Proving with PODs

How do PODs and GPCs work?

Andrew Twyman, Ahmad Afuni

0xPARC

Section 1: Provable Data

- How do PODs work?
- How to go from POD to circuit

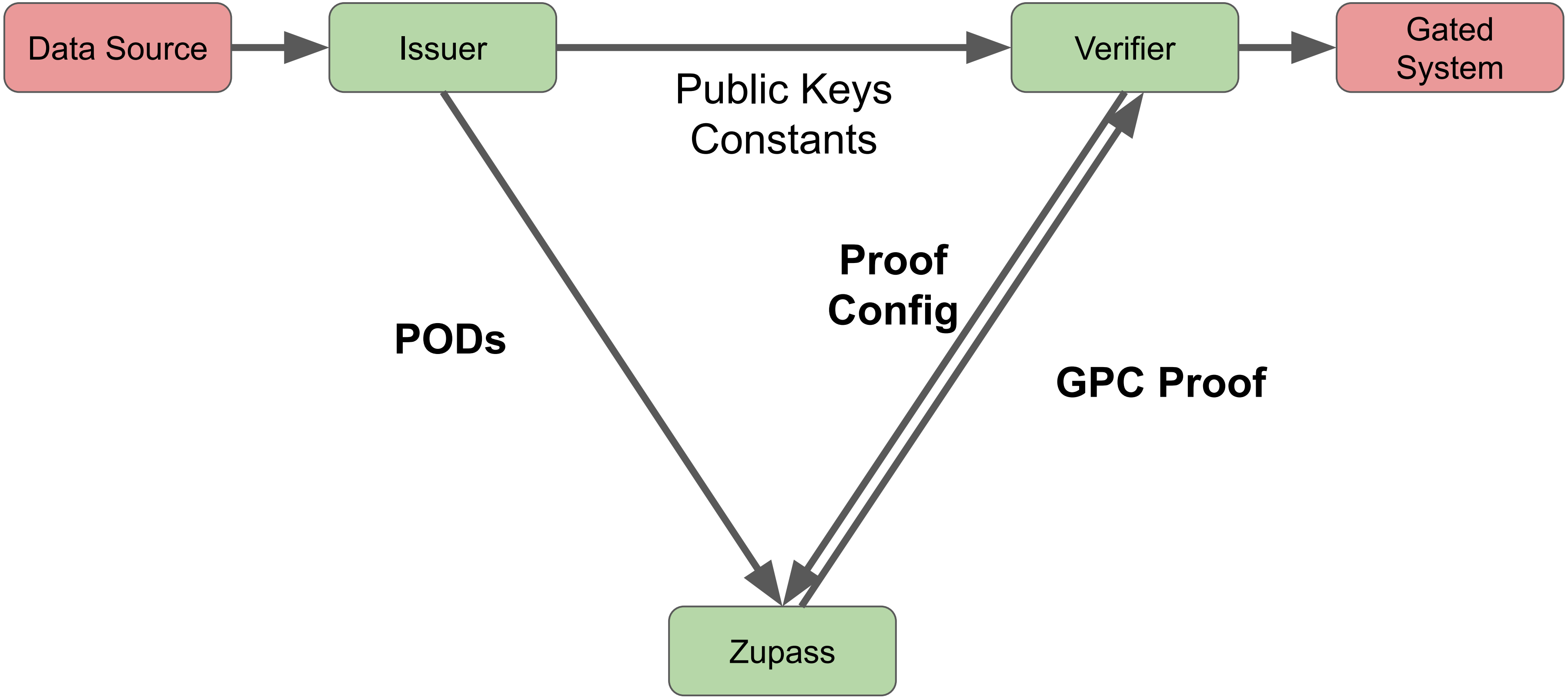
Andrew

Section 2: Modular Circuits

- How do GPCs work?
- How to configure a circuit

Ahmad

POD Ecosystem



What makes it Provable?

- POD = A data format which makes ZK proofs easy
- Merklization
  - Parts are separately verifiable
  - Deterministic and repeatable
- ZK-Friendly primitives
  - Baby-Jubjub prime field math
  - Poseidon hash
  - EdDSA signatures
- Non-ZK-friendly parts can be verified outside the circuit

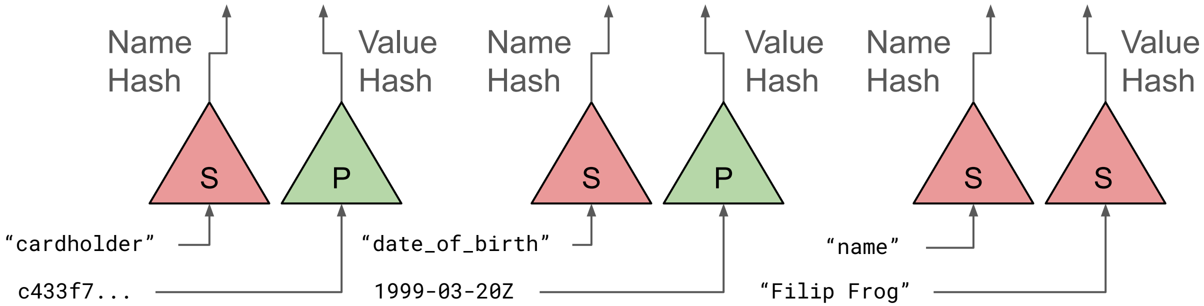
ID Card Entries (slide-sized)

Name	**Value**	**Type Hint**
name	Filip Frog	string
date_of_birth	March 20, 1999	date
cardholder	Semaphore ID	eddsa_pubkey

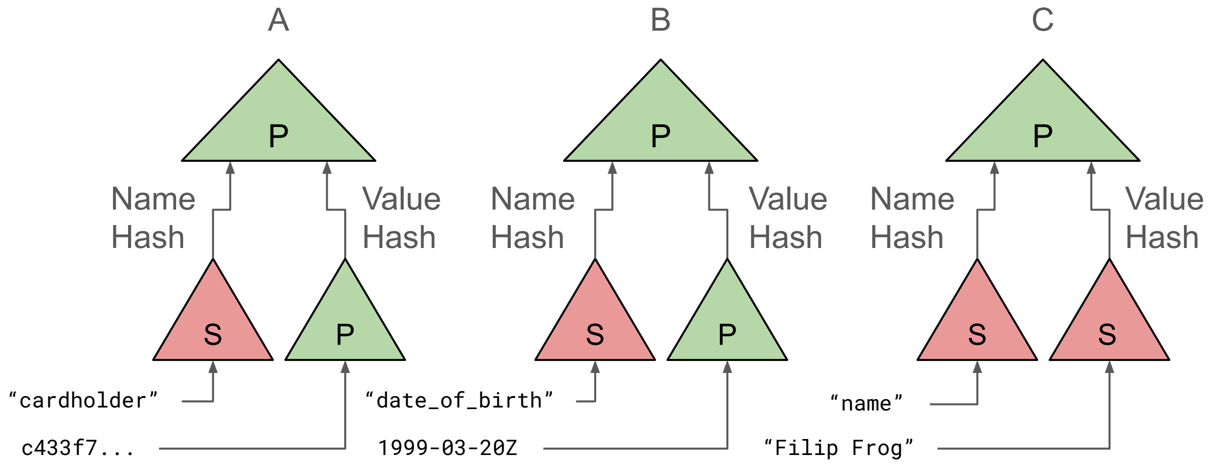
```
{
  "cardholder": {
    "eddsa_pubkey": "c433f7a696b7aa3a5224..."
  },
  "date_of_birth": {
    "date": “1999-03-20T00:00:00.000Z”
  },
  "name": “Filip Frog”,
}
```

- How do we hash and sign this in a ZK-friendly way?

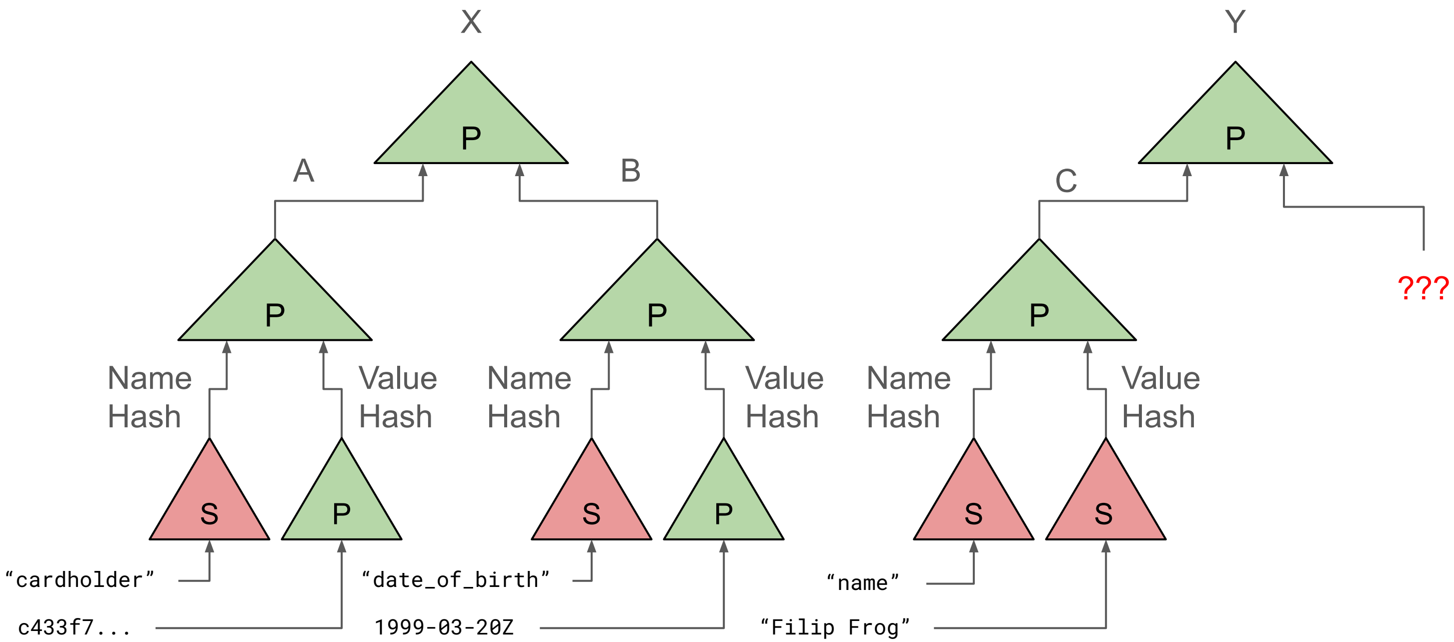
Hashing Entries



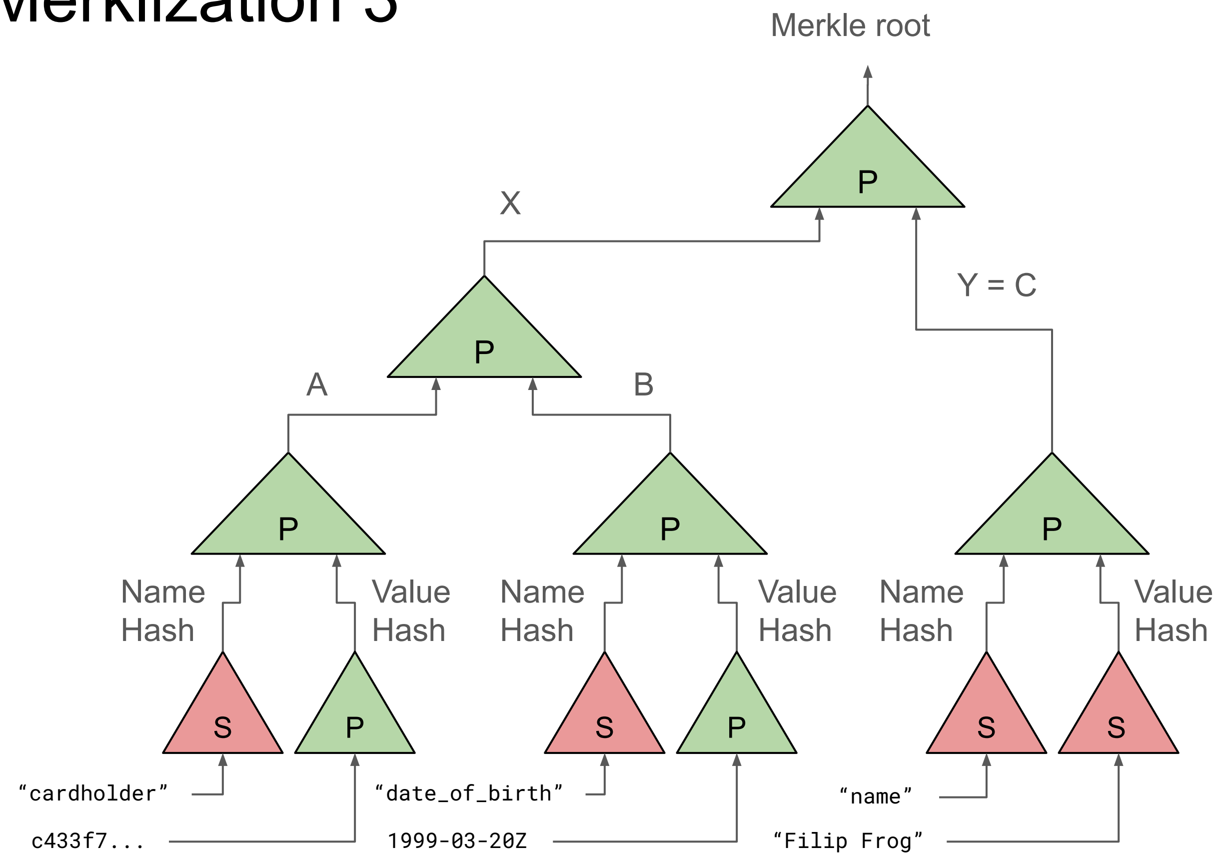
Merklization 1



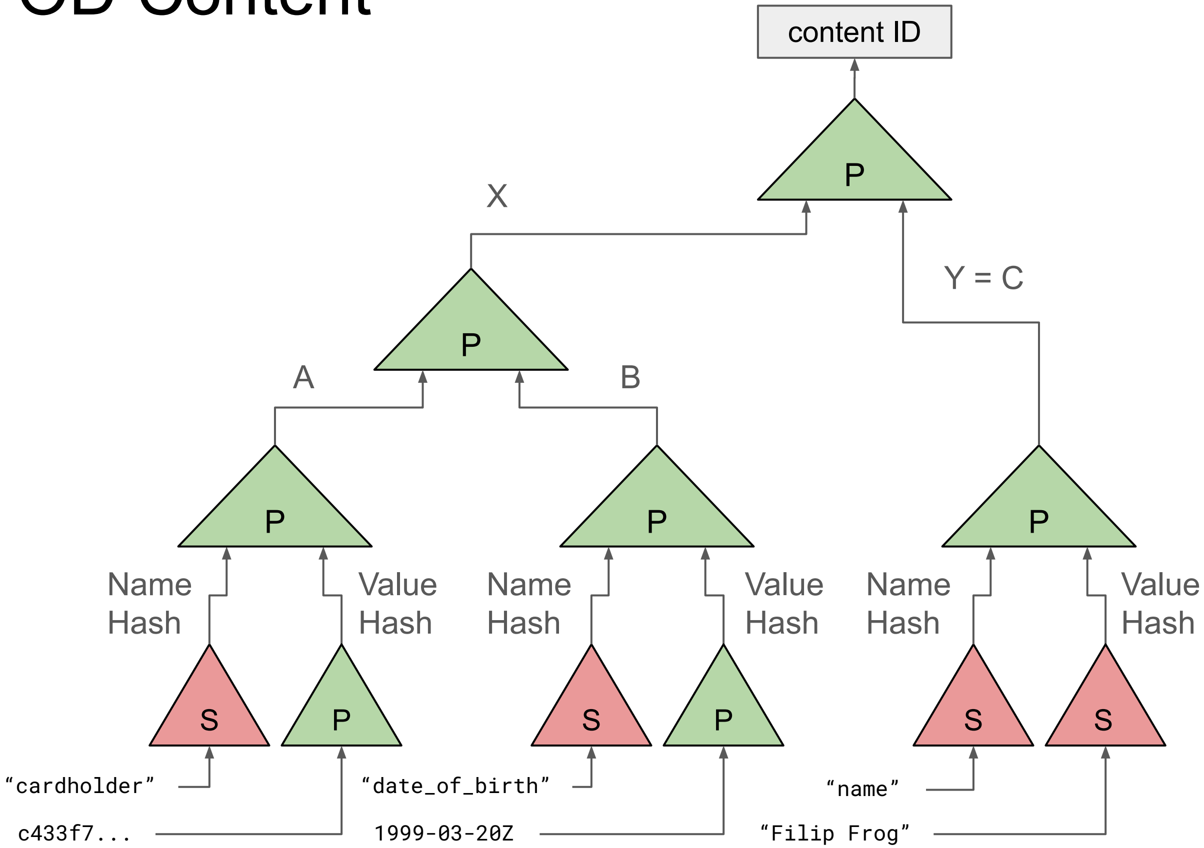
Merklization 2



Merklization 3

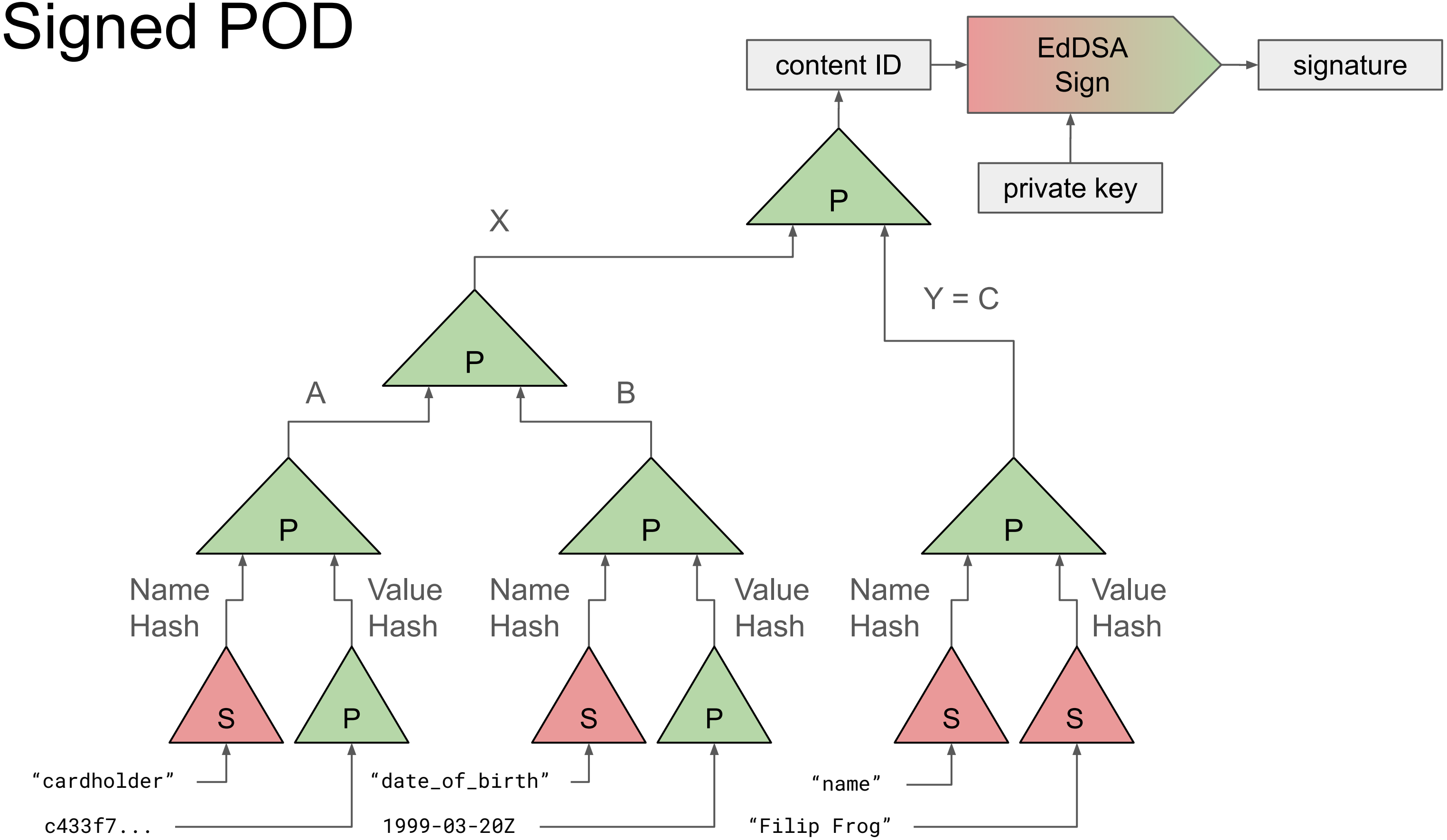


POD Content

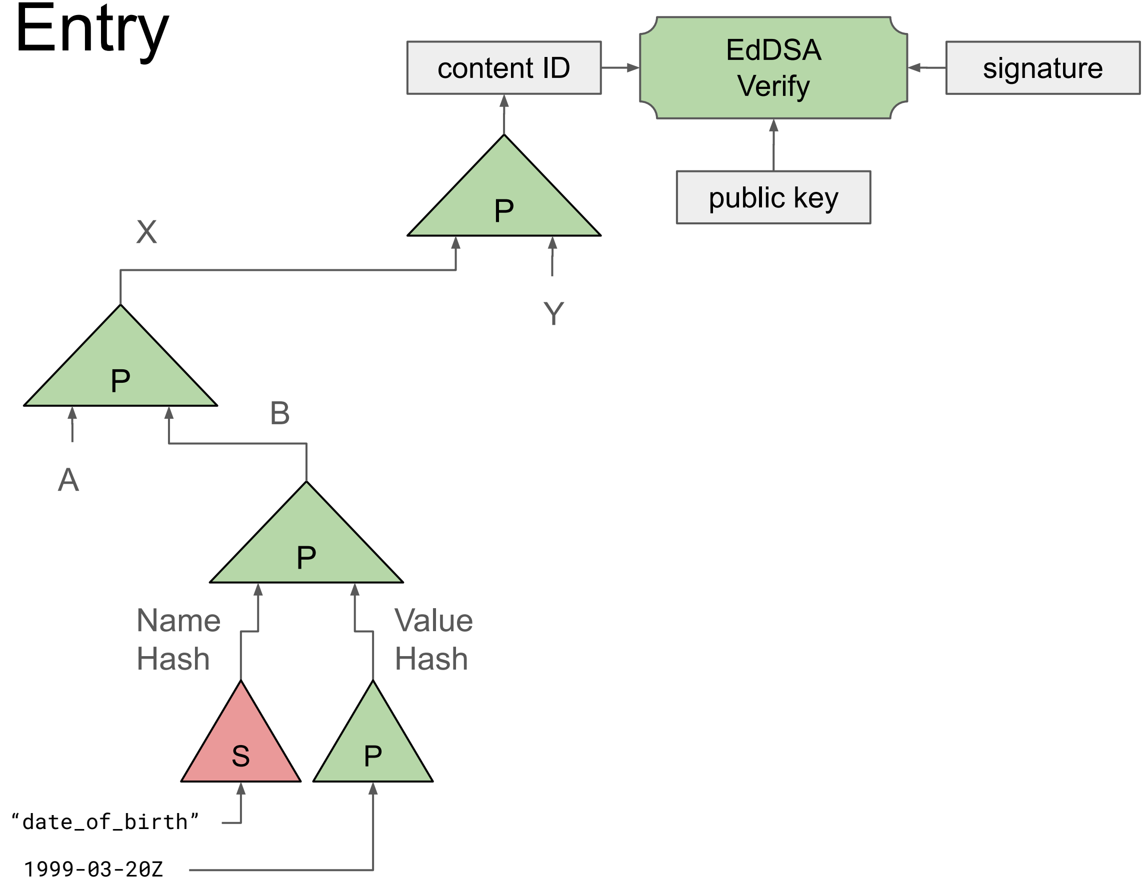


content ID
P
X
Y = C
P
A
B
P
P
P
Name
Hash
Value
Hash
Name
Hash
Value
Hash
Name
Hash
Value
Hash
S
P
S
P
S
S
“cardholder”
c433f7...
“date_of_birth”
1999-03-20Z
“name”
“Filip Frog”

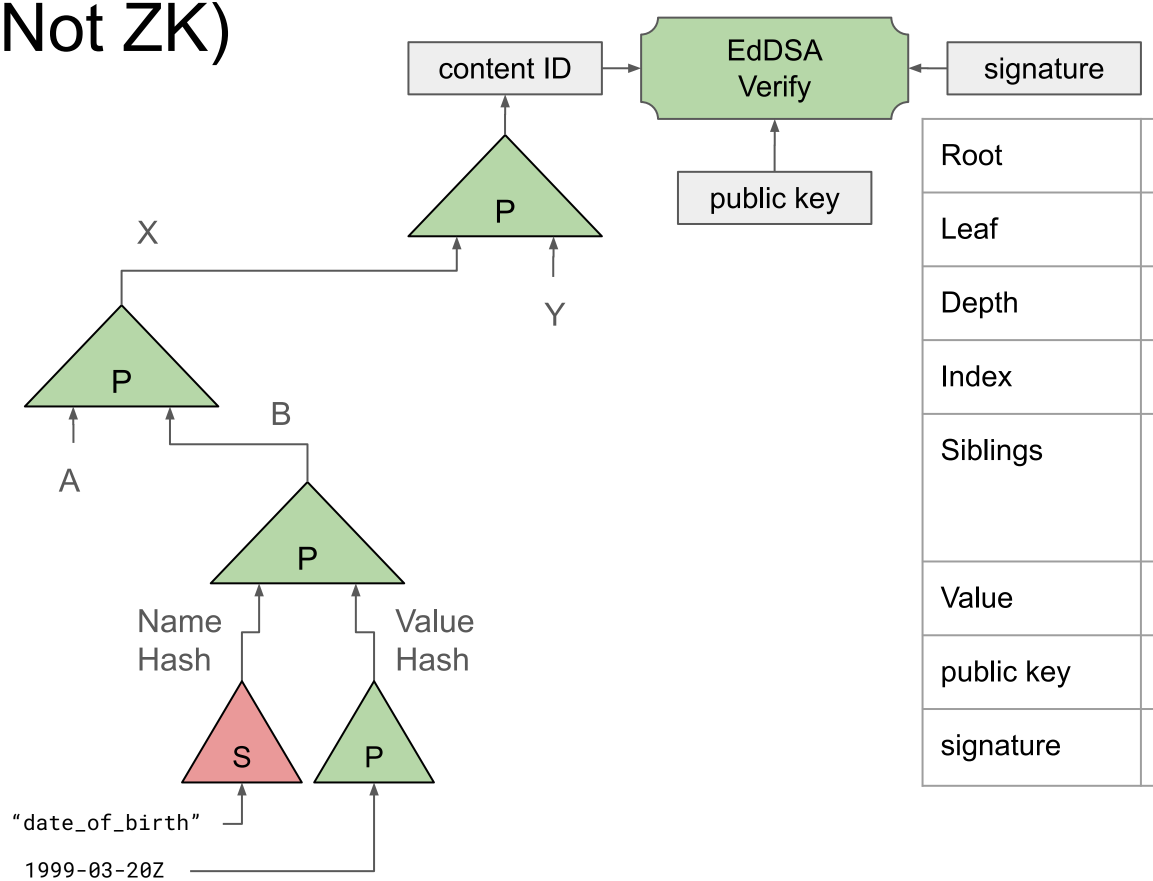
Signed POD


content ID
EdDSA Sign
signature
private key
P
X
Y = C
P
A
B
P
P
P
Name Hash
Value Hash
Name Hash
Value Hash
Name Hash
Value Hash
S
P
S
P
S
S
"cardholder"
c433f7...
"date_of_birth"
1999-03-20Z
"name"
"Filip Frog"

Proving One Entry

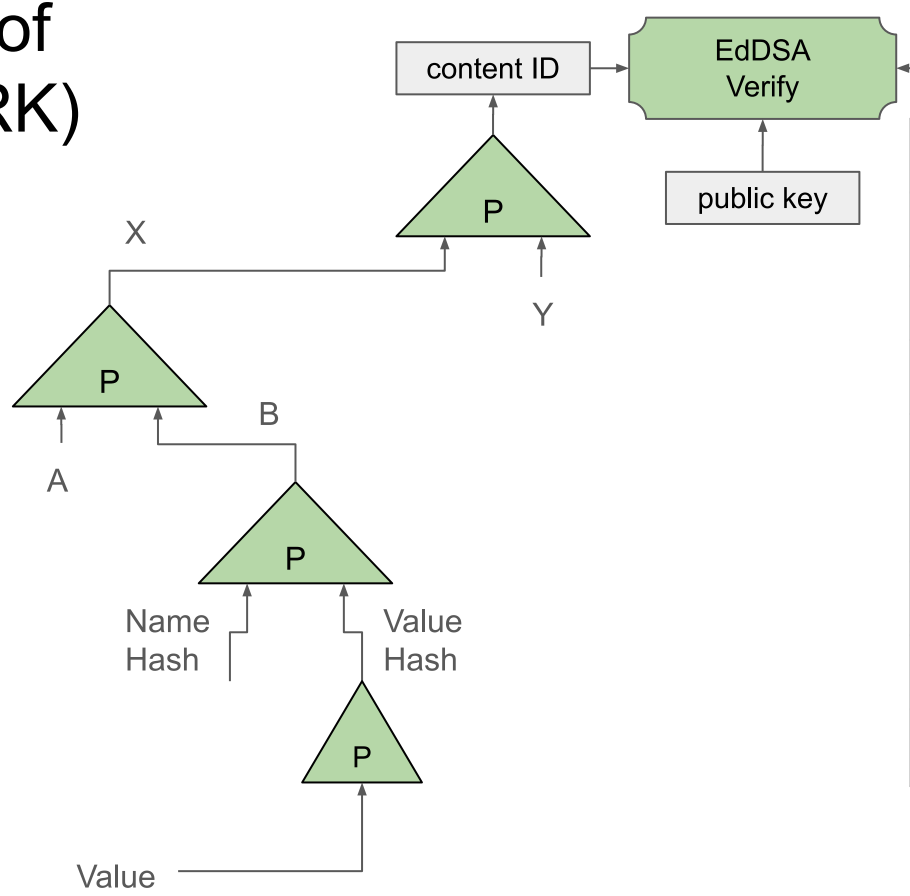


Entry Proof (Not ZK)



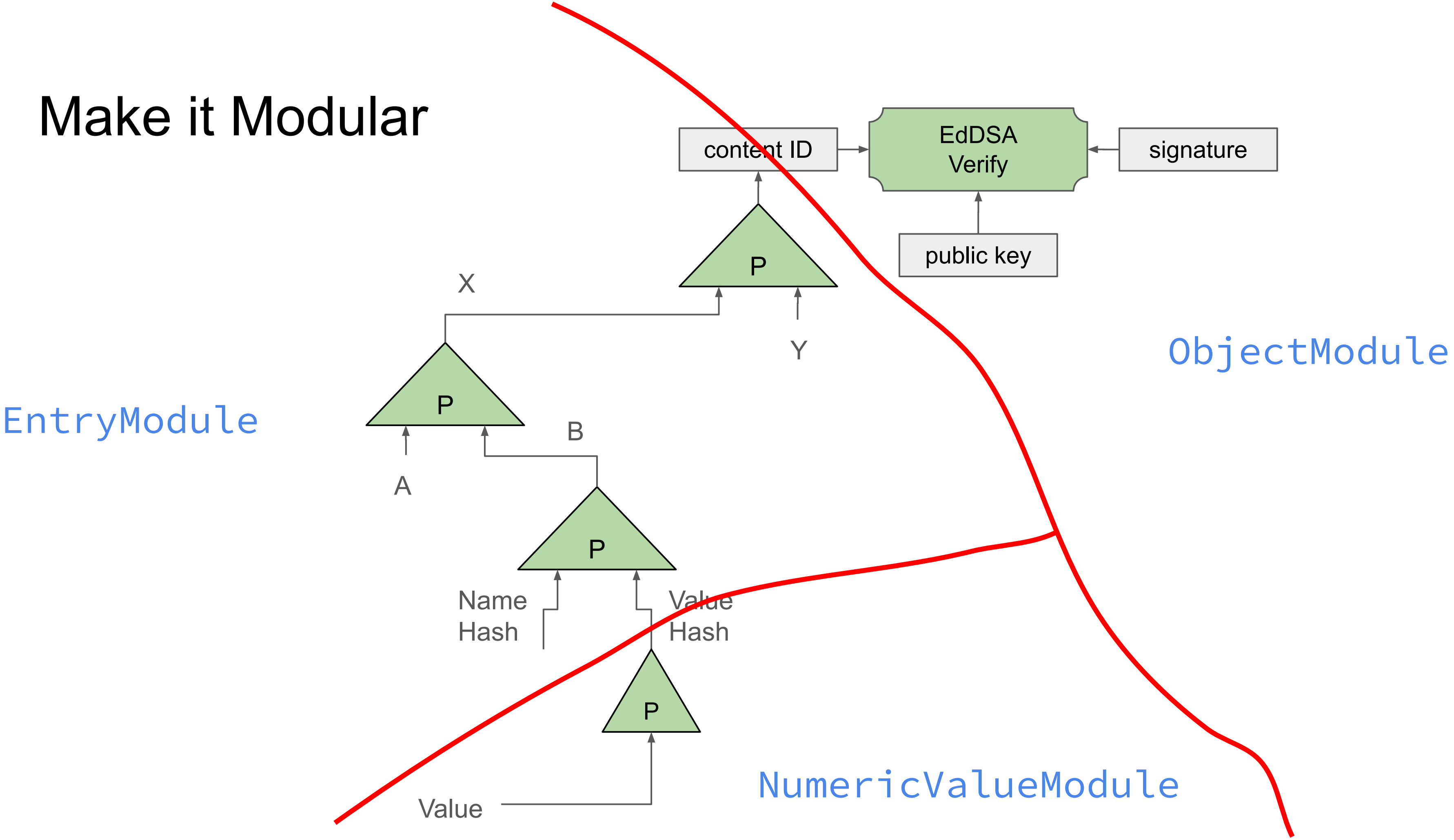
Root	Content ID
Leaf	Name Hash
Depth	3
Index	2 (010)
Siblings	Value Hash, A, Y
Value	1999-03-20Z
public key	NnGAciO/OIz...
signature	kKt/qddVepE...

Entry ZK Proof (in a zkSNARK)



Root	Content ID
Leaf	Name Hash
Depth	3
Index	2 (010)
Siblings	Value Hash, A, Y
Value	1999-03-20Z
public key	NnGAciO/OIz...
signature	kKt/qddVepE...

Make it Modular



POD = Object Data

- Object is made of Entries (name/value pairs)
  - Name is a string, identified by hash
  - Value is a scalar, identified by hash
  - Type hints determine how to hash each value
- Content ID generated from the data (deterministic)
  - Merkle-tree hashing keeps entries separable
- Signature on Content ID makes an attestation

POD = Portable

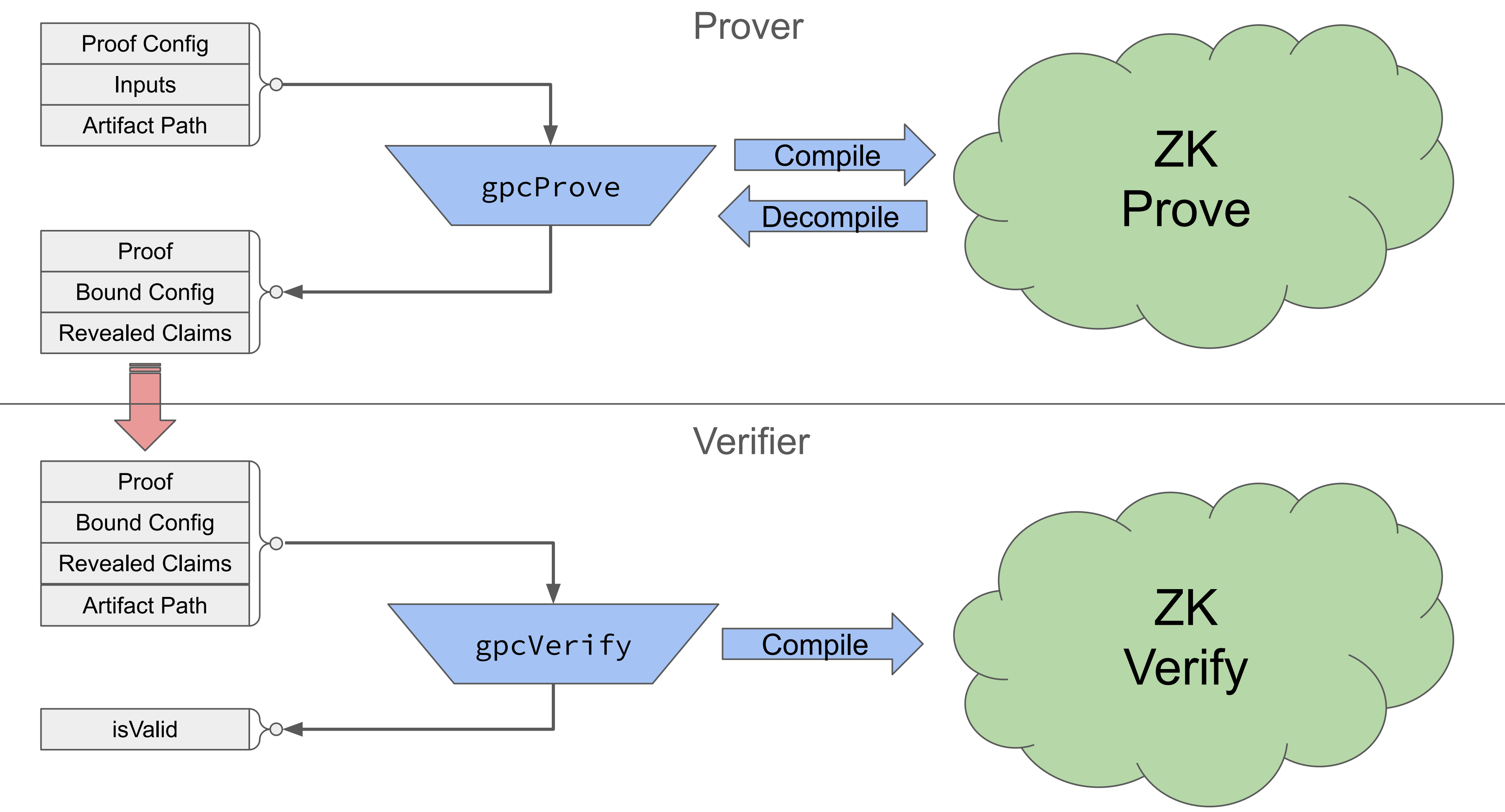
- A data format defined by math
- Not just TypeScript, but any language
- Not just JSON, but any format
- If you can calculate the Content ID and Signature, it's a POD

POD = Provable

- Format allows for efficient ZK circuits
- Signature check on Content ID validates the whole POD
    - Fixed cost per POD, no need for all the data
- Merkle proofs verify entries independently
    - Cost per entry scales with log(POD size)
- zkSNARK can verify all of this in modules
    - Configuration selects what is proven

POD Value Types

Value Type	TS type	Hash	Value/Range	Equals	Ordered
cryptographic	bigint	Poseidon1	0…(p-1)	number	no
int	bigint	Poseidon1	64-bit signed	number	yes
boolean	boolean	Poseidon1	true/false	number	yes
date	Date	Poseidon1	epoch ± ms	number	yes
string	string	SHA	utf8	string	no
bytes	Uint8Array	SHA	bytes	string	no
eddsa_pubkey	string	Poseidon2	EC Point	-	no
null	null	constant	-	-	no



Prover
Proof Config
Inputs
Artifact Path
gpcProve
Compile
Decompile
ZK
Prove
Proof
Bound Config
Revealed Claims
Verifier
Proof
Bound Config
Revealed Claims
Artifact Path
gpcVerify
Compile
ZK
Verify
isValid

GPC = Modular Circuits

- Each circuit contains multiple of modules
  - ObjectModule checks signature
  - EntryModule checks Merkle membership
  - NumericValueModule module checks value hash and bounds checks
  - ListMembershipModule checks if a value is in a list
  - Etc…
- Modules are connected by verified signals (numbers)
  - Content ID
  - Value Hash
  - Numeric Value
- Public inputs determine how the modules are wired together

GPC = Circuit Family

- We pre-generate circuits with different combinations of modules
    - 1 object, 5 entries, 2 numeric values, 1 list of size 20, etc…
- Proving framework picks the smallest (fastest) circuit for each config
- Pre-compiled artifacts for each circuit available for download
    - For proving: key and witness generator are large (12-50MB)
    - For verification: verification key is small (12-50KB)

Proof Inputs

- Configuration
  - What do you want to prove?
- Inputs
  - PODs
  - User identity (private key)
  - Lists of acceptable values
- Artifact Path
  - Where to find binaries for circuits
  - Download URL for browser
  - Local file for server

Proof Outputs

- Cryptographic Proof
  - Cryptographic numbers
- Bound Configuration
  - Same as config input
  - Plus includes a circuit identifier
- Revealed Claims
  - Revealed entry values
  - Signers of PODs
  - Lists of acceptable values from input

```
const { proof, boundConfig, revealedClaims } = await gpcProve(
  proofConfig,
  proofInputs,
  GPC_ARTIFACTS_PATH
);
```

Proof Compiler

- Check configuration and inputs
- Determine circuit requirements (how many modules)
- Pick the smallest circuit which fits
- Hash and Merklize all inputs
- Format inputs and configuration as circuit input signals
- Download artifacts (proving key, witness generator)
- Generate ZK proof
- Decompile circuit output signals into claims

Verify Compiler

- Check configuration and inputs
- Determine circuit requirements (how many modules)
- **Confirm the bound circuit fits**
- Hash and Merklize all inputs
- Format inputs and configuration as circuit input signals
- Download artifacts (**verification key**)
- **Verify ZK proof**

- Repeating steps is required for security (prover can’t be trusted)

Verifier Inputs

- Proof outputs
  - Cryptographic Proof
  - Bound Configuration
  - Revealed Claims
- Artifact Path
  - Same as for proving

Security Checks (in App)

- Check the configuration
  - Is it the one you expect?
- Check Revealed Claims
  - Signed by the right issuer?
  - Are the values acceptable?

```
const isValid = await gpcVerify(
  proof,
  boundConfig,
  revealedClaims,
  GPC_ARTIFACTS_PATH
);
```

What’s inside the circuits?

- Ahmad’s deep dive (live)

Want to learn more?

CLS all day tomorrow
Workshop in the afternoon

Telegram: t.me/zupass
Docs: pod.org

https://dc7.getfrogs.xyz/scanner/FrogProof