Cryptography for Tezos

Marc Beunardeau, Vincent Herbert, Anne-Laure Schmitt, Marco Stronati, Danny Willems – Nomadic Labs

Journée scientifique Inria - Nomadic Labs

Paris, September 21, 2020

Context

Current work

- Privacy
- Aggregatable signatures
- Multiparty randomness generation
- Vote

Privacy, trust and decentralisation

- Blockchains enable the decentralisation of an open network
- A necessary condition for trust is that everyone can check the correctness of transactions

$\Rightarrow$ Privacy has been sacrificed

Cryptography to the rescue

- Advanced cryptographic techniques allow controlled interaction from a third party with 'hidden data'
- Theoretical solutions have been there for long
- Very rich state of the art $\Rightarrow$ no research needed here
- Still not widely used in industry (starts to be in blockchains)

Zero-knowledge proofs (zkp)

- Let $P(x, w)$ a reasonably sized ($\approx$ 1 000 000 gates) boolean circuit
- Given x the prover can shows that he knows w such that $P(x, w)$
- Nothing is revealed about w

$P(\cdot, \cdot)$ is called the statement

Can be non interactive with very small proof size and verification time

Commitment

A commitment scheme is a pair of polynomial algorithm $Commit(x, r) = c$ and $Reveal(x, r, c) = b$ that respects:

- Hiding: $Commit(x, r)$ does not reveal x
- Binding: One cannot find x, r, x', r' with $x \neq x'$st $Reveal(x', r', Commit(x, r)) = \top$

General principle to use zkp in blockchains/ Sapling protocol

- The blockchain is a set of committed assets
- Create a new set resulting of a valid transaction
- Prove in zk that the you know the opening of some commitments such that the changed part of the new set is valid wrt the old one

Other primitives are used when general zkp are not necessary:

- Hommomorphic commitments
- Randomisable signatures

asset

An asset is anything which has an owner and which creation, destruction and transfer is controlled by a smart contract

What we have today in Tezos

Functionality

- Integration of Z-Cash's protocol (Sapling) : privacy preserving transfer of an asset
- Can publicly create and destroy an asset
- Can transfer it in a privacy preserving fashion without any control
- Cannot exchange two assets

Efficiency

- Forging a transaction: ≈ 2 s
- Verifying a transaction: ≈ 20 ms

What we want

- Prove additional properties on a transaction (eg. amount $<$ 100)
- Prove that the transaction is done by someone in a white list (without revealing himself)
- Prove that some encrypted on chain information reveals our transaction (so that the key can be asked for by authorities)
- Exchange different assets

Challenges

Trusted setup

- Non interactive zkp need public parameters $pp = Setup(r)$
- The setup is said to be trusted if knowing r allows to break the system
- The setup is said to be universal if one setup can be used for different statement
- Sapling's nizk has a non universal trusted setup (for better efficiency)

Since we plan to add several (small) statements, we would like to have a universal or trustless setup

Plonk

- Pairing based, efficient, universal trusted setup nizk
- We already have a pairing implementation and the trusted setup from Sapling
- Security of pairing might evovle in the near future

We are implementing the rest in Ocaml and independant library for aritmhetic and polynomials
Very modular construction for which we can leverage Ocamls module system

IOP's based nizk

- Trustless setup
- Simpler cryptography (hash functions)
- Post quantum
- Not as efficient
- More complex for us

Exploration of other libraries, can't find for now satisfying implementations

Statements in R1CS

- Statement are expressed in a 'weird' circuit form working in a finite field (constrained by the nizk)
- Our statements contains cryptographic primitives (eg. commitments to open them 'in zk')
- We have one elliptic curve in R1CS for hashes and commitment from Sapling (JubJub)

We would like to explore that by having a PhD student on this topic
We would like different primitives (eg. encryption)
Lots of optimisations to explore
We would like to be able to switch finite fields and circuit languages

R1CS semantic verification

A project will be done (by the Kestrel Institute) to functionally verify the R1CS JubJub implementation

We hope the reuse it for new primitives implementations

Proofs of protocol's security

- We have a pen and paper proof for Sapling
- We could write more proofs for 'augmented' Sapling with more properties

A project (by INRIA) will be done to make a symbolic verification of some of Sapling 's properties
We hope to reuse that for some versions of 'augmented Sapling'

Questions