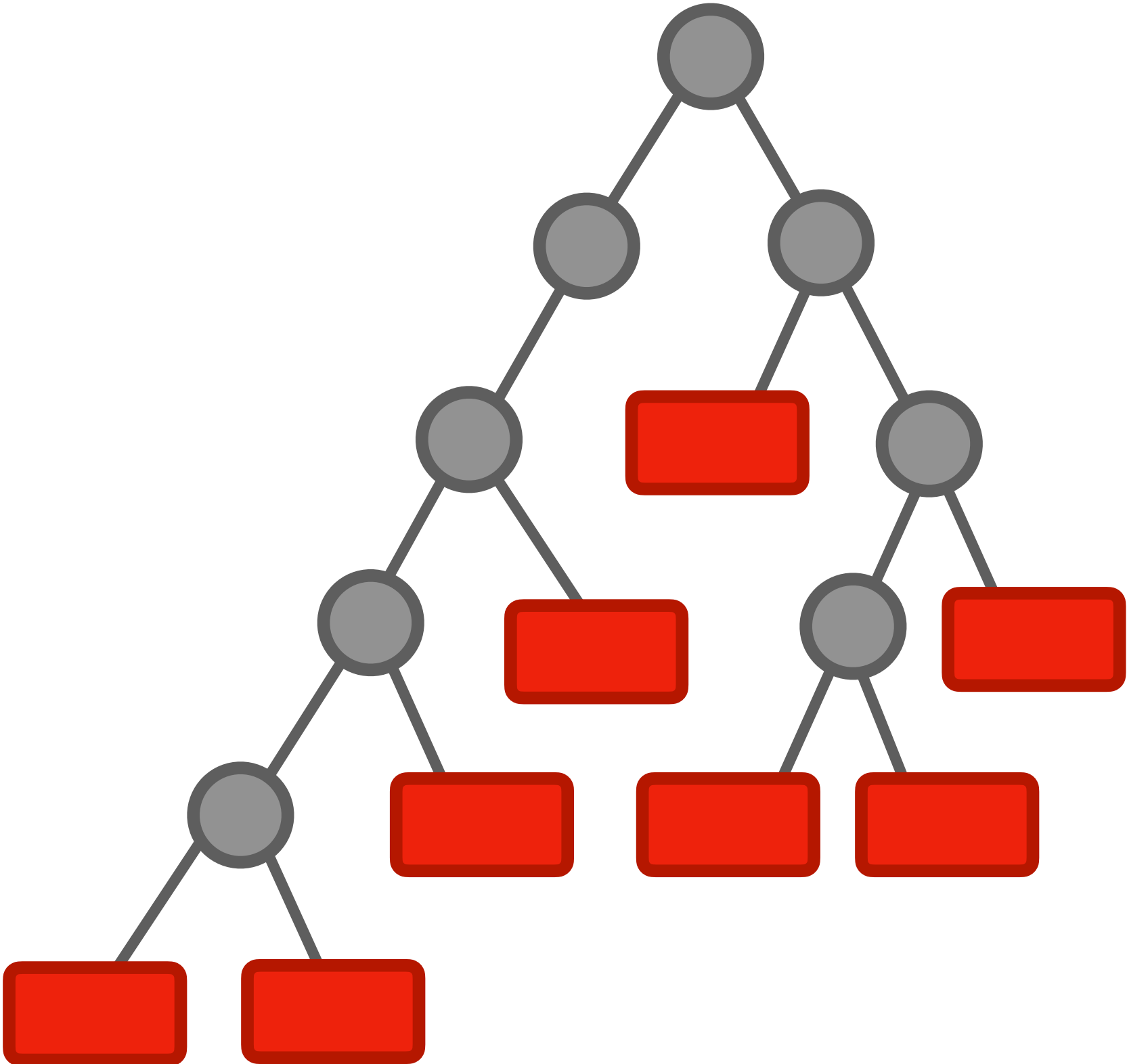
- Ratchet keys for backwards secrecy
 - Spiral ratchet for quick fast forwards

```
37-36-35-34-33-32-31
|
38 17-16-15-14-13 30
|
39 18 5-4-3 12 29
|
40 19 6 1-2 11 28
|
41 20 7-8-9-10 27
|
42 21-22-23-24-25-26
|
43-44-45-46-47-48-49...
```



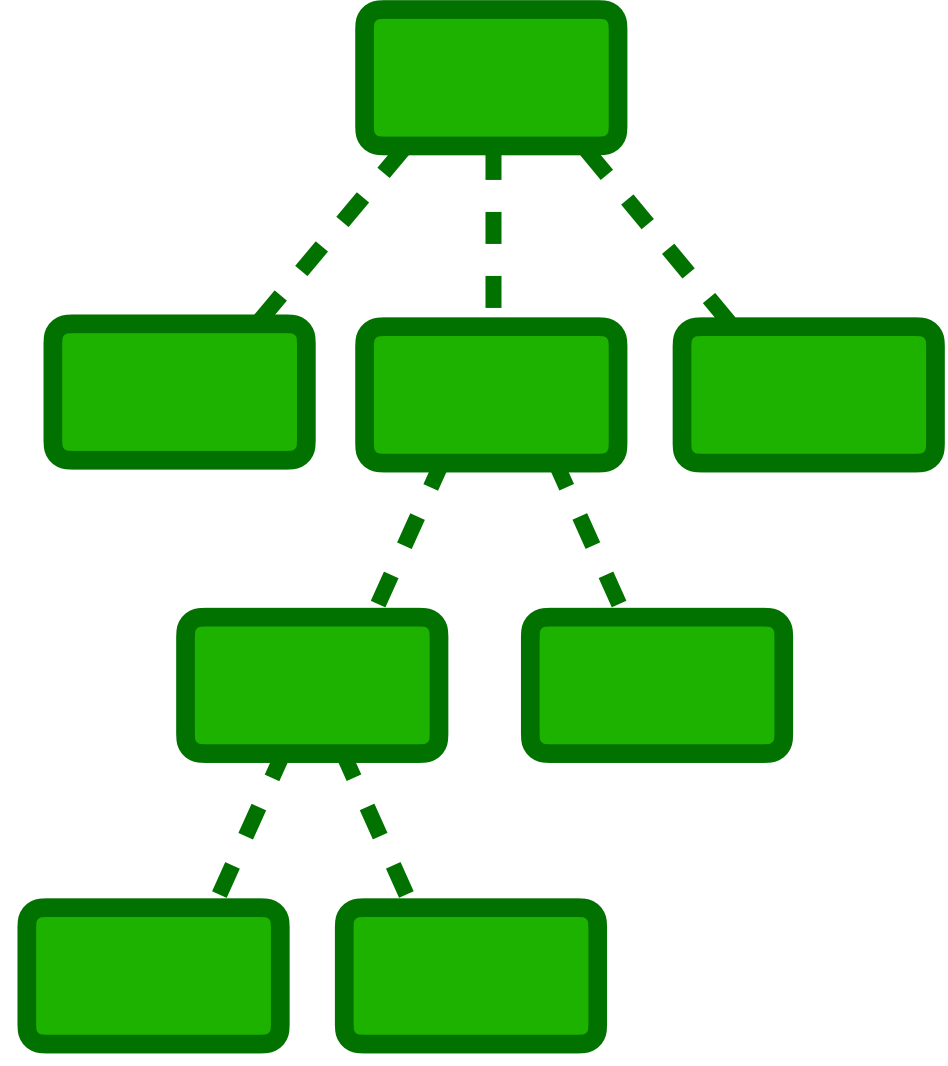
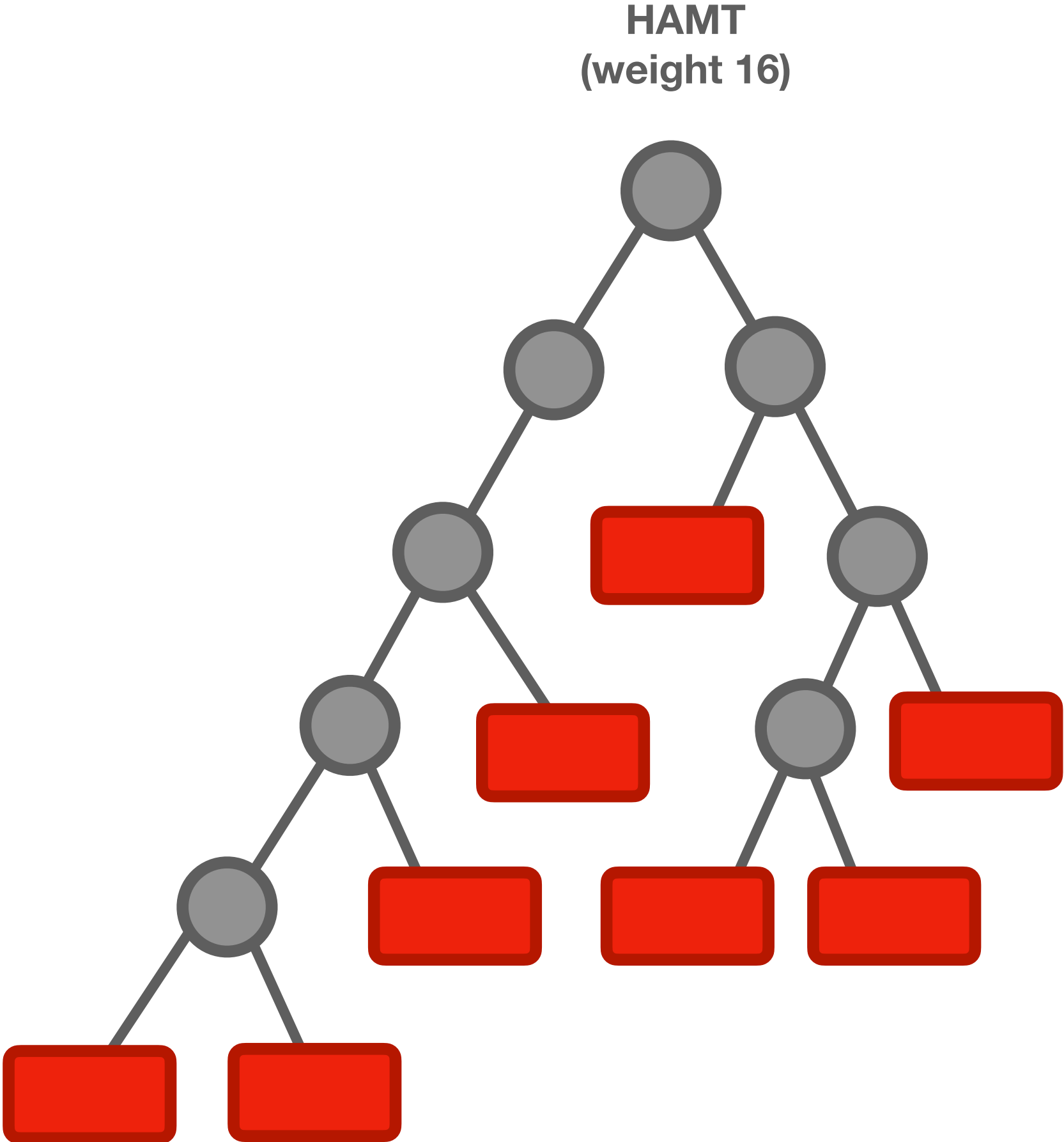
Securing Data Access

Encrypted Tree is Surprisingly Efficient



Securing Data Access

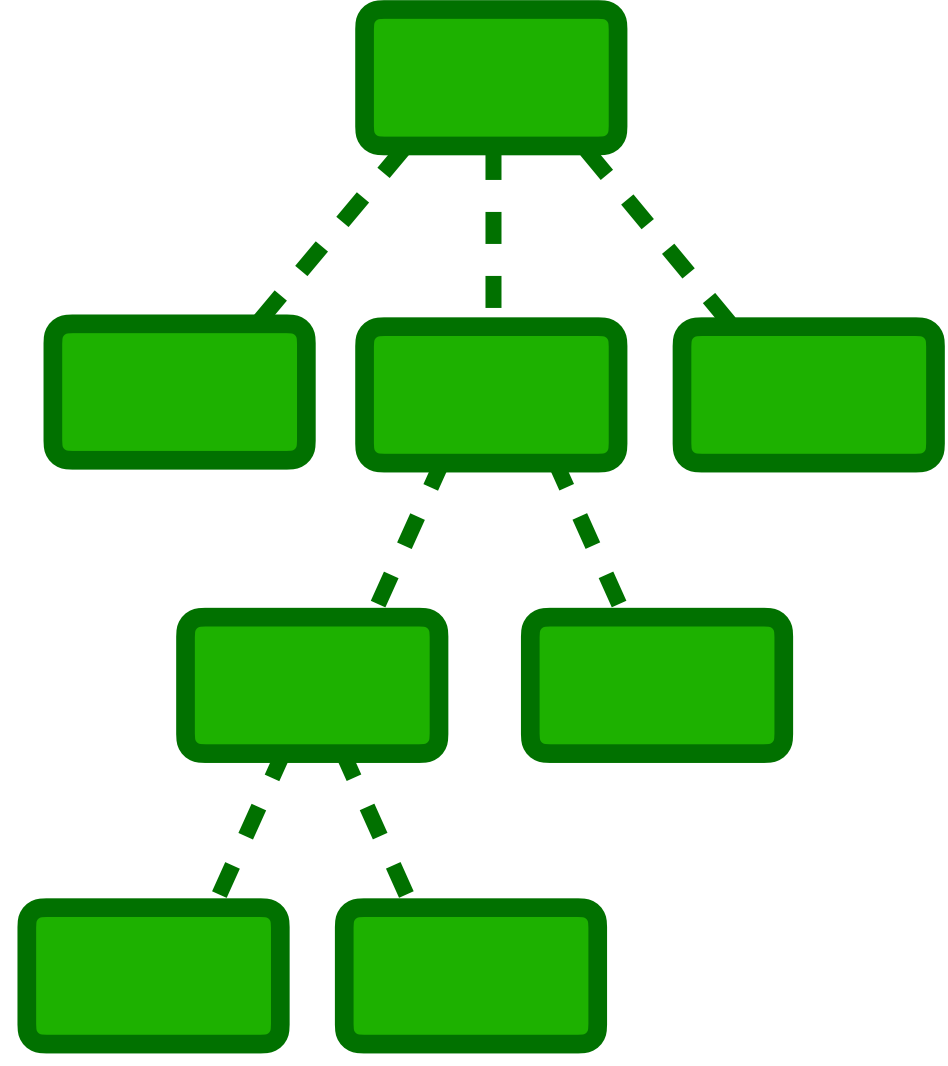
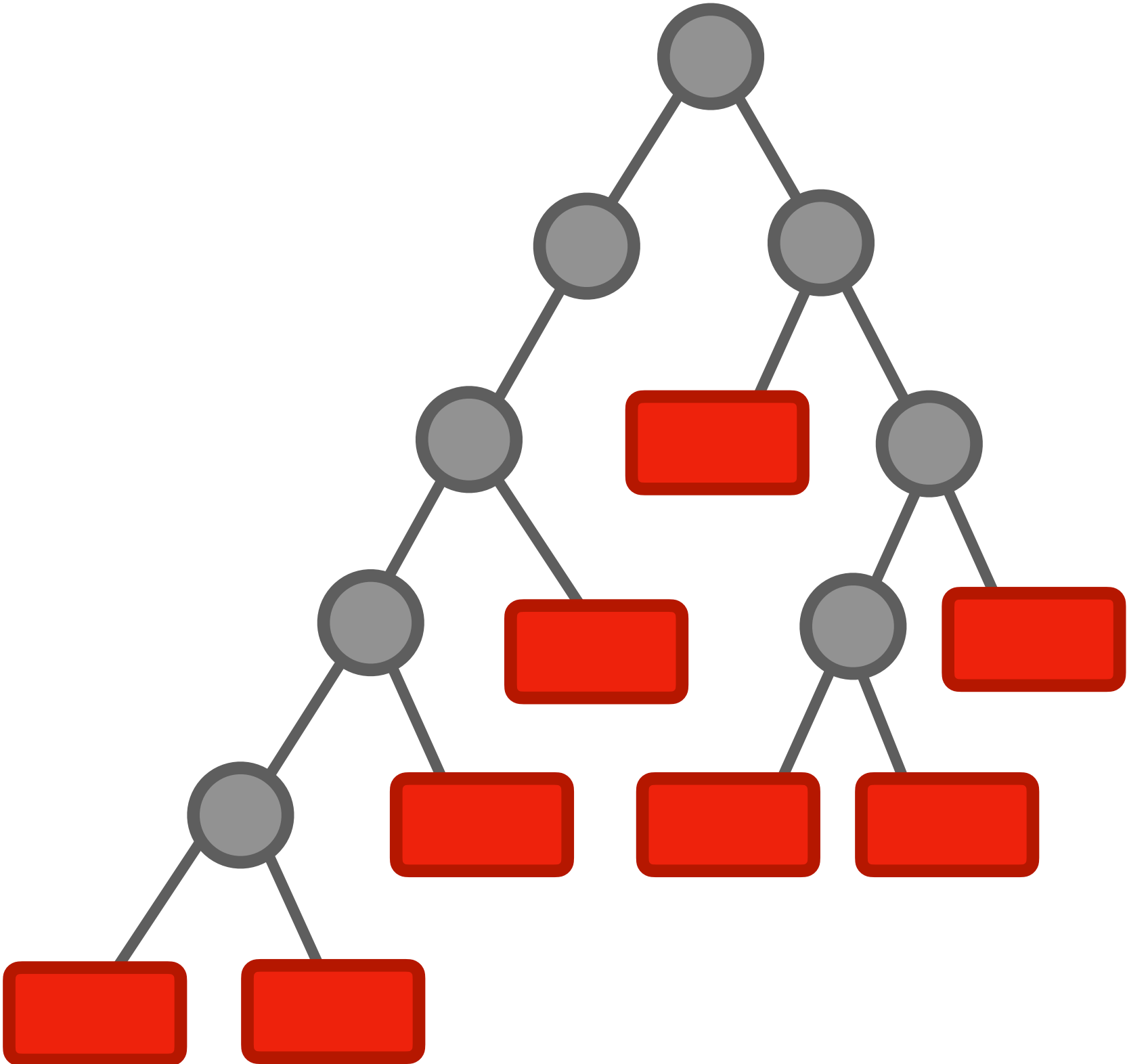
Encrypted Tree is Surprisingly Efficient



Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4,096$ items
(weight 16) $16^4 = 65,536$ items

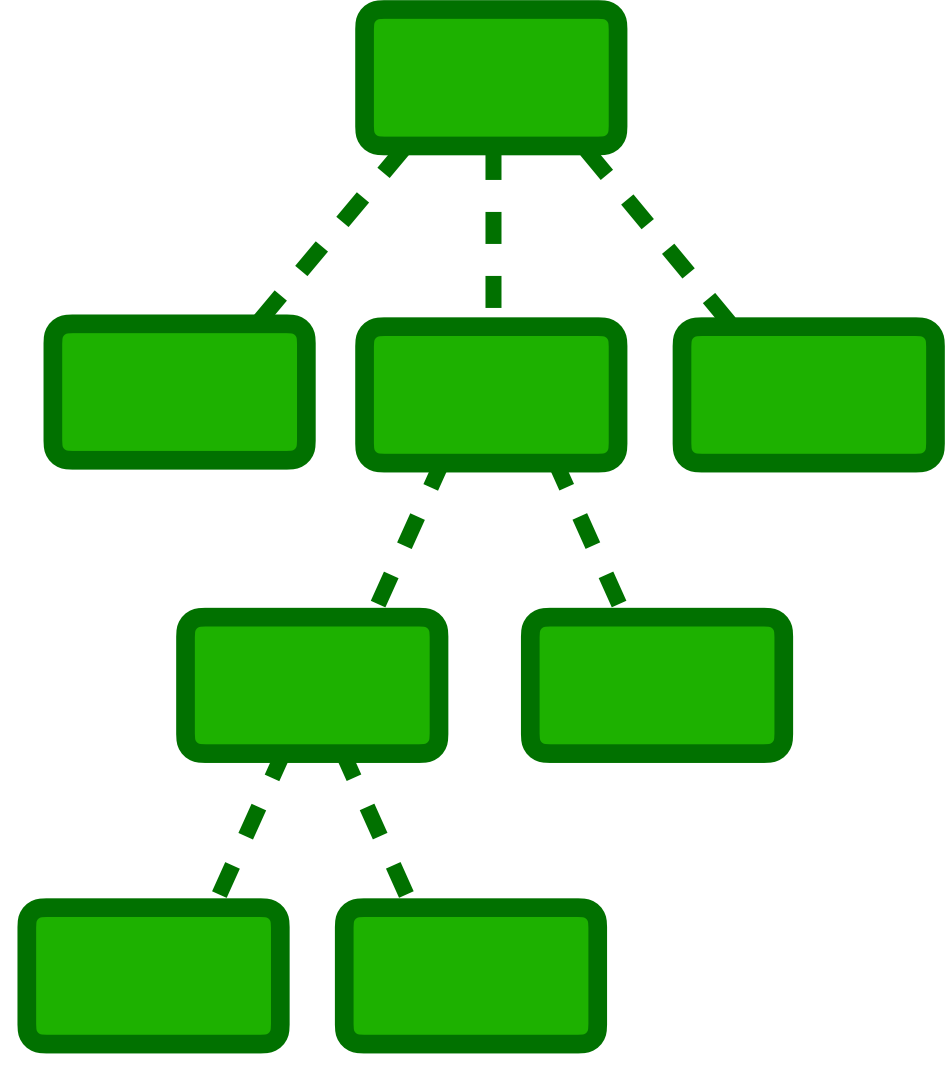
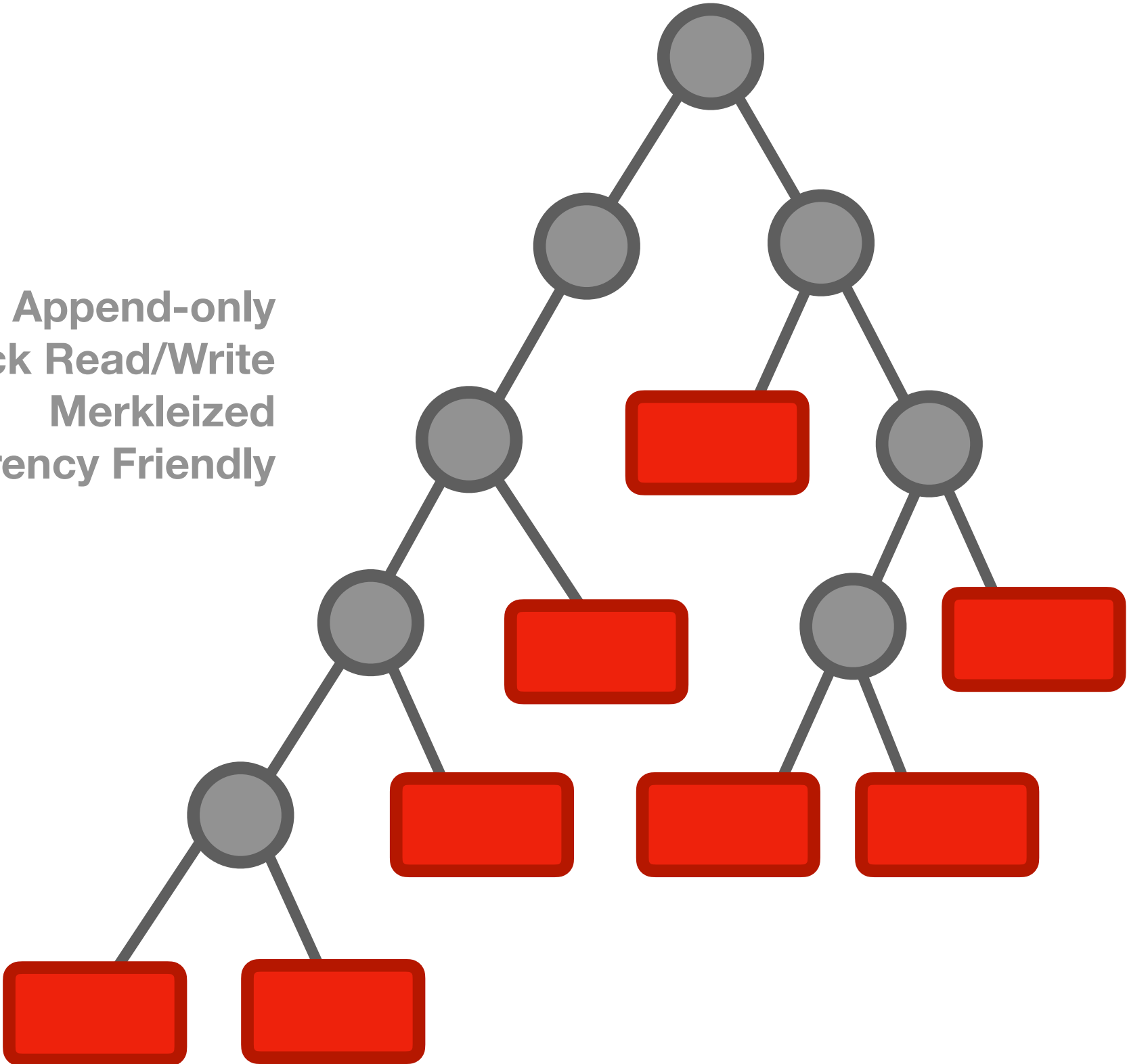


Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4,096$ items
(weight 16) $16^4 = 65,536$ items

Append-only
Quick Read/Write
Merkleized
Concurrency Friendly

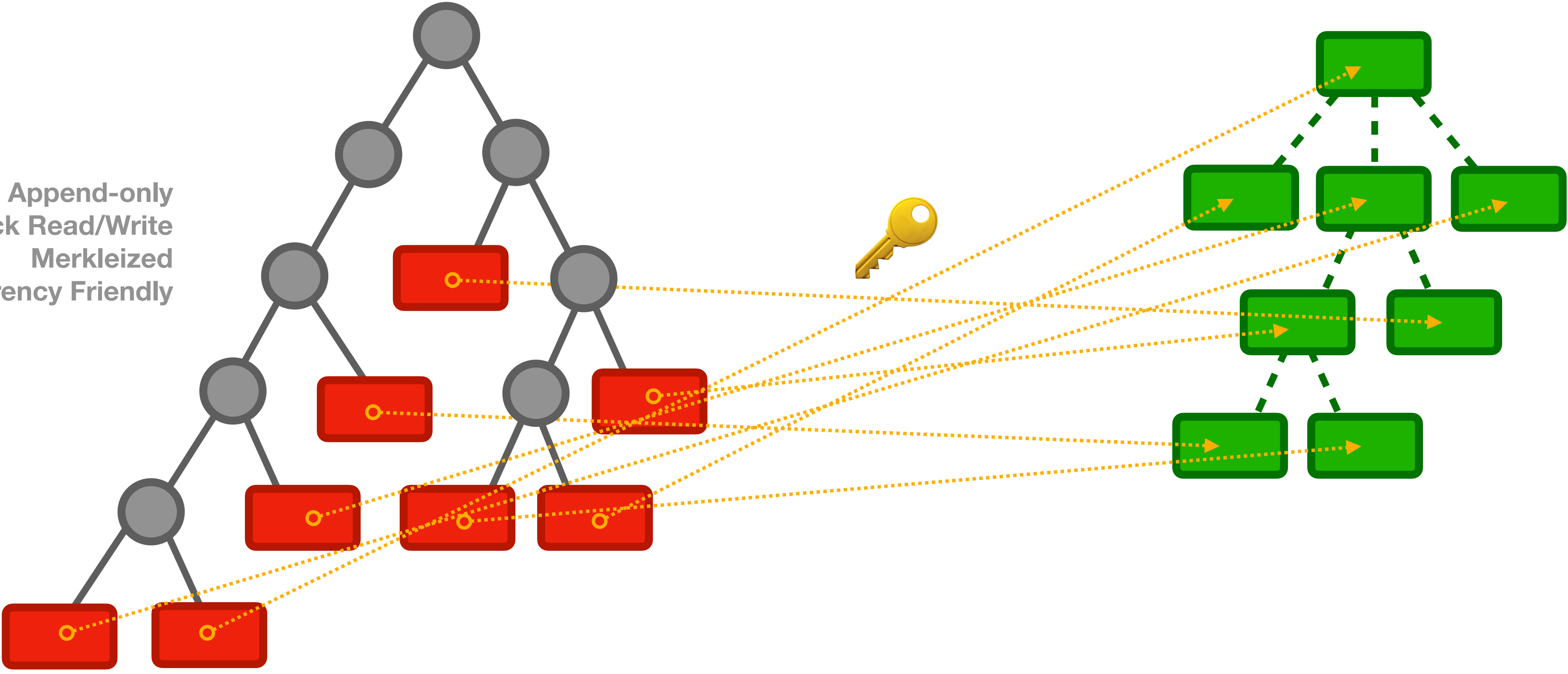


Securing Data Access

Encrypted Tree is Surprisingly Efficient

HAMT $16^3 = 4,096$ items
(weight 16) $16^4 = 65,536$ items

Append-only
Quick Read/Write
Merkleized
Concurrency Friendly



Securing Data Access

Namefilters & Hidden Paths

Securing Data Access

Namefilters & Hidden Paths

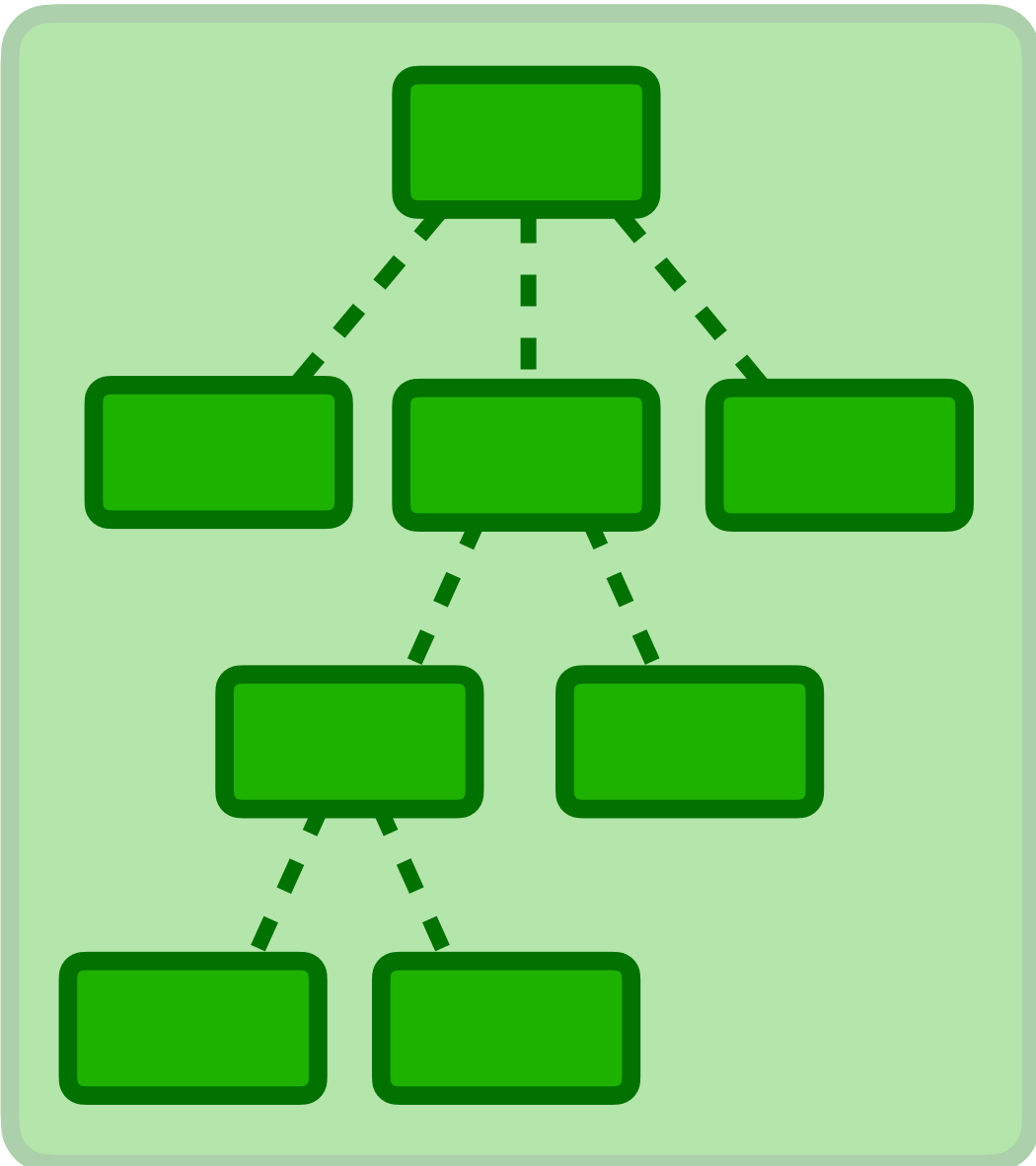
- Bare Filter
 - `parentFilter`
 - `AND bloom(SHA(aesKey))`
 - `AND bloom(SHA(aesKey ++ revisionRatchet))`
- Saturation
 - `nameFilter AND bloom(SHA(nameFilter))`
 - Repeat until threshold bits flipped

Securing Data Access

Access-Mediated Collaborative Rooting

Securing Data Access

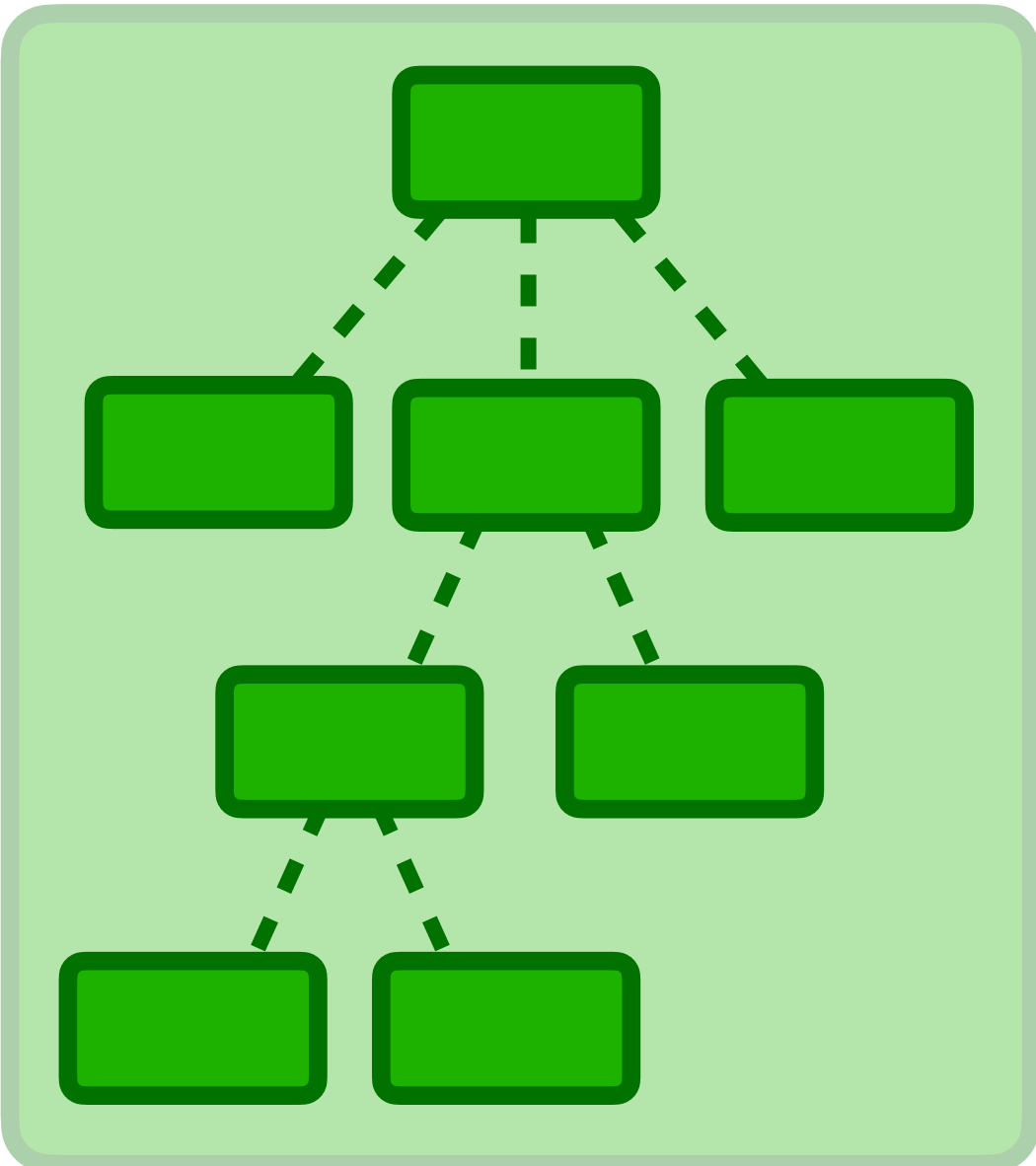
Access-Mediated Collaborative Rooting



Rev 0

Securing Data Access

Access-Mediated Collaborative Rooting

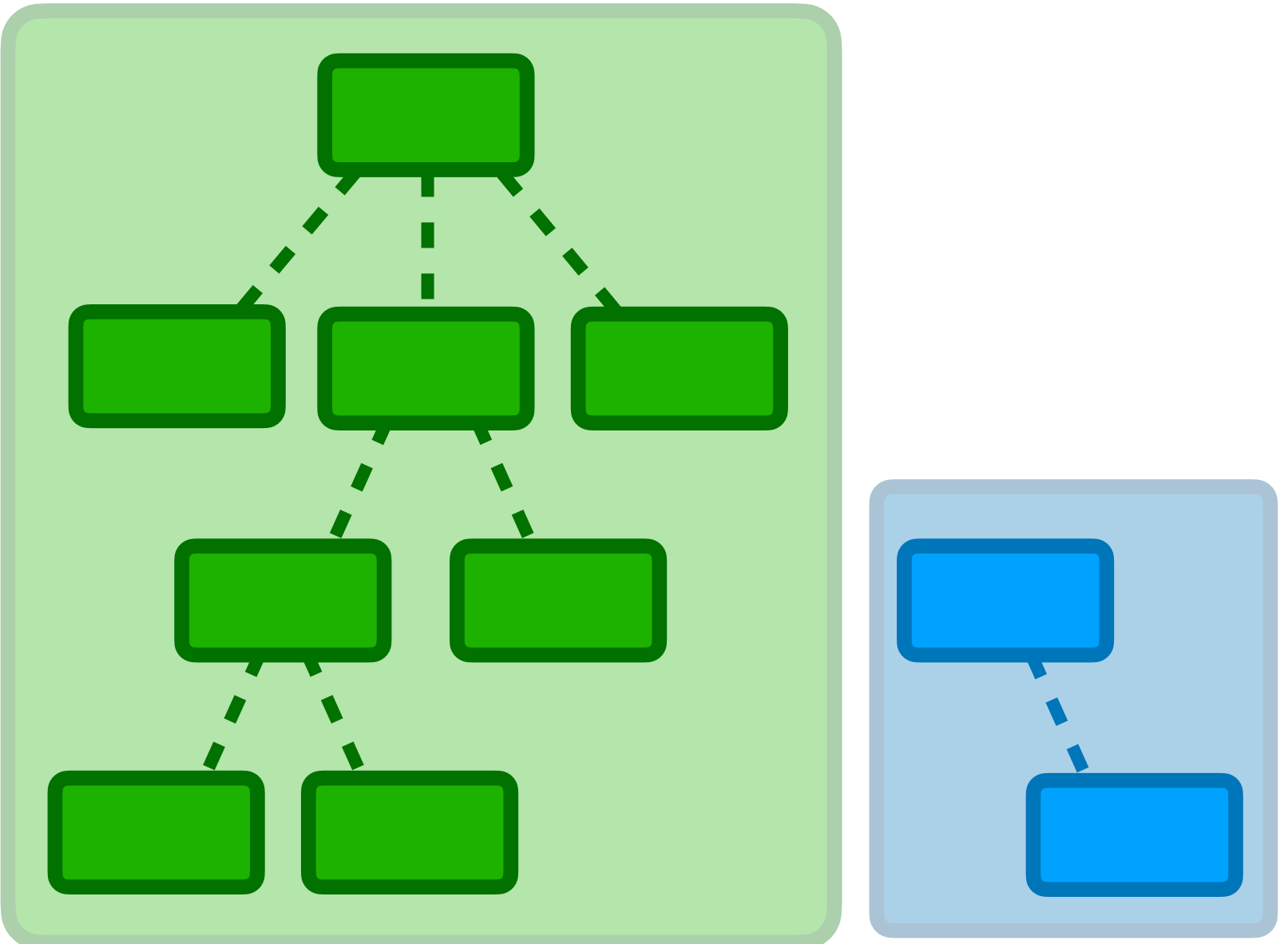


Rev 0



Securing Data Access

Access-Mediated Collaborative Rooting



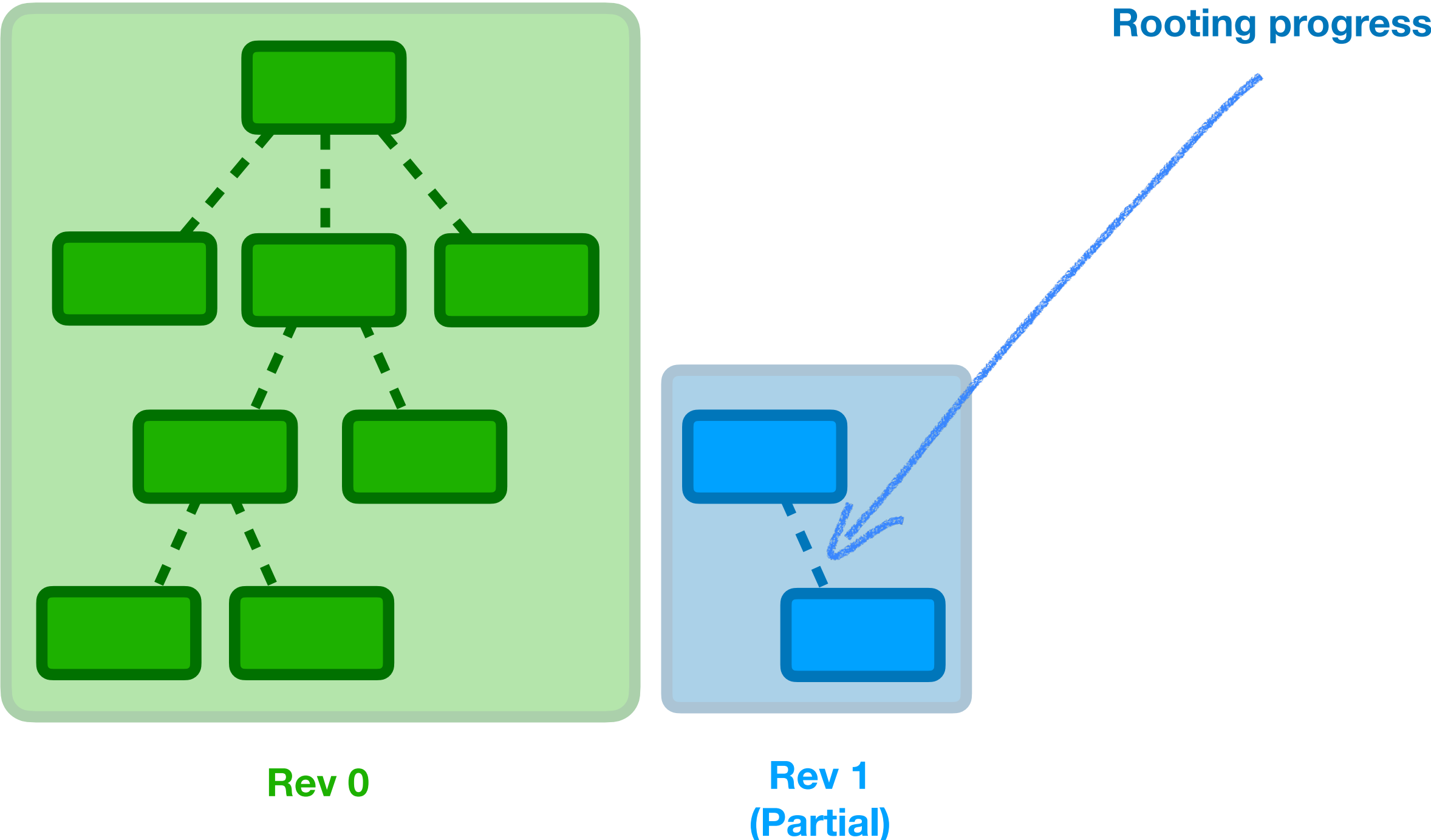
Rev 0

Rev 1
(Partial)



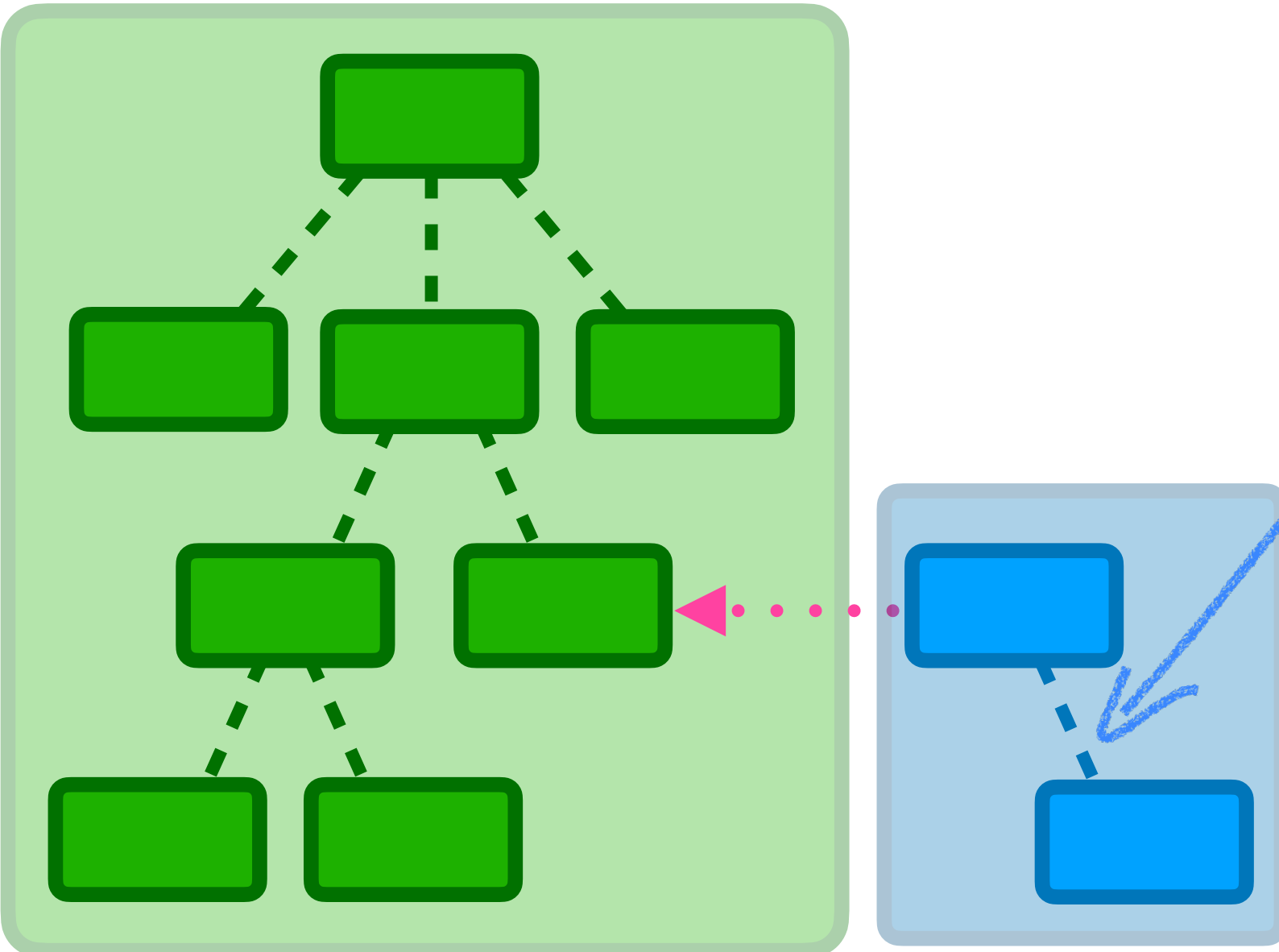
Securing Data Access

Access-Mediated Collaborative Rooting



Securing Data Access

Access-Mediated Collaborative Rooting



Rev 0

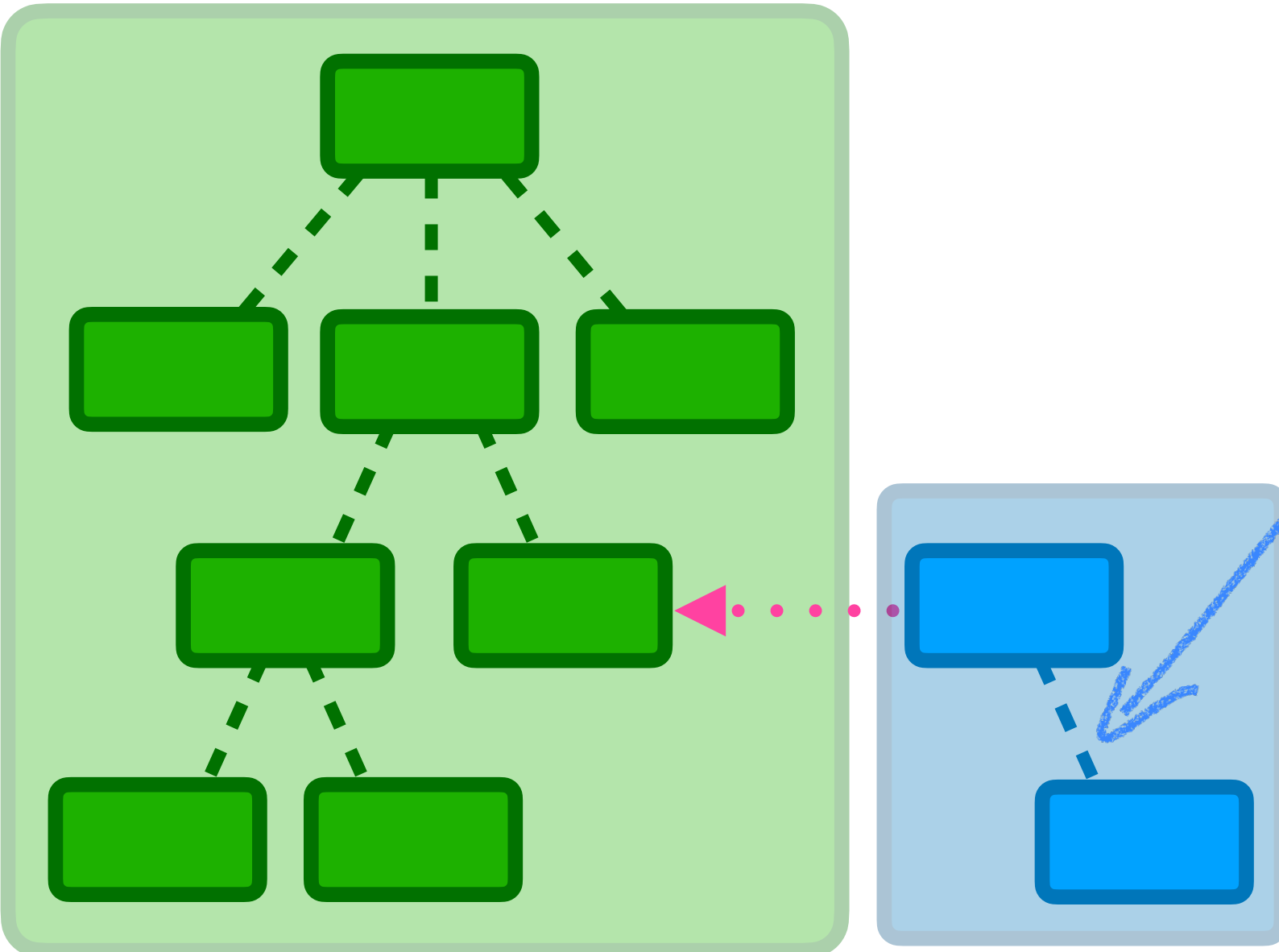
Rev 1
(Partial)

Rooting progress



Securing Data Access

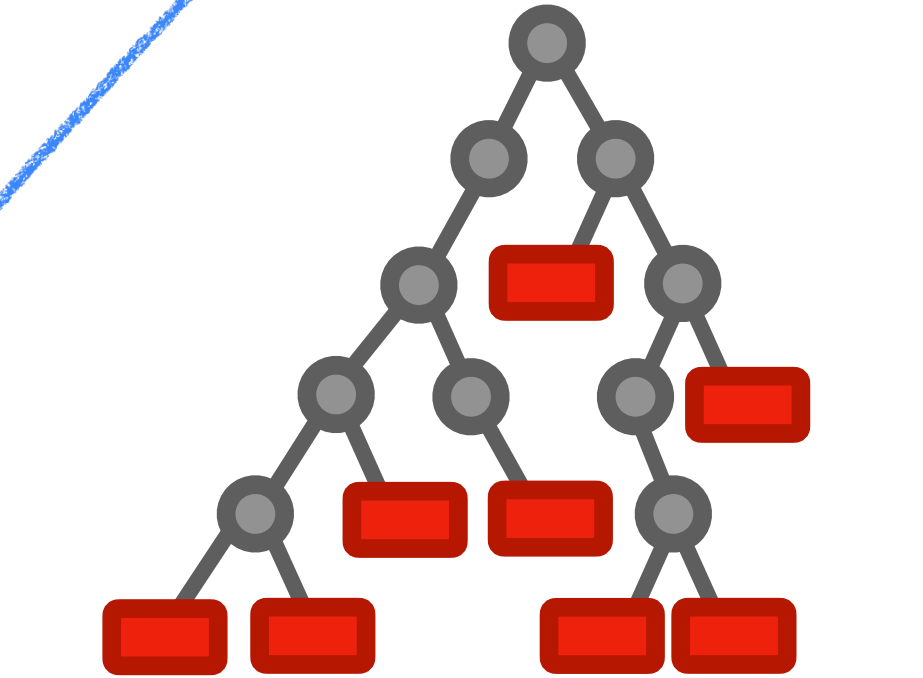
Access-Mediated Collaborative Rooting



Rev 0

Rev 1
(Partial)

Rooting progress

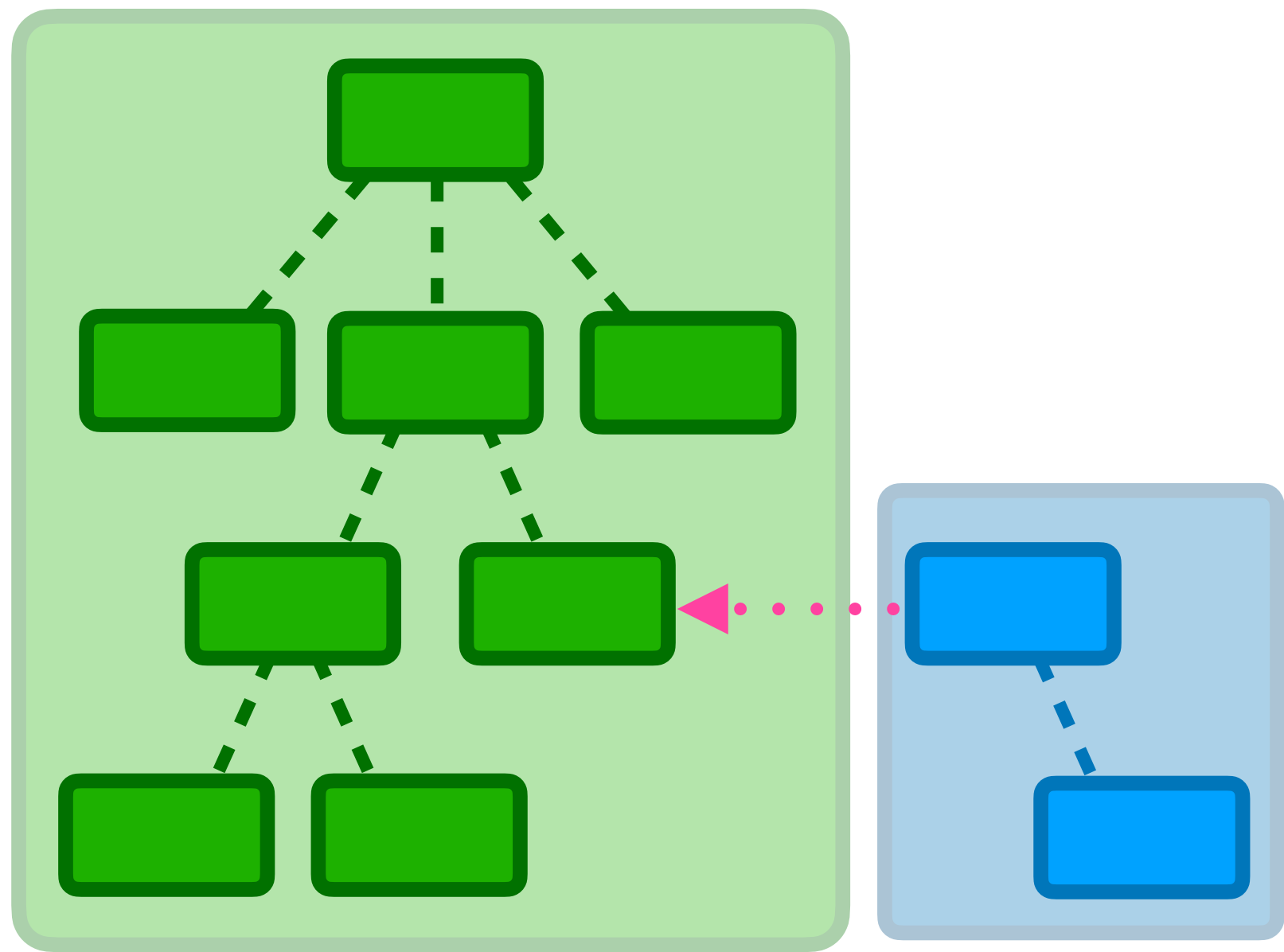
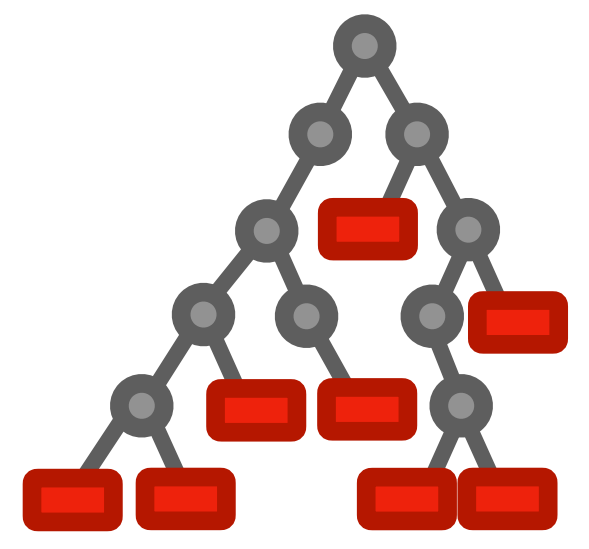


No common root at this layer!
Attached via HAMT



Securing Data Access

Progressive Fast Forward

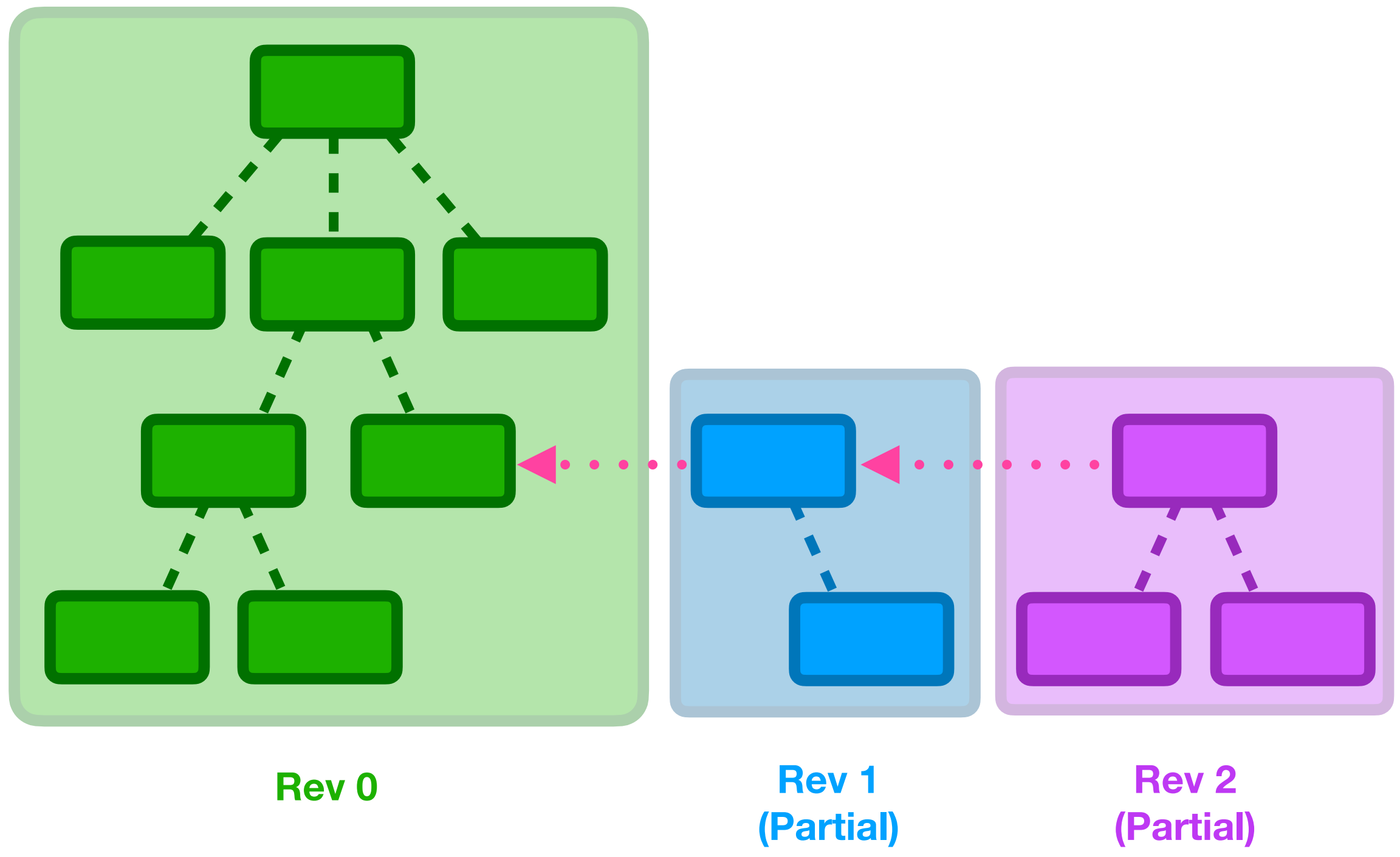
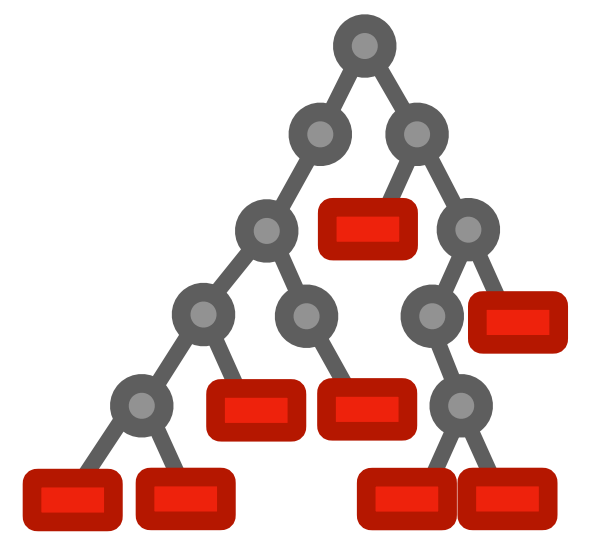


Rev 0

Rev 1
(Partial)

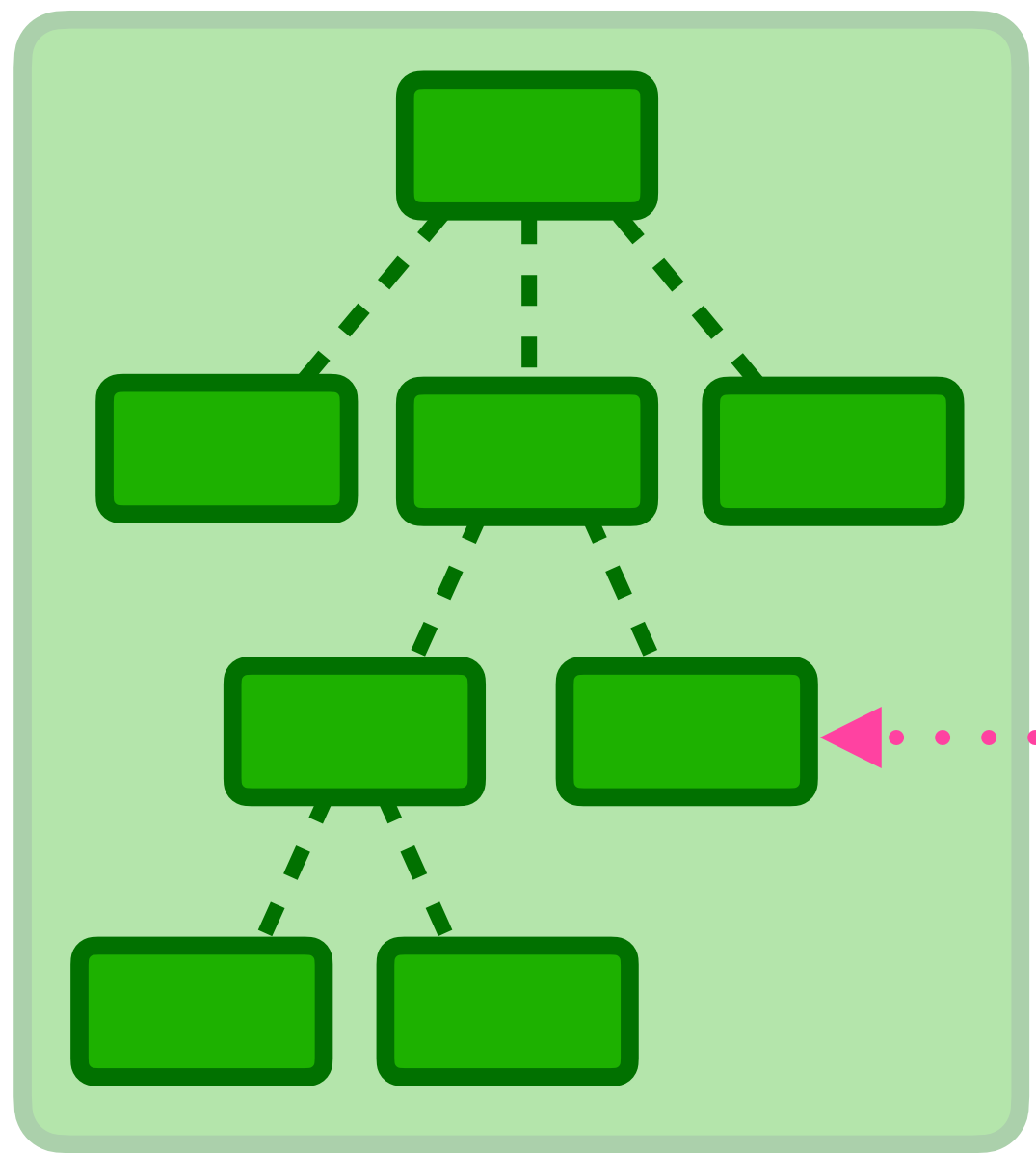
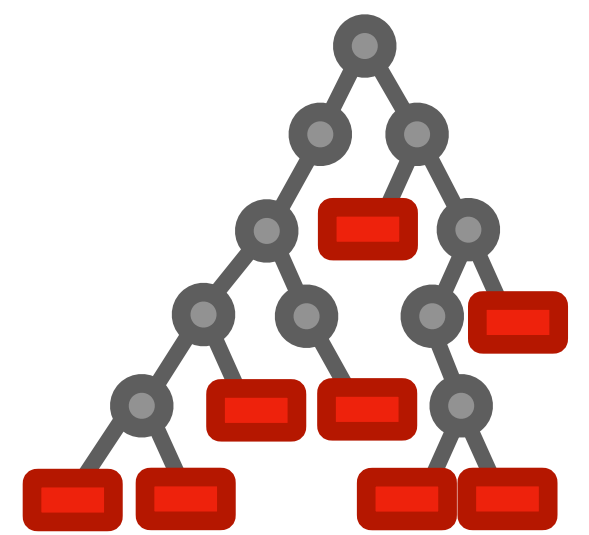
Securing Data Access

Progressive Fast Forward

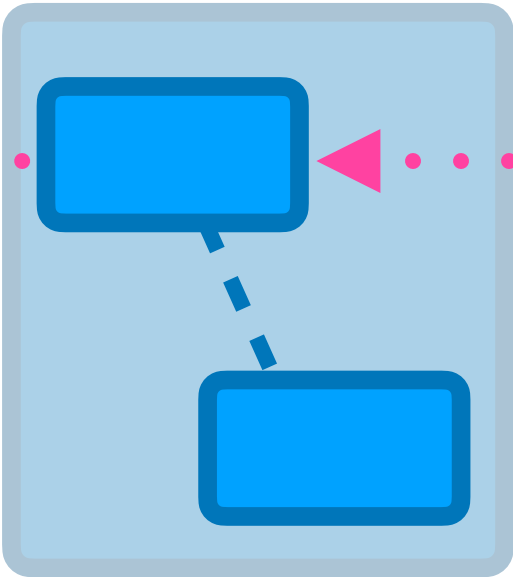


Securing Data Access

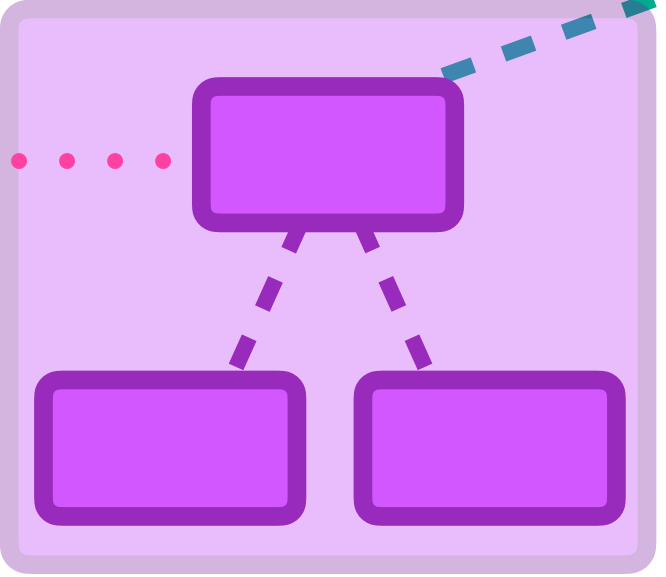
Progressive Fast Forward



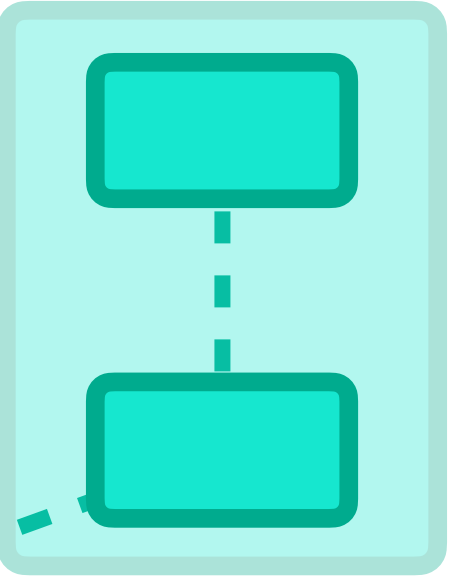
Rev 0



Rev 1
(Partial)



Rev 2
(Partial)

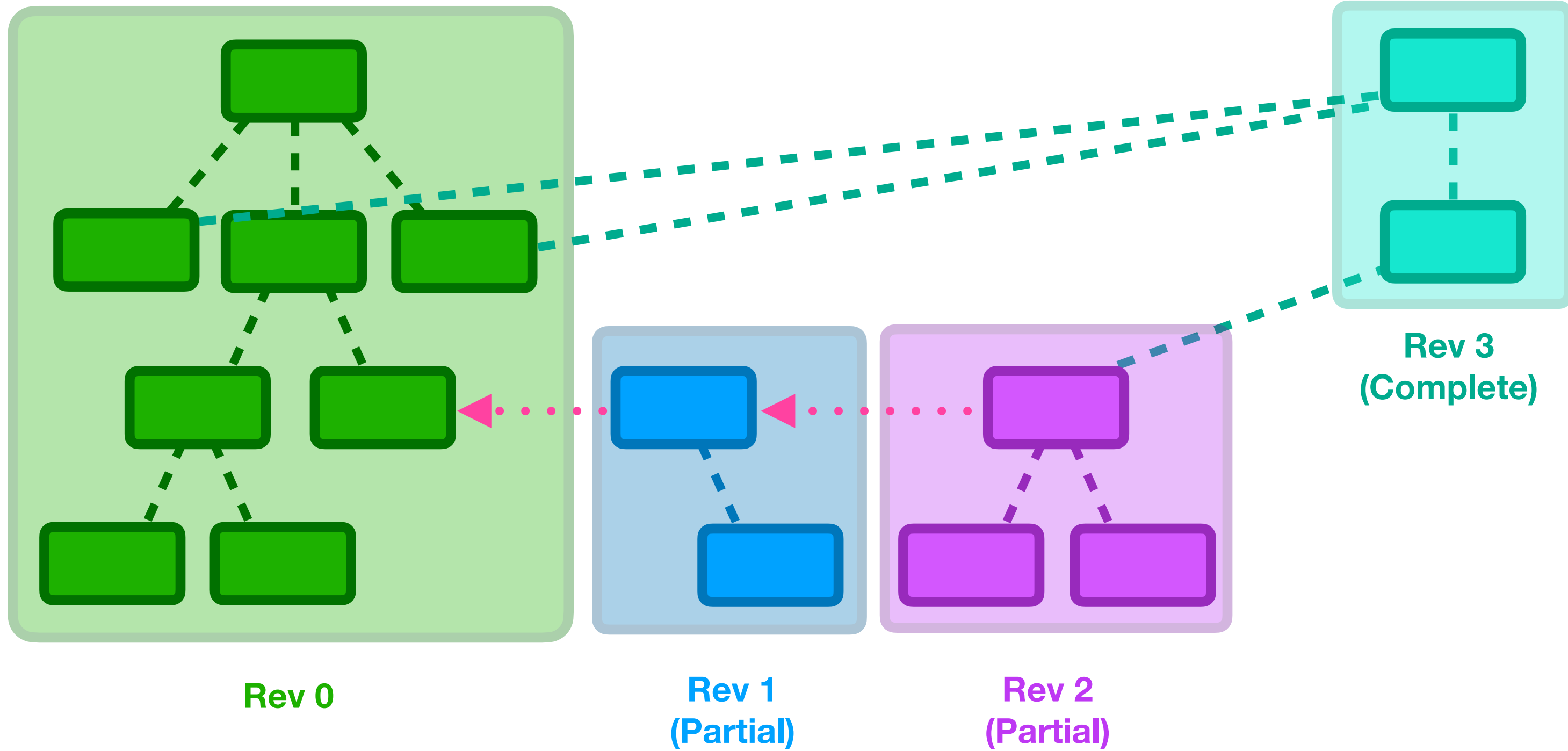
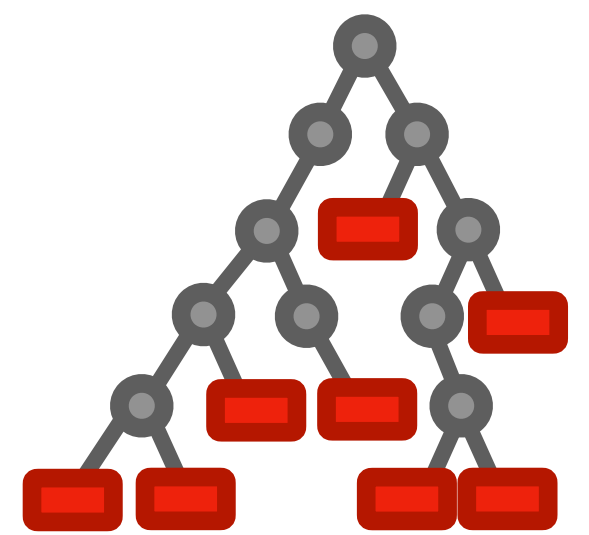


Rev 3
(Complete)



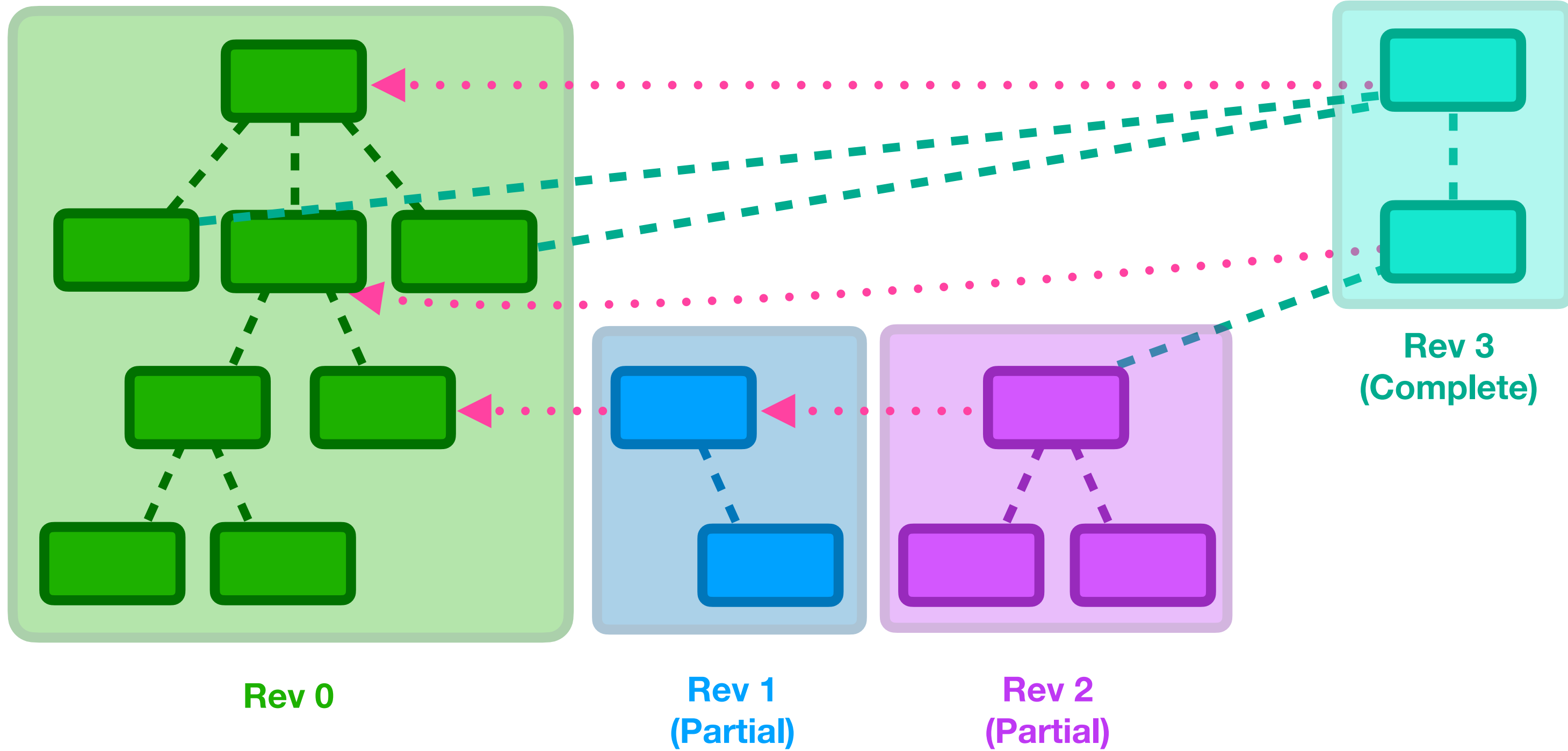
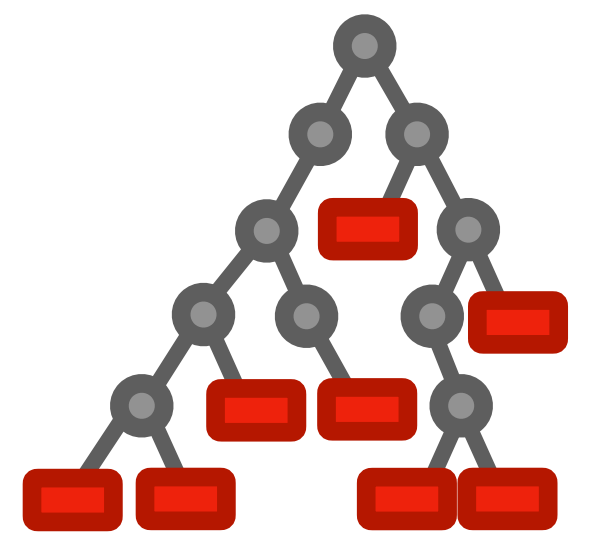
Securing Data Access

Progressive Fast Forward



Securing Data Access

Progressive Fast Forward



Securing Data Access

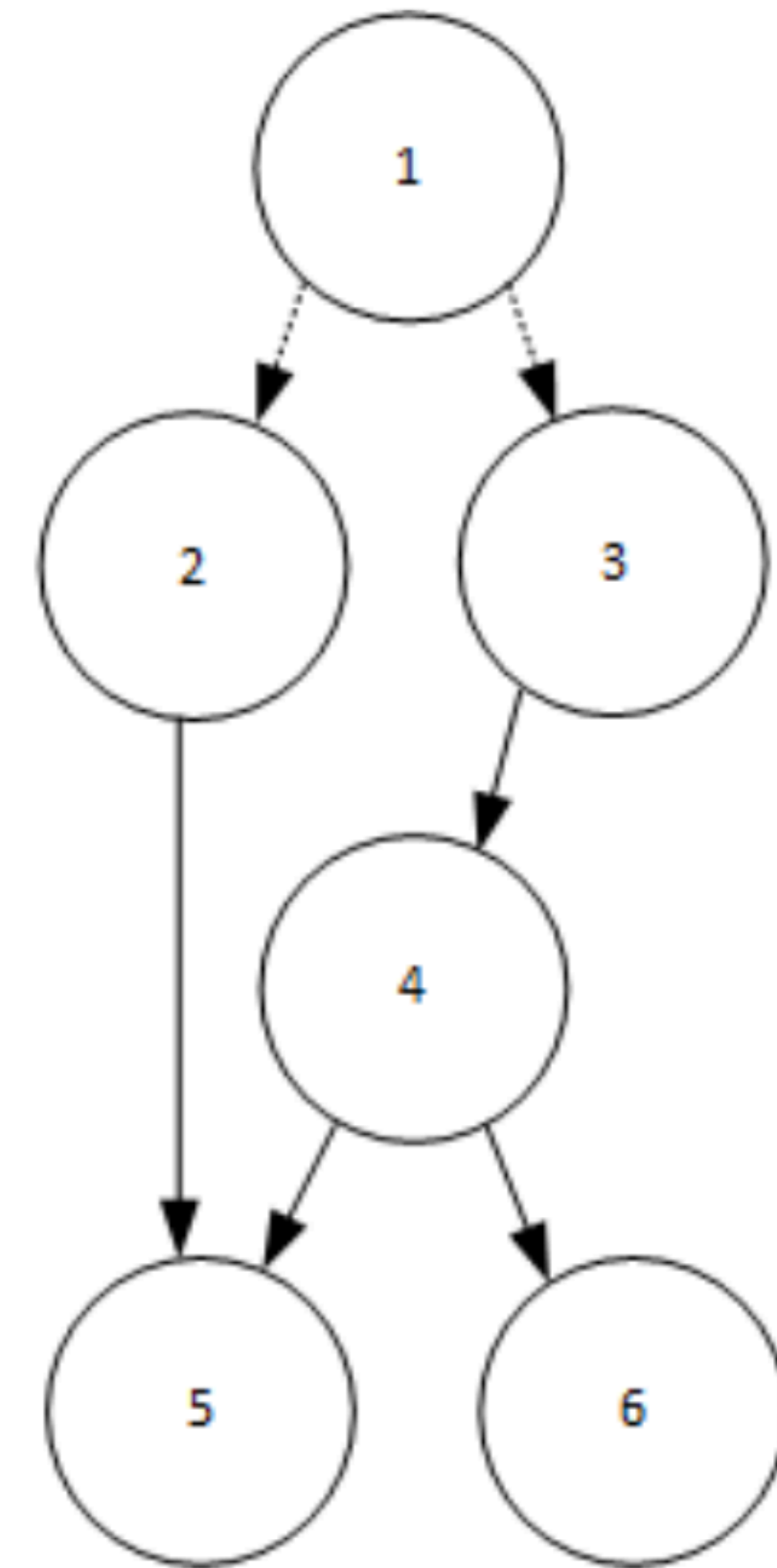
Merkle CRDT

Single File Version Shadow

Securing Data Access

Merkle CRDT

- Original paper from PL
- Persistent data structure by default
- Confluent with automated reconciliation
- Innate causal clock via Merkle DAG
- Coarse grained (path-level)



Single File Version Shadow

Securing Data Access

Async Granting Read & Write

Securing Data Access

Async Granting Read & Write



Securing Data Access

Async Granting Read & Write

Shared *by* Me

did:key:zStEksDrxkwYmpzqB
dAQjjx1PRbHG3fq4ChGeJcYU
YU44a4CBUExTTjeCbop6Uur


Securing Data Access

Async Granting Read & Write

Shared *by Me*

did:key:zStEksDrxkwYmpzqB
dAQjjx1PRbHG3fq4ChGeJcYU
YU44a4CBUExTTjeCbop6Uur

Human Readable Name

 **Symlink**


Securing Data Access

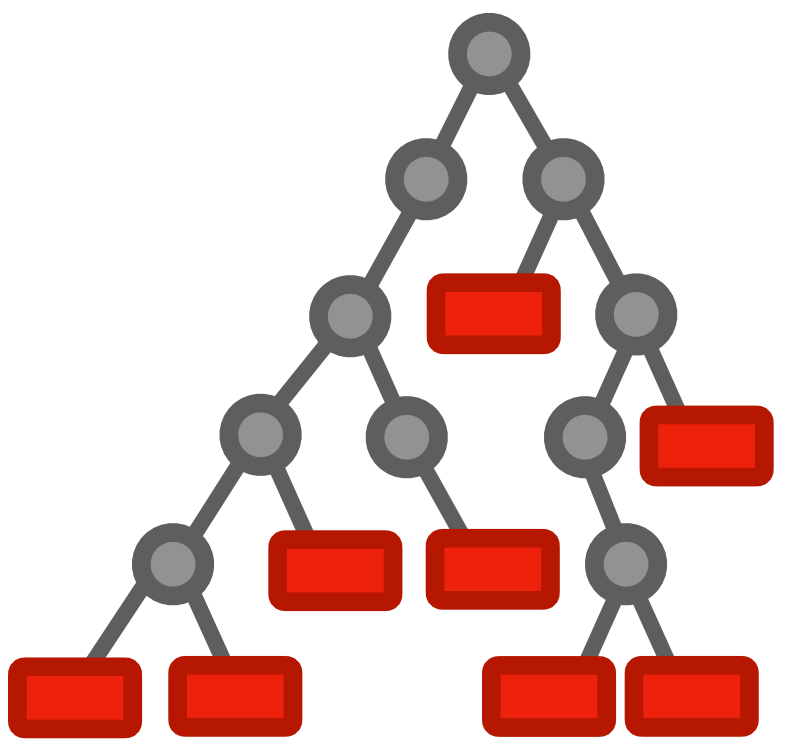
Async Granting Read & Write

Shared *by Me*

did:key:zStEksDrxkwYmpzqB
dAQjjx1PRbHG3fq4ChGeJcYU
YU44a4CBUExTTjeCbop6Uur

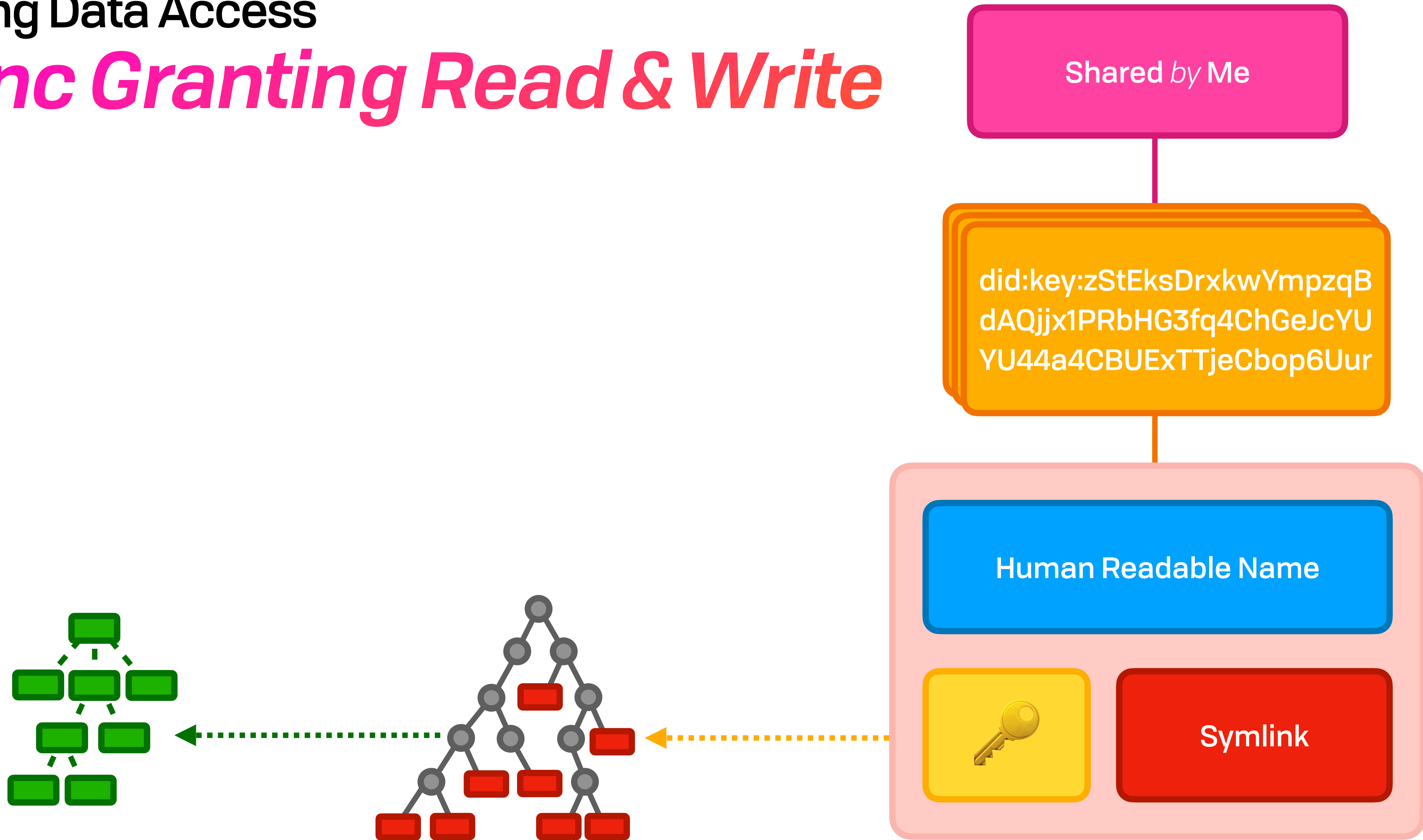
Human Readable Name

 **Symlink**



Securing Data Access

Async Granting Read & Write



Securing Data Access

Async Granting Read & Write

