Blockchains & Cryptocurrencies

Anonymity - III

Image from cryptonomad.info

Instructor: Abhishek Jain

Johns Hopkins University - Spring 2021

Agenda

- **Last Time:**
  - Confidential Transactions for hiding transaction values
  - Homomorphic Commitments and ZK Proofs
  - ZeroCoin
- **Today:** Complete discussion on anonymity (hopefully)
  - ZK Proofs example for Sudoku
  - ZK-SNARGs and ZCash Cryptocurrency

Recap: Zerocoin (MGGR14)

- Proposed as an extension to Bitcoin in 2014
  - Requires changes to the Bitcoin consensus protocol
- **Main Advantage**: Huge anonymity set (potentially, all transactions)
- Key Tools: Commitments, accumulators and ZK Proofs

Recap: Zero-Knowledge Proofs for NP

- Powerful Theorem by Goldreich-Micali-Wigderson from 1980s:
  Anything in NP can be proven in zero-knowledge

- How do we show this?
  - Design a ZK proof for an "NP-Complete" Language (e.g., CircuitSAT)

ZK Proof for Sudoku Puzzles

- On the Whiteboard!
- Generalized nxn Sudoku is NP-Complete

Limitations of Zerocoin

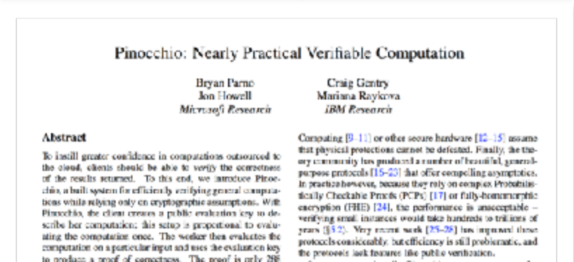
- Proofs are big
- Must convert “zerocoins” to “bitcoins” to spend them

Zcash

Ben-Sasson, Chiesa, Garman, Green, Miers, Tromer, Virza
(Oakland '14)

A better tool

- Better, smaller arguments of knowledge:
  - **S**uccinct **N**on-Interactive **AR**guments of **K**nowledge (zkSNARKs) (Bitansky *et al*., Parno *et al.*, Ben-Sasson *et al.)*
  - 288 byte proof for arbitrary-sized arithmetic circuits
  - And there are C compilers!



Pinocchio: Nearly Practical Verifiable Computation

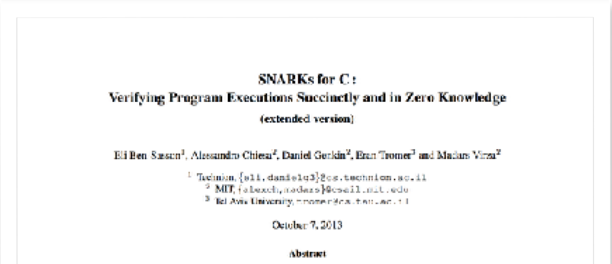
Bryan Parno
Jon Howell
Microsoft Research

Craig Gentry
Mariana Raykova
IBM Research

Abstract

To instill greater confidence in computations outsourced to the cloud, clients should be able to verify the correctness of the results returned. To this end, we introduce Pinocchio, a built system for efficiently verifying general computations while relying only on cryptographic assumptions. With Pinocchio, the client creates a public evaluation key to describe her computation; this setup is proportional to evaluating the computation once. The worker then evaluates the computation on a particular input and uses the evaluation key to produce a proof of correctness. The proof is only 288

Computing [9–11] or other secure hardware [12–15] assume that physical protections cannot be defeated. Finally, the theory community has produced a number of beautiful, general-purpose protocols [15–23] that offer compelling asymptotics. In practice however, because they rely on complex Probabilistically Checkable Proofs (PCPs) [17] or fully-homomorphic encryption (FHE) [24], the performance is unacceptable – verifying small instances would take hundreds to trillions of years (§5.2). Very recent work [25–28] has improved these protocols considerably, but efficiency is still problematic, and the protocols lack features like public verification.



SNARKs for C:
Verifying Program Executions Succinctly and in Zero Knowledge
(extended version)

Eli Ben-Sasson[1], Alessandro Chiesa[2], Daniel Genkin[2], Eran Tromer[3] and Madars Virza[2]

[1] Technion, {eli,danielg3}@cs.technion.ac.il
[2] MIT, {alexch,madars}@csail.mit.edu
[3] Tel Aviv University, tromer@cs.tau.ac.il

October 7, 2013

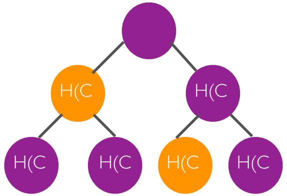
Abstract

ZK-SNARKs

- **SNARKs**: Succinct, non-interactive arguments of knowledge
  - Succinctness: proof size (and verification time) is independent of witness size
  - Non-interactive: One message (very convenient for blockchains!)
  - Argument of Knowledge: Soundness against *efficient* cheating provers*
- **ZK-SNARGs**: SNARGs that are also zero knowledge!
  - Fiat-Shamir approach: Start from succinct "public-coin" interactive ZK and make them non-interactive using Hash Functions
  - Other approaches also known. Extensive area of research!

ZCash Design

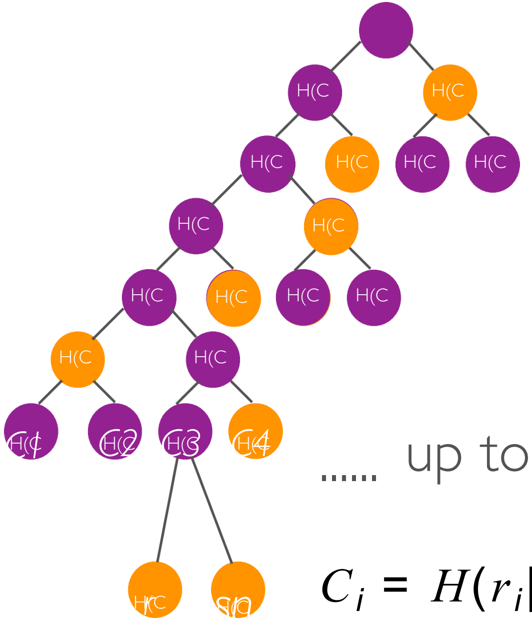
- Somewhat similar to ZeroCoin, but with important differences:
  - Hash functions for commitments
  - Hash trees for accumulator
  - SHA-256 as the hash function
  - Circuits for the proofs hand-optimized



C_{SHA256} (circuit for SHA256)	Gate count
Message schedule	8 032
All rounds	21 632
1 round (of 64)	338
Finalize	288
Total	29 952

Figure 2: Size of circuit C_{SHA256} for SHA256.

ZCash tree

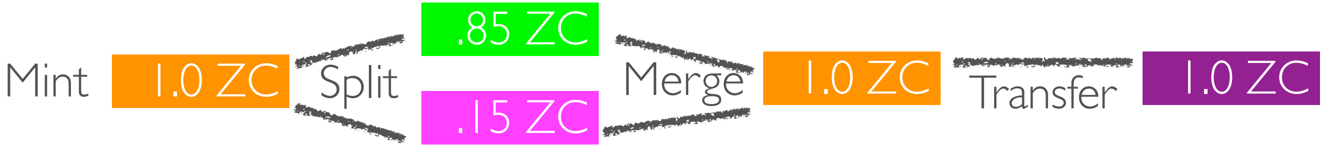


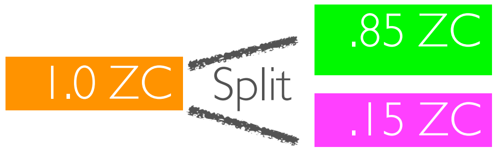
...... up to 2^{64} coins

$$C_i = H(r_i || sn)$$

But wait a second...

- If the proofs are general & efficient, why do we need Bitcoin anymore?
  - Let's add hidden values to the coin: $C_i = H(r_i || v || sn)$
  - Create transactions to split/merge coins
  - Allow payments (from Alice to Bob) that don't reveal value
    - Pay to individuals, pay to address





To split a coin:

1. "Spend" the input coin (by revealing its serial number)
2. "Mint" two new coins
3. Prove that the new coins total to the value of the first coin

<u>To merge two coins</u>:

- "Spend" the input coins (by revealing their serial numbers)
- "Mint" a new coin
- Prove that the old coins total to the value of the new coin

1.0 ZC $\xrightarrow{\text{Transfer}}$ 1.0 ZC

<u>To pay a coin</u>:

1. Transfer the coin secrets to the target user
2. Embed the recipient's 'address' $A = H(x)$
3. User must prove knowledge of x to redeem

Some Finer Points

- Regular ZK-SNARK only guarantees Soundness & ZK
- ZK-SNARKs can be "malleable" (i.e., one could potentially transform a ZK-SNARK for one statement into one for another statement without a valid witness)
- ZCash uses other tools to "build" non-malleability

Result: Zerocash

- A fully untraceable, divisible electronic cash system
  - Coins are anonymous starting from Coinbase transaction
  - Coins can be split/joined, paid and revealed
  - The only place where coin values need be public is when one offers transaction fees

So what's the catch?

- The public parameters are quite large
  - **About 1.2 GB** (a non-trivial portion of total blockchain size)
  - Significant follow-up research on ZK-SNARKs with small public parameters
- Public parameters must be generated by a trusted party (or by running a "secure multiparty computation" protocol)
  - A party who knows a *trapdoor* can forge proofs
  - But cannot de-anonymize transactions
  - Subsequent research on this topic for stronger guarantees

Anonymous Credentials

- Due to Chaum *et al.*
  - Allow us to prove statements about identity <u>without revealing it</u>
    - E.g., "I am an authorized user", "I am a subscriber"
    - Example: TPM anonymous attestation
  - Usually requires a trusted anonymous credential issuer

Anonymous Credentials

- Observation: e-Cash is just a form of anonymous credential
  - By adding similar commitments to the identities/attributes we can prove statements about our identity
  - No trusted credential issuer
  - Can use this to, e.g., implement decentralized anonymous reputation systems

Questions to think over…

- What are the ethics of anonymous cryptocurrencies?
  - Can we distinguish "good" use from "bad" use?
  - Do the "good" uses outweigh potential "bad" uses?
- Can anonymity guarantees be an "add-on" (instead of building new systems from scratch)?
- What about DeFi, where we may need "selective" anonymity to comply with regulations (KYC, etc) ?