Blockstream

Smart Signatures: Experiments in Authorization

Christopher Allen

ChristopherA@Blockstream.com

Digital Signatures

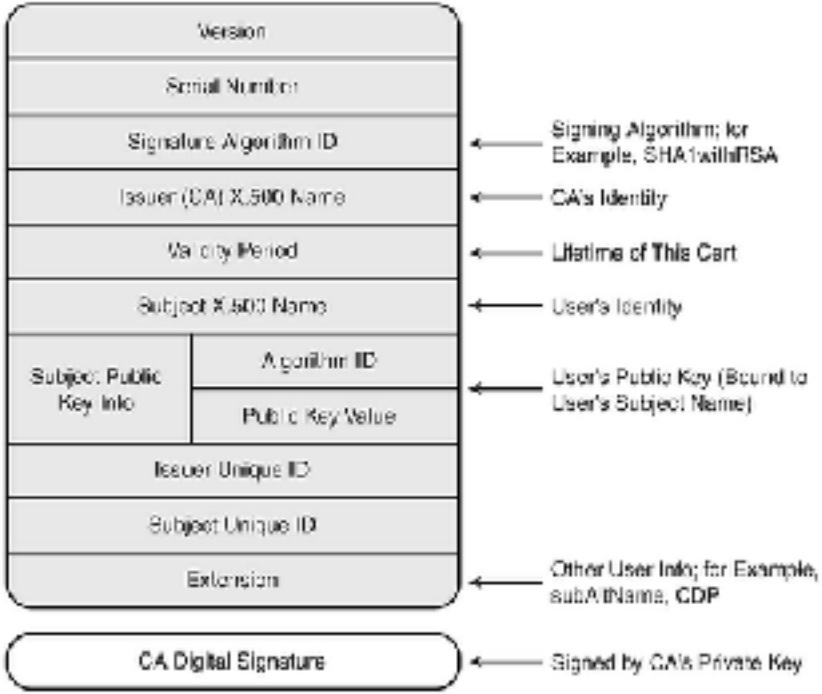
- **Demonstrates the validity of a message**
    - **1976:** Concept invented by Diffie & Hellman
    - **1980:** Digital Signatures Patent
    - **1983:** Made practical by Rivest, Shamir & Adleman
    - **1988:** First X.509 Digital Signature Standard issued
- **Architecture not changed significantly for 40 years!**


United States Patent [19]
Hellman et al.
[11] 4,200,770
[45] Apr. 29, 1980
[54] CRYPTOGRAPHIC APPARATUS AND METHOD
ABSTRACT

United States Patent [19]
Rivest et al.
[11] 4,405,829
[45] Sep. 20, 1983
CRYPTOGRAPHIC COMMUNICATIONS SYSTEM AND METHOD
ABSTRACT

Traditional Digital Signatures

- **To validate a message:**
  - Canonicalize message
  - Hash the message
  - Encrypt hash with private key
  - Validate with public key
- **Embody in a Certificate Data Format**
  - Typically ASN.1/X.509
- **Signed by other Certificates**
- **Confirmed using Trust Policy**



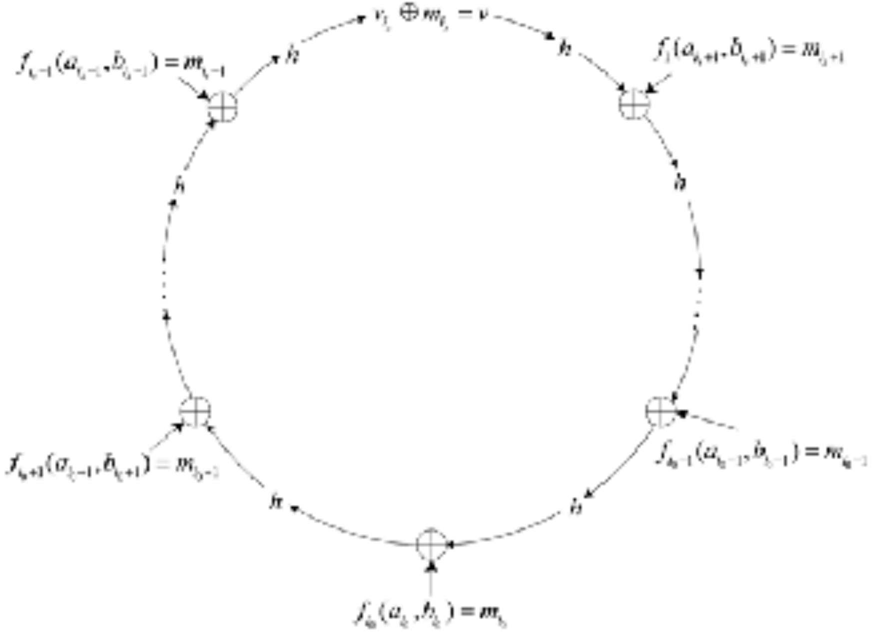
Blockstream

The Trust Policy

- **The Trust Policy is defined and limited by third-parties**
  - A Certificate Authority
  - An App, a Browser or OS
- **The Trust Policy is NOT defined by the signer or verifier!**
  - Is the intent of the signer fully expressed?
  - Does the verifier understand the intent of the signer?
  - Does the CA or App understand the trust requirements of the verifier?

New Kinds of Signatures

- **Modern crypto now allows:**
  - Multi-Signatures
  - Ring Signatures
  - Blind Signatures
  - Aggregated Signatures
  - Confidential Signatures
- **Traditional digital signature data formats have had difficulty adapting to these new forms.**



Traditional Authorization

- **Core use — Authorization!**
  - A Trust Policy ensures that the conditions required for a task are met
- **Traditional Signatures**
  - Authenticate that a specific party signed a message
  - Certify that the signing party is authorized to do the task

Traditional Authorization

- **Core use — Authorization!**
  - A Trust Policy ensures that the conditions required for a task are met
- **Traditional Signatures**
  - Authenticate that a specific party signed a message
  - ~~Certify that the signing party is authorized to do the task~~

Smart Signatures

- **Core use — Also Authorization!**
  - Signature Script ensures that all conditions required for a task are met
- **Smart Signatures**
  - Additional parties can be authorized
  - Parties can delegate authorization
  - AND/OR expressions
  - Conditions can be more than who signed!

Smart Signatures

- **The Difference**
  - Trust Policy is interpreted not by a CA, or code executed by an App, Browser or OS.
  - The Trust Policy is embodied by the signer into the signature itself
- **Conceived at first *#RebootingWebOfTrust Design Workshop* December 2015**
  - Christopher Allen, Greg Maxwell, Peter Todd, Ryan Shea, Pieter Wuille, Joseph Bonneau, Joseph Poon, and Tyler Close

Our Inspiration

- **Bitcoin Transaction Signature**
  - Uses a stateless predicate language (aka “Script”)
  - Created by the signer
  - Based on the signer’s Trust Policy
  - Supports ANDs, ORs, multi-sigs, time-locks, puzzles, or even other scripts
- **Many other possible use cases**

```
OP_DEPTH 1 OP_EQUAL
IF
    <pubKeyPresident>
    OP_CHECKSIGNATURE
ELSE
    2 <pubKeyVicePresidentA>
    <pubKeyVicePresidentB>
    <pubKeyVicePresidentC>
    3 OP_CHECKMULTISIG
ENDIF
```

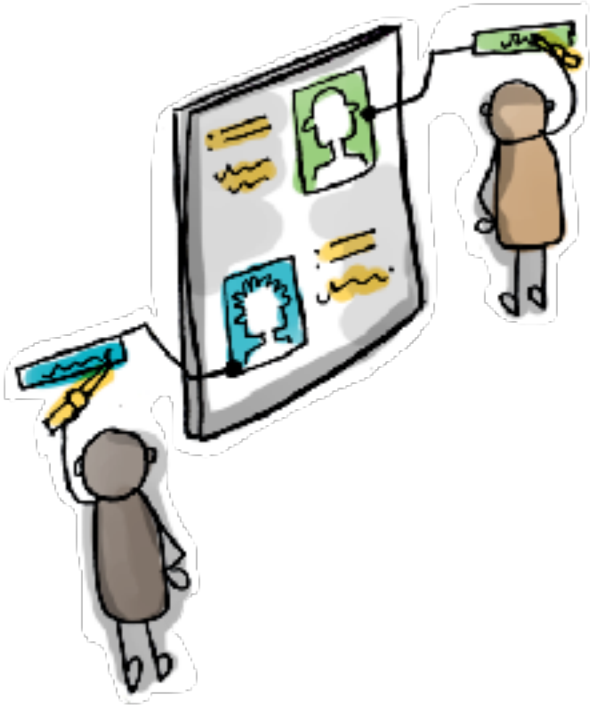
1 key OR 2 of 3 keys

Use Case: Multifactor Expressions

- **Multiple parties within a single smart signature**
  - N of N signatures
  - M of N signatures
  - Logical AND and ORs
- **Other possible elements**
  - biometric signatures
  - proof of hardware control
  - etc.

Use Case: Signature Delegation

- **Signers should be able to:**
  - Delegate to another party
  - Limit delegated usage based on
    - Time (*"1 month"*)
    - Function (*"only purchases"*)
    - Content (*"not more than $5K"*)
  - Optionally to permanently pass control if usage of a key ceases



Use Case: Multiple Combinations

- **Multiple Combinations**
  - multifactor & delegation & message context
- **For instance:**
  - Development Release / Continuous Integration Toolchain
    - multifactor 3-of-5 signatures
      - one signer has authorized his assistant because he's on leave
      - another signer requires 2-of-2 keys for his signature
        - one of which is stored on a hardware token.

Use Case: Transactional Support

- **Signatures are often part of a larger process**
  - Prove specific transactional states exist
  - Test against Oracles
- **For instance**
  - “No more than $5K has already been spent this month”
  - Transactional history of a painting to ensure provenance

Requirements

- **Smart Signatures are complex and thus have security pitfalls**
  - The script language
  - The signatures & the system
- **Six categories of requirements**
  - Composable
  - Inspectable
  - Provable

  } language
  - Deterministic
  - Bounded
  - Efficient

  } system

Requirement: Composable

- **A smart signature language should be Composable**
  - Aggregate simple behaviors into more complex ones
  - Simple data structures: stacks, lists, etc.
  - Constrained set of operations to allow security review
  - Inspiration: Forth, Scheme, Haskell, etc.

Requirement: Inspectable

- **A smart signature language should be Inspectable**
  - Understandable by a qualified programmer
  - Make visible the many elements of the signature script and how they will be verified
  - Help the programmer evaluate the function and purpose of script

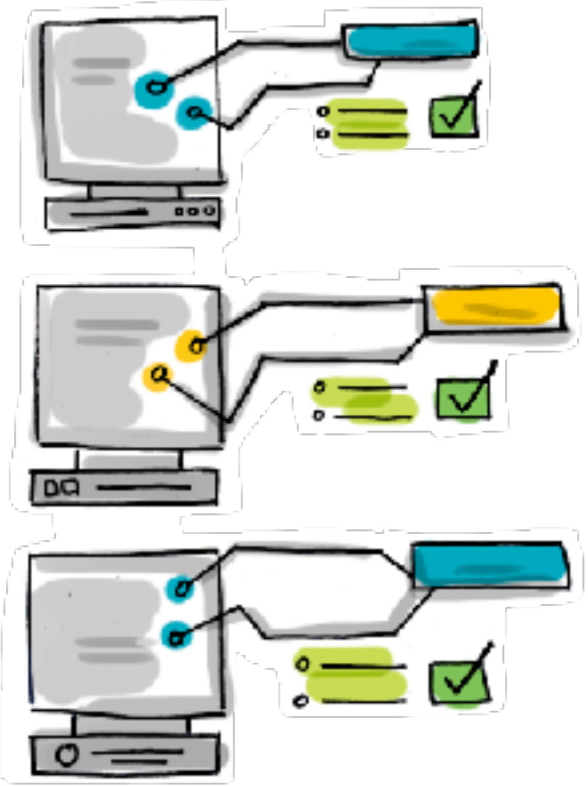
Blockstream

Requirement: Provable

- **A smart signature language should be Provable**
  - Formally analyzable to prove correctness
  - Support expert tools to discover hidden bugs

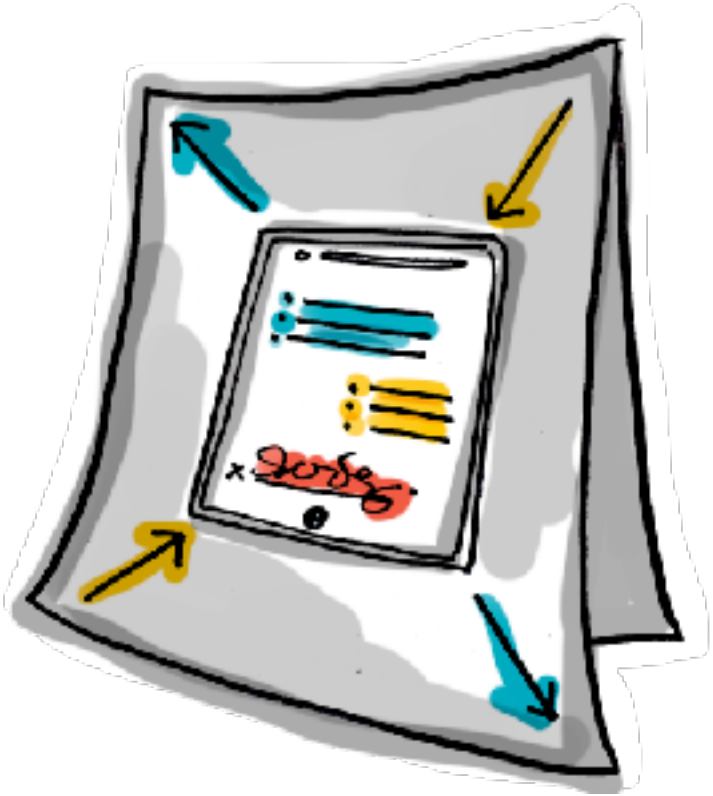
Requirements: Deterministic

- **A smart signature system should be Deterministic**
  - Scripts should always produce the same result
  - Even on different OS or hardware



Requirement: Bounded

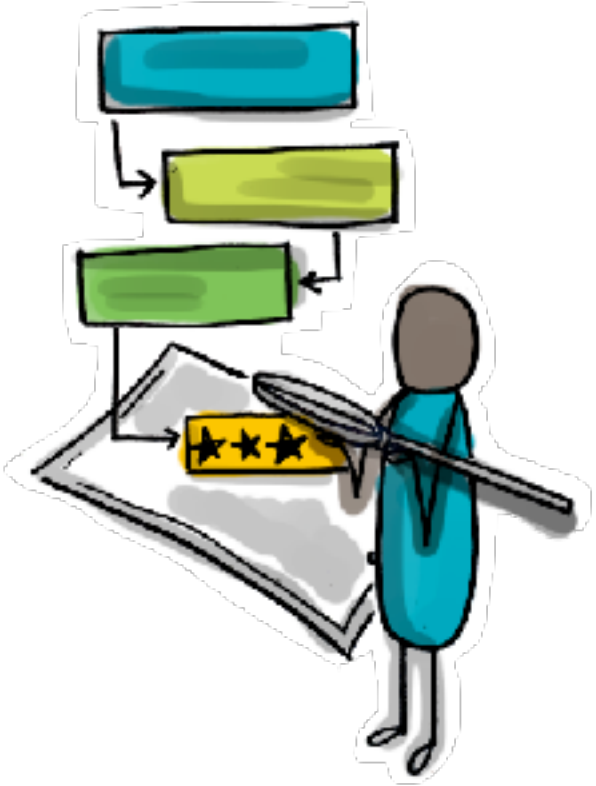
- **A smart signature system should be Bounded**
  - Execution must not exceed appropriate CPU or memory limitations
  - Minimize the size of scripts in order to limit bandwidth and storage costs.
  - Enforcement of these limitations must also be deterministic.



Blockstream

Requirement: Efficient

- **A smart signature system should be Efficient**
  - ***No*** requirements on the difficulty of creating signatures
  - The cost of verifying should be very low



A Challenge: Privacy?

- **Always a trade-off between flexibility & privacy**
  - Reveals information about Signers
  - Smart signature functionality may allow correlation
  - Reduces substitutability, and thus may break fungibility & bearer aspects
- **A consideration, not a requirement**
  - Limit sharing, execute off-chain
  - Be transparent & be deliberate

Experiments: Bitcoin Script

- **Bitcoin Script**
  - A Forth-Like Language
  - Stack-Based
  - Well-Tested, Well-Trusted
  - Currently limited capabilities
    - MAST & Schnorr coming
  - \+ Deterministic, Bounded, Efficient
  - ~ Composable, Inspectable
  - – Provable

```
OP_DEPTH 1 OP_EQUAL
IF
    <pubKeyPresident>
    OP_CHECKSIGNATURE
ELSE
    2 <pubKeyVicePresidentA>
    <pubKeyVicePresidentB>
    <pubKeyVicePresidentC>
    3 OP_CHECKMULTISIG
ENDIF
```

1 key OR 2 of 3 keys

https://en.bitcoin.it/wiki/Script

Experiments: Ivy

- **The Ivy Approach**
  - By Chain.com
  - Compiles to Bitcoin Script
  - Easier syntax
  - Adds named variables
  - Static types
  - + Inspectable Bitcoin “Script”
  - Same limitations as Bitcoin Script
  - – Provable

```
contract LockWithMultisig(
  pubKey1: PublicKey,
  pubKey2: PublicKey,
  pubKey3: PublicKey,
  val: Value
) {
  clause spend(sig1: Signature,
sig2: Signature) {
    verify checkMultiSig([pubKey1,
pubKey2, pubKey3], [sig1, sig2])
    unlock val
  }
}
```

Blockstream

http://docs.ivy-lang.org/bitcoin/

Conditional script for debug build

Experiments: Dex

- **The Dex Approach**
  - Deterministic Predicate Expressions by Peter Todd
  - Scheme-like Lambda Calculus
  - Optimized for Hash Tree
  - Partial proofs are supported
  - Built to support state machines
  - + Composable, Deterministic, Efficient, Bounded
  - ~ Inspectable, Provable

```
(or (checksig release-pubkey sig (hash build))
    (and (checksig dev-pubkey sig (hash build))
         (== build-type "debug")))
```

https://petertodd.org/2016/state-machine-consensus-building-blocks

Blockstream

Conditional script for debug build

Experiments: Simplicity

- **The Simplicity Approach**
  - By Russell O'Connor, Blockstream
  - Sequent Calculus
  - Finitary functions with bounded complexity
  - Formal Provable Semantics
  - Scripts formally provable via Coq
  - + Provable, Deterministic, Bounded, Efficient, Composable
  - ~ Inspectable

```
basicSigVerify b c :=
comp (pair(witness b)
     (pair pubKey (comp
(witness c)sighash)))
     (comp (pair checkSig
unit) (case fail unit))
```

Blockstream https://blockstream.com/simplicity.pdf ***Basic signature verify***

Experiments: Simplicity

SESSION TOMOROW 10:50am!

Experiments: Σ–State

- **The Σ–State Approach**
  - By Alexander Chepurnoy
  - Uses Sigma–Protocols
    - Optimized for zk-proofs
    - Ring & Threshold Sig
  - Strong Types
  - + Inspectable, Composable, Deterministic, Efficient
  - ~ Provable, Bounded

```
(height ≥ 100 ∧ dlog_g
backerPK) ∨ (height < 100
∧ tx.has_output (amount ≥
100000, proposition =
dlog_g projectPK)
```

Blockstream

https://github.com/ScorexFoundation/sigmastate-interpreter

Cost Limit plus Timelock

Experiments: Michelson

- **The Michelson Approach**
  - By Tezos
  - Inspired by OCaml
  - Like “Script” is Stack-Based
  - Strongly Typed
  - + Composable, Inspectable, Efficient
  - ~ Provable, Bounded, Deterministic

```
parameter key_hash;
storage (pair timestamp (pair tez
key_hash));
return unit;
code {
DUP; CDAR; DUP; NOW; CMPGT; IF {FAIL}
{}; SWAP;
DUP; CAR; DIP{CDDR}; AMOUNT; PAIR;
SWAP; DIP{SWAP; PAIR};
DUP; CAR; AMOUNT; CMPLE; IF {FAIL} {};
DUP; CAR;
DIP{CDR; DEFAULT_ACCOUNT}; UNIT;
TRANSFER_TOKENS;
PAIR }
```

Crowdfunding Script

Blockstream

https://www.tezos.com/static/papers/language.pdf

Experiments: Crypto Conditions

- **The Crypto Conditions Approach**
  - By Ripple for Interledger
  - Not a language, A JSON description!
  - Deterministic Boolean Algebra
  - Easier Testing, Limited Flexibility
  - + Bounded, Efficient, Deterministic
  - ~ Inspectable
  - – Composable, Provable

```
const conditionDescription = {
  type: 'threshold-sha256',
  threshold: 2,
  subconditions: [{
    type: 'prefix-sha256',
    prefixUtf8: '...',
    subcondition: {
      type: 'ed25519',
      publicKey: '...' } }, {
    type: 'preimage-sha256',
      preimage: '...' }] }
```

Blockstream

Experiments: Status

- **Bitcoin Script** - Integrated into Bitcoin, no full stand-alone version (github.com/kallewoof/btcdeb debugger is a start)
- **Ivy** - Whitepaper, Full Ivy script playground available
- **Dex** - No Whitepaper, no implementation
- **Simplicity** - Whitepaper available, no public code yet
- **Σ–State** - Whitepaper soon, code in-progress
- **Michelson** - Whitepaper, alpha script playground
- **Crypto Conditions** - Whitepaper, part of Interledger reference

Watching: Smarm

- **The Smarm Approach**
  - By Christopher Lemmer Webber
  - Designed for Smart Signatures, maybe on top of Simplicity
  - Subset of Scheme R5RS, but Typed
  - Restricted Environment & Lexical Scope based on Reese's W7
  - Can compile to Native Code
  - + Composable, Inspectable, Deterministic,
  - ~ Provable, Bounded, Efficient

?

http://bit.ly/SmarmRequirements

Watching: Frozen Realms

- **The Frozen Realms Approach** ?
  - By Miller, Morningstar, Patiño
  - A “safe” subset of Javascript
  - Limited Primordials
  - May compile to WASM(?)
  - + Composable, Inspectable,
  - ~ Provable, Efficient, Deterministic
  - – Bounded (Turing Complete!)

Blockstream https://github.com/tc39/proposal-frozen-realms

Watching: Bamboo/EVM

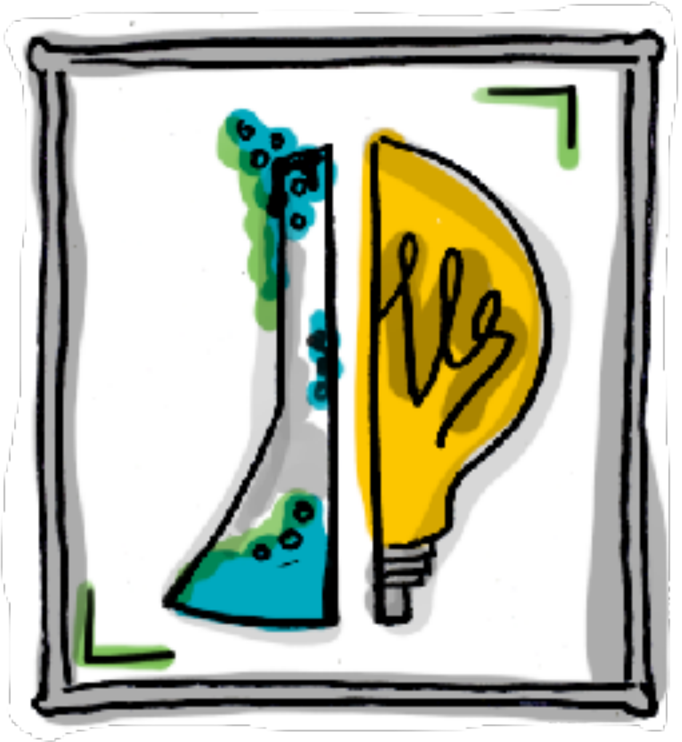
- **The Bamboo Approach**
  - Designed for Ethereum
  - Javascript-like
  - Explicit state transitions
  - Avoids reentrancy
  - + Composable, Inspectable
  - ~ Deterministic, Efficient
  - – Provable, Bounded (Turing Complete!)

```
contract Vault(address hotwallet, address
vaultKey, address recoveryKey) {
  case(void unvault(uint256 amount)) {
    if (sender(msg) != vaultKey) abort;
    uint256 unvaultPeriod = 60 * 60 * 24 * 7 *
2; // two weeks
    if (now(block) + unvaultPeriod < now(block))
abort;
    return then become UnVaulting(now(block) +
unvaultPeriod, amount, hotwallet, vaultKey,
recoveryKey); }
   case(void recover(address _newHotWallet)) {
    if (sender(msg) != recoveryKey) abort;
    return then become Vault(_newHotWallet,
vaultKey, recoveryKey); }
  case(void destroy()) {
    if (sender(msg) != recoveryKey) abort;
    return then become Destroyed(); } }
```


https://github.com/pirapira/bamboo

Open Questions

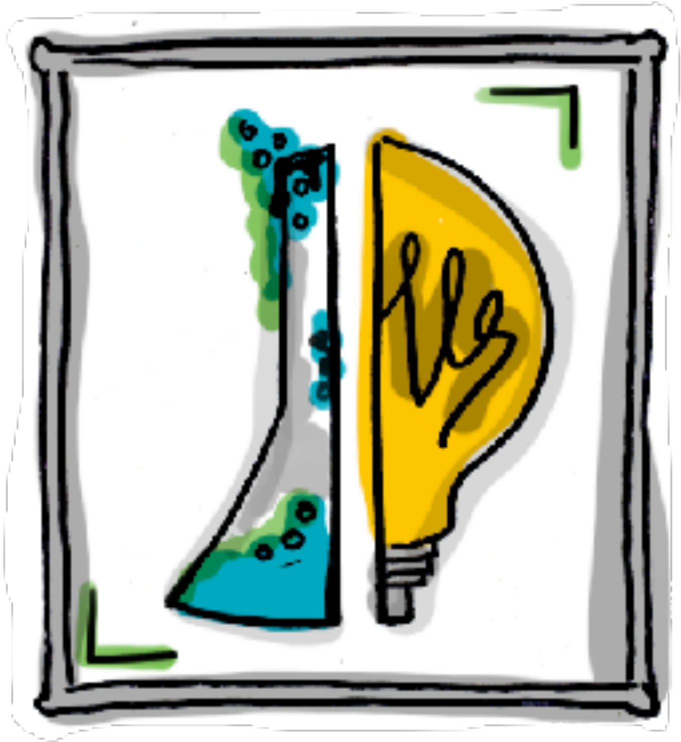
- **Context**
  - Internal references?
    - Lists, trees, acyclic graphs
  - Run-time context?
  - External process state?
- **Oracles**
  - Preserving execution boundedness?
  - What are simple MVP oracles?
- **Revocation**
  - Proof of non-revocation?
  - Short-life vs. revocation?



Blockstream

Open Questions

- **Object Capabilities**
  - Are “ocap” and Least Authority architectures another use case?
- **Cryptographic Primitives**
  - HD Keys?
  - Poelstra’s “Scriptless Scripts”?
- **Smart Contracts**
  - Non-predicate scripts?
  - None of the experiments above are Turing-complete, but where exactly is the line between?



References

C. Allen, G. Maxwell, P. Todd, R. Shea, P. Wuille, J. Bonneau, J. Poon, and T. Close. “Smart Signatures”. Rebooting the Web of Trust I. https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/smart-signatures.pdf. 2015.

C. Allen, S. Appelcline. “Smarter Signatures: Experiments in Verification”. Rebooting the Web of Trust II. https://github.com/WebOfTrustInfo/ID2020DesignWorkshop/blob/master/final-documents/smarter-signatures.pdf. 2016.

bit.ly/SmarterSignatures **#SmartSignatures**

Blockstream

Are you a Language Geek? Come to Next #RebootingWebOfTrust

“To influence the future of decentralized trust and self-sovereign identity through the establishment & promotion of decentralized identity technology. This is done via the collaborative creation of white papers and specifications & by public presentations of these ideas.”

REBOOTING the WEB of TRUST

March 6-8th in Santa Barbara

https://rwot6.eventbrite.com

Blockstream

Christopher Allen

PGP: **FDA6C78E**

ChristopherA@Blockstream.com

http://www.Blockstream.com

Blockstream