

POD2



Outline

- The problem
- POD2 description
- Some technical details



Problem

- Interoperability of programmable cryptography
 - Selective disclosure (reveal some part of signed data)
 - Private property proofs (reveal some property of signed data)
 - Properties of mixed data (I have this government ID & this ethereum key)
- User friendly
 - As a user I want to see the properties, not a zk circuit
 - Can run it on my phone
- Dev friendly
 - Framework for interoperable data
 - No need to write new circuits for combinations of properties
 - Future use-cases without changing the framework



Problem: Examples

1. Make an app that requires proving that you're over 18 according to your government: OK, I can use privado ID.
 2. Make an app that requires proving you have some tokens: OK, I can write a circuit using a storage proof circuit library.
- Make an app that requires (1) or (2), without disclosing which one: uh...
 - Wouldn't it be great if the check result of (1) and (2) wasn't "terminating" and we could continue making new checks out of them?



Problem: Examples

Online game with API to query player stats and facts (with some private info via player API key)

- Building meta-games on top of the game, considering player privacy



POD2: What is it?

- Interoperable cryptographic data format
- Extendable
- With programmable privacy
- Client (smartphone) friendly



SignedPod

- Signed key-value object

```
{  
  _type: Signed  
  _signer: 0xGov  
  birth_year: 1989  
  name: "Alice"  
} + signature
```



MainPod

- Proved private/public statements

```
[  
    exists verified gov_pod: SignedPod  
    gov_pod["_signer"] in WHITELIST  
    gov_pod["birth_year"] < (2025-21)  
    gov_pod["name"] == "Alice"  
] + proof
```



Introduction Pods

- Adapter for foreign cryptographic data to pod format

```
{
```

```
  _type: Ed25519
```

```
  ed25519_pk: 0xpublic_key
```

```
  signed_msg: 0xmsg_hash
```

```
} + proof
```



Custom Predicates

- The MainPod supports a fixed set of native predicates:
 - Equal, NotEqual, LtEq, Lt, Contains, NotContains, Sum, Product, Max, Hash
- Extendable Custom Predicates:
 - Conjunction (AND) of statements
 - Disjunction (OR) of statements



Custom Predicates: Example

```
// src, dst: PubKey, attestation_pod: Pod
eth_dos_friend(src, dst, private: attestation_pod) =
AND(
  ValueOf(?attestation_pod[KEY_TYPE], SIGNATURE)
  Equal(?attestation_pod[KEY_SIGNER], src)
  Equal(?attestation_pod["attestation"], dst)
)
```

```
// src, intermed, dst: PubKey, distance, shorter_distance: Int
eth_dos_distance(src, dst, distance,
  private: shorter_distance, intermed) =
OR(
  AND(
    eth_dos_distance(?src, ?intermed, ?shorter)
    SumOf(?distance, ?shorter_distance, 1)
    eth_friend(?intermed, ?dst)
  )
  AND(
    Equal(?src, ?dst)
    Equal(?distance, 0)
  )
)
```



Recursion

- The MainPod can take other MainPods or intro pods as inputs

[

exists verified pod_a: Ed25519Pod

exists verified pod_b: Sha512Pod

pod_a["signed_msg"] == pod_b["sha512_image"]

pod_b["sha512_preimage"] == "Hello World"

pod_a["ed25519_pk"] in AUTH_LIST

] + proof

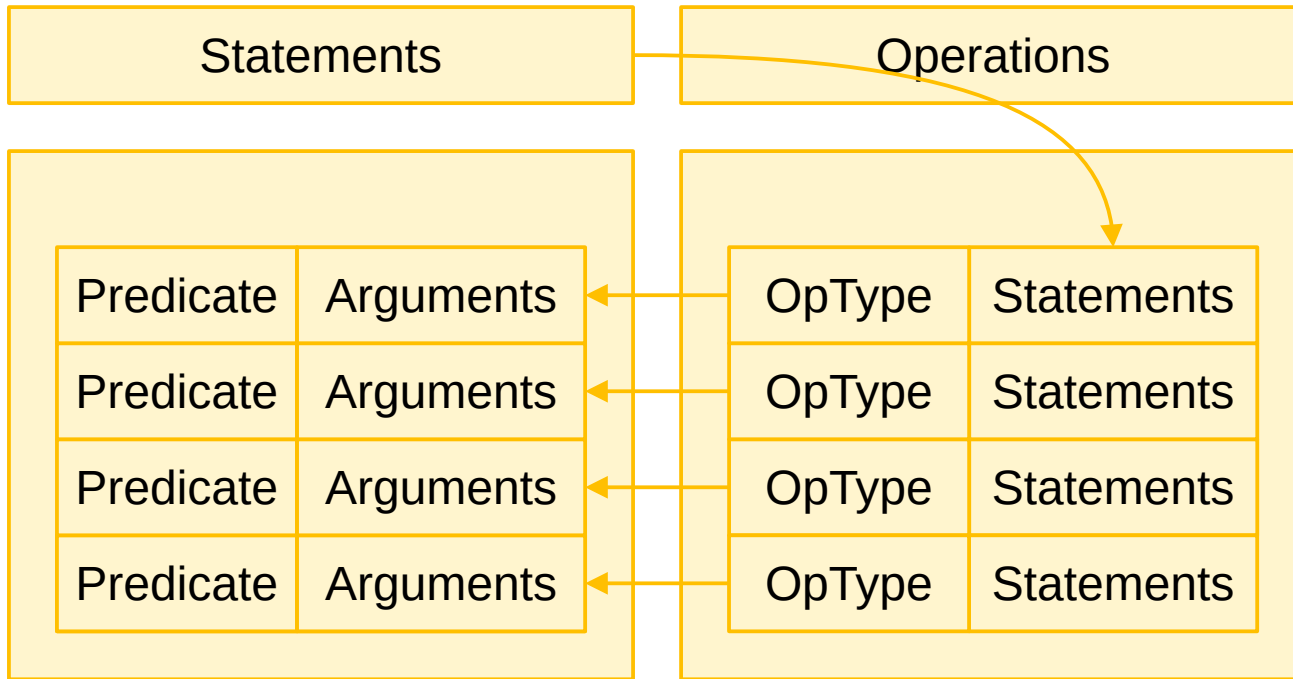


Stack

- Plonky2
 - Very fast recursion
 - Succint proofs
 - Client friendly (CPU & RAM)



Circuit design



Example:

Statements	Operations
0: None	← None
1: Equal(x, 10)	← NewEntry
2: Equal(y, 20)	← NewEntry
3: Lt(x, y)	← LtFromEntries 1 2
4: NotEqual(x, y)	← LtToNotEqual 3