Crypto-Conditions

A Standard for Composable Signatures

Adrian Hope-Bailie

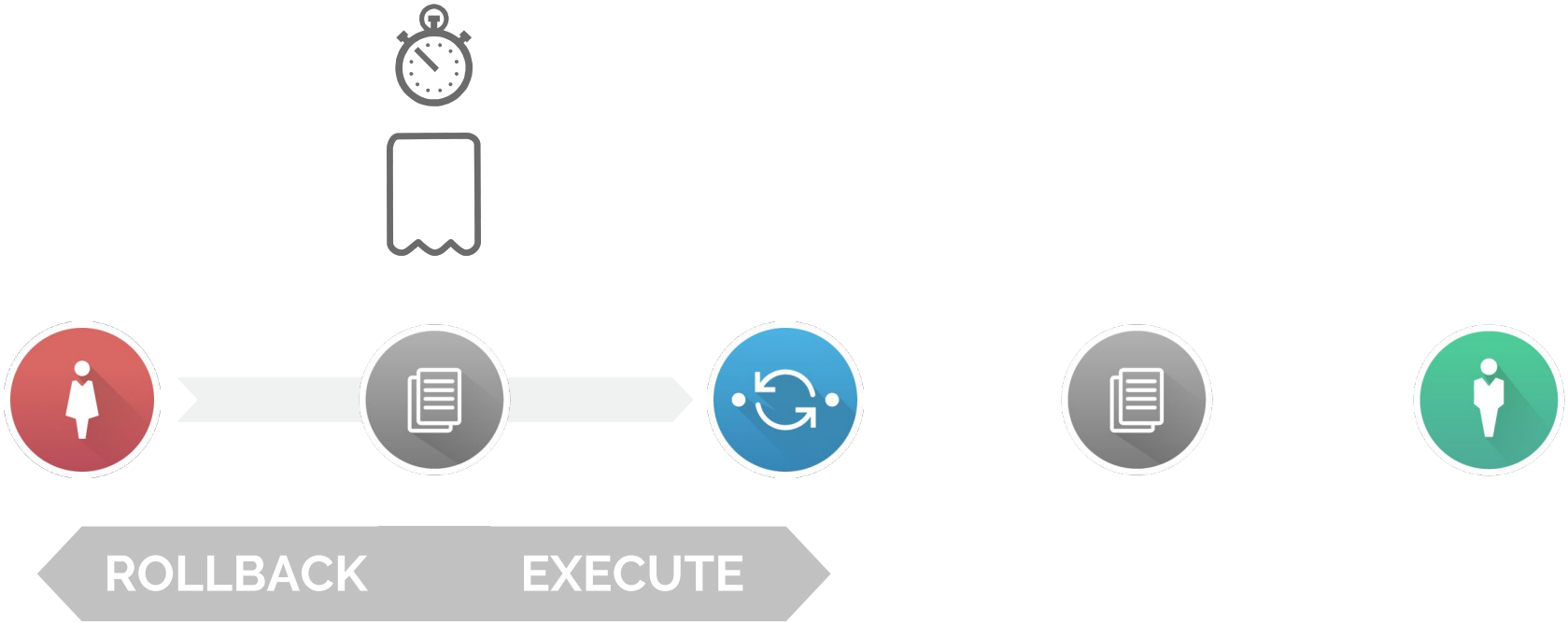
Crypto-Conditions

Standard for composable signatures

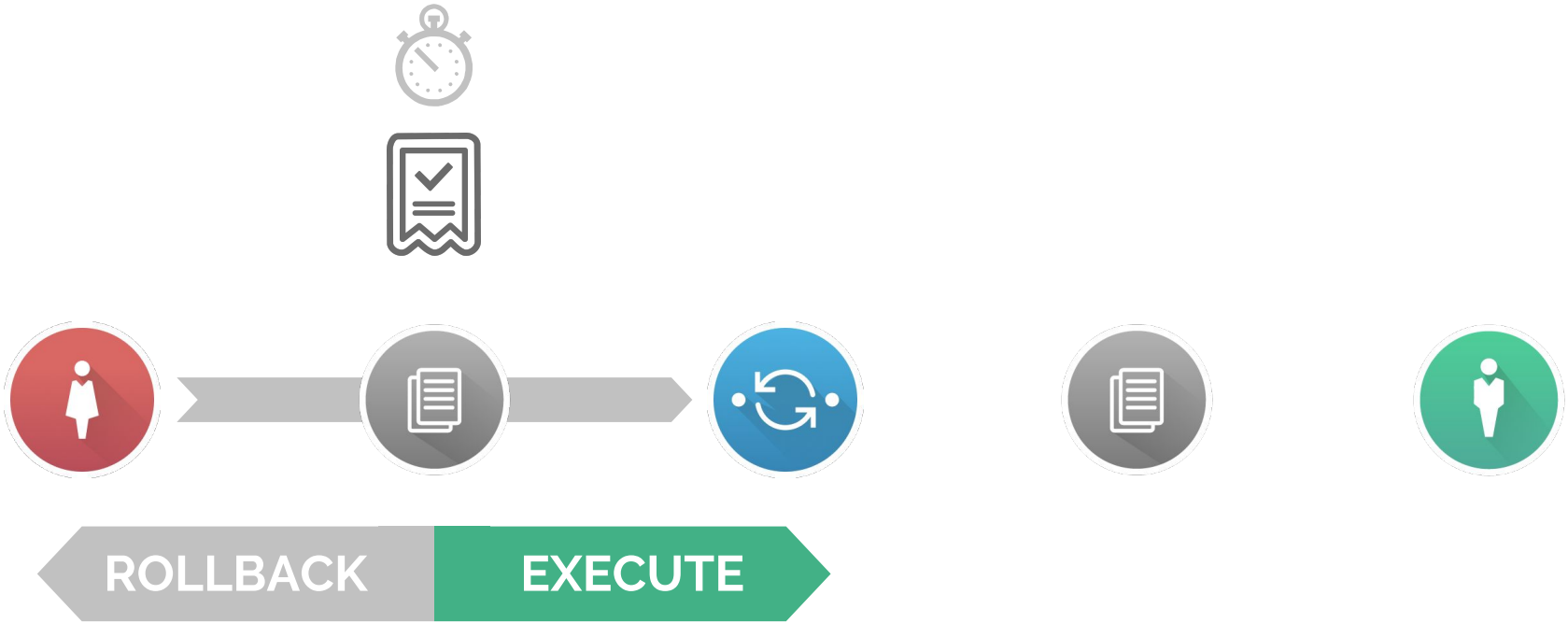
(multi-sig, but more...)

- Minimal
- Composable
- Extensible/Upgradable
- Efficient (using Merkle circuits)

Invented for use in the Interledger Protocol

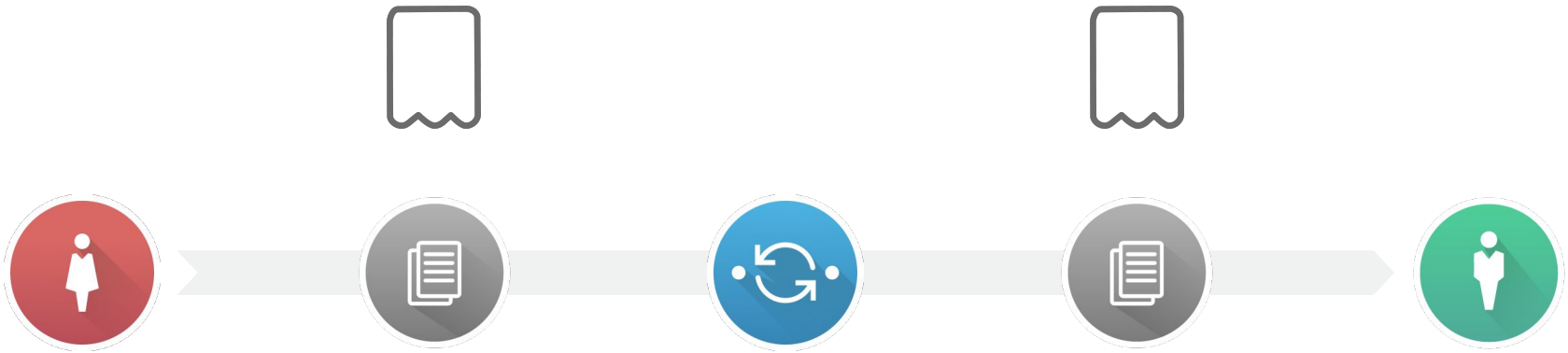


Fulfillment of a Condition Executes Transfer

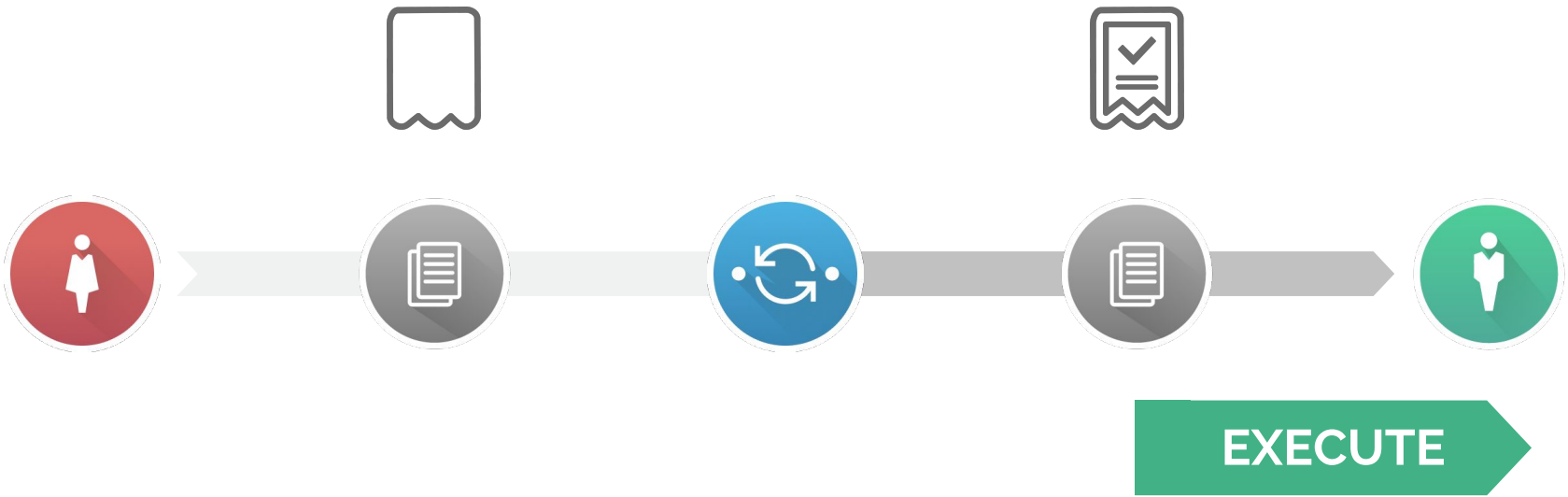


“Smart Contract” Complexity Spectrum

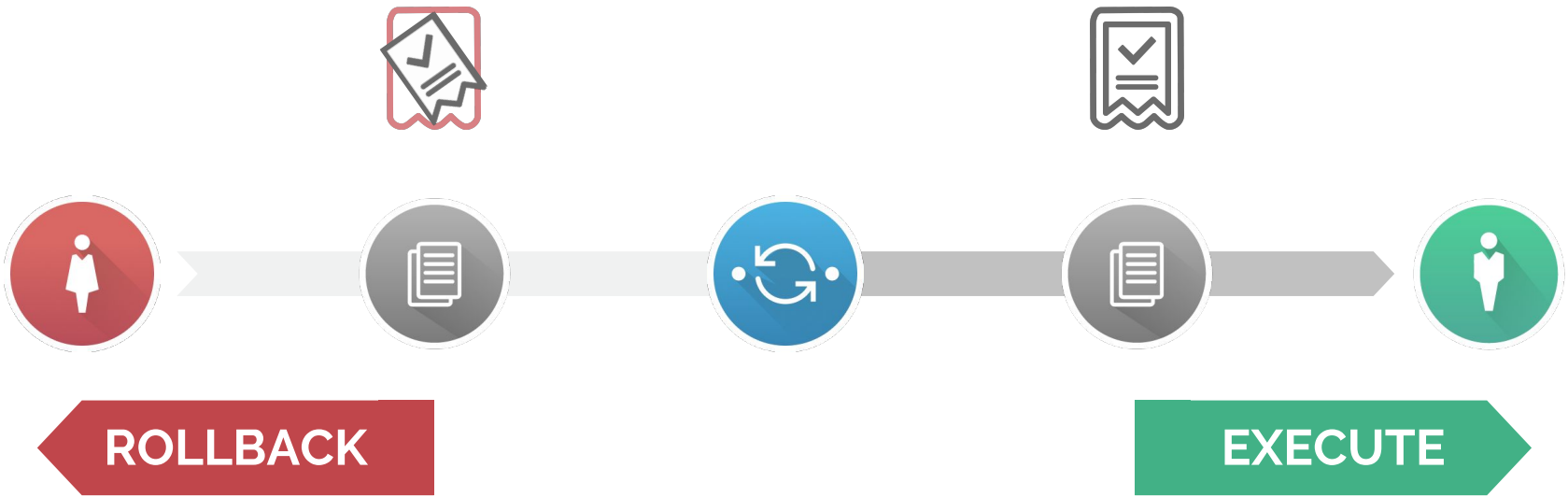
Interledger has some specific feature requirements



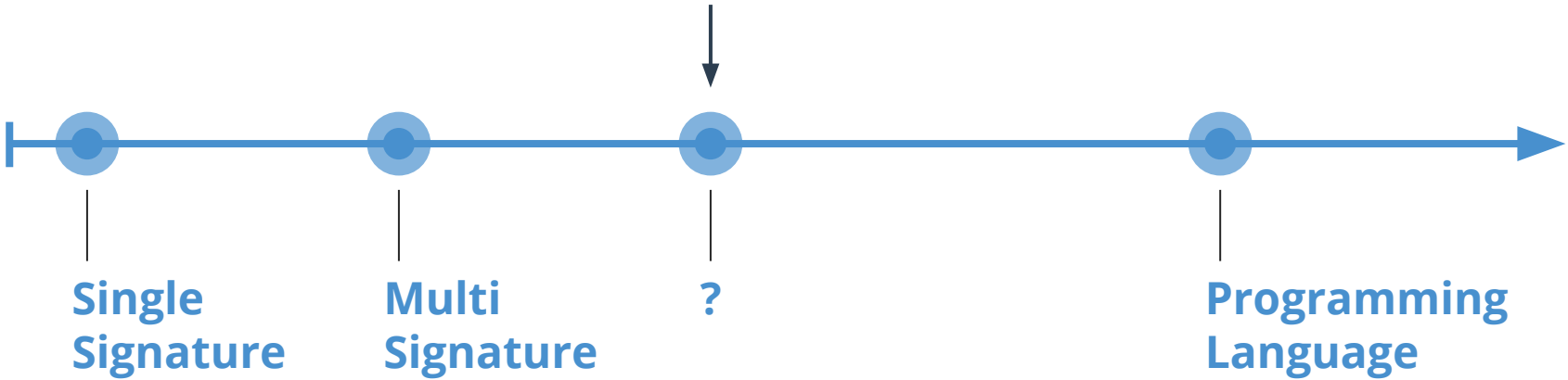
We Care About Minimalism and



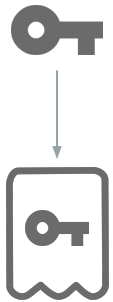
Conditions MUST Be **Bit-Perfect** with **Predictable** Evaluation



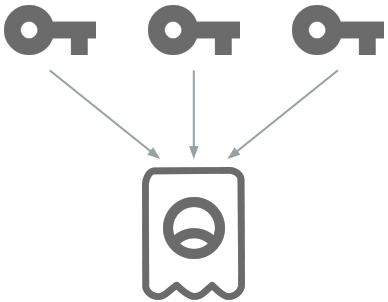
Flexibility vs Simplicity



Condition Types

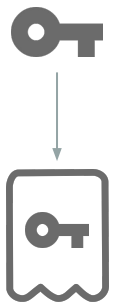


Signatures

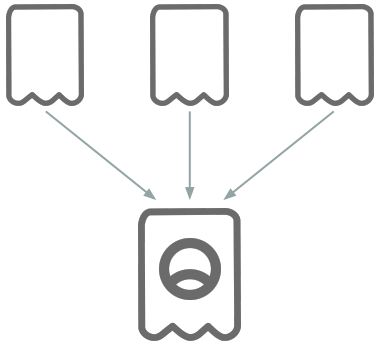


Thresholds

Make Everything a Crypto-Condition for Recursion



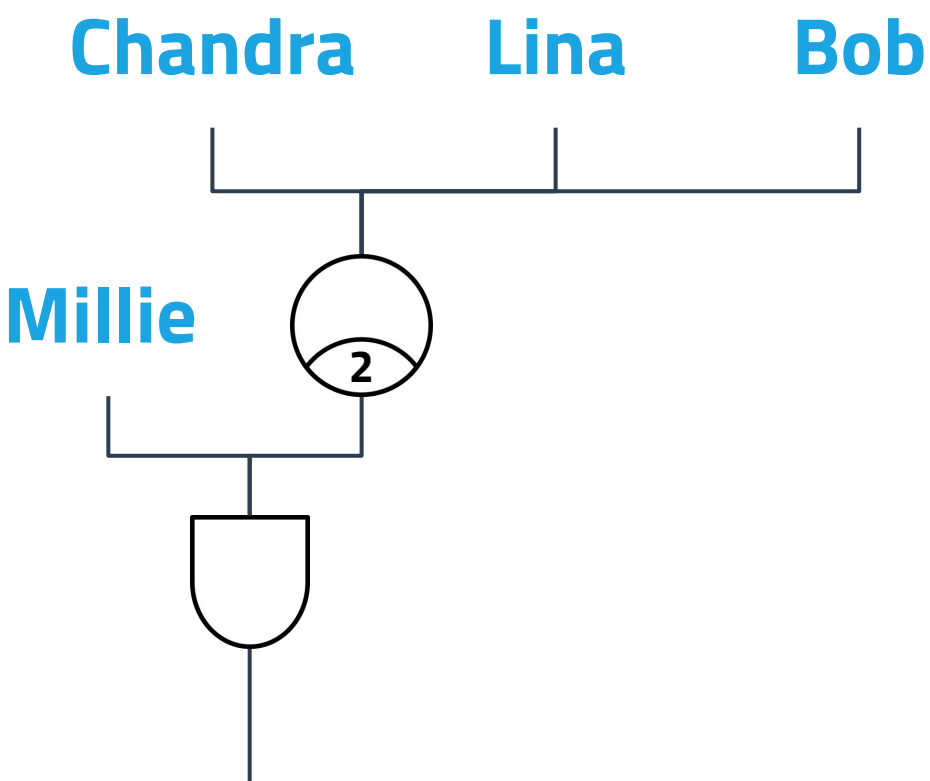
Simple



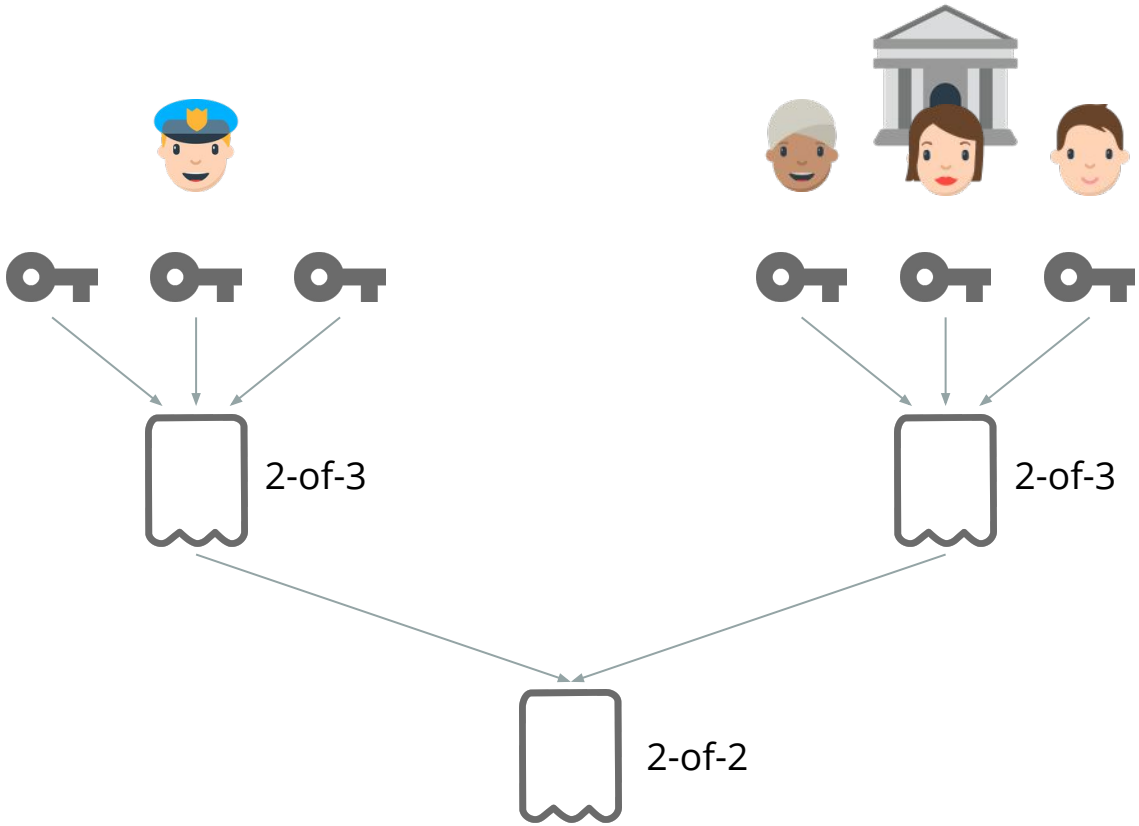
Compound

Boolean Threshold Circuits

Crypto-Conditions as the Logic Gates



Composition Use Case Example



Call for Standardization

May 21st 2016

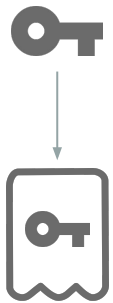
Smart Signatures

Christopher Allen & Shannon Appelcline

Internet Draft

draft-thomas-crypto-conditions-01

Condition — Binary Encoding



```
Condition ::= SEQUENCE {
     type                    ConditionType,
     featureBitmask          OCTET STRING,
     fingerprint             OCTET STRING,
     maxFulfillmentLength    INTEGER (0..MAX)
}


ConditionType ::= INTEGER {
     preimageSha256(0),
     rsaSha256(1),
     prefixSha256(2),
     thresholdSha256(3),
     ed25519(4)
} (0..65535)
```

Fulfillment — Binary Encoding

```
Fulfillment ::= SEQUENCE {
     type                    ConditionType,
     payload                 OCTET STRING
}


Ed25519FulfillmentPayload ::= SEQUENCE {
     publicKey               OCTET STRING (SIZE(32)),
     signature               OCTET STRING (SIZE(64))
}
```

Signature Scheme

Signature Schemes Are a (Cryptographic) Condition

- Condition … like a public key; provided up-front
- Fulfillment … like a signature; cryptographic proof
- Message … actual data to validate against (possibly contextual)

Types of Crypto-Conditions

Simple

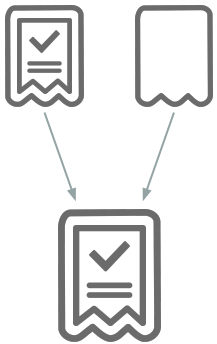
```
preimage   = hash                          ⇒ preimage        ⇒ ()       ⇒ …
rsa        = pubkey                        ⇒ signature       ⇒ message  ⇒ …
ed25519    = pubkey                        ⇒ signature       ⇒ message  ⇒ …
```

Compound

```
prefix     = (subcondition, prefix)        ⇒ subfulfillment  ⇒ message  ⇒ …
threshold  = (subconditions, threshold)    ⇒ subfulfillments ⇒ message  ⇒ …
```

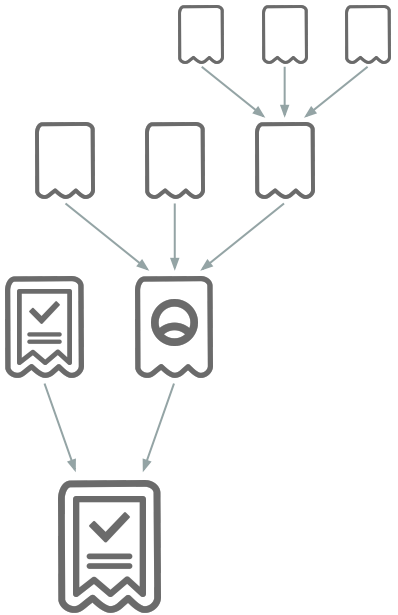
Merkle Circuits

Threshold Fulfillment



Contains conditions for the unfulfilled branches

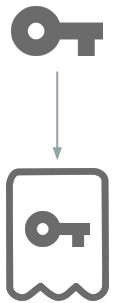
Threshold Fulfillment



Always generates the same, fixed size fingerprint

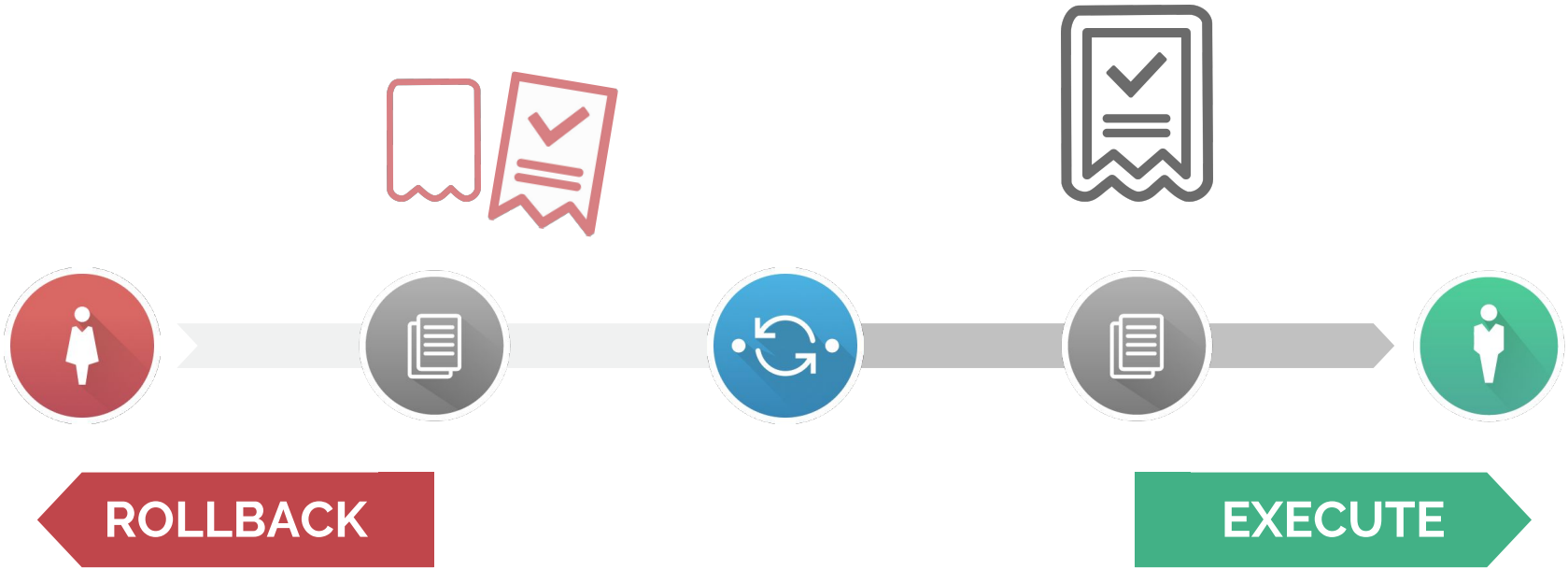
(Max) Fulfillment Length

Condition Contains Fulfillment Length



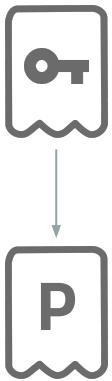
```
Condition ::= SEQUENCE {
     type                    BIT STRING,
     subtypes                BIT STRING,
     fingerprint             OCTET STRING,
     maxFulfillmentLength    INTEGER (0..MAX)
}
```

Fulfillment Size Must Be Supported By Everyone



Prefix Conditions

Prefix Condition



Same key, but pretend the message has an additional prefix.

- Create a more limited scoped key
- Works with any condition type
- Enables fixed message conditions

Related Standards

RFC 5652, RFC 5752

Multiple Signatures in Cryptographic Message Syntax (CMS)

RFC 7515 — Jones, Bradley, Sakimura

JSON Web Signature (JWS)

- Provide basic support for multiple signatures
- **No structured keys**

Other Related Work

- RFC 5752 — Turner and Schaad
- Pay-to-script-hash (P2SH) — Andresen
- Tree Signatures — Wuille
- Merkleized Abstract Syntax Trees (MAST) — Rubin et al
- Smart Signatures — Allen et al
- Deterministic Expressions (DEX) — Todd
- State Channels — Coleman
- Multihash — Benet et al

Implementation Status

Github	Language	Implementer	Status
bigchaindb/cryptoconditions	Python	Ascribe/BigchainDB	Complete
interledger/five-bells-condition	JavaScript	Ripple	Complete
jtremback/crypto-conditions	Go	Althea	Partial
interledger/java-crypto-conditions	Java	Ripple & Everis	Partial

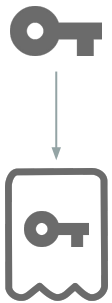
Possible Future Extensions

- Larger hash sizes (512-bits)
- Quantum-secure signatures (SPHINCS, etc.)
- Subdelegation Condition
- Rehash/HMAC Condition
- Homomorphic Hashes
- ...

Open Questions

- Should the bitmask and type be variable or fixed length?
  (currently proposed: variable and drop features bitmask)
- Should we support ECDSA (e.g. P-256), Ed25519 or both?
  (currently defined: Ed25519)
- Is OER encoding the right choice?
  (What about CBOR?)
- Should this become an IETF standards track RFC?

Proposed new Binary Encoding



```
Condition ::= SEQUENCE {
     type                    BIT STRING,
     subtypes                BIT STRING,
     fingerprint             OCTET STRING,
     maxFulfillmentLength    INTEGER (0..MAX)
}

Fulfillment ::= SEQUENCE {
     type                    BIT STRING,
     payload                 OCTET STRING
}
```

Discussion

http://interledger.org

ledger@ietf.org