ZKP in Corda

Our experience and more

Aleksei Koren

ZKProof Community Event, October 28, Amsterdam

ING

Agenda

Corda privacy model

Security/privacy tradeoff

Denial-of-State attack

Zero-Knowledge Notary

Pains and insights

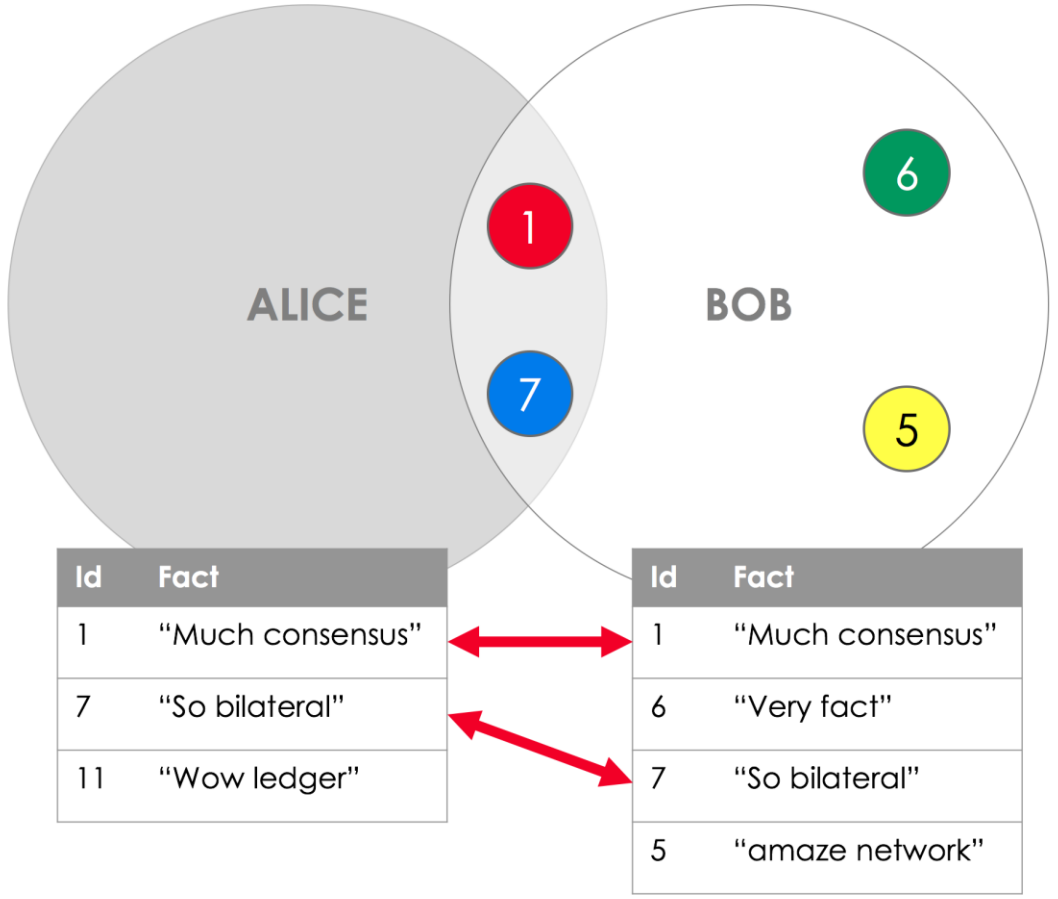
Corda privacy model

DAG UTXO

Transaction data (and chain history) is visible only to participants

Notary service provides protection from double-spending

Validating vs non-validating notary



Non-validating Notary: denial-of-state attack

Does NOT validate:
Smart contract rules, or signatures

DOES check if inputs are already consumed

Attackers can consume any unspent state they know ref of,
no matter who owns it.

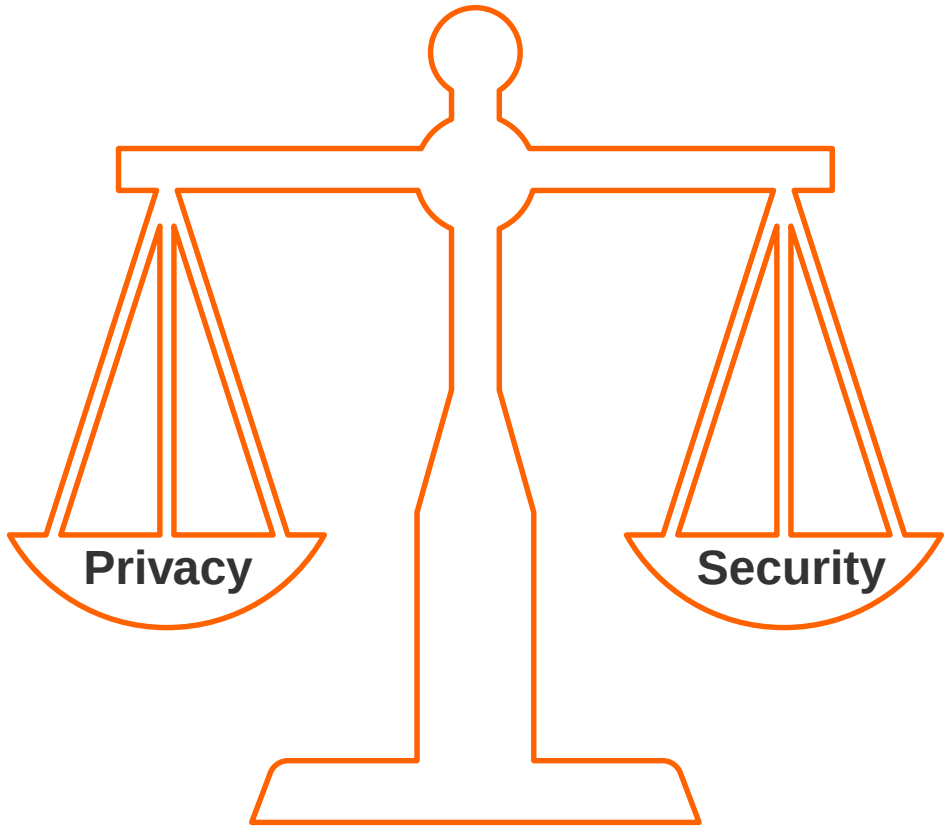
Real owner will be denied use of the state **for ever***

So there is a trade-off

We want to share our transaction details on a need-to-know basis only.

Trade-off:

- non-validating: denial-of-state risk
- validating: privacy



So can we have our cake and eat it too?

Solution 1: Ignore!

Pro: it's what we do now

Con: If you close your eyes and don’t see a problem, a problem still sees you

Solution 2: Lawyer up!

Pro: will fix situation eventually

Con: resolution might not be fast

Con: during investigation you might have to reveal everything

Con: (centralized) arbitration needed

Con: undefined how to 'revert' transactions

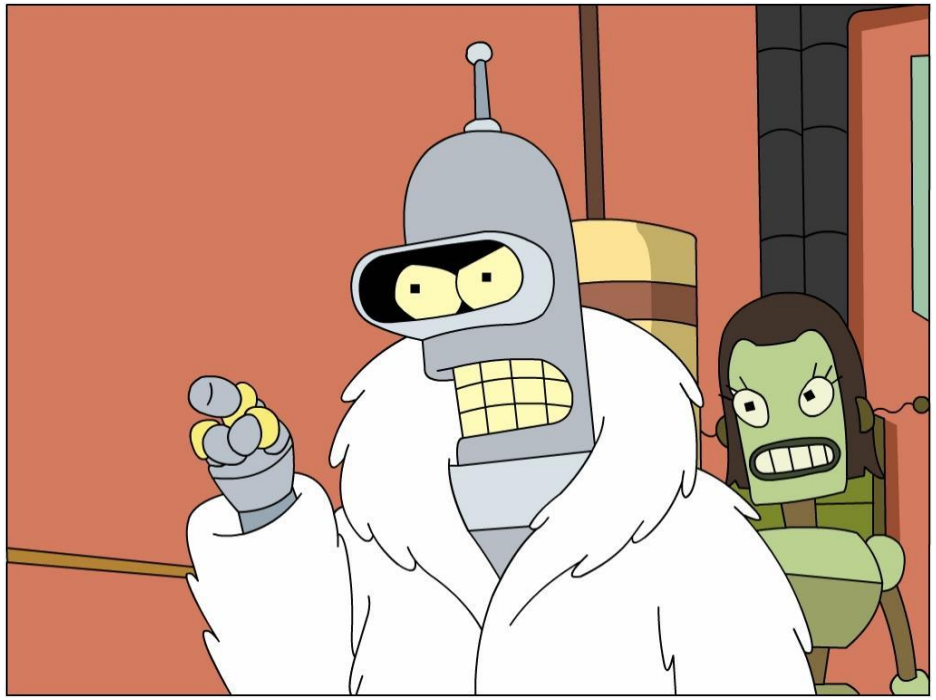
Solution 3: Trusted Hardware (SGX)

Pro: solves everything!

Con: vendor lock-in

Con: more complex development

Fact: need to trust vendor (current security issues, backdoors)



Solution 4: ZKP (no knowledge - no problem)

Pro: solves everything as well

Con: also more complex development

Fact: need to trust global scientific community

Extra bonus: verifier scalability

Agenda

Corda privacy model

Security/privacy tradeoff

Denial-of-State attack

Zero-Knowledge Notary

Pains and insights

ZK Notarisation: Overview

Non-validating Notary as base. Same privacy, same checks.

Initiator creates a *zero-knowledge proof* that:

- for the id of the tx we want to verify,
- they know a matching tx data structure
- that satisfies the smart contract rules
- and that they know the matching signatures

The Notary verifies this proof and is convinced of these facts.

This prevents Denial of State attack: attacker can no longer spend a state that they do not own.

ZK Notarisation: Toolchain

Setup Once: Notary

1) Translates Corda smart contract to C *(manually at the moment 😔)*
2) Compiles C code to libsnark-compatible R1CS using Pepper
3) Performs trusted setup for libsnark *(spoiler: it's no problem!)*
4) Distributes prover key

Proving: Tx Initiator

1) From Corda, generates a proof for tx with libsnark using prover key, with tx id as public input and transaction data and signatures as secret input.

   Secret input is a serialised version of a full LedgerTransaction

Verification: Notary

1) Verifies the proof with libsnark using verifier key, with only tx id as input

Agenda

Corda privacy model

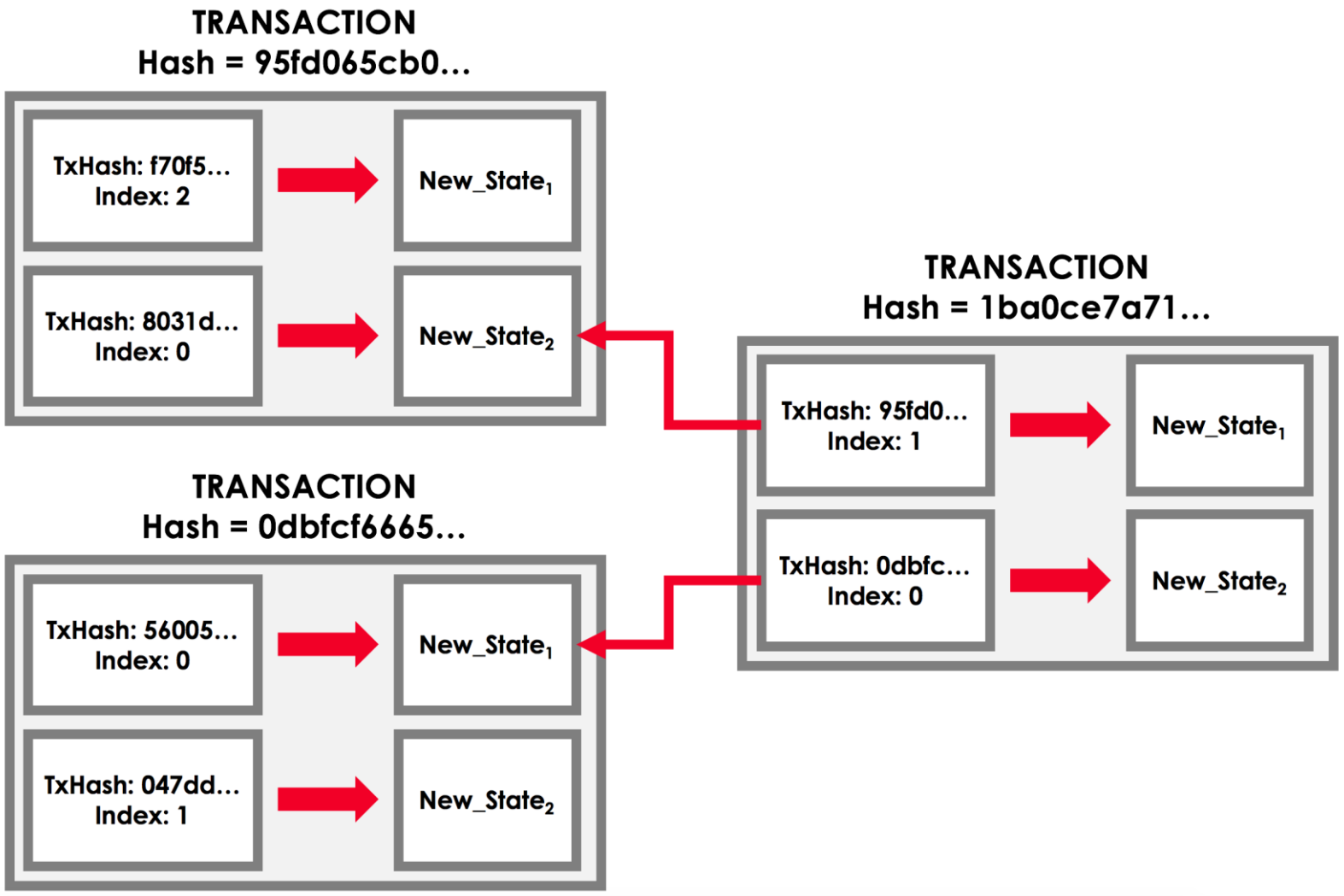
Security/privacy tradeoff

Denial-of-State attack

Zero-Knowledge Notary

Pains and insights

Checking transaction history



Gadget Central

Compilation is a pain

Pepper is cool, C is not:

- serialisation/deserialization
- re-creation and support of data model
- manual process of translating the Kontract into C

Bright future of compilation

ZKP DSL

e.g. xJSnark, Zokrates

Limited and low-level

Still 2 (or more) codebases for contracts

Or VM for all target platforms

OR

Kotlin

Or its subset (like R3's DJVM)

Multiplatform compilation to JVM, JS, Native, WASM

You can also do DSL on top of it

Good part of compiler's boilerplate is already there

?

Agenda

Corda privacy model

Security/privacy tradeoff

Denial-of-State attack

Zero-Knowledge Notary

Pains and insights

Thank you!