ZKP and Corda – current state and path to adoption

KASIA STREICH – SOFTWARE ENGINEER AT R3

DLT platform written in Kotlin developed at R3

Smart contracts in any JVM language

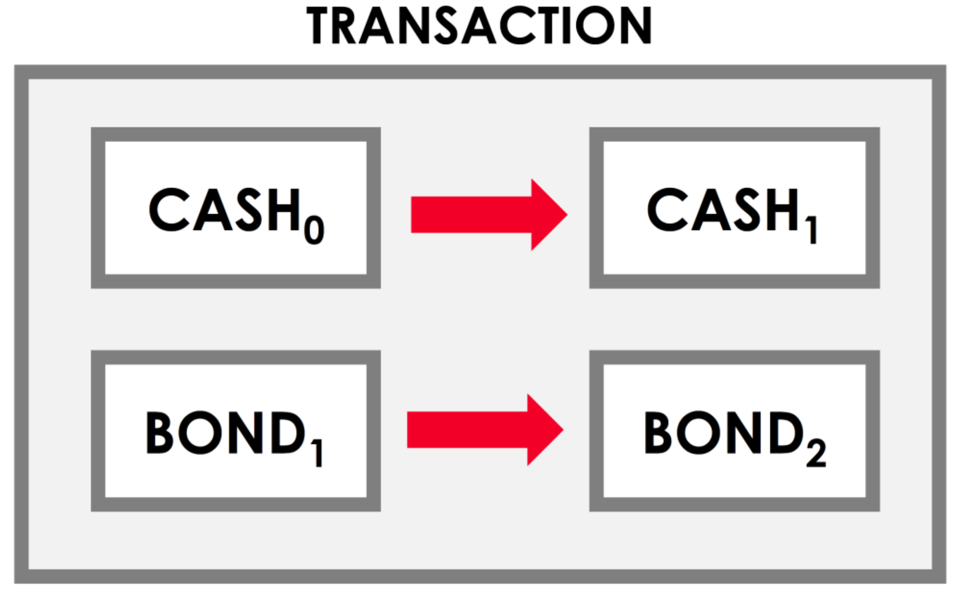
Open source:
https://github.com/corda/corda

Agenda

- Corda data model
- Transaction validity
- Privacy in Corda
- SGX
- Finding ZKP shaped problems
- Path to adoption of ZKP

Corda

- Classic UTXO model
- Immutable states
- Transactions mark states as committed



Corda

Example verification in Kotlin:

- Contracts in JVM languages
- Verify function
- CorDapp

```
class CarContract : Contract {
    override fun verify(tx: LedgerTransaction) {
        val command = tx.commands.requireSingleCommand<Commands>().value

        when(command) {
          is Commands.Issue -> requireThat {
            "There should be no input state" using (tx.inputs.isEmpty())
            "There should be one input state" using (tx.outputs.size == 1)
            "The output state must be of type CarState" using
(tx.outputs.get(0).data is CarState)
            val outputState = tx.outputs.get(0).data as CarState
            "The licensePlateNumber must be seven characters long" using
(outputState.licensePlateNumber.length == 7)
          }
        }
    }
```

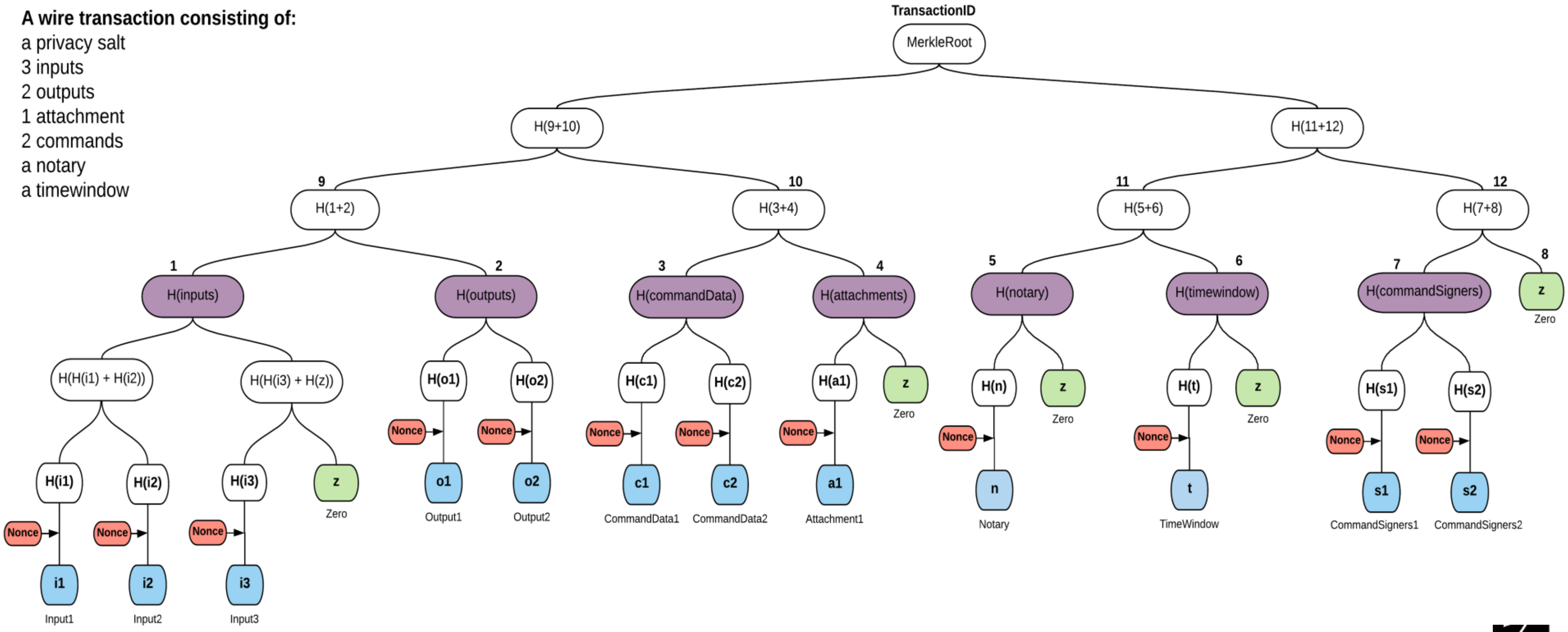
Corda – Uniqueness consensus

- Double spend using notaries
- Validating and non validating nodes
- Privacy hiding for non-validating notaries was implemented using partial Merkle trees

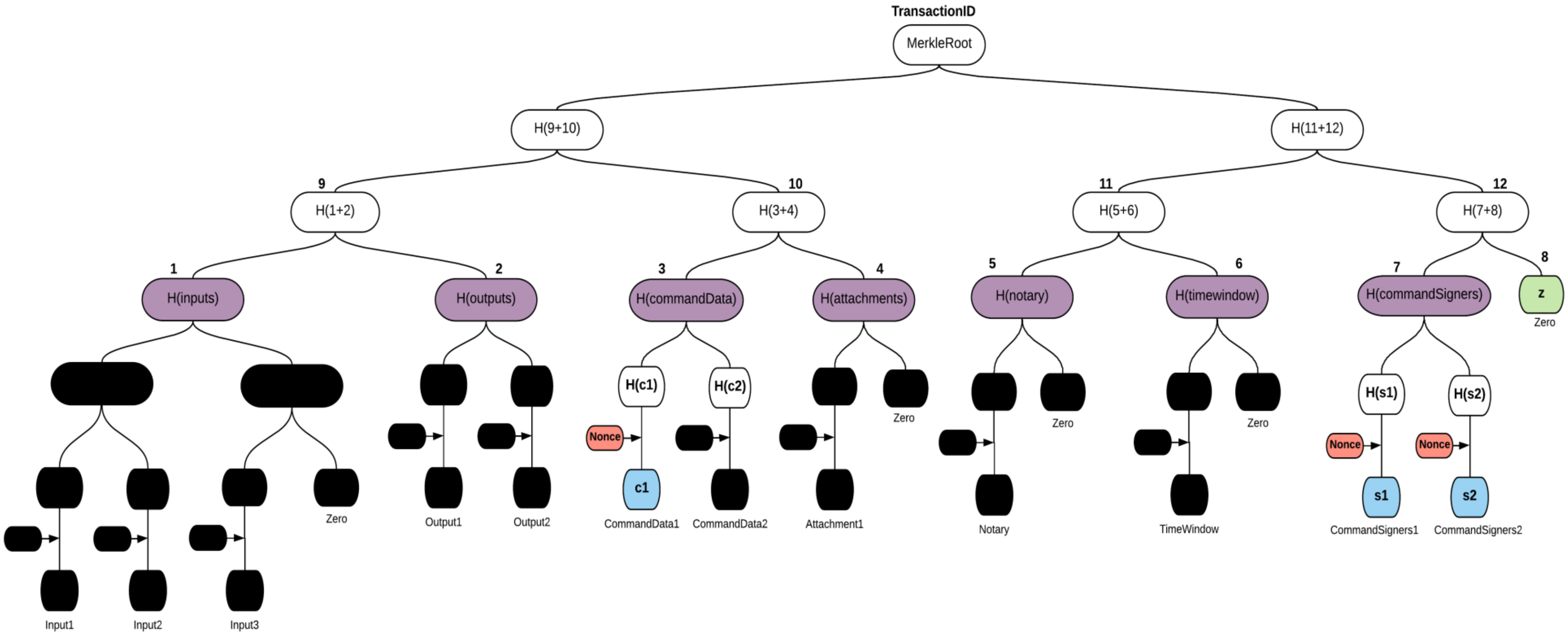
Merkle trees in Corda

A wire transaction consisting of:

a privacy salt
3 inputs
2 outputs
1 attachment
2 commands
a notary
a timewindow



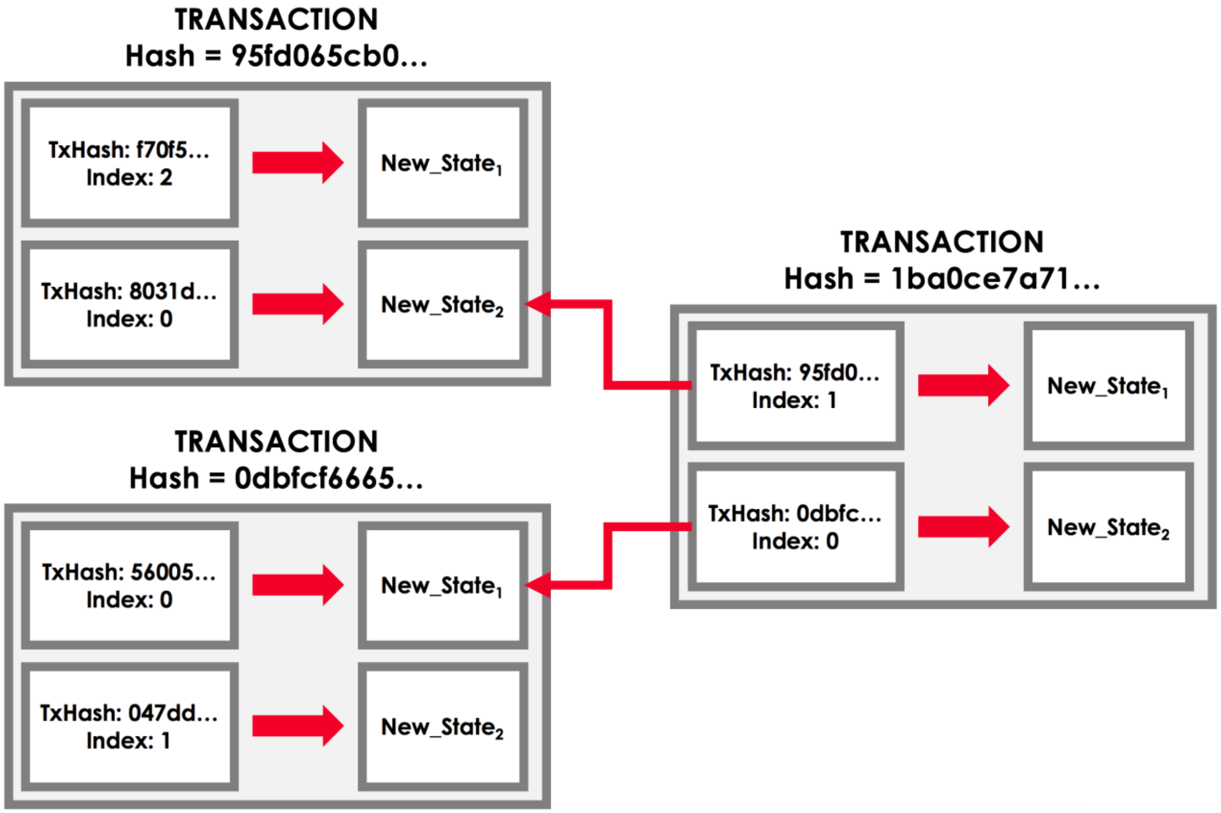
Merkle trees in Corda – tear-offs



What makes a transaction valid

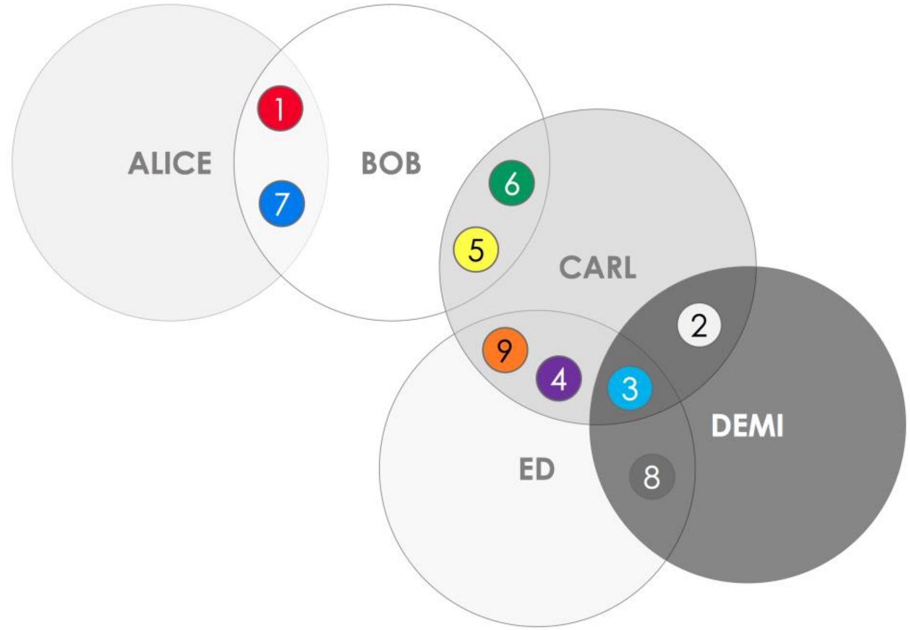
- Contract verification
- Signatures from participants
- Double spend protection – notary signature
- Merkle tree verification
- Back-chain resolution

Back-chain resolution



Corda - privacy

- Data shared on need to know basis
- With back-chain resolution for fungible assets – problematic
- Transactions propagate – not good!



What to do?

INTEL SOFTWARE GUARD EXTENSIONS

- SGX provides a way to offload sensitive data processing to remote untrusted machines
- Hardware-based memory encryption which isolates application and data in memory
- Enclaves are protected from any process running at higher privilege level
- Protocol that lets client check validity of computation run in remote enclave – remote attestation

Why SGX?

PRACTICAL CONCERNS

- Aggressive timelines
- R3 SGX SDK - Framework to write own enclaves using JVM languages with remote attestation – version 1.0
- Easier to use for domain experts in finance
- No need to check all signatures in a back-chain

ZKPs – possible use cases for our business

- In preparation for adoption of ZKPs we design our platform having it in mind
- Same use case as SGX
- Hardness of breaking the hardware vs hardness of breaking a mathematical problem
- Three problems arise immediately:
  - Back-chain validation
  - Non-validating notaries
  - Oracles
- Non-interactive - proofs that everyone on the network can verify
- We may try evaluate approach using one of stable cordapps – Token SDK seems like a good choice

Path to adoption

- Code arithmetisation
- Common reference string problem
- New signature schemes
- Algorithmic agility
- Training and documentation

JVM languages -> R1CS?

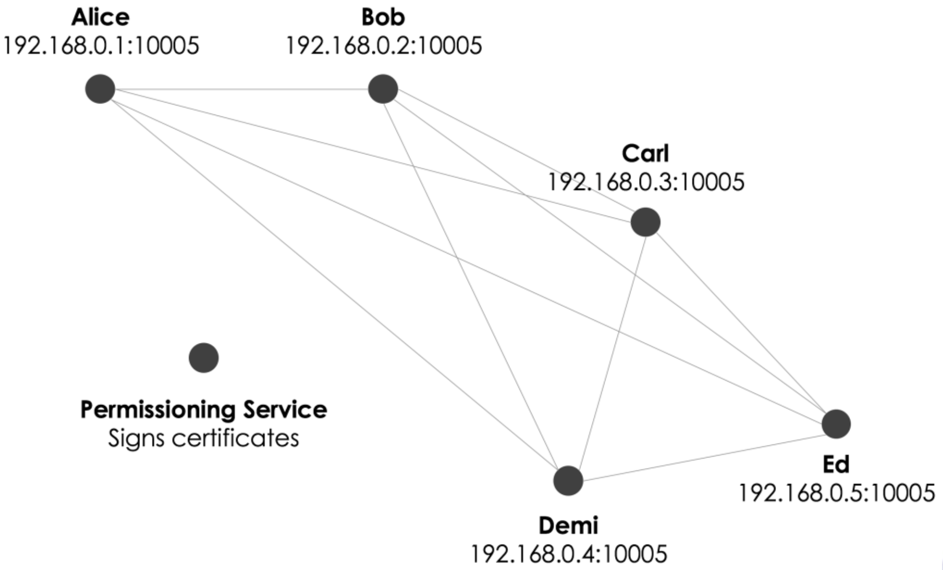
- Graal VM intermediate representation seems to be a good choice
- Graal project already converts bytecode to graph in SSA form and does aggressive memory read/writes optimization – good start for code flattening
- By doing this work once we get Java, Kotlin, Scala, Clojure (lisp), Haskell, Python, Ruby, JavaScript etc.

What about non-determinism? Deterministic JVM!

- All nodes need to agree what is valid and what not
- JVM contracts, execution in different environments can yield different results
- Custom build JVM sandbox – static analysis and byte code rewriting
- IntelliJ plugin
- Deterministic subset of Java:
  - Files, I/O, network, etc.
  - Random number generators

Trusted setup

- Sonic updateable CRS seems like a good solution
- We use x509 certificates
- Well known identities with X500 name
- Low possibility of sybil attack on network in this case
- We need to prepare infrastructure



What about signature validation?

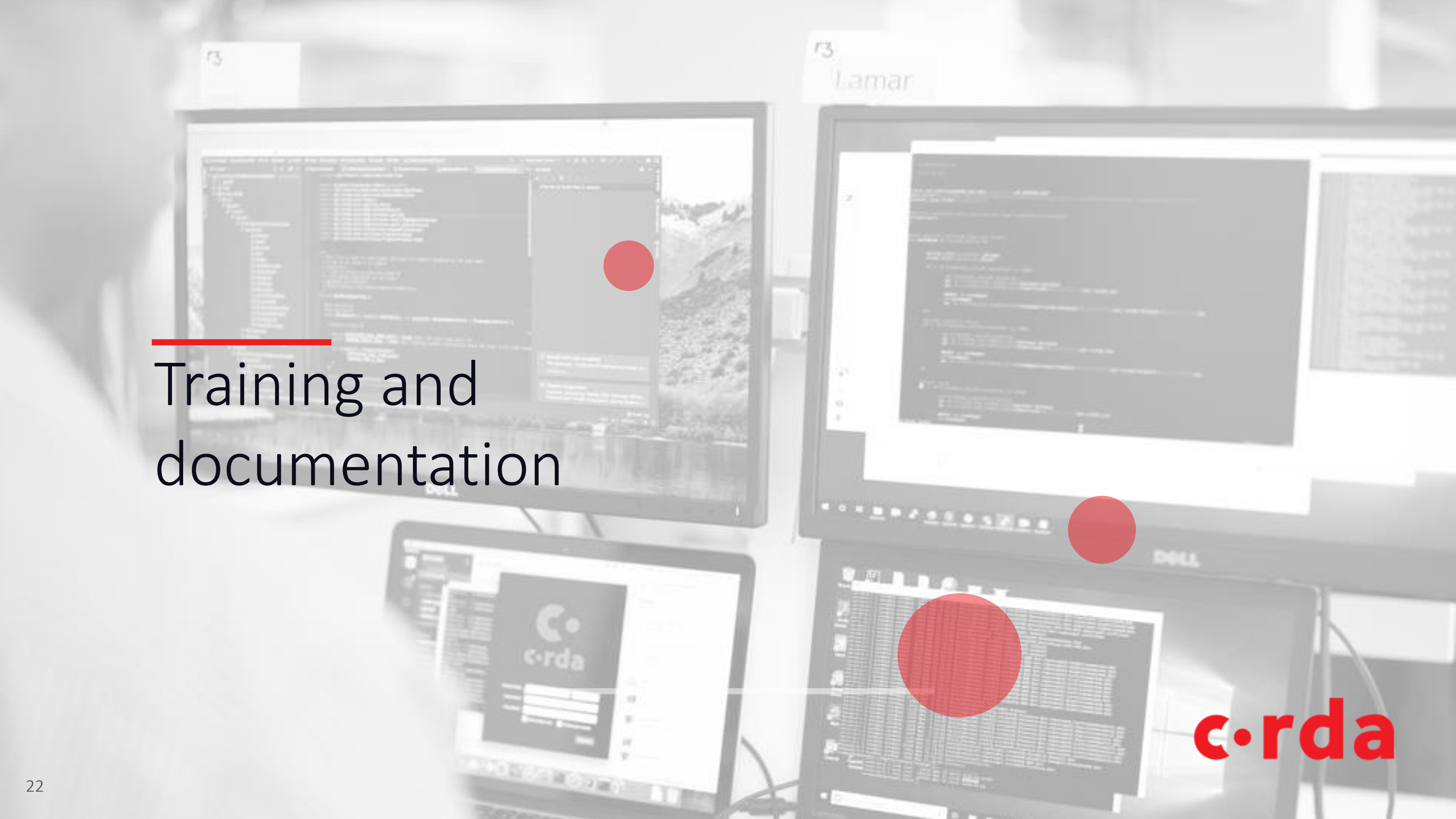
- So far the signature algorithms that are ZKP friendly are not widely adopted
- Cipher suites supported in Corda:
  - Pure EdDSA using the ed25519 curve and SHA-512
  - ECDSA using the NIST P-256 curve (secp256r1) and SHA-256
  - ECDSA using the Koblitz k1 curve (secp256k1) and SHA-256
  - RSA (3072bit) PKCS#1 and SHA-256
  - SPHINCS-256 and SHA-512 (experimental)

What is needed from business perspective

- Clarity and standardization on what EC algorithms are optimal
- Various factors when designing Corda:
  - Security-level
  - Adoption
  - Compatibility with HSM vendors
  - Business demand
  - Side channel security
  - Post quantum resistance
  - Rigorous testing

Algorithmic agility

- Introduction of new signature algorithms means global upgrade of all nodes, because it’s part of consensus!
- Research in ZKP area moves fast
- To be able to introduce new protocols we need update path
- In particular proof algorithm N+1 should be able to verify proofs from the previous ones



Training and documentation

Thank you

www.r3.com

New York
11 West 42nd Street, 8th Floor
New York, NY 10036

London
2 London Wall Place,
London, EC2Y 5AU

Brazil
Av. Angélica, 2529 - Bela Vista
6th Floor
São Paulo - SP, 01153-000, Brazil

Singapore
18 Robinson Road, Level #14-02
Singapore, 048547

Hong Kong
Bonham Strand, 7F Office 18-121
Hong Kong