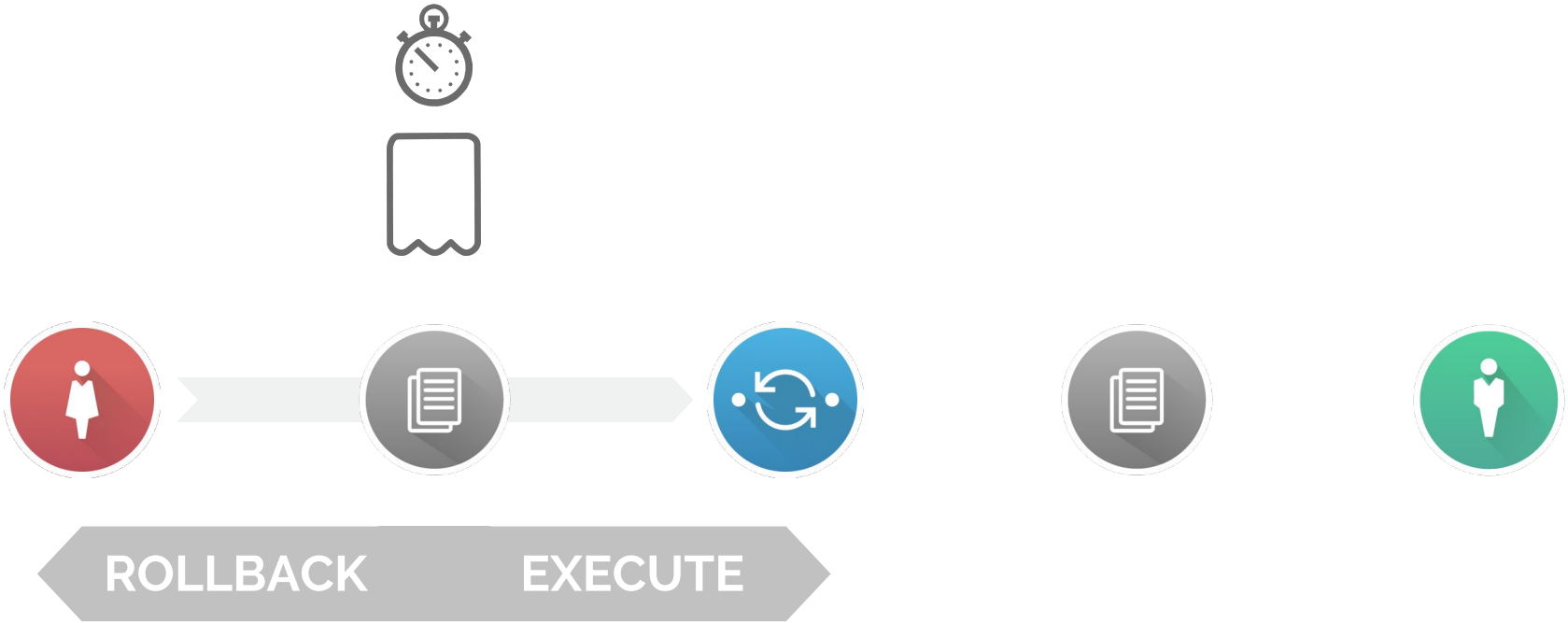
Crypto-Conditions

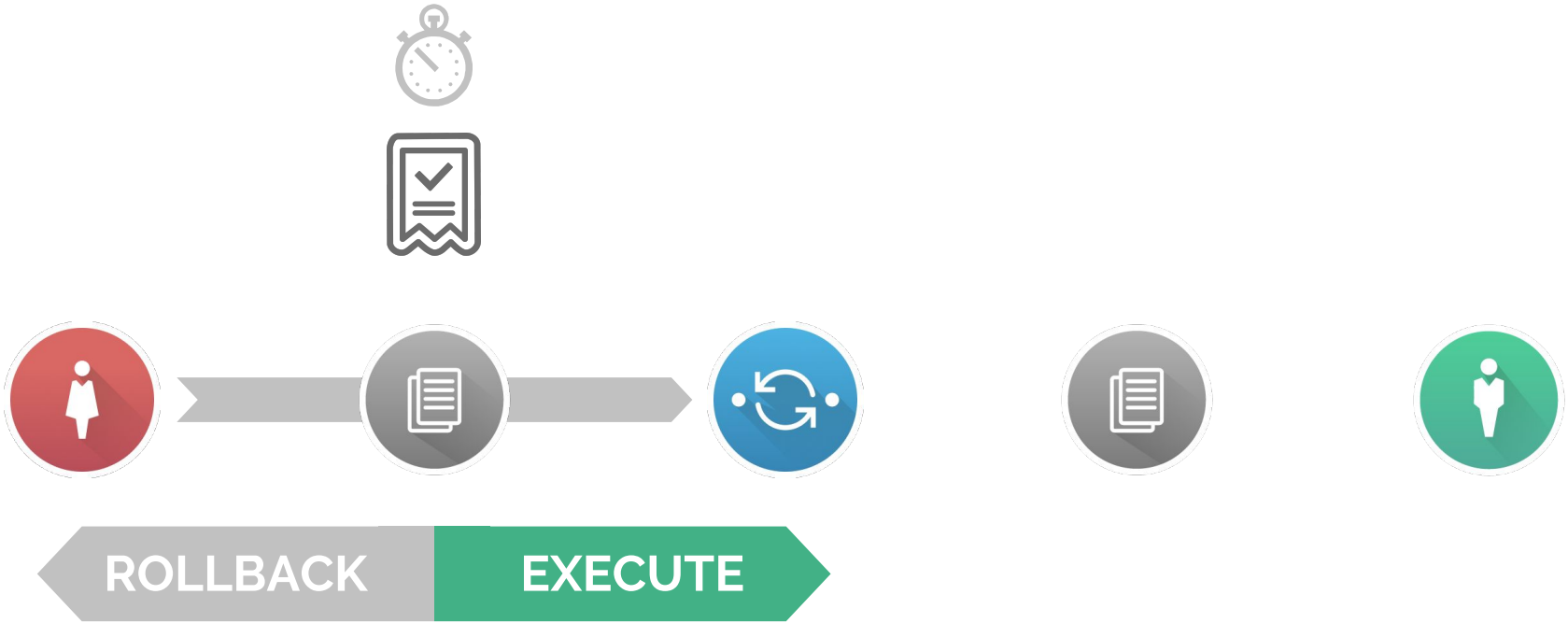
A Standard for Composable Signatures

Stefan Thomas

Holds Are Dependent on Conditions + Expiries



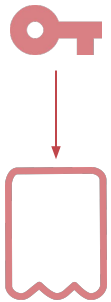
Condition Fulfillment Executes Transfer



Cryptography

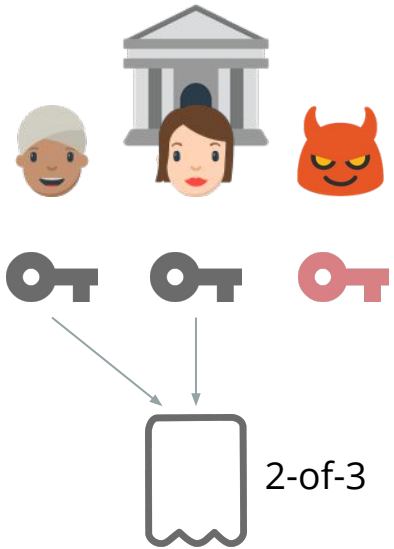
Are Simple Signatures Sufficient?

Are Simple Signatures Sufficient?

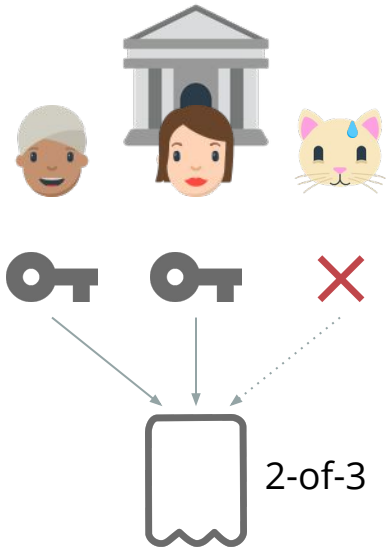


Are Simple Signatures Sufficient?

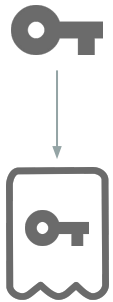
How Multi-Signatures Can Help



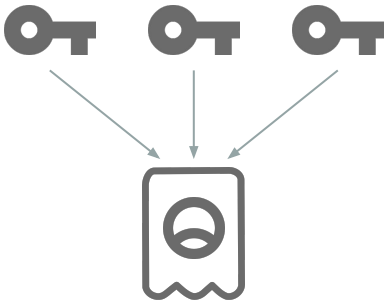
How Multi-Signatures Can Help



Condition Types

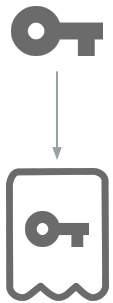


Signatures

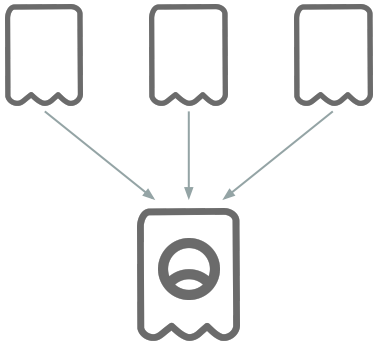


Thresholds

Condition Types

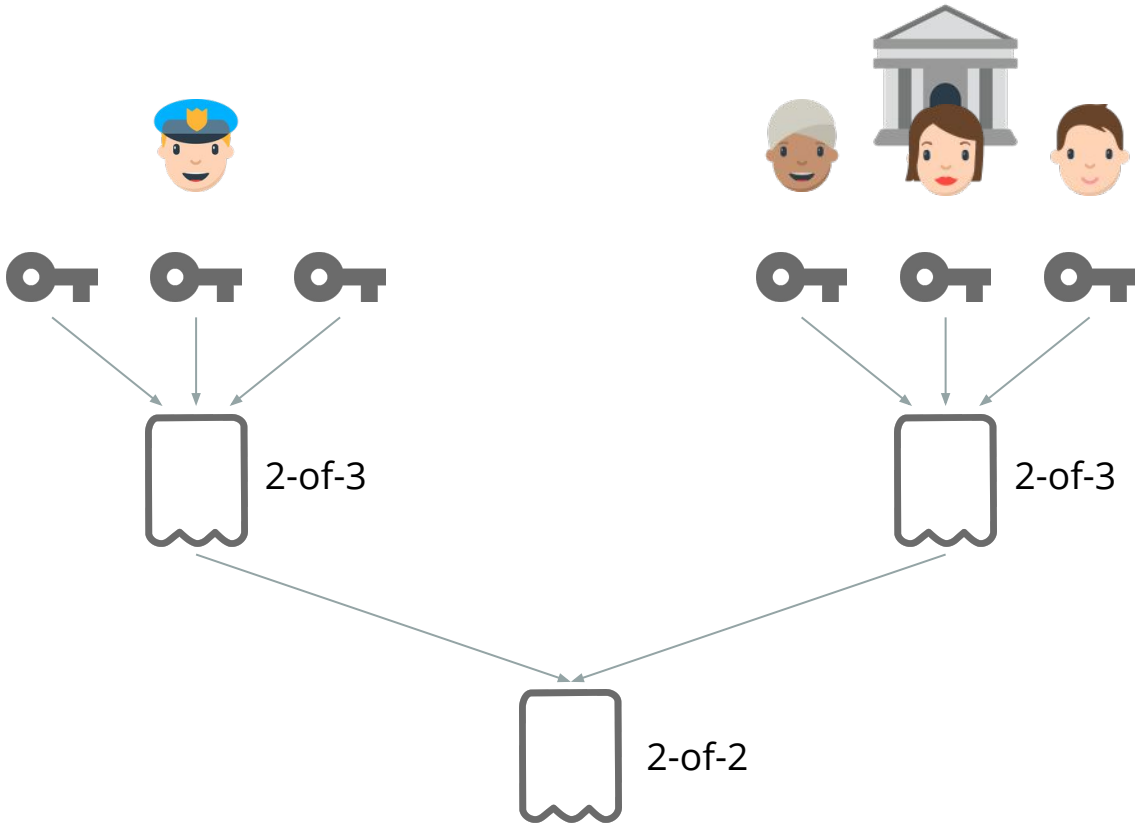


Signatures



Thresholds

Composition Use Case Example



Prior Art: Bitcoin Scripts

```
2 <K1> <K2> <K3> 3 CHECKMULTISIGVERIFY
```

- Forth-like language
- Many opcodes disabled
- Primary use: m-of-n multi-signature

Related Standards

RFC 5652, RFC 5752

Multiple Signatures in Cryptographic Message Syntax (CMS)

RFC 7515 — Jones, Bradley, Sakimura

JSON Web Signature (JWS)

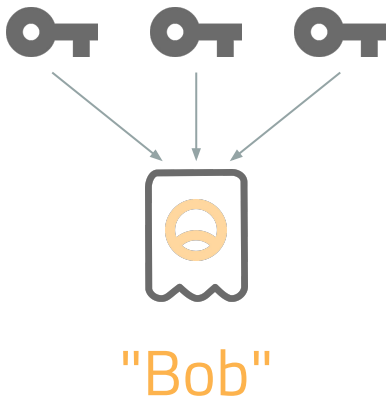
- Minimal support for multiple signatures
- **No structured keys**

Structured Keys

Suppose Alice receives a key from Bob...

Bob

Structured Keys



Suppose Alice receives a key from Foo Bank.

The **structure** of Foo Bank's key shouldn't matter to her.

(Within reason — we'll get to that later.)

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

Call for Standardization

May 21st 2016

Smart Signatures

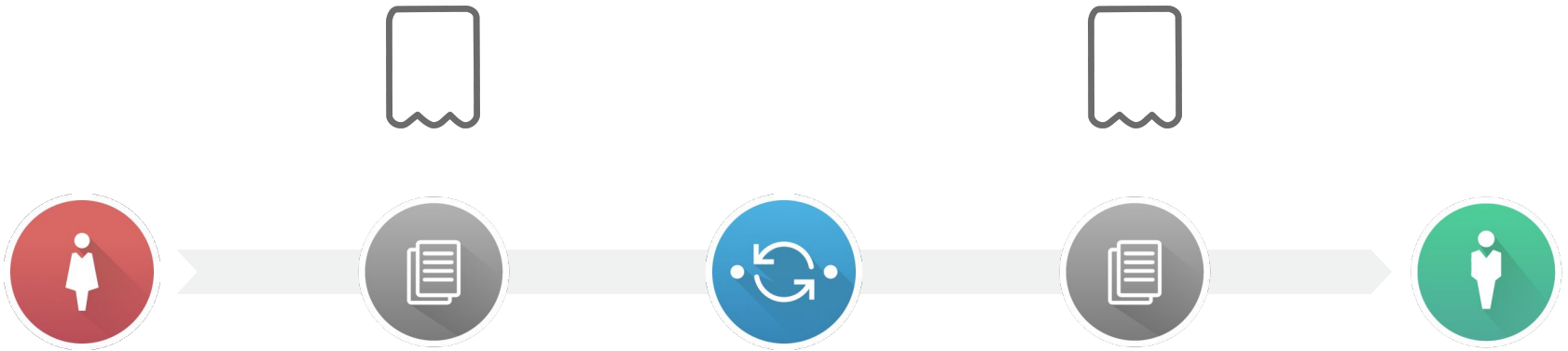
Christopher Allen & Shannon Appelcline

Complexity Spectrum

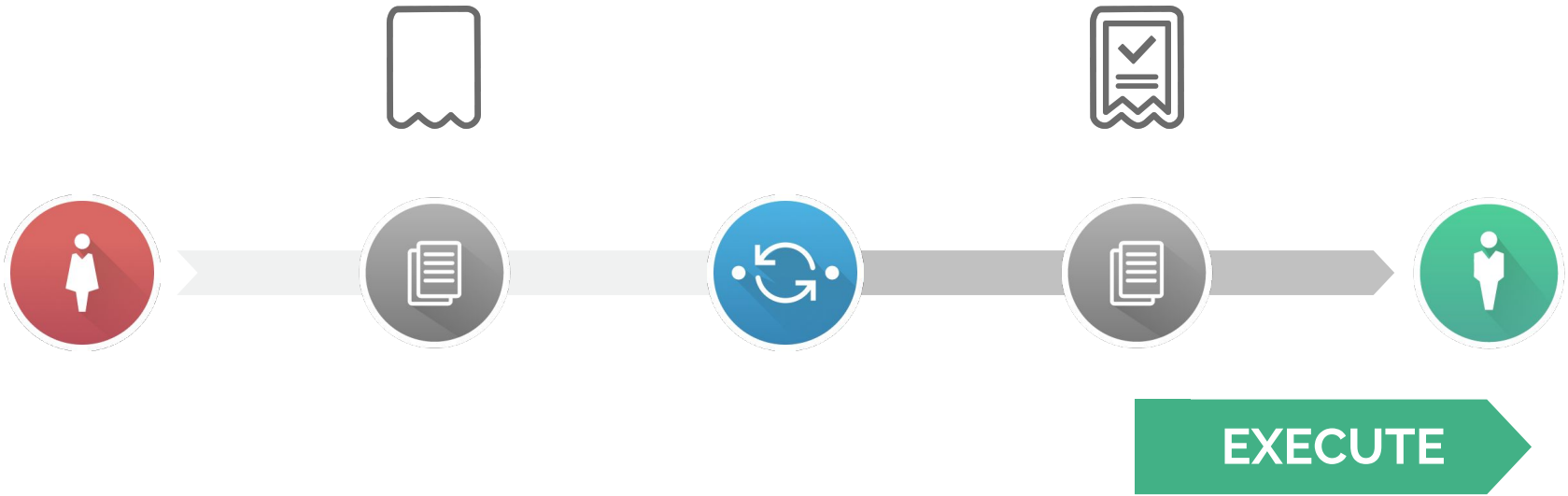
Complexity Spectrum

Complexity Spectrum

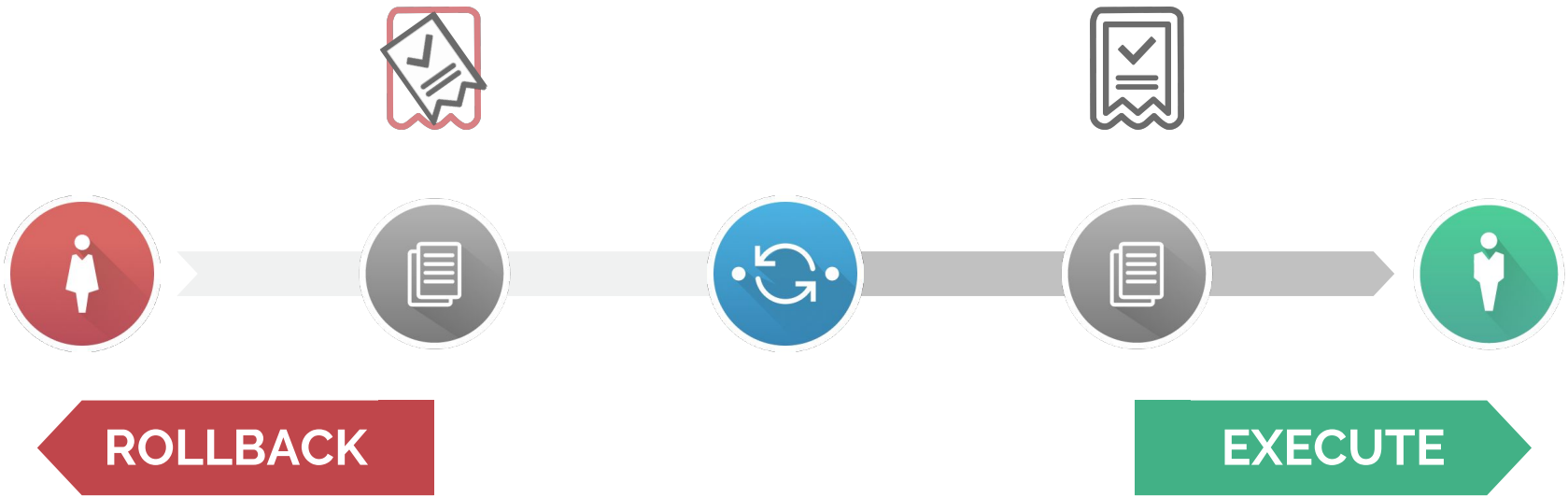
Why Do We Care About Minimalism?



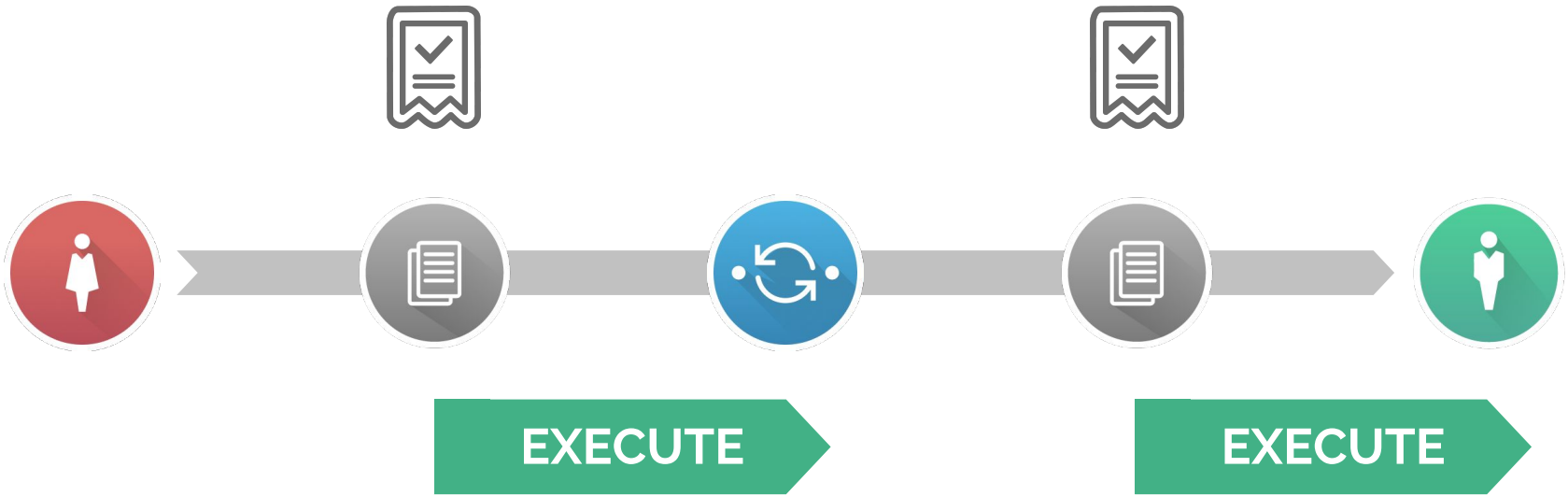
Why Do We Care About Minimalism?



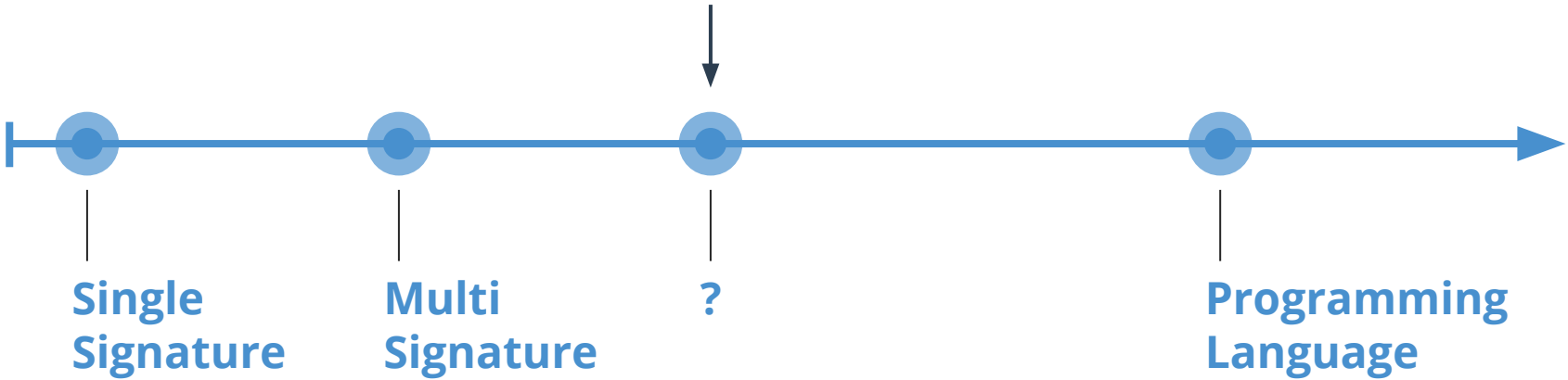
Conditions MUST Be **Bit-Perfect**



Condition Implementations MUST Be **Bit-Perfect**



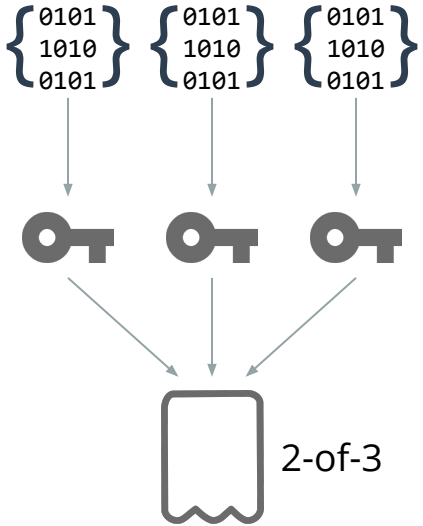
Flexibility vs Simplicity



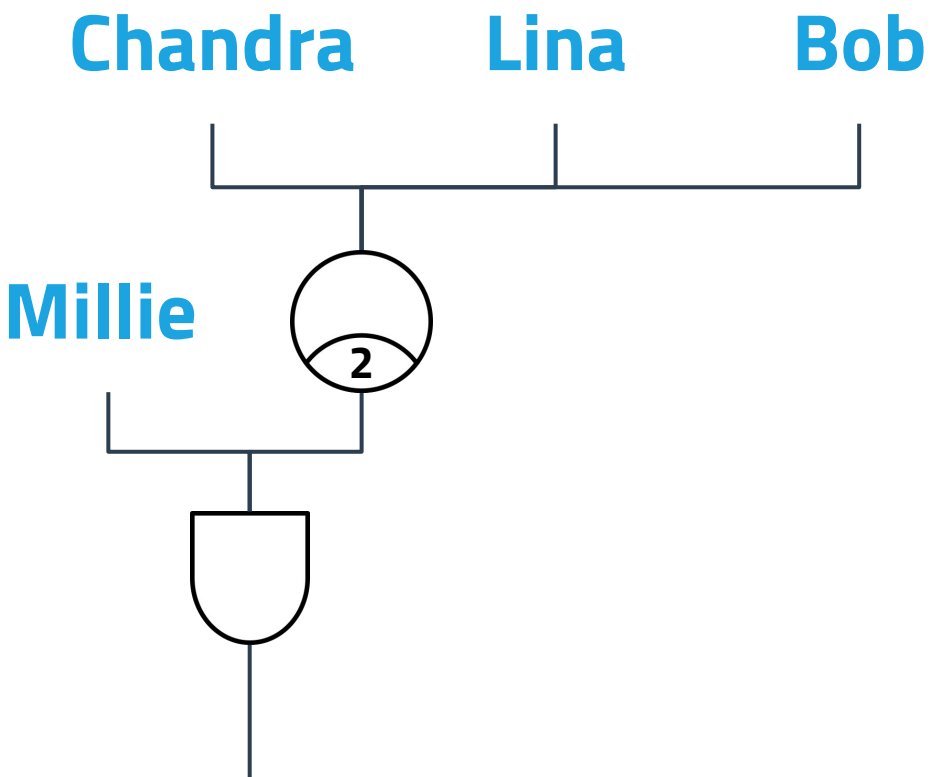
Delegation

Smart Oracle

Smart Oracle



Boolean Threshold Circuits



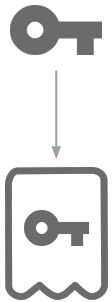
Proposal

draft-thomas-crypto-conditions-00

Implementation Status

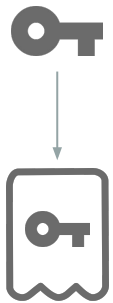
Github	Language	Implementer	Status
bigchaindb/cryptoconditions	Python	Ascribe/BigchainDB	Complete
interledger/five-bells-condition	JavaScript	Ripple	Complete
jtremback/crypto-conditions	Go	Althea	Partial

Condition — Printable Encoding



```
cc:4:20:RCmTBlAEqh5MSPTdAVgZTAI0m8xmTNluQA6iaZGKjVE:96
```

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 — Printable Encoding

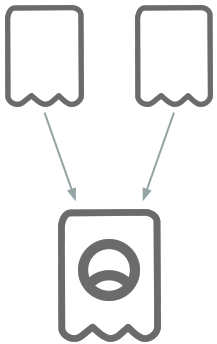
```
cf:4:
RCmTBlAEqh5MSPTdAVgZTAI0m8xmTNluQA6iaZGKjVGfTbzglso5
Uo3i2O2WVP6abH1dz5k0H5DLylizTeL5UC0VSptUN4VCkhtbwx3B
00pCeWNy1H78rq6OTXzok-EH
```

Fulfillment — Binary Encoding

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

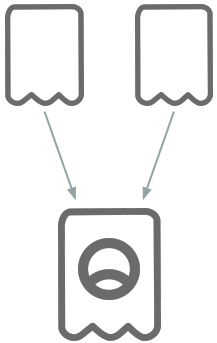
```
Ed25519FulfillmentPayload ::= SEQUENCE {
  publicKey OCTET STRING (SIZE(32)),
  signature OCTET STRING (SIZE(64))
}
```

Threshold Condition



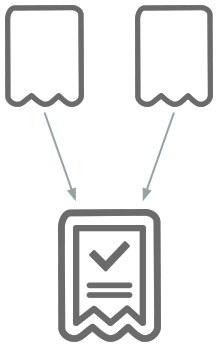
```
ThresholdSha256FingerprintContents ::= SEQUENCE {
  threshold INTEGER (0..4294967295),
  subconditions SEQUENCE OF ThresholdSubcondition
}

ThresholdSubcondition ::= SEQUENCE {
  weight INTEGER (0..4294967295),
  condition Condition
}
```

Threshold Condition



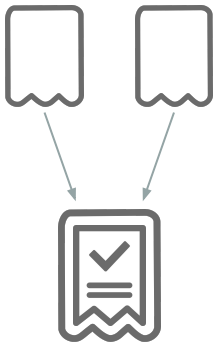
```
cc:2:2b:d3O4epRCo_3rj17Bf3v8hp5ig7vq84ivPok07T9Rdl0:146
```

Threshold Fulfillment



```
ThresholdSha256FulfillmentPayload ::= SEQUENCE {
  threshold INTEGER (0..4294967295),
  subfulfillments SEQUENCE OF ThresholdSubfulfillment
}

ThresholdSubfulfillment ::= SEQUENCE {
  weight INTEGER (0..4294967295),
  condition Condition,
  fulfillment Fulfillment
}
```

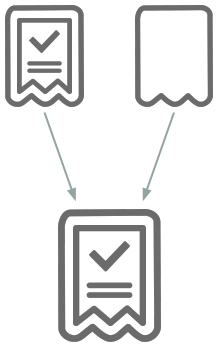
Threshold Fulfillment



```
cf:2:
AQEBAgEBBAAAAQAAAQEAJwAEASAgIHahWSBEpuT1ESZbynOmBNkLBS
nR32Ar4woZqSV2YNEBYA
```

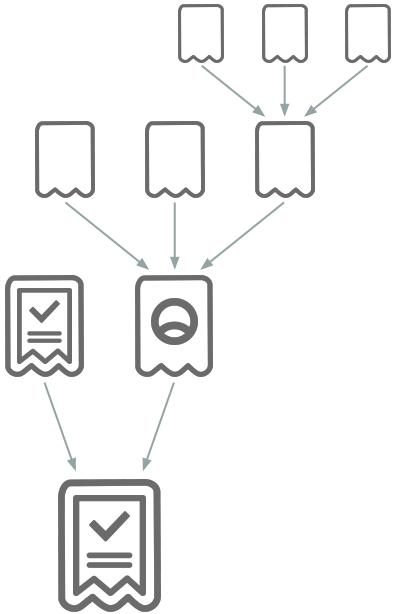
Merkle Circuits

Threshold Fulfillment



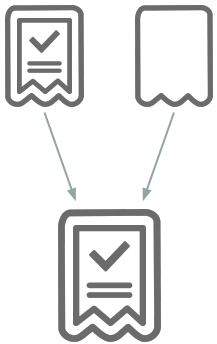
```
cf:2:
AQEBAgEBBAAAAQAAAQEAJwAEASAgIHahWSBEpuT1ESZbynOmBNkLBS
nR32Ar4woZqSV2YNEBYA
```

Threshold Fulfillment



```
cf:2:
AQEBAgEBBAAAAQAAAQEAJwAEASAgIHahWSBEpuT1ESZbynOmBNkLBS
nR32Ar4woZqSV2YNEBYA
```

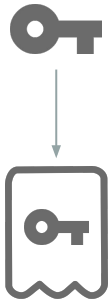
Threshold Fulfillment

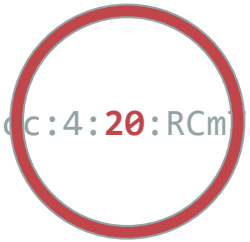


```
cf:2:
AQEBAgEBBAAAAQAAAQEAJwAEASAgIHahWSBEpuT1ESZbynOmBNkLBS
nR32Ar4woZqSV2YNEBYA
```

Bitmask

Condition Contains Feature Bitmask





dc:4:20:RCmBlAEqh5MSPTdAVgZTAI0m8xmTNluQA6iaZGKjVE:96

Bitmask — Simple Approach

Types

0	preimageSha256
1	rsaSha256
2	prefixSha256
3	thresholdSha256
4	ed25519

Feature Bits

_____1	preimageSha256
_____1_	rsaSha256
_____1__	prefixSha256
_____1___	thresholdSha256
_____1____	ed25519

Bitmask — Simple Approach

Types

0	preimageSha256
1	rsaSha256
2	prefixSha256
3	thresholdSha256
4	ed25519
5	preimageSha512
6	rsaSha512
7	prefixSha512
8	thresholdSha512

Feature Bits

________1	preimageSha256
_______1_	rsaSha256
______1__	prefixSha256
_____1___	thresholdSha256
____1____	ed25519
___1_____	preimageSha512
__1______	rsaSha512
_1_______	prefixSha512
1________	thresholdSha512

Bitmask — Feature Suites

Types

0	preimageSha256
1	rsaSha256
2	prefixSha256
3	thresholdSha256
4	ed25519

Feature Bits

________1	sha256
_______1_	preimage
______1__	prefix
_____1___	threshold
____1____	rsa
___1_____	ed25519

Bitmask — Feature Suites

Types

0	preimageSha256
1	rsaSha256
2	prefixSha256
3	thresholdSha256
4	ed25519
5	preimageSha512
6	rsaSha512
7	prefixSha512
8	thresholdSha512

Feature Bits

________1	sha256
_______1_	preimage
______1__	prefix
_____1___	threshold
____1____	rsa
___1_____	ed25519
__1______	sha512

Bitmask — Allows Feature Restriction

Types

0	preimageSha256
1	rsaSha256
2	prefixSha256
3	thresholdSha256
4	ed25519

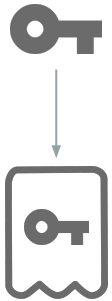
Feature Bits

________1	sha256
_______1_	preimage
______1__	prefix
_____1___	threshold
____1____	rsa
___1_____	ed25519

⇒ Smaller Implementation Size, Smaller Attack Surface

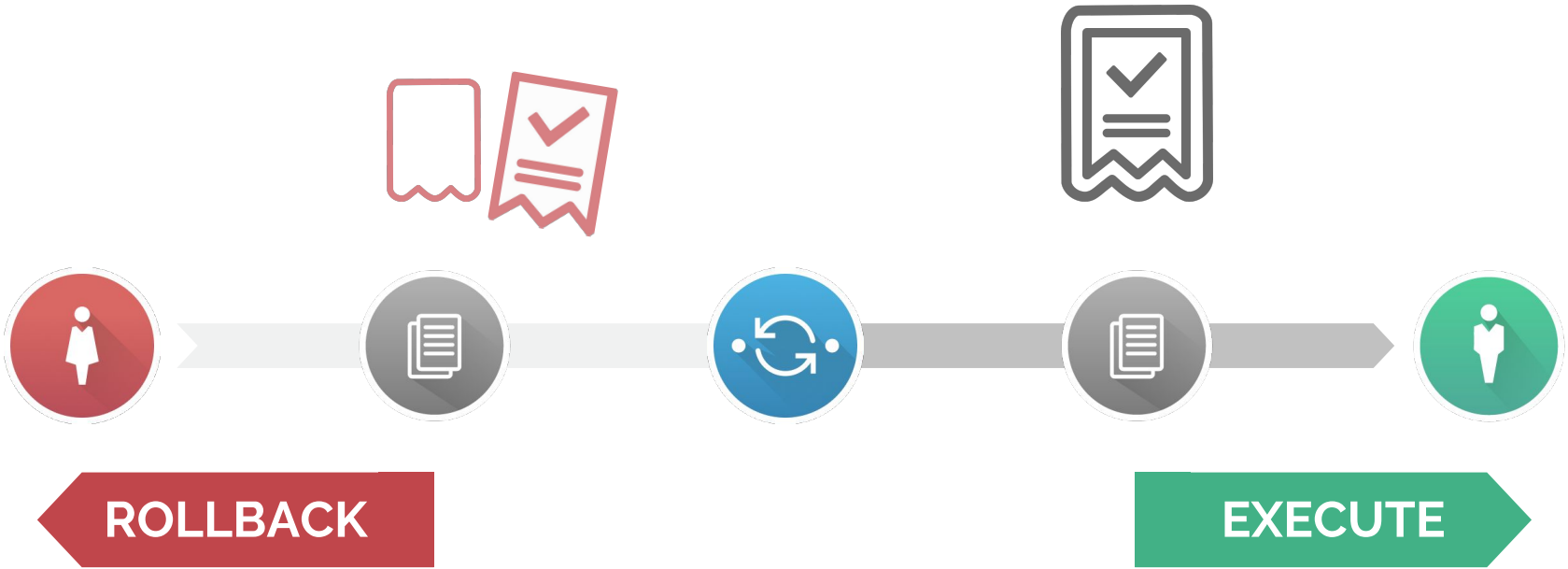
(Max) Fulfillment Length

Condition Contains Fulfillment Length



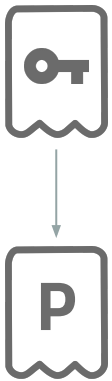
cc:4:20:RCmTBlAEqh5MSPTdAVgZTAI0m8xmTNluQA6iaZGKjVE:**96**

Fulfillment Size Must Be Supported By Everyone



Prefix Conditions

Prefix Condition



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

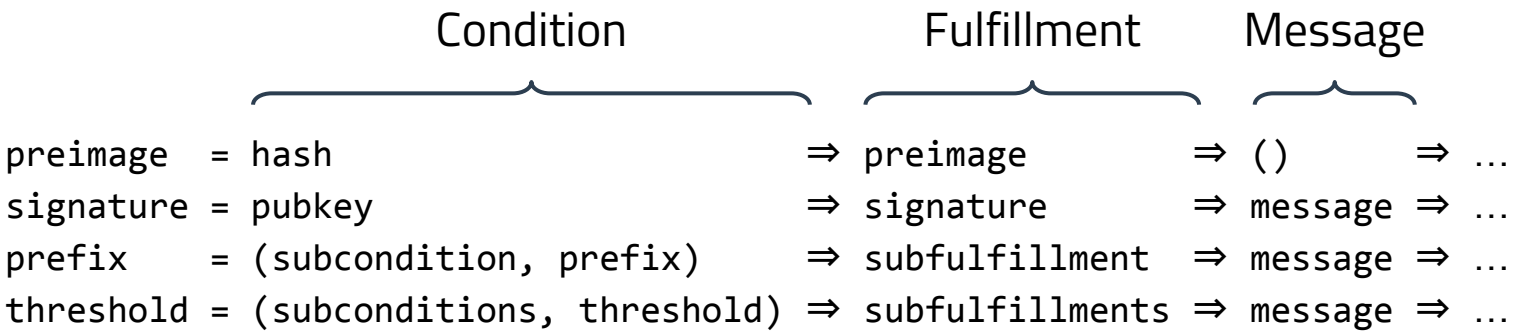
- Create a more limited scoped key
- Works with any condition type
- Enables fixed message conditions

Terminology

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

Other Types of Crypto-Conditions



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

Extensibility

Possible Future Extensions

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

Recap

Crypto-Conditions

Standard for composable signatures

(aka multi-sig on steroids)

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

Open Questions

- Should the bitmask be variable or fixed length? (currently: variable)
- Would a rehash/HMAC condition be safer than a prefix condition?
- Should we support ECDSA (e.g. P-256), Ed25519 or both?
- Is OER encoding the right choice?

Discussion