AUTHORIZING USERS *WITHOUT* A BACKEND

…AND UCAN TOO 😉

fission

AUTHORIZING USERS WITHOUT A BACKEND

BROOKLYN ZELENKA, @expede

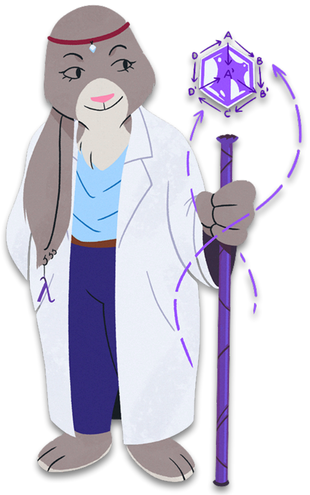
fission

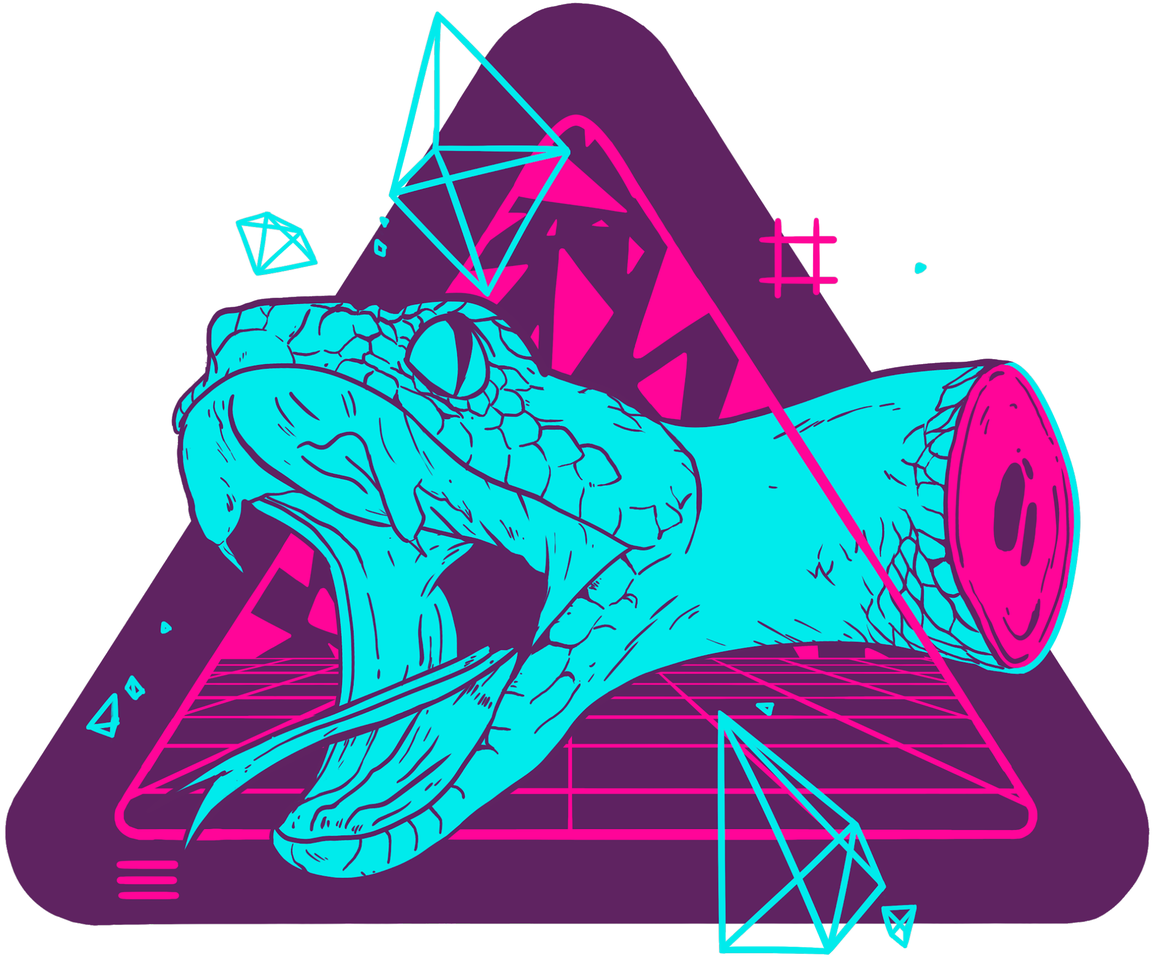
AUTHORIZING USERS WITHOUT A BACKEND

BROOKLYN ZELENKA, @expede

- Cofounder/CTO at Fission
  - https://fission.codes
- PLT & VMs
- Previously an Ethereum Core Dev
  - EIPs 615, 902, 1066, 1444
  - ECIP 1050
- VanFP, Code & Coffee YVR
- Witchcraft, Algae, Exceptional, & others

fission

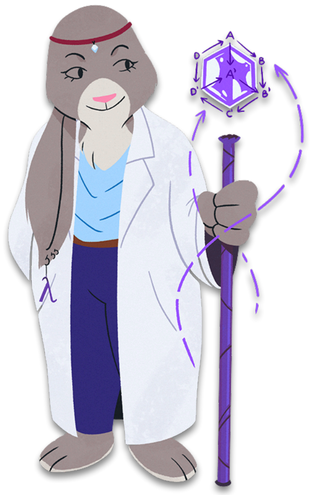


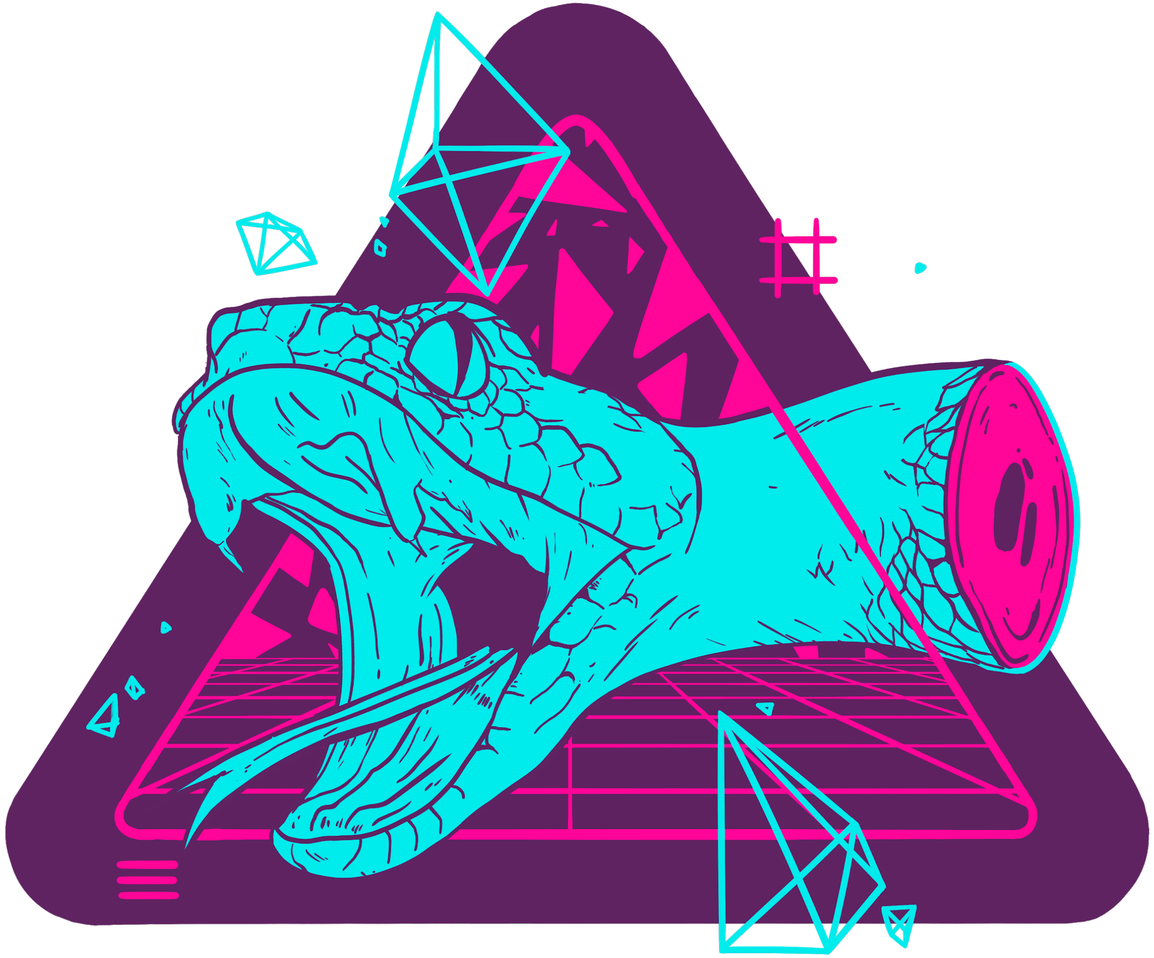


SCREAMING_SNAKE_CASE

WE HAVE STICKERS!

fission





S C R E A M I N G _ S N A K E _ C A S E

WE HAVE STICKERS!

PING ME AND WE'LL MAIL SOME

fission

SOME BACKGROUND CONTEXT

SOME BACKGROUND CONTEXT

WHAT SET OF PROBLEMS IS FISSION SOLVING?

SOME BACKGROUND CONTEXT

SHIPPING A WEB APP IN 2020 IS TOO HARD!

Backends

- Multi-tenant
- Increasingly sharded
- Highly concurrent
- Data leaks everywhere 😱
- **ACL complexity & GDPR**

DevOps

- Expensive & complex
- Very much its specialty
- We're close to peak Kubernetes

SOME BACKGROUND CONTEXT

SHIPPING A WEB APP IN 2020 I

Backends

- Multi-tenant
- Increasingly sharded
- Highly concurrent
- Data leaks everywhere 😱
- **ACL complexity & GDPR**

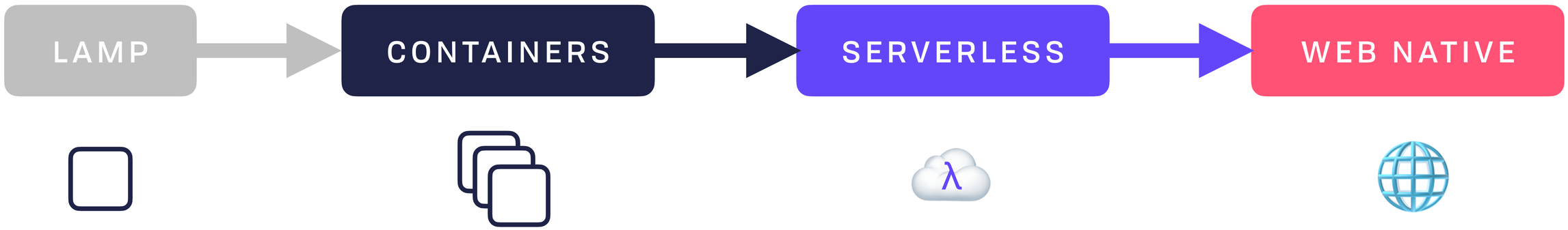
DevOp

- Expe
- Very
- We'

SOME BACKGROUND CONTEXT

FRONTEND IS EATING THE BACKEND 🍔😋

- Frontend is never going away
- Browsers keep getting *more powerful* (e.g. WebAssembly, WebAuthN)
- Trend to more granular edge — Cloudflare Workers / Fastly Edge Cloud
- Empower front end devs / full stack web apps for the 20's and beyond 🚀



SOME BACKGROUND CONTEXT

CONSTRAINTS

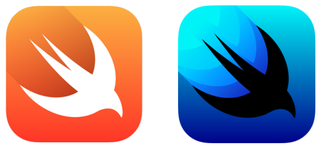
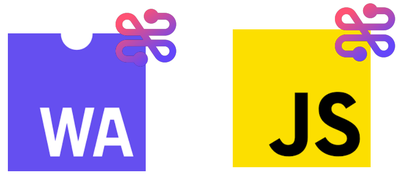
SOME BACKGROUND CONTEXT

CONSTRAINTS

- Everything for a modern web app directly in the browser
- Vanilla browsers only — no plug-ins
- As secure *or better* than with traditional cloud infra
- UX should feel the same or easier

SOME BACKGROUND CONTEXT

"WEB NATIVE"

	iOS	+ fission
COMPUTE		
IDENTITY		
STORAGE		

Build web apps more like native mobile & desktop

Password-less login, end-to-end encryption, secure by default

Local-first, secure, user controlled, global file & hosting platform

</shill>

OKAY, THE BACKEND GOES AWAY 👍

…NOW WHAT?

NOW WHAT?

WE HAVE SOME NEW BUILDING BLOCKS!

- Start thinking “universally”
- WebCrypto API 🔐
- Self-sovereign identity / DID 🛂
- Content addressing #️⃣
- Macaroons 🍪
- Resurrecting SPKI auth 🧟‍♀️👻
- CQRS applied to authZ (separate methods)

(Disclaimer: taken care of under the hood, but interoperable)

NOW WHAT?

WE HAVE SOME NEW BUILDING BLOCKS!

- Start thinking “universally”
- WebCrypto API 🔐
- Self-sovereign identity / DID 🛂
- Content addressing #️⃣
- Macaroons 🍪
- Resurrecting SPKI auth 🧟‍♀️👻
- CQRS applied to authZ (separate methods)

(Disclaimer: taken care of under the hood, but interoperable)

STEP ONE

USER IDS WITHOUT A DATABASE

USER IDS WITHOUT A DATABASE

STANDARDIZATION 🏢

USER IDS WITHOUT A DATABASE

STANDARDIZATION 🏢

- W3C, Microsoft, BC, etc
- For users, devices, and more
- Based on public-key cryptography
- Truly “universal” UUIDs
- Agnostic about backing

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"
  }]
}
```

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🤦‍♀️

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🦹‍♀️

- Generate your own globally-unique, verifiable user ID!

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🦹‍♀️

- Generate your own globally-unique, verifiable user ID!
- As many as you like 🤷‍♀️

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🦹‍♀️

- Generate your own globally-unique, verifiable user ID!
- As many as you like 🤷‍♀️
- Many methods — we're starting with "did:key"

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🦹‍♀️

- Generate your own globally-unique, verifiable user ID!
- As many as you like 🤷‍♀️
- Many methods — we're starting with "did:key"
- Not super readable, so publicize over DNS TXT record

```
_did.USERNAME.fission.name
```

USER IDS WITHOUT A DATABASE

SELF-SOVEREIGN IDENTITY (SSI) 🔑🤦‍♀️

- Generate your own globally-unique, verifiable user ID!
- As many as you like 🤷‍♀️
- Many methods — we're starting with "did:key"
- Not super readable, so publicize over DNS TXT record

```
_did.USERNAME.fission.name
```

```
did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUHzngyNKmKx4VKWEJE6sk4SE4Ka3kH92MxU2YC7CcePHy77GzZy8
```

```
Ed25519 — AAAAC3NzaC1lZDI1NTE5AAAAIB7/gFUQ9llI1BTrEjW7Jq6fX6JLsK1J4wXK/dn9JMcO
```

STEP TWO

DISTRIBUTED READ CONTROL

DISTRIBUTED READ CONTROL

OCAP / READ KEYS

DISTRIBUTED READ CONTROL

OCAP / READ KEYS

- ACLs
  - "Reactive access control”
  - Authority by association

DISTRIBUTED READ CONTROL

OCAP / READ KEYS

- ACLs
  - "Reactive access control”
  - Authority by association
- OCAP
  - “Proactive" access control
  - Authority by possession
    - “You either have the key, or you don’t”

DISTRIBUTED READ CONTROL

OCAP / READ KEYS

- ACLs
  - "Reactive access control”
  - Authority by association
- OCAP
  - “Proactive" access control
  - Authority by possession
    - “You either have the key, or you don’t”
- Normal AES-256 keys

DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

- Public keys playing double duty: IDs and secure key exchange!

DISTRIBUTED READ CONTROL

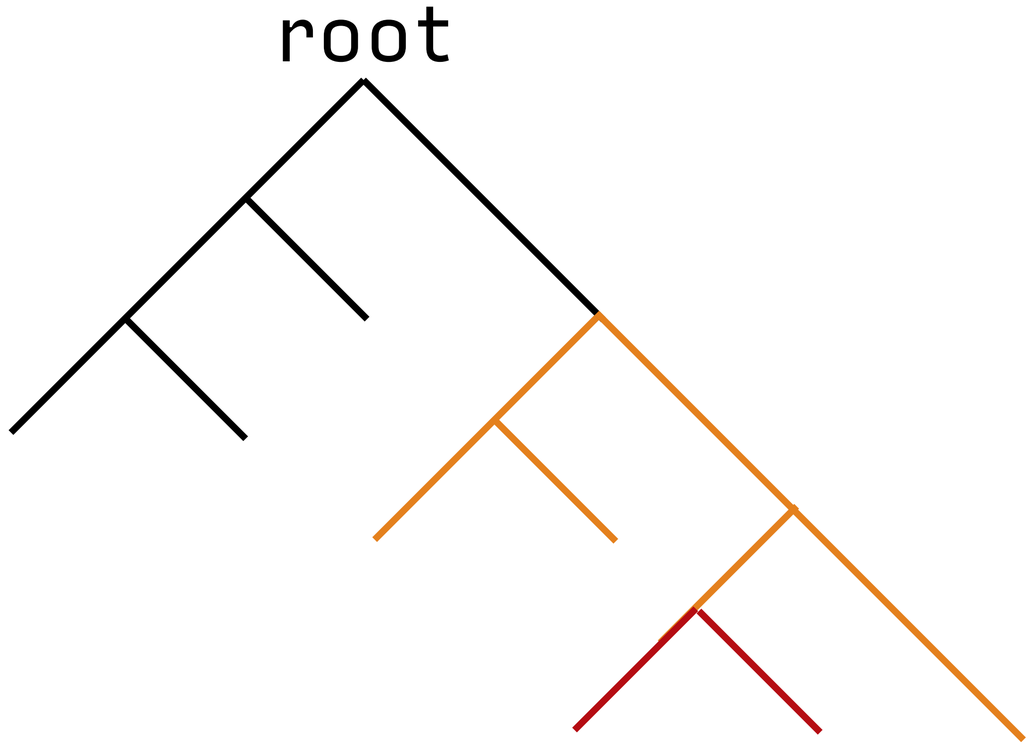
MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

- Public keys playing double duty: IDs and secure key exchange!
- Encrypt the encryption with more encryption
  - Each layer (file or dir) is encrypted with a key
  - Dirs contain keys for each sub dir / file
  - Recurse!

DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

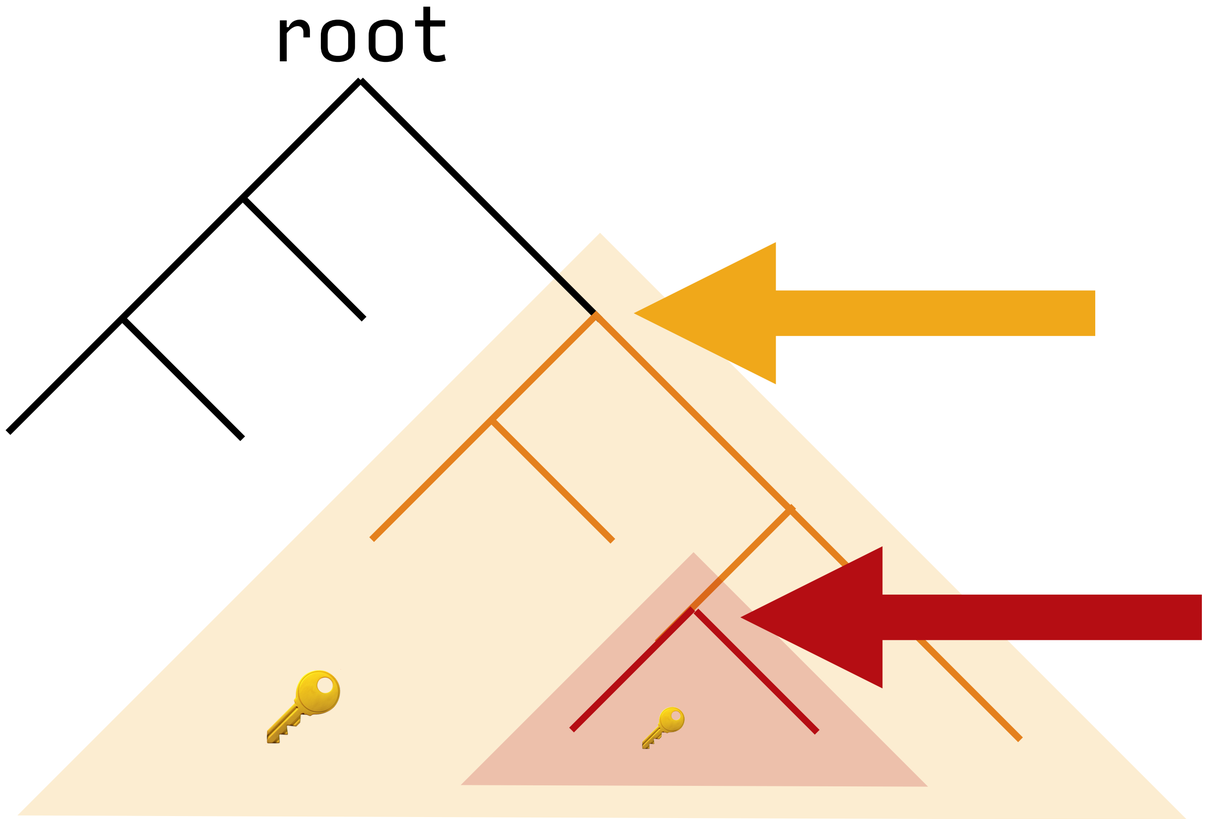
- Public keys playing double duty: IDs and secure key exchange!
- Encrypt the encryption with more encryption
  - Each layer (file or dir) is encrypted with a key
  - Dirs contain keys for each sub dir / file
  - Recurse!



DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

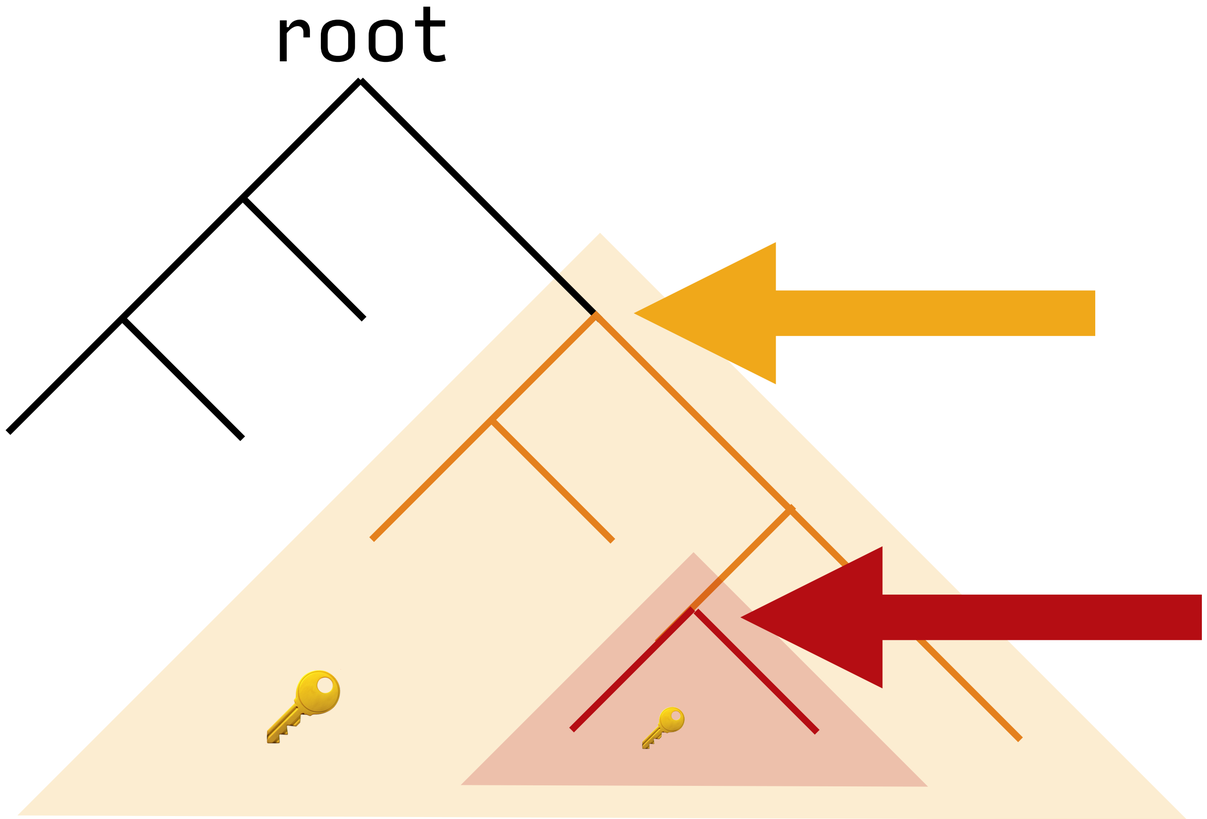
- Public keys playing double duty: IDs and secure key exchange!
- Encrypt the encryption with more encryption
  - Each layer (file or dir) is encrypted with a key
  - Dirs contain keys for each sub dir / file
  - Recurse!



DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

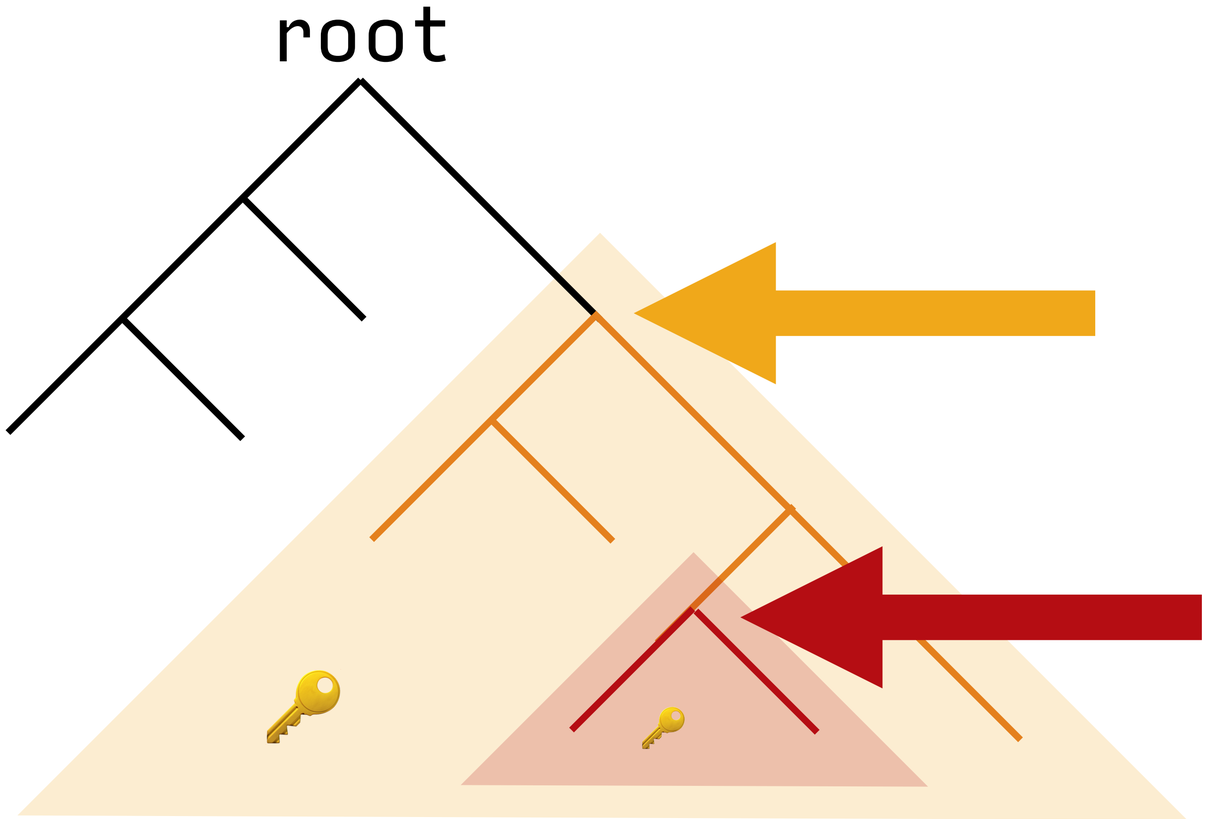
- Public keys playing double duty: IDs and secure key exchange!
- Encrypt the encryption with more encryption
  - Each layer (file or dir) is encrypted with a key
  - Dirs contain keys for each sub dir / file
  - Recurse!
- Access granted to a directory and below
  - i.e. Same UX Dropbox/Google Drive
  - Full user controlled



DISTRIBUTED READ CONTROL

MORE GRANULAR ACCESS: CRYPTREES 🔐🌳

- Public keys playing double duty: IDs and secure key exchange!
- Encrypt the encryption with more encryption
  - Each layer (file or dir) is encrypted with a key
  - Dirs contain keys for each sub dir / file
  - Recurse!
- Access granted to a directory and below
  - i.e. Same UX Dropbox/Google Drive
  - Full user controlled
- Revocation = key rotation & DH exchange

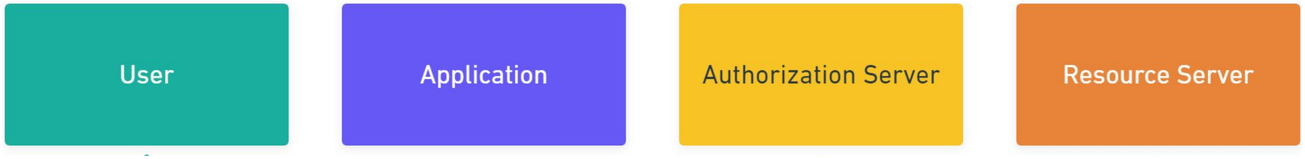


STEP THREE

DELEGATED WRITE ACCESS

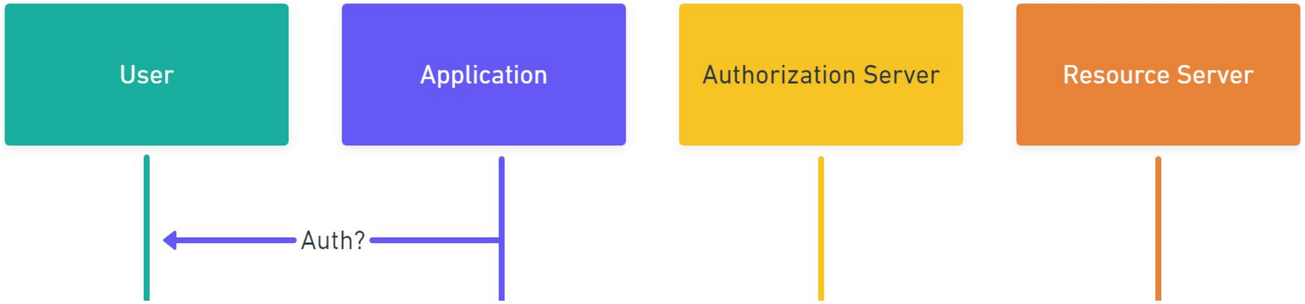
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



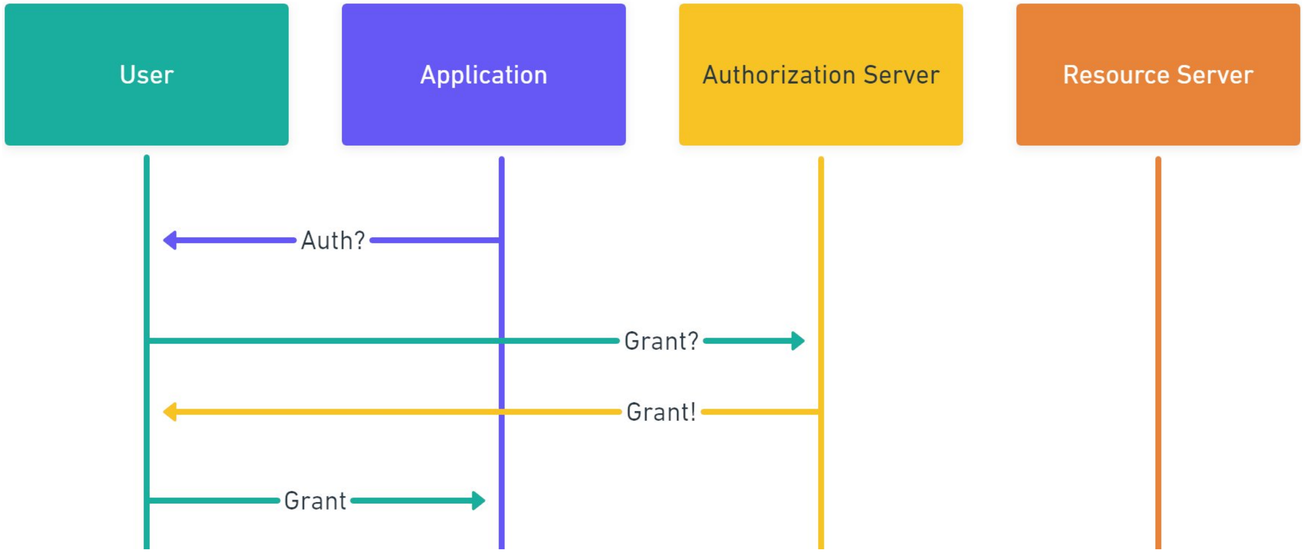
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



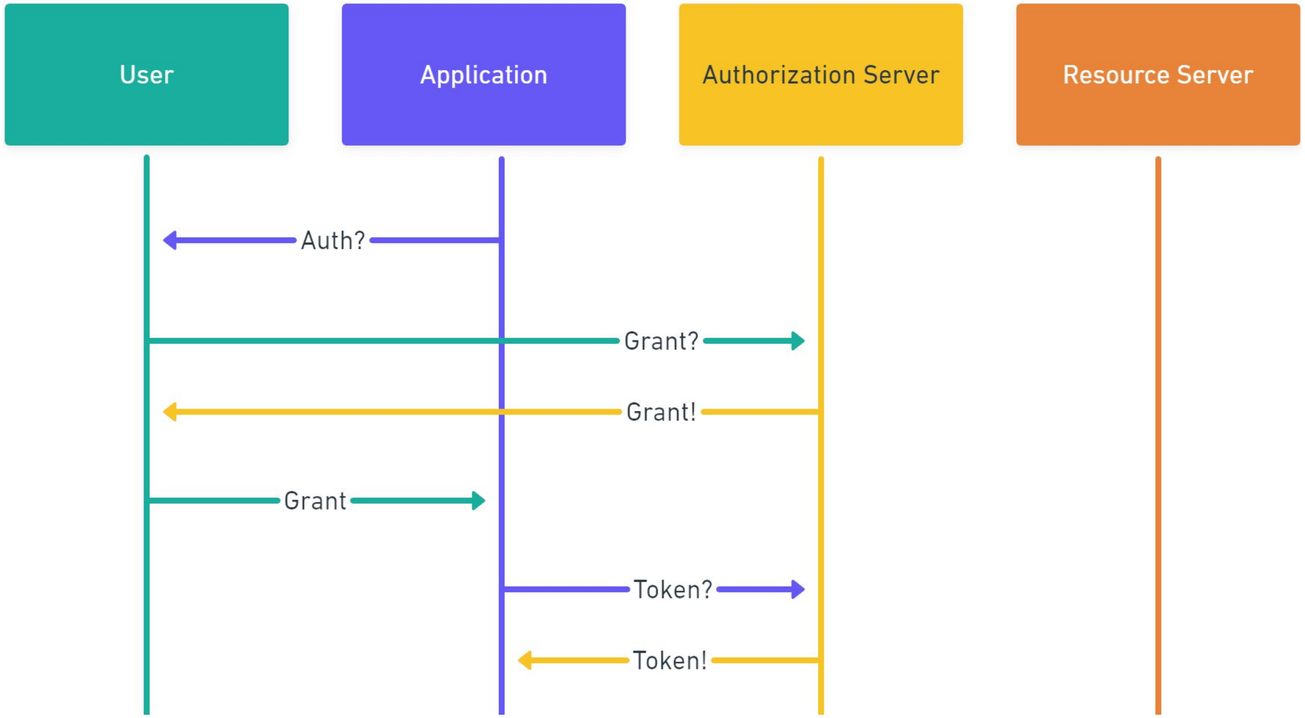
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



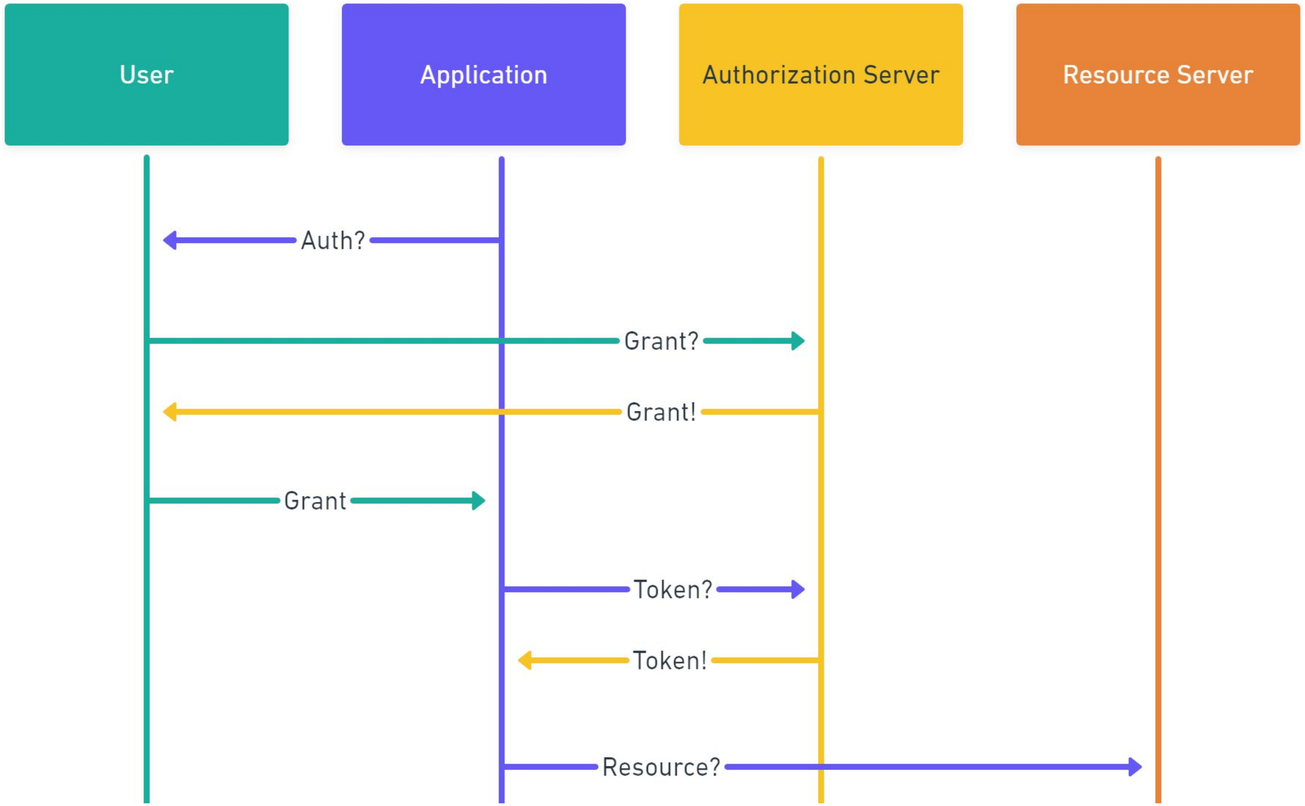
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



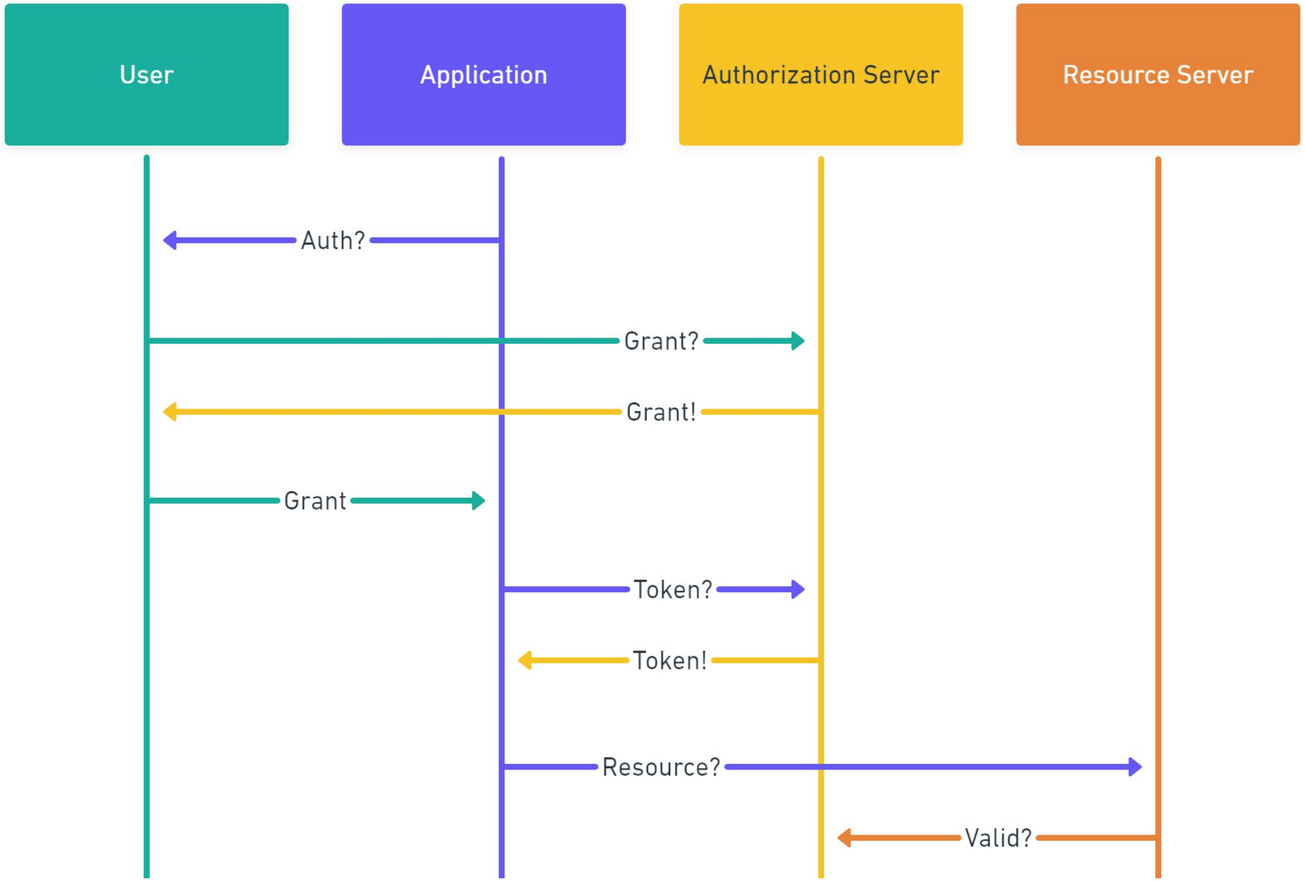
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



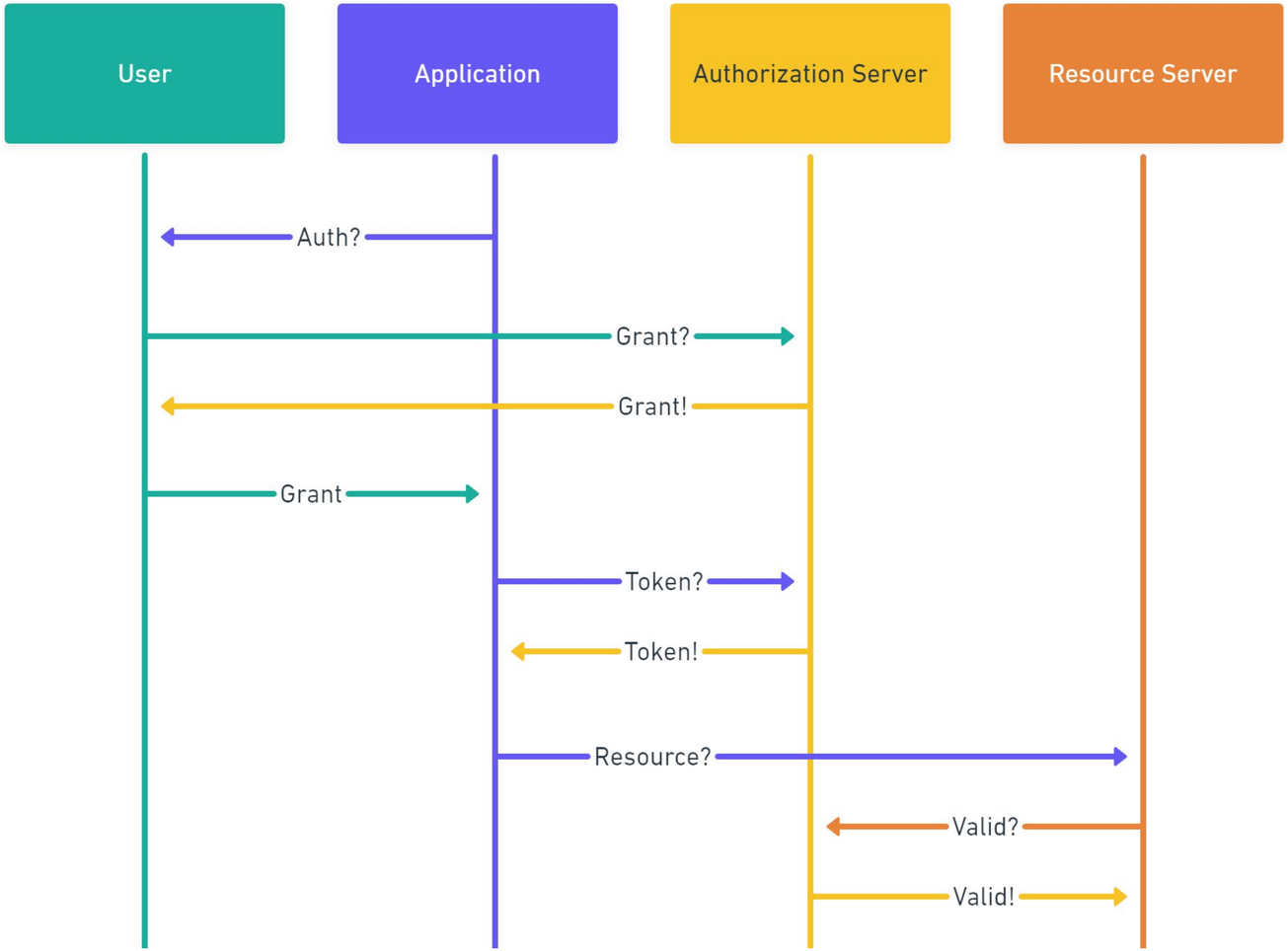
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



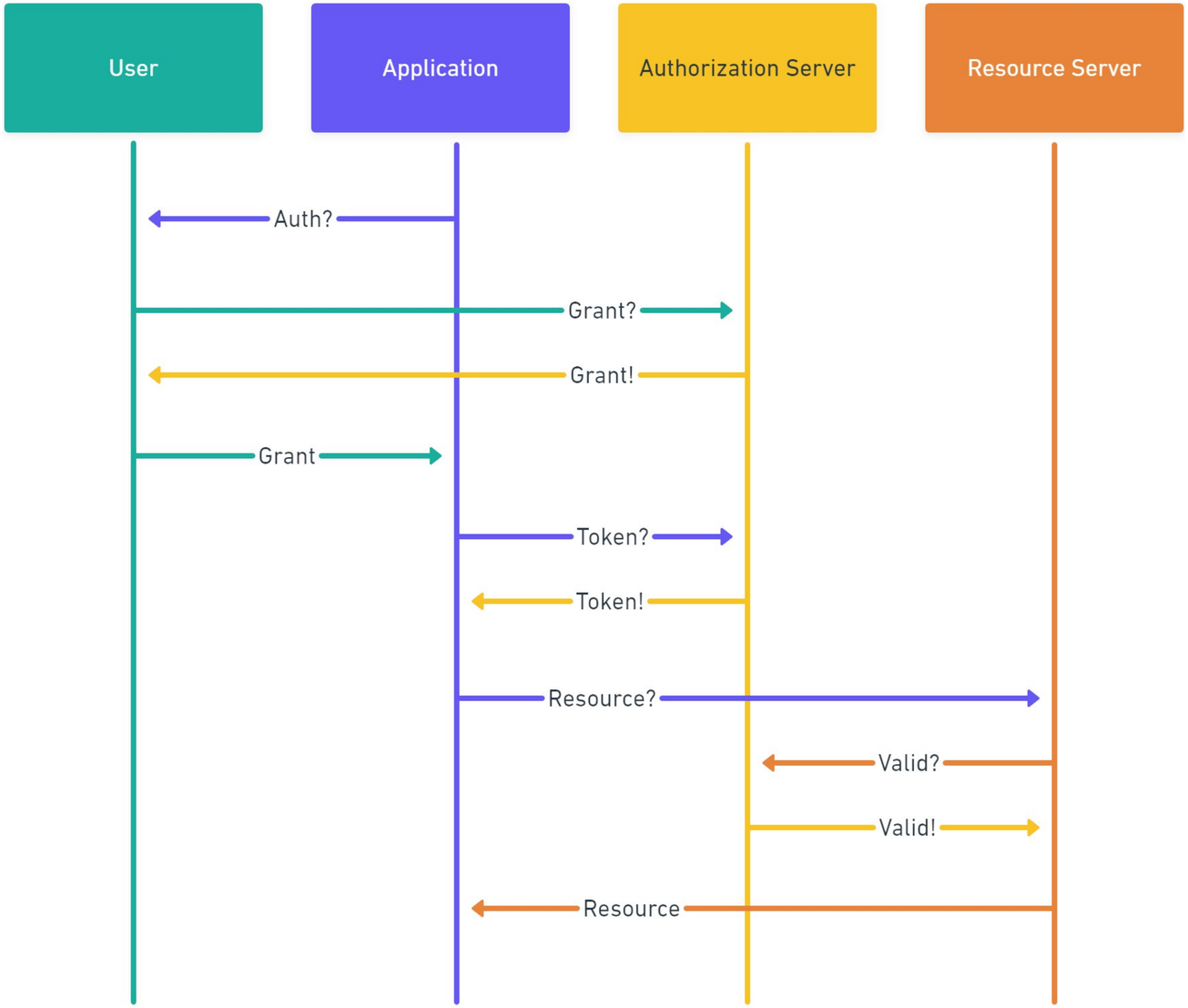
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



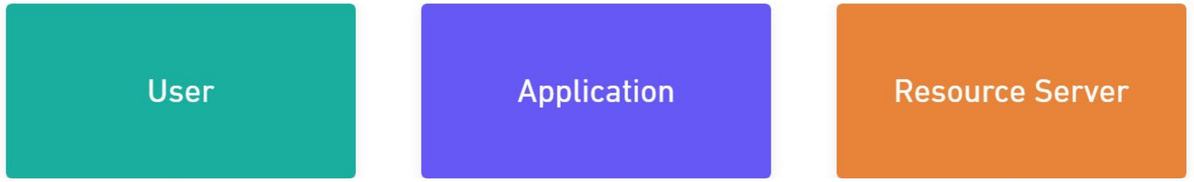
DELEGATED WRITE ACCESS

STATUS QUO: OAUTH



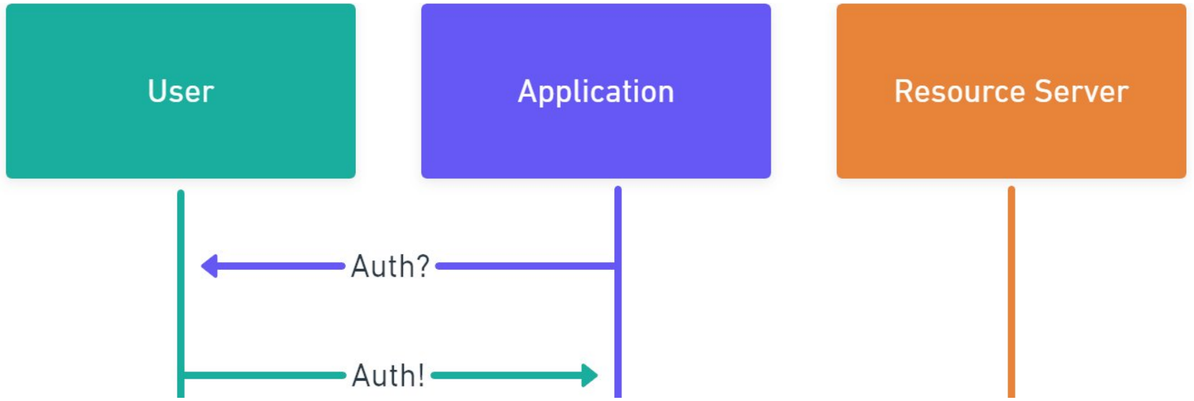
DELEGATED WRITE ACCESS

SELF-SIGNED TOKENS (UCAN)



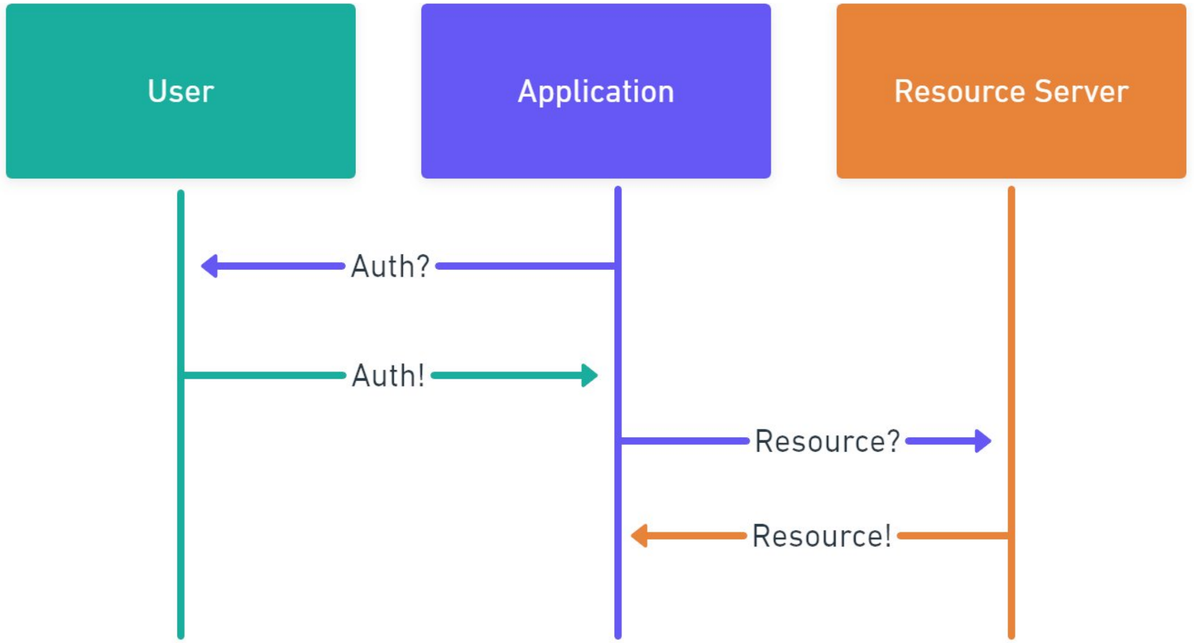
DELEGATED WRITE ACCESS

SELF-SIGNED TOKENS (UCAN)



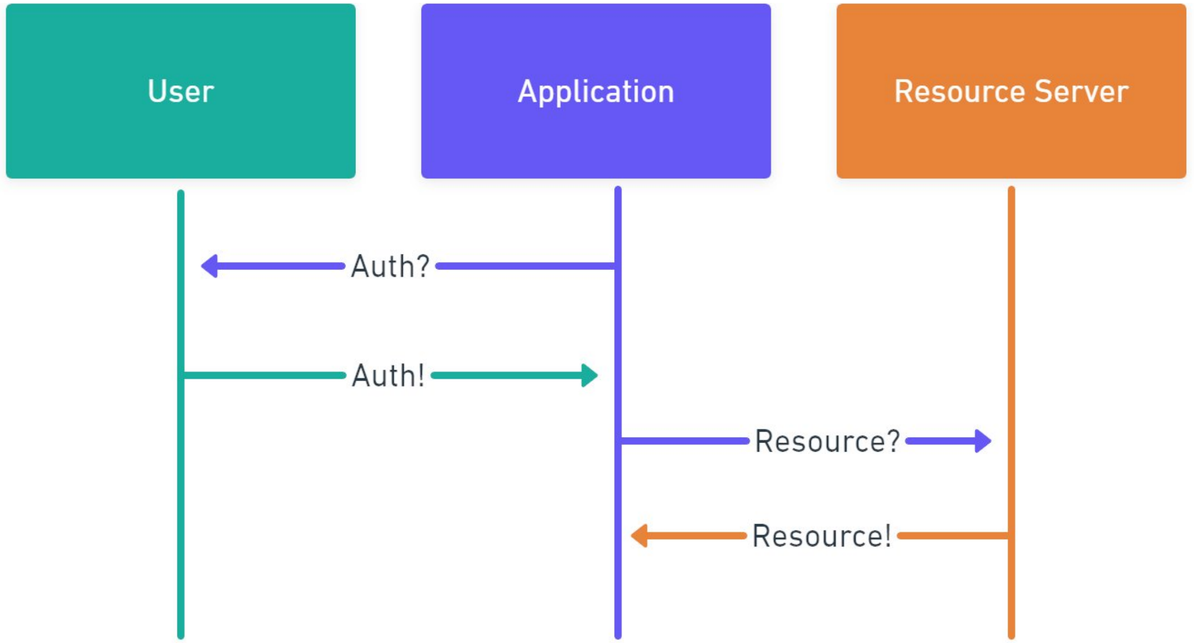
DELEGATED WRITE ACCESS

SELF-SIGNED TOKENS (UCAN)



DELEGATED WRITE ACCESS

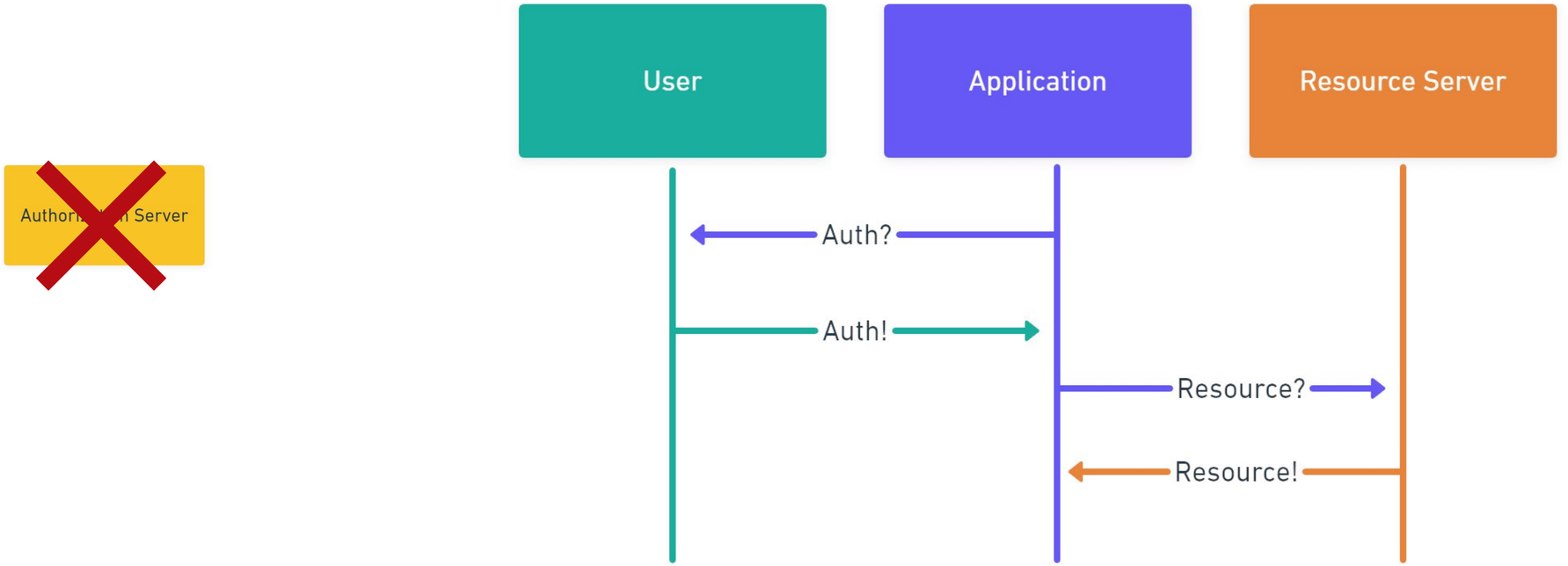
SELF-SIGNED TOKENS (UCAN)



DONE!

DELEGATED WRITE ACCESS

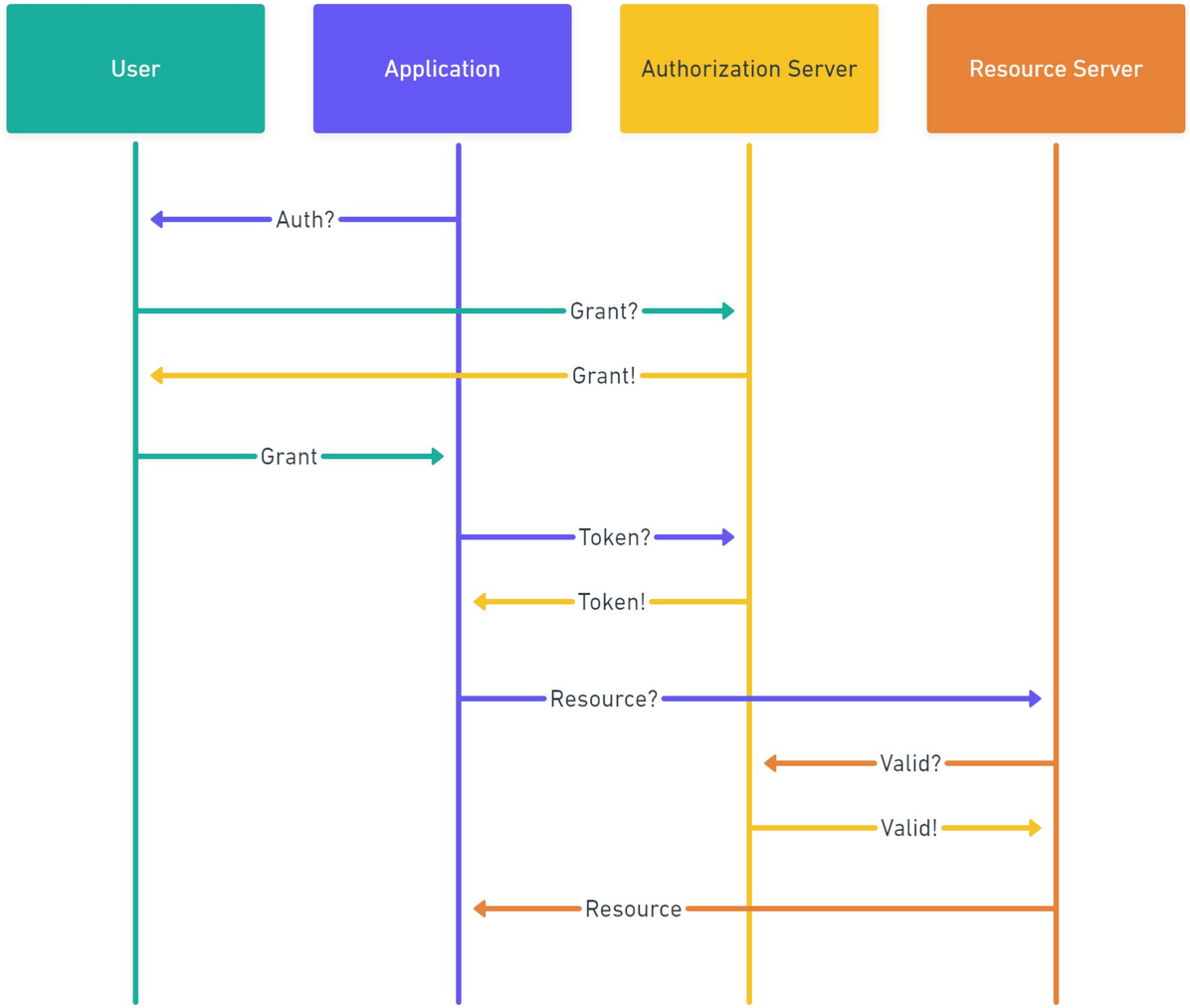
SELF-SIGNED TOKENS (UCAN)

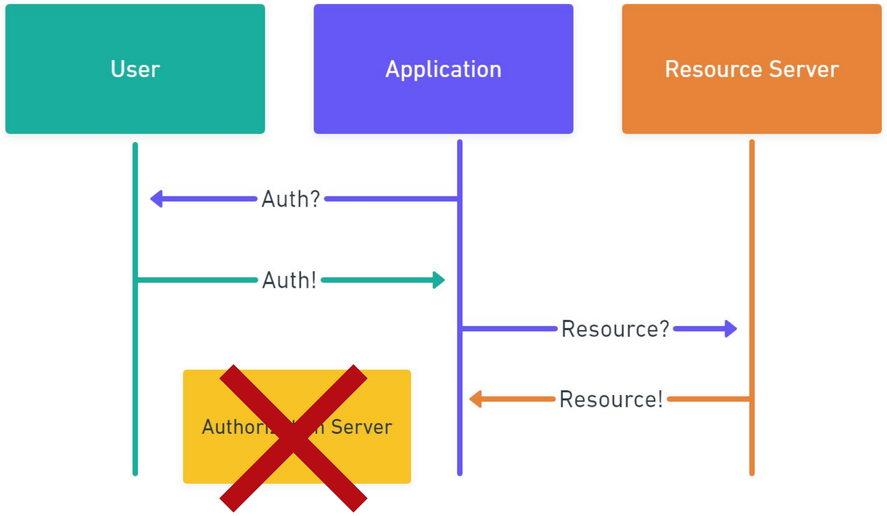


DONE!

DELEGATED WRITE ACCESS

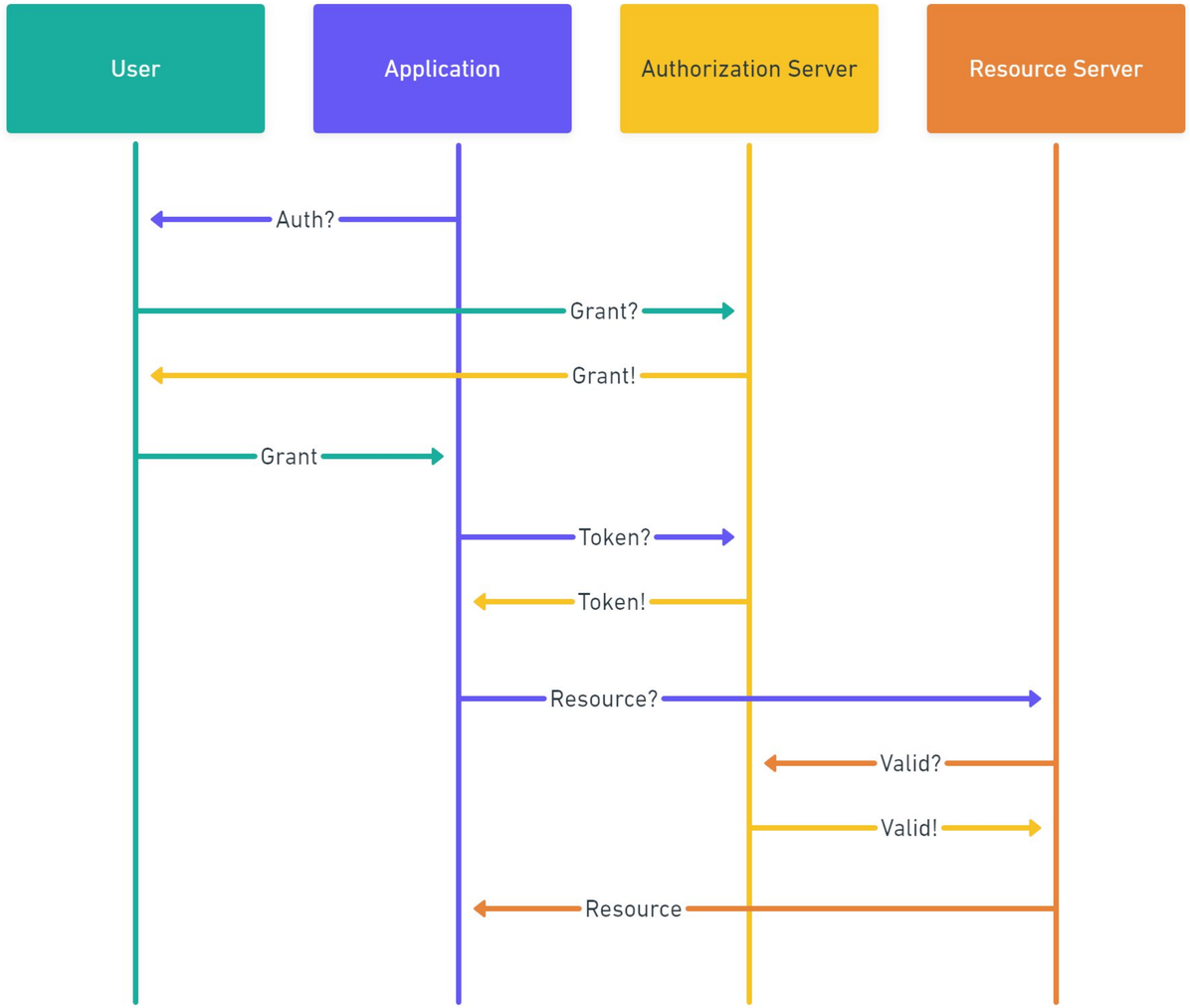
SIDE-BY-SIDE

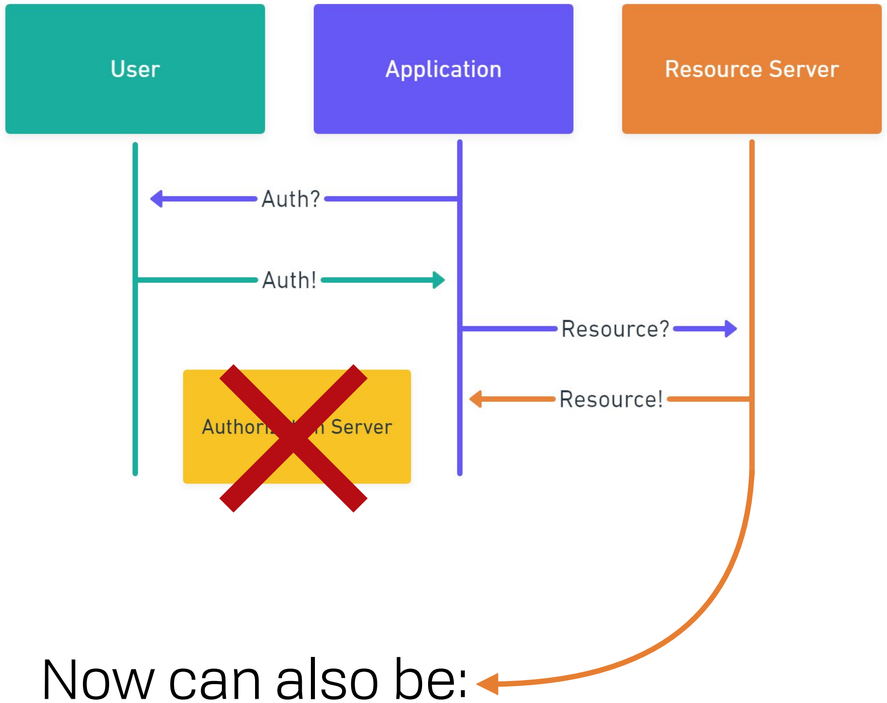




DELEGATED WRITE ACCESS

SIDE-BY-SIDE





Now can also be:

- Another device (same human)
- A user's peer (different human)
- Some service

DELEGATED WRITE ACCESS

GOOGLE'S MACAROONS: "STACKED COOKIES" 🍪 🍪 🍪

Root Proof

DELEGATED WRITE ACCESS

GOOGLE'S MACAROONS: "STACKED COOKIES" 🍪 🍪 🍪

Root Proof

- Solves for Google's infra
- Decentralized delegation
- Attenuation
- Shrink size with HMACs
- Assumes auth servers

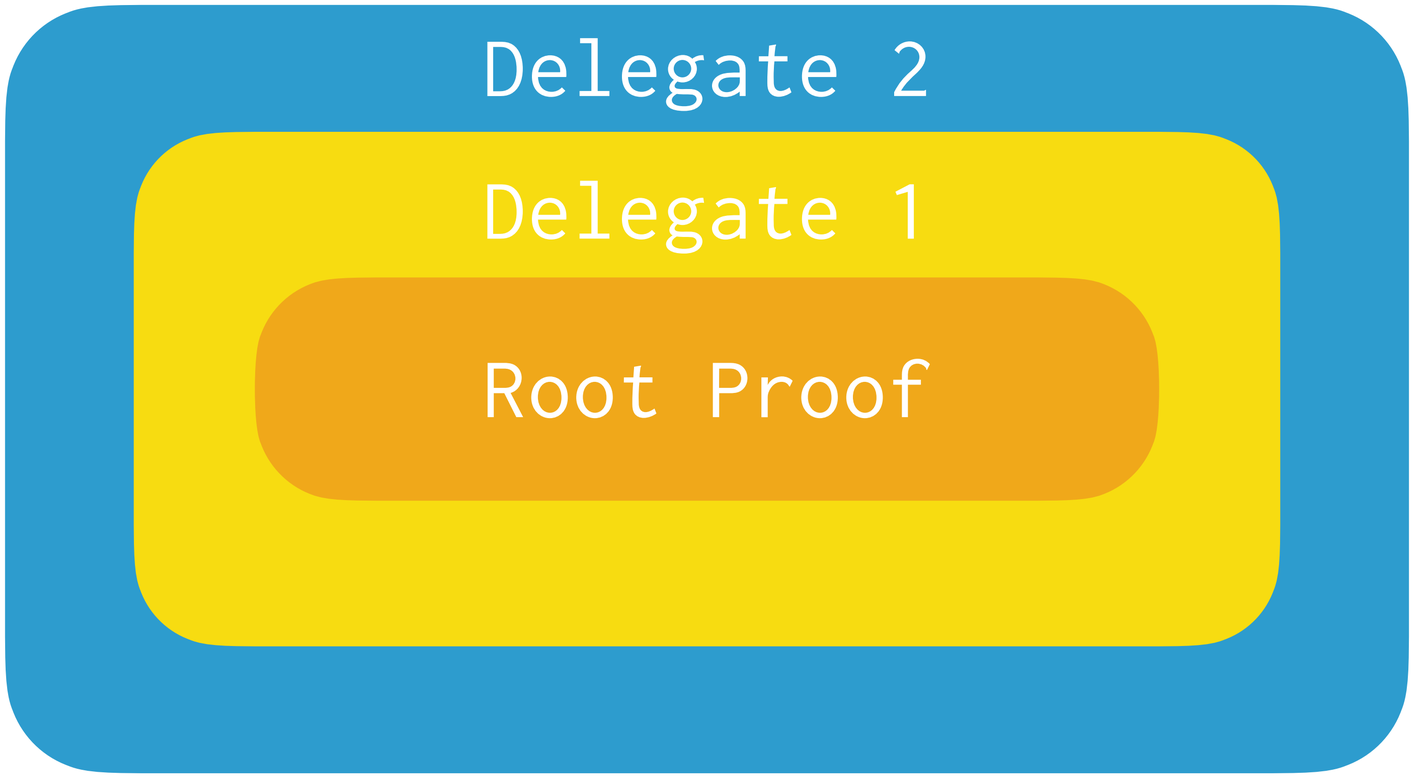
DELEGATED WRITE ACCESS

GOOGLE'S MACAROONS: "STACKED COOKIES" 🍪 🍪 🍪

- Solves for Google's infra
- Decentralized delegation
- Attenuation
- Shrink size with HMACs
- Assumes auth servers

DELEGATED WRITE ACCESS

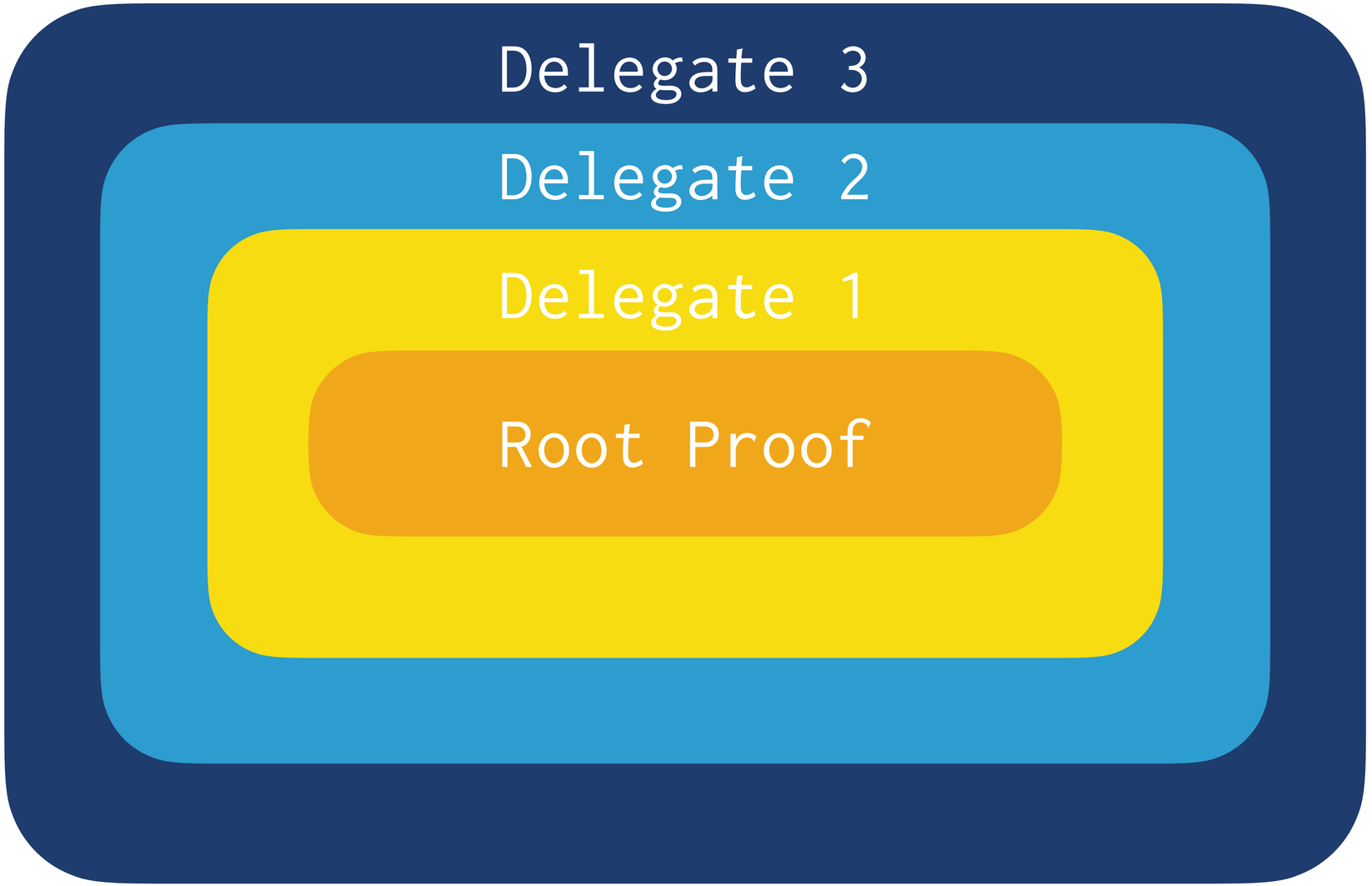
GOOGLE'S MACAROONS: "STACKED COOKIES" 🍪 🍪 🍪



- Solves for Google's infra
- Decentralized delegation
- Attenuation
- Shrink size with HMACs
- Assumes auth servers

DELEGATED WRITE ACCESS

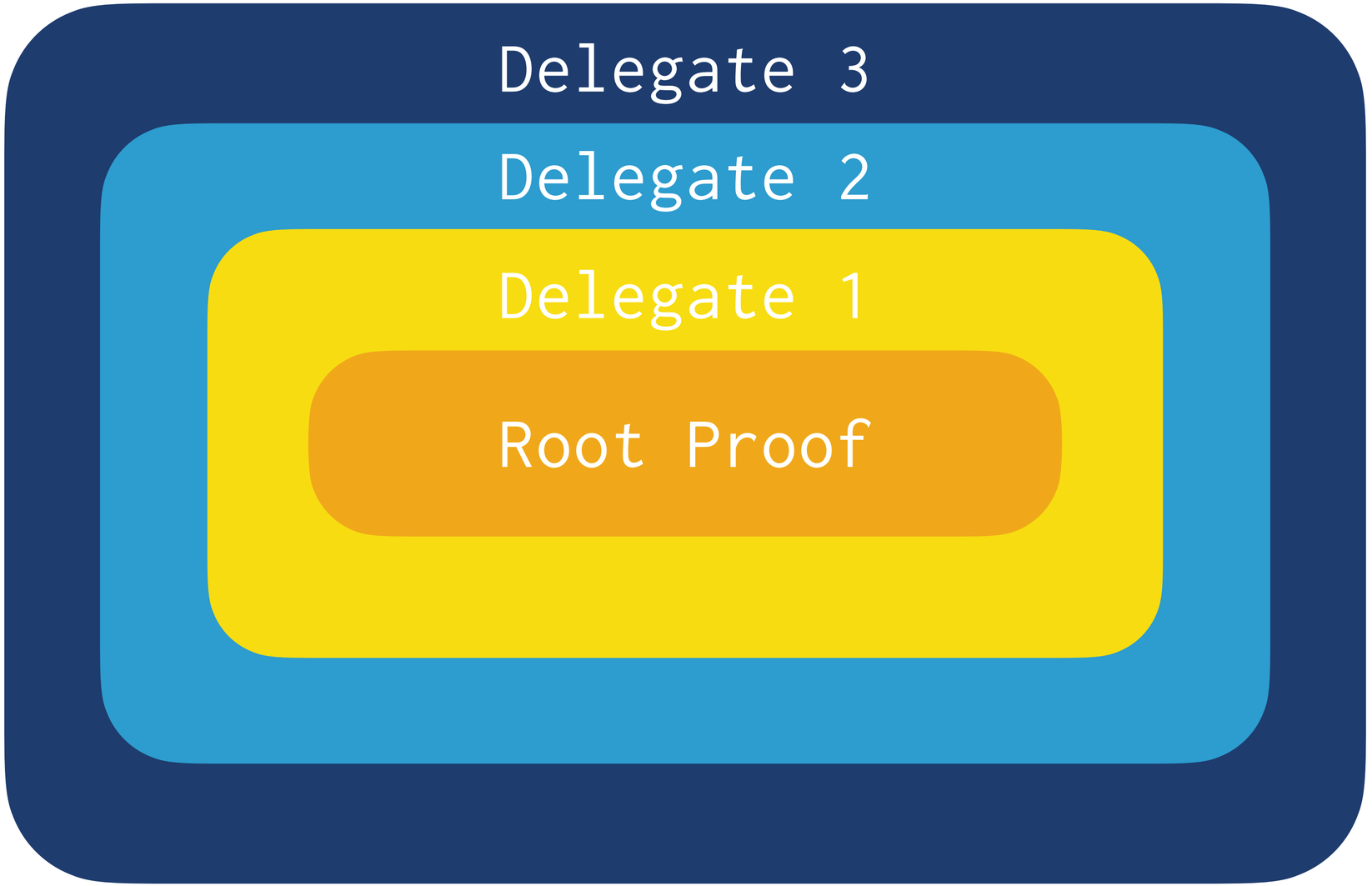
GOOGLE'S MACAROONS: "STACKED COOKIES" 🍪 🍪 🍪



- Solves for Google's infra
- Decentralized delegation
- Attenuation
- Shrink size with HMACs
- Assumes auth servers

DELEGATED WRITE ACCESS

UCAN: USER CONTROLLED AUTHORIZATION NETWORK 🦜



- Solves for user-centrism
- Decentralized delegation
- Attenuation
- Shrink size with CIDs
- Assumes PKI

DELEGATED WRITE ACCESS

EACH LAYER FOLLOWS THIS FORM

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:<JWT PROOF>,
  “exp”:1589423547
}
<SIGNATURE>
```

DELEGATED WRITE ACCESS

EACH LAYER FOLLOWS THIS FORM

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:<JWT PROOF>,
  “exp”:1589423547
}
<SIGNATURE>
```

DELEGATED WRITE ACCESS

EACH LAYER FOLLOWS THIS FORM

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:<JWT PROOF>,
  “exp”:1589423547
}
<SIGNATURE>
```

DELEGATED WRITE ACCESS

EACH LAYER FOLLOWS THIS FORM

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:<JWT PROOF>,
  “exp”:1589423547
}
<SIGNATURE>
```

Recursive

Problem: gets pretty big

DELEGATED WRITE ACCESS

HASHING IT DOWN

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:”QmaEmBULputJ5sAJX4bRQYwwWV2DUPnwNSz2R2eTvHV4DT”,
  “exp”:1589423547
}
<SIGNATURE>
```

DELEGATED WRITE ACCESS

HASHING IT DOWN

```
{
  “alg”: “RS256”,
  “typ”: “JWT“,
  “cty”: “JWT“
}
{
  “iss”:“did:key:z1MdJPaWBebKxtE33AszRWYTF67wCLeFdcsqc3R87hyLKzBK…”,
  “aud”:“did:key:zBR4m3DNZHT1G8Nb2RHzgKK7TrWxEmJjZskgvFdncTthzUH…”,
  “scp”:“/public/photos/covid2020/”,
  “pty”:”APPEND_ONLY”,
  “prf”:”QmaEmBULputJ5sAJX4bRQYwwWV2DUPnwNSz2R2eTvHV4DT”,
  “exp”:1589423547
}
<SIGNATURE>
```

RECAP

RECAP

WELL THAT WAS A LOT OF CONCEPTS

RECAP

WELL THAT WAS A LOT OF CONCEPTS

- Fully client-side auth
- User controlled / sharding logical conclusion
- A “universal” user ID table
- Infinite scale 📈
- No need for an auth server
- Online, offline, P2P, or traditional cloud infra ✅✅✅✅
- Crypto keys… crypto keys everywhere!

https://fission.codes
https://talk.fission.codes

🌍🌍🌍 THANK YOU, CODING EARTH 🎉

brooklyn@fission.codes
github.com/expede
@expede