Notes of the 2nd ZKProof Workshop

December 21, 2019

editors@zkproof.org

The 2nd ZKProof Workshop took place
on April 10–12, 2019, in Berkeley CA, USA

The initial draft was open to public comments since July 31, 2019. No comment was received.

Changes in the final version (December 21, 2019): revised cover page; this page; no line numbers.

Proposed citation: ZKproof. *Notes of the 2nd ZKProof Workshop*. Ed. by D. Benarroch, L.T.A.N. Brandão, E. Tromer. Pub. by zkproof.org, December 2019

Editors' note

This document is a compilation of notes about the 2nd ZKProof Workshop. It serves as a descriptive memory of the content discussed in the workshop, it includes hyperlinks and web addresses to related external content, and it transcribes informal notes scribed during (and parsed after) the discussion sessions held at the workshop. The content included here is mostly based on information publicly available online, namely in the following online resources:

- ZKProof main website: https://zkproof.org
- 2nd ZKProof Workshop website: https://zkproof.org/workshop2/main.html
- ZKProof community forum: https://community.zkproof.org/
- ZKProof GitHub repository: https://github.com/zkpstandard/
- Youtube channel: https://www.youtube.com/channel/UC79GUI9SBNnfmJOQyHDrrPQ
- List of slide-decks: https://community.zkproof.org/t/zkproof-2019-speaker-slides/167

Several of the hyperlinks in this document were produced by the Internet Archive Wayback Machine. Some elements were lightly edited for readability, or omitted when redundant. Text marked by double square brackets ("[[...]]") represents annotations by the editors.

This compilation is within the scope of the "ZKProof editorial process" proposed in the workshop. A main goal is to help distinguish "workshop notes" (in this document, expected to contain informal notes of open discussions taken during the workshop) from other more formal content to be put together, such as the "ZKProof Community Reference" document that will conglomerate technical material about zero knowledge proofs.

This document includes the following material related to the workshop:

- Logistics: logistic information about the workshop, including the program schedule
- Talks and Panel: speaker, title and abstract of each presentation, and external links
- Breakouts: informal notes (retrieved from the ZKProof forum) about each "breakout" session
- Proposals: informal notes (retrieved from the ZKProof forum) about each "proposal" session

The ZKProof editors (editors@zkproof.org):

Daniel Benarroch, Luís Brandão and Eran Tromer.

July 31, 2019

Acknowledgments

The material in this compilation has multiple sources, acknowledged throughout the document. The work and support of all participants, committees and sponsors is appreciated. We may be unable to acknowledge everyone, but point out at least the following contributions:

- **Speakers:** For each talk or panel, we include the available title and abstract and identify the speaker(s), which is/are assumed to be the corresponding author(s), and/or to have obtained proper permission to present the material.

- **Moderators, scribes, contributors:** There were multiple open discussions ("breakouts" and "proposals"), moderated by moderator(s) identified in the workshop schedule. Knowingly to the participants, in each session there were volunteered scribe(s) taking (informal) notes of the discussion, for posterior publication (e.g., in the ZKProof community forum and in this compilation), possibly associating the name of participants to the scribed comments.

- **Organization, oversight, sponsoring:** The workshop was organized by two "organizers" (chairs). The organization had the oversight of the ZKProof steering committee, composed of eleven members, who provided advice and selected the presentations. A proposals committee of twelve members evaluated and selected submitted "proposals" for presentation at the workshop. The workshop was sponsored by thirteen entities (company or foundation). All these members are identified in the "Logistics" section of this compilation.

- **Editors:** This compilation was produced within the scope of a proposed (and ongoing) editorial process that intends to gather reference material about ZKProof. This document was compiled by the current ZKProof editors.

Figure 1: Group photo. Taken on April 12, 2019, during the 2nd ZKProof Workshop at Berkeley Marina, Berleley CA, USA. Permission to reuse this photo is restricted to the ZKProof context.

Table of Contents

List of Figures

1 Logistics

1.1 Workshop organization, oversight and sponsoring

The 2nd ZKProof Workshop took place on:

- Dates: Wednesday April 10 through Friday April 12, 2019
- Location: DoubleTree by Hilton Hotel, Berkeley California, USA

The workshop was publicly announced in advance, for example in the ZKProof website (2018-Nov-14) and in a post in the ZKProof community forum (2019-Feb-11).

The following teams were involved in organizing this 2nd ZKProof workshop:

- **Steering Committee:** Dan Boneh, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Jens Groth, Yuval Ishai, Yael Kalai, Elaine Shi, Eran Tromer, Muthu Venkitasubramaniam and Aviv Zohar.
- **Proposals Committee:** Daniel Benarroch, Sean Bowe, Ran Canetti, Alessandro Chiesa, Jens Groth, Yael Kalai, Mariana Raykova, abhi shelat, Justin Thaler, Eran Tromer, Muthu Venkitasubramaniam and Aviv Zohar.
- **Workshop Organizers:** Daniel Benarroch and Dan Lipiec.

The Workshop was sponsored by:

- **Platinum sponsors:** QEDIT; Zcash Foundation
- **Gold sponsors:** Vmware; Beam; Deloitte
- **Silver sponsors:** CPIIS; Clearmatics; Ethereum Foundation; Protocol Labs; Stratumn; ING
- **Bronze sponsors:** HACERA
- **Special contributors:** Microsoft

The schedule presented below is based on the general schedule obtained from the ZKProof webpage and also integrates the parallel sessions of breakout discussions and proposal presentations. Compared with the original schedule, this also updates the notation (proposed for the editorial process) "community standards" to "Presentation proposal".

1.2 Schedule of Day 1 — Wednesday, April 10, 2019

08:00	**Registration & Light Breakfast**

09:00	**Welcoming Remarks** Jonathan Rouach & Ruben Arnold (QEDIT) and Josh Cincinnati (Zcash Foundation)
09:10	Talk 1: ZKP for Audits of Unsolicited Consumer Communication Hitarshi Buch and Joshua Satten, Wipro
09:35	Talk 2: Applications of Zero Knowledge Proofs in the Banking Industry Eduardo Moraes, ING
10:00	Talk 3: Bringing ZKP to Traditional Industries, Physical World Use-cases Shiri Lemel, QEDIT
10:25	Talk 4: Privacy Pass: a Lightweight Zero Knowledge Protocol Designed for the Web Nick Sullivan, Cloudflare
10:50	**COFFEE BREAK**
11:10	Talk 5: Tooling Infrastructure for Zero-Knowledge Proofs Henry de Valence, Zcash Foundation
11:35	Talk 6: An R1CS based Implementation of Bulletproofs Cathie Yun, Interstellar
12:00	Talk 7: Fragile Nonce Selection and ZKPs as a Solution Andrew Poelstra, Blockstream
12:25	Talk 8: Notes from the SNARKonomicon: Techniques for Writing SNARK Programs Izaak Mekler, O(1) Labs
12:50	Talk 9: Zero Knowledge Proofs and Self-Sovereign Identity Jordi Baylina, Iden3
13:15	**LUNCH**
14:30	Talk 10: zk-SHARKs: Combining Succinct Verification and Public-Coin Setup Madars Virza, MIT
14:55	Talk 11: LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs Dario Fiore, IMDEA
15:20	Talk 12: Sonic: zk-SNARKs from Linear-Size Universal and Updatable SRS Sean Bowe, Electric Coin Company
15:45	Talk 13: DIZK: a Distributed Zero-Knowledge Proof System Howard Wu, Berkeley & Dekrypt Capital
16:10	**COFFEE BREAK**
16:30	Talk 14: Enterprise Features for Confidential Asset Transfers Ori Wallenstein, QEDIT
16:55	Talk 15: Zether: Towards Privacy in a Smart Contract World Shashank Agrawal, Visa Research
17:20	Talk 16: Succinct Proofs in Ethereum Barry Whitehat, Ethereum Foundation
17:45	Talk 17: Aurora: Transparent Succinct Arguments for R1CS Nick Spooner, UC Berkeley

18:10	**END OF DAY**
18:40	**ZKProof Reception at Venue**

1.3 Schedule of Day 2 — Thursday, April 11, 2019

08:00	**Coffee & Light Breakfast**
09:00	Introducing the Standards Workshop Steering Committee
09:10	Talk 18: Public Accountability vs. Secret Laws: Can They Coexist? Shafi Goldwasser, UC Berkeley, MIT and Weizmann
09:40	Talk 19: Efficient Zero-Knowledge Protocols: The Modular Approach Yuval Ishai, Technion
10:20	Talk 20: Privacy-enhancing Cryptography at NIST Rene Peralta, NIST
11:00	**COFFEE BREAK**
11:25	Talk 21: ZKProof Proceedings: Review & Process Moderators: Daniel Benarroch, Luís Brandão & Eran Tromer
12:05	Breakout 1: ZKProof Proceedings Moderators: Daniel Benarroch & Luís Brandão Breakout 2: Recursive Composition of SNARKs Leads: Izaak Meckler & Eran Tromer Breakout 3: Structure Reference String Generation Leads: Sean Bowe & Mary Maller
12:45	**LUNCH**
14:00	Panel 1: Panel: Zero Knowledge in the Enterprise Moderator: Jonathan Rouach (QEDIT) Panelists:[a] Carlos Kuchkovsky (BBVA), Yael Kalai (Microsoft Research), Mike Hearn (R3) and Jonathan Levi (HACERA)
15:00	Talk 22: Bilinear Pairings based Zero-Knowledge Proofs Jens Groth, DFINITY
15:40	Talk 23: MPC-in-the-Head based Zero-Knowledge Proofs Amit Sahai, UCLA
16:20	**COFFEE BREAK**

[a] David Archer (Galois) was initially scheduled to participate but was not able to attend.

16:45	**Proposals 1&2: Interoperability of Zero-Knowledge Systems** Moderators: Sean Bowe and Abhi Shelat
	Breakout 4: Security Assumptions Underpinning Zero-Knowledge Proofs Leads: Jens Groth & Yuval Ishai
	Breakout 5: Formal Verification for ZK Systems Leads: Izaak Meckler
18:00	**END OF DAY**

1.4 Schedule for Day 3 — Friday, April 12, 2019

08:00	**Coffee & Light Breakfast**
09:00	Talk 24: GKR based Zero-Knowledge Proofs Yael Kalai, Microsoft Research
09:40	Talk 25: IOP based Zero-Knowledge Proofs Alessandro Chiesa, UC Berkeley
10:20	Talk 26: Discrete Log based Zero-Knowledge Proofs Dan Boneh, Stanford
11:00	**COFFEE BREAK**
11:25	**Proposal 3: Commit-and-Prove Functionality** Moderators: Jens Groth, Yael Kalai, Mariana Raykova & Muthu Venkitasubramaniam
	Breakout 8: Domain-Specific Languages for ZK Leads: Stefan Deml & Ahmed Kosba
12:45	**LUNCH**
14:00	Talk 27: From Public-Key Cryptography to PKI: Reflections on Standardizing the RSA Algorithm Jim Bidzos and Burt Kaliski, Verisign
15:00	Talk 28: Zero Knowledge Ideal Functionality Muthu Venkitasubramaniam, University of Rochester and Ligero
15:40	**Open Discussion about ZKProof** Moderators: Aviv Zohar & Daniel Benarroch
16:20	**COFFEE BREAK**
16:45	**Proposal 4: Deterministic Generation of Elliptic Curves for ZK Systems** Moderators: Sean Bowe & Alessandro Chiesa
	Breakout 7: Interactive Zero Knowledge Leads: Yael Kalai & Justin Thaler
	Breakout 8: ZK Protocol Security Analysis & Proofs of Correctness Leads: Ariel Gabizon & Ran Canetti

18:00	**Closing Remarks**
18:05	**END OF 2ND ZKPROOF WORKSHOP** See you next year!

1.5 ZKProof charter and code

The booklet of the event also included notice to the ZKProof Charter and Code of Conduct.

1.5.1 ZKProof Charter

The goal of the ZKProof Standardardization effort is to advance the use of Zero-Knowledge Proof technology by bringing together experts from industry and academia and producing Community Standards for the technology.

We set the following guiding principles:[1]

- We seek to represent all aspects of the technology, research and community in an inclusive manner.
- Our goal is to reach consensus where possible, and to properly represent conflicting views where consensus was not reached.
- As an open initiative, we wish to communicate our results to the industry, the media and to the general public, with a goal of making all voices in the event heard.
  - Participants in the event might be photographed or filmed.
  - We encourage you to tweet, blog and share with the hashtag #ZKProof. - Our official twitter handle is @ZKProof.

All participants, speakers and sponsors of the ZKProof Standards Workshop shall adhere to the following code of conduct to ensure a safe and productive environment for everybody*:

1.5.2 ZKProof Code of Conduct

At the workshop, you agree to:

- Respect the boundaries of other attendees.
- Respect the opinions of other attendees even if you are not in agreement with them.
- Avoid aggressively pushing your own services, products or causes.
- Respect confidentiality requests by participants.
- Look out for one another.

These behaviors don't belong at the workshop:

[1] Editor's note: Compared with the charter published in the ZKProof webpage, the booklet was missing the item "*The initiative is aimed at producing documents that are open for all and free to use. As an open initiative, all content issued from the ZKProof Standards Workshop is under Creative Commons Attribution 4.0 International license.*" This is resolved by having these notes also placed under CC Attribution 4.0.

- Invasion of privacy
- Being disruptive, drinking excessively, stalking, following or threatening anyone.
- Abuse of power (including abuses related to position, wealth, race or gender).
- Homophobia, racism or behavior that discriminates against a group or class of people.
- Sexual harassment of any kind, including unwelcome sexual attention and inappropriate physical contact.

If you have any question, please refer to contact@zkproof.org

*This code of conduct is adapted from that of TEDx

2 Talks and Panel

For this workshop, there was a call for contributed talks, whose submissions were reviewed by the Steering Committee and workshop organizers. Five of the talks during the first day were talks accepted through submissions. The rest of the talks were all by invited speakers.

This section lists the titles, abstracts and speakers names, of the talks and panel presented during the three days of the workshop. The information is based on the workshop webpage.

2.1 Day 1

2.1.1 Talk 1: ZKP for audits of Unsolicited Consumer Communication

Speakers: Hitarshi Buch & Harihara Natarajan, Wipro
Time: 2019-April-10, 09:00
Hyperlinks: Youtube video
Abstract: The menace of Unsolicited Commercial Communication (text messages / voice calls) is growing day by day as phone subscribers keep receiving unwanted messages and calls. Telecom regulatory bodies have now mandated that this kind of communication should be filtered based on subscriber's preferences & consent. These preferences are managed by the respective telecom service providers (TSP) like Vodafone, AT&T; etc. Exposing this data would result in the subscriber being subjected to more spam and also targeted advertising etc. Therefore the TSP needs to provide evidence that messages are segregated based on subscriber's preferences so that a level playing field is maintained, without compromising the user preference data. Our solution leverages a combination of ZKP and blockchain to enable the prover (TSP) to construct a proof which can be used by the verifier (Auditor) to validate the proof of delivery / exclusion . We will demonstrate a prototype using zk-SNARKs for on-demand proof generation and describe a technique leveraging Merkle trees for pre-computed proofs that can be applied to this real-life use case. We will also discuss the merits & demerits of each approach and the practical limitations.

2.1.2 Talk 2: Applications of Zero Knowledge Proofs in the Banking Industry

Speaker: Eduardo Moraes, ING
Time: 2019-April-10, 09:35
Hyperlinks: Youtube video, slide deck
Abstract: In this talk we will present the implementation of ZKPs at ING. We will show interesting use cases in the banking industry, describing findings and challenges that we faced during this effort.

2.1.3 Talk 3: Bringing ZKP to Traditional Industries, Physical World Use-cases

Speaker: Shiri Lemel, QEDIT
Time: 2019-April-10, 10:00
Hyperlinks: Youtube video

Abstract: Taking Zero Knowledge into traditional industries requires an end-to-end overview of the use-case and deep understanding of processes and flows. In this talk we will review use-cases in supply chain management and user consent systems to understand the role of ZKP in enabling digitalisation, and how QEDIT is addressing the needs of these industries in our Asset Transfer solution.

2.1.4 Talk 4: Privacy Pass: a Lightweight Zero Knowledge Protocol Designed for the Web

Speaker: Nick Sullivan, Cloudflare

Time: 2019-April-10, 10:25

Hyperlinks: Youtube video

Abstract: Privacy Pass is a project launched in 2017 to help make solving CAPTCHAs online less painful using zero-knowledge cryptography. The core of Privacy Pass is a 1-RTT cryptographic protocol (based on an implementation of an oblivious pseudorandom function) that allows users to receive a significant amount of anonymous tokens in exchange for solving a challenge. These tokens can be exchanged in the future for access to services without having to interact with a challenge and without the service knowing which specific challenge was originally solved. Privacy Pass is now in use by over a hundred thousand monthly active users in the form of the Privacy Pass browser extension for Chrome and Firefox. In this talk I'll explore both the mathematical underpinnings of this project and its future directions.

2.1.5 Talk 5: Tooling Infrastructure for Zero-Knowledge Proofs

Speaker: Henry de Valence, Zcash Foundation

Time: 2019-April-10, 10:50

Hyperlinks: Youtube video

Abstract: Developing and deploying zero-knowledge proofs into production systems without catastrophic failures along the way is a significant challenge. This talk will discuss tooling for zero-knowledge proofs, aiming at reducing the time and effort to produce high-quality implementations, as well as some challenges for standardization.

2.1.6 Talk 6: An R1CS based Implementation of Bulletproofs

Speaker: Cathie Yun, Interstellar

Time: 2019-April-10, 11:35

Hyperlinks: Youtube video, slide deck

Abstract: I will discuss our Bulletproofs constraint system implementation, explaining the R1CS API and an extension to Bulletproofs which takes advantage of the lack of setup to select a circuit from a family parameterized by Fiat-Shamir challenges. I will illustrate these concepts by walking through the construction of a shuffle gadget, which proves that one list of Pedersen commitments is a permutation of another. I'll also talk about the multiparty computation protocol for interactive aggregation, and how we encode the protocol states into the Rust type system, ensuring that any compilable instantiation of the protocol executes in the correct order, with no possibility of replay

attacks. Lastly, I'll discuss the applications we are building using Bulletproofs R1CS proofs - a confidential assets protocol (Cloak), and a confidential smart contract language (ZkVM).

2.1.7 Talk 7: Fragile Nonce Selection and ZKPs as a Solution

Speaker: Andrew Poelstra, Blockstream

Time: 2019-April-10, 12:00

Hyperlinks: Youtube video

Abstract: Elliptic-curve-based signature schemes such as Schnorr and ECDSA require fresh uniform randomness for every signature. Deviations from uniform, even by fractions of a bit, can be exploited by lattice-based attacks to extract secret key material. On the other hand, uniform randomness is difficult to reliably obtain in many constrained environments, such as virtual machines or cheap hardware devices. The traditional solution to this, for signatures, is to derive randomness deterministically by hashing a secret key and message to be signed (i.e. using RFC6979). The message and nonce determine the Fiat-Shamir challenge, ensuring that distinct signatures will always have independently uniform randomness. However, in the multiparty setting, the Fiat-Shamir challenge is now determined by input from all participants. This means that individual participants can no longer ensure unique randomness by hashing data known at nonce-generation time. Safe nonce generation requires some unique input, such as a monotonic counter, or direct access to uniform randomness; neither of which are simple to obtain in a robust manner. If all parties were guaranteed to generate their nonces deterministically, we would again have the simple situation where the message and signing keys completely determined the Fiat-Shamir challenge, again allowing the simple solution of hashing the message with some secret. Such a guarantee is hard to obtain directly, but easy (in principle) to obtain by modifying the signing protocol to require all signers provide zero-knowledge proofs of their nonce generation. We discuss challenges in such an approach, including the difficulties with provable security and the limitations of using popular zero-knowledge proof schemes in resource-constrained environment such as hardware signing keys.

2.1.8 Talk 8: Notes from the SNARKonomicon: Techniques for Writing SNARK Programs

Speaker: Izaak Mekler, O(1) Labs

Time: 2019-April-10, 12:25

Hyperlinks: Youtube video

Abstract: Writing programs for SNARKs is, for now, somewhat of an obscure art. This showcase will cover the techniques learned in the process of writing the large SNARK programs used in Coda, as a way to spread this knowledge to the community at large. There are several kinds of knowledge that comprise "how to program SNARKs"?. One is algorithmic techniques for efficiently encoding certain kinds of functions. Another is ways of thinking or mindsets that are useful in effectively structuring large non-deterministic programs. The primary goal of this showcase is to widely disseminate both the algorithmic techniques and the ways of thinking that we have found to be useful. The showcase will also acquaint the audience with concrete, real-world examples of different techniques, which they can use going forward as a "cook-book"?.

2.1.9 Talk 9: Zero Knowledge Proofs and Self-Sovereign Identity

Speaker: Jordi Baylina, Iden3

Time: 2019-April-10, 12:50

Hyperlinks: Youtube video, slide deck

Abstract: We will introduce iden3 project with a quick overview of our identity technology including the circom compiler and snarkJS library. Then we will talk about a proposal of how such ZK identity could be standardized with specific modules, protocols and data structures. Then we will talk about a proposal of a self sovereign identity standard with the ZK technology in mind. We will also explain the advantages and possibilities of such a system and will give an overview about the specific modules, protocols and data structures.

2.1.10 Talk 10: zk-SHARKs: Combining Succinct Verification and Public-Coin Setup

Speaker: Madars Virza, MIT

Time: 2019-April-10, 14:30

Hyperlinks: Youtube video, slide deck

Abstract: We investigate the goal of deploying efficient non-interactive zero-knowledge proofs for moderately complex statements. Motivated by the desire to deploy ZK proofs in real world distributed ledger systems. we seek ZK proof systems that have fast verification time, short proof size, and no trusted setup. However, in terms of concrete parameters, we can achieve only two out of the three. In particular, it was not known how to achieve all of the following for million-gate circuits: * fast (milliseconds) verifier; * short proofs (couple kilobyte-long); and * does not reply on a structured reference string (e.g., trusted setup) for soundness or zero-knowledge. We propose a new form of proof systems: zk-SHARK (zero-knowledge Succinct Hybrid ARguments of Knowledge). These combine the fast verification of zk-SNARKs with the no-trusted-setup of some non-succinct NIZKs. A zk-SHARK has two verification modes: a prudent mode (relying on a uniform random string), and an optimistic mode (relying on a structured reference string). Crucially, even complete corruption of the setup used by optimistic verification does not invalidate the prudent verification. Moreover, old “prudent proofs”? can be re-accelerated with a new optimistic mode setup (in case the old setup becomes unconvincing or compromised). We propose a construction of zk-SHARKs, tailored for efficiency of both modes: it is competitive with both state-of-the-art SNARKs (in terms of prover and verifier time) and NIZKs (in terms of proof size). Our zk-SHARK construction acieves all three properties outlined above. We also discuss the applicability to transaction and block verification in blockchain applications. Joint work with Mariana Raykova and Eran Tromer.

2.1.11 Talk 11: LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs

Speaker: Dario Fiore, IMDEA

Time: 2019-April-10, 14:55

Hyperlinks: Youtube video, slide deck

Abstract: We study the problem of building SNARKs modularly by linking small specialized ”proof gadgets” SNARKs in a lightweight manner. Our motivation is both theoretical and practi-

cal. On the theoretical side, modular SNARK designs would be flexible and reusable. In practice, specialized SNARKs have the potential to be more efficient than general-purpose schemes, on which most existing works have focused. If a computation naturally presents different "components" (e.g. one arithmetic circuit and one boolean circuit), a general-purpose scheme would homogenize them to a single representation with a subsequent cost in performance. Through a modular approach one could instead exploit the nuances of a computation and choose the best gadget for each component. In this talk I will present LegoSNARK, a "toolbox" (or framework) for commit-and-prove zkSNARKs (CP-SNARKs) that includes: (1) General composition tools: build new CP-SNARKs from proof gadgets for basic relations, simply. (2) A "lifting" tool: add commit-and-prove capabilities to a broad class of existing zkSNARKs, efficiently. This makes them interoperable (linkable) within the same computation. For example, one QAP-based scheme can be used prove one component; another GKR-based scheme can be used to prove another. (3) A collection of succinct proof gadgets for a variety of relations. Additionally, through our framework and gadgets, we are able to obtain new succinct proof systems. Notably: – LegoGro16, a commit-and-prove version of Groth16 zkSNARK, that operates over data committed with a classical Pedersen vector commitment, and that achieves a 5000x speedup in proving time. – LegoUAC, a pairing-based SNARK for arithmetic circuits that has a universal, circuit-independent, CRS, and proving time linear in the number of circuit gates (vs. the recent scheme of Groth et al. (CRYPTO'18) with quadratic CRS and quasilinear proving time). This is a joint work with Matteo Campanelli and Anais Querol.

2.1.12 Talk 12: Sonic: zkSNARKs from Linear-Size Universal and Updatable SRS

Speaker: Sean Bowe, Electric Coin Company

Time: 2019-April-10, 15:20

Hyperlinks: Youtube video

Abstract: Ever since their introduction, zero-knowledge proofs have become an important tool for addressing privacy and scalability concerns in a variety of applications. In many systems each client downloads and verifies every new proof, and so proofs must be small and cheap to verify. The most practical schemes require either a trusted setup, as in (pre-processing) zk-SNARKs, or verification complexity that scales linearly with the complexity of the relation, as in Bulletproofs. The structured reference strings required by most zk-SNARK schemes can be constructed with multi-party computation protocols, but the resulting parameters are specific to an individual relation. Groth et al. discovered a zk-SNARK protocol with a universal and updatable structured reference string, but the string scales quadratically in the size of the supported relations. Here we describe a zero-knowledge SNARK, Sonic, which supports a universal and continually updatable structured reference string that scales linearly in size. Sonic proofs are constant size, and in the batch verification context the marginal cost of verification is comparable with the most efficient SNARKs in the literature. We also describe a generally useful technique in which untrusted "helpers"? can compute advice that allows batches of proofs to be verified more efficiently.

2.1.13 Talk 13: DIZK: a Distributed Zero Knowledge Proof System

Speaker: Howard Wu, UC Berkeley and Dekrypt Capital

Time: 2019-April-10, 15:45

Hyperlinks: Youtube video

Abstract: Recently there has been much academic and industrial interest in practical implementations of *zero knowledge proofs*. These techniques allow a party to *prove* to another party that a given statement is true without revealing any additional information. In a Bitcoin-like system, this allows a payer to prove validity of a payment without disclosing the payment's details. Unfortunately, the existing systems for generating such proofs are very expensive, especially in terms of memory overhead. Worse yet, these systems are "monolithic", so they are limited by the memory resources of a single machine. This severely limits their practical applicability. We describe DIZK, a system that *distributes* the generation of a zero knowledge proof across machines in a compute cluster. Using a set of new techniques, we show that DIZK scales to computations of up to billions of logical gates (100x larger than prior art) at a cost of 10μs per gate (100x faster than prior art). We then use DIZK to study various security applications.

2.1.14 Talk 14: Enterprise Features for Confidential Asset Transfers

Speaker: Ori Wallenstein, QEDIT

Time: 2019-April-10, 16:30

Hyperlinks: Youtube video, slide deck

Abstract: Enterprise Blockchains are substantially different than public Blockchains. Businesses looking to adopt zero-knowledge proofs for preserving privacy have unique constraints and requirements that allow offering new features and taking different trade-offs that would not be possible in a public Blockchain. In this talk I will discuss the zero-knowledge proof based solution QEDIT offers for confidential asset transfer. I will share details on some of the features of our solution and highlight how they were tailored for the specific needs of enterprise customers.

2.1.15 Talk 15: Zether: Towards Privacy in a Smart Contract World

Speaker: Shashank Agrawal, Visa Research

Time: 2019-April-10, 16:55

Hyperlinks: Youtube video

Abstract: Blockchain-based smart contract platforms like Ethereum have become quite popular as a way to remove trust and add transparency to distributed applications. While different types of important applications can be easily built on such platforms, there does not seem to be an easy way to add a meaningful level of privacy to them. In this talk, I will describe Zether, a fully-decentralized, confidential payment mechanism that is compatible with Ethereum and other smart contract platforms. Zether takes an account-based approach similar to Ethereum for efficiency and usability. It consists of a new smart contract that keeps the account balances encrypted and exposes methods to deposit, transfer and withdraw funds to/from accounts through cryptographic proofs. Zether also incorporates a mechanism to enable interoperability with arbitrary smart contracts. This helps to make several popular applications like auctions, payment channels, voting, etc. confidential. I will also talk about Sigma-Bullets, an improvement of the existing zero-knowledge proof system, Bulletproofs, which helps to make Zether more efficient. Sigma-Bullets make Bulletproofs more inter-operable with Sigma protocols, which is of general interest. We implemented Zether as an Ethereum smart contract and measured the amount of gas used. A Zether confidential transaction costs about 0.014 ETH or approximately $1.51 (as of early Feb, 2019). I will discuss how small changes to Ethereum, which are already being discussed independently of Zether, would

drastically reduce this cost.

2.1.16 Talk 16: Succinct Proofs on Ethereum

Speaker: Barry Whitehat, Ethereum Foundation
Time: 2019-April-10, 17:20
Hyperlinks: Youtube video
Abstract: The ability to validate zero-knowledge proofs using bilinear pairings was added to Ethereum in October 2017. Since then the community has built 2 high-level languages to make writing zero-knowledge applications easier, and made many more advances. They have built solutions for: 1. private identity and credentials 2. anonymous voting/signaling 3. scalablity 4. private transactions and proposed solutions for anonymous reputation systems. Come hear about what we have built and what we are planning to build :)

2.1.17 Talk 17: Aurora, Transparent Succinct Arguments for R1CS

Speaker: Nick Spooner, UC Berkeley
Time: 2019-April-10, 17:45
Hyperlinks: Youtube video
Abstract: We design, implement, and evaluate a zero knowledge succinct non-interactive argument (SNARG) for Rank-1 Constraint Satisfaction (R1CS), a widely-deployed NP language undergoing standardization. Our SNARG has a transparent setup, is plausibly post-quantum secure, and uses lightweight cryptography. A proof attesting to the satisfiability of n constraints has size $O(\log 2n)$; it can be produced with $O(n \log n)$ field operations and verified with $O(n)$. At 128 bits of security, proofs are less than 250kB even for several million constraints, more than 10x shorter than prior SNARGs with similar features. A key ingredient of our construction is a new Interactive Oracle Proof (IOP) for solving a univariate analogue of the classical sumcheck problem [LFKN92], originally studied for multivariate polynomials. Our protocol verifies the sum of entries of a Reed–Solomon codeword over any subgroup of a field. We also provide libiop, a library for writing IOP-based arguments, in which a toolchain of transformations enables programmers to write new arguments by writing simple IOP sub-components. We have used this library to specify our construction and prior ones, and plan to open-source it.

2.2 Day 2

2.2.1 Talk 18: Public Accountability vs. Secret Laws: Can They Coexist?

Speaker: Shafi Goldwasser, UC Berkeley, MIT and Weizmann
Time: 2019-April-11, 09:00
Hyperlinks: Youtube video
Abstract: Post 9/11, journalists, scholars and activists have pointed out that secret laws — a body of law whose details and sometime mere existence is classified as top secret — were on the rise in all three branches of the US government due to growing national security concerns. Amid

heated current debates on governmental wishes for exceptional access to encrypted digital data, one of the key issues is: which mechanisms can be put in place to ensure that government agencies follow agreed-upon rules in a manner which does not compromise national security objectives? This promises to be especially challenging when the rules, according to which access to encrypted data is granted, may themselves be secret. In this work we show how the use of cryptographic protocols, and in particular, the use of zero-knowledge proofs can ensure accountability and transparency of the government in this extraordinary, seemingly deadlocked, setting. We propose an efficient record-keeping infrastructure with versatile publicly verifiable audits that preserve perfect (information-theoretic) secrecy of record contents as well as of the rules by which the records are attested to abide. Our protocol is based on existing blockchain and cryptographic tools including commitments and zero-knowledge SNARKs, and satisfies the properties of indelibility (i.e., no back-dating), perfect data secrecy, public auditability of secret data with secret laws, accountable deletion, and succinctness. We also propose a variant scheme where entities can be required to pay fees based on record contents (e.g., for violating regulations) while still preserving data secrecy. Our scheme can be directly instantiated on the Ethereum blockchain (and a simplified version with weaker guarantees can be instantiated with Bitcoin).

2.2.2 Talk 19: Efficient Zero-Knowledge Protocols: The Modular Approach

Speaker: Yuval Ishai, Technion

Time: 2019-April-11, 09:40

Hyperlinks: Youtube video

Abstract: I will discuss the following modular approach for the design of efficient zero-knowledge protocols: (1) design an "information-theoretic"? probabilistic proof system; (2) apply a cryptographic compiler to obtain a zero-knowledge protocol whose security is based on cryptographic assumptions. Most efficient zero-knowledge protocols from the literature can be cast into the above framework. The talk will give an overview of different types of proof systems and cryptographic compilers that give rise to a wide range of efficiency and security tradeoffs.

2.2.3 Talk 20: Privacy-enhancing Cryptography at NIST

Speaker: Rene Peralta, NIST

Time: 2019-April-11, 10:20

Hyperlinks: Youtube video, slide deck

Abstract: We will talk about crypto standards at NIST and how it may relate to the ZKProof initiative. We will overview some of the history of crypto standards development at NIST. This informs our thinking regarding crypto areas that currently fall in uncharted territory with respect to standardization. An area of current interest at NIST is privacy-enhancing cryptography (PEC). Within this scope, we believe that it would be useful to develop reference material (documentation and implementations). Zero-knowledge techniques are an important component of this, along with techniques from Secure Multiparty Computation. Considering some PEC use cases as a basis, we will ask whether the current ZKProof proposal (as spelled out in the various public documents) is a good match for the anticipated needs.

2.2.4 Talk 21: ZKProof Proceedings: Review & Process

Speakers: Daniel Benarroch, Luís Brandão & Eran Tromer
Time: 2019-April-11, 11:25–12:05[2]
Hyperlinks: slide deck
Abstract: TBA

2.2.5 Panel 1: Zero Knowledge in the Enterprise

Panelists: Carlos Kuchovsky (BBVA), Yael Kalai (Microsoft Research), Mike Hearn (R3) and Jonathan Levi (HACERA)
Moderator: Jonathan Rouach (QEDIT)
Time: 2019-April-11, 14:00
Hyperlinks: Youtube video
Abstract: TBA

2.2.6 Talk 22: Bilinear Pairings based Zero-Knowledge Proofs

Speaker: Jens Groth, DFINITY
Time: 2019-April-11, 15:00
Hyperlinks: Youtube video
Abstract: TBA

2.2.7 Talk 23: MPC-in-the-Head based Zero-Knowledge Proofs

Speaker: Amit Sahai, UCLA
Time: 2019-April-11, 15:40
Hyperlinks: Youtube video
Abstract: TBA

2.3 Day 3

2.3.1 Talk 24: GKR based Zero-Knowledge Proofs

Speaker: Yael Kalai, Microsoft Research
Time: 2019-April-12, 09:00
Hyperlinks: Youtube video
Abstract: TBA

[2]Editors' note: this was originally mentioned in the scheduled of breakouts; we now identify it as a regular talk, to distinguish it better from the subsequent "breakout" discussions.

2.3.2 Talk 25: IOP based Zero-Knowledge Proofs

Speaker: Alessandro Chiesa, UC Berkeley
Time: 2019-April-12, 09:40
Hyperlinks: Youtube video
Abstract: TBA

2.3.3 Talk 26: Discrete Log based Zero-Knowledge Proofs

Speaker: Dan Boneh, Stanford
Time: 2019-April-12, 10:20
Hyperlinks: Youtube video
Abstract: TBA

2.3.4 Talk 27: From Public-Key Cryptography to PKI: Reflections on Standardizing the RSA Algorithm

Speakers: Jim Bidzos and Burt Kaliski, Verisign
Time: 2019-April-12, 14:00
Hyperlinks: Youtube video, slide deck
Abstract: When Rivest, Shamir and Adleman invented the RSA public-key cryptosystem in 1977, security applications — to the extent they employed encryption at all — relied on symmetric-key cryptography and infrastructures, such as the Data Encryption Standard (DES). Public-key cryptography had just been discovered by Diffie, Hellman and Merkle, and it would still take another decade or so until all the elements were in place — from algorithms to protocols to public-key infrastructure (PKI) — that have made public-key cryptography the standard way to secure applications today. Specifying those elements involved a collaborative effort where industry not only learned the new language of public-key cryptography in a technical sense but also started speaking it in products and policies. The story of the standardization of the RSA algorithm offers insights both for understanding how today's public-key infrastructure came to be, and for approaching standards efforts for additional cryptographic technologies such as zero knowledge proofs.

2.3.5 Talk 28: Zero Knowledge Ideal Functionality

Speaker: Muthu Venkitasubramaniam, University of Rochester and Ligero
Time: 2019-April-12, 15:00
Hyperlinks: Youtube video
Abstract: TBA

3 Breakout sessions

This section transcribes notes from the breakout sessions. For each breakout session (1–8), a thread was created in the ZKProof community forum, for possible followup public discussion related to the topic, including posting the scribed notes. The scribed notes are informal, and were mostly taken in real-time during the discussion, though some were edited or extended later.

BREAKOUT SESSIONS

DAY 2
APRIL 11, 2019

	MAIN STAGE	BREAKOUT 1	BREAKOUT 2
11:25-12:05	ZKProof Proceedings: Review & Process Moderators: Daniel Benarroch, Luis Brandão & Eran Tromer		
12:05-12:45	ZKProof Proceedings: Discussion Moderators: Daniel Benarroch & Luis Brandão	Recursive Composition of SNARKs Leads: Izaak Meckler & Eran Tromer	Structure Reference String Generation Leads: Sean Bowe & Mary Maller
16:45-18:00	*Community Standards:* Interoperability of Zero-Knowledge Systems Moderators: Sean Bowe & Abhi Shelat	Security Assumptions Underpinning Zero-Knowledge Proofs Leads: Jens Groth & Yuval Ishai	Formal Verification for ZK Systems Leads: Izaak Meckler & David Archer

DAY 3
APRIL 12, 2019

	MAIN STAGE	BREAKOUT 1	BREAKOUT 2
11:25-12:45	*Community Standards:* Commit-and-Prove Functionality Moderators: Jens Groth, Yael Kalai, Mariana Raykova & Muthu Venkitasubramaniam	Domain-Specific Languages for ZK Leads: Stefan Deml & Ahmed Kosba	
16:45-18:00	*Community Standards:* Deterministic Generation of Elliptic Curves for ZK Systems Moderators: Sean Bowe & Alessandro Chiesa	Interactive Zero Knowledge Leads: Yael Kalai & Justin Thaler	ZK Protocol Security Analysis & Proofs of Correctness Leads: Ariel Gabizon & Ran Canetti

Figure 2: Schedule of breakout proposal sessions

3.1 Breakout 1: ZKProof Proceedings: Discussion

Time: Thursday April 11, 2019, 12:05–12:45

Moderators: Daniel Benarroch & Luís Brandão

Abstract: In this session we aim to review the documents, as well as the newly drafted Community Reference, which is a LaTeX version of the proceedings [[...]]. We will discuss 3 different topics, to be decided by the participants during the session.

Scribe: Daniel Benarroch & Eduardo Moraes

Informal notes from the ZKProof community forum:

Post #1: danib31 — May 16, 2019, 1:22pm

This session was about reviewing the existing proceedings from the first workshop — see zkproof.org/documents.html

[[...]]

NIST also took the time to create a detailed list of comments around the Reference document.

Here are the slides of the presentation about the editorial process: https://docs.google.com/presentation/d/1_W5qEkAxJtB86KbJNhQl8C65NplHjm-_hVt7ZI7VboM/edit?usp=sharing

And see this folder for all the ZKProof public documentation

For all note taker, here is the view only document for SCRIBES:

[[https://web.archive.org/web/20190725212318/https://docs.google.com/document/d/1GBCjGJ87n_PPG8U7-EbgvwtlQeRxevxG5ngtW5gfZo0/view]]

Again, here is the LaTeX PDF of the ZKProof Reference Document v0.1:

ZKProofCommunityReference.pdf Google Drive file.

ZKProofCommunityReference (1).pdf (790.4 KB)

Post #2: danib31 — May 16, 2019, 9:46am

Here are my notes from the session:

Moderators: Luis Brandao and Daniel Benarroch **Scribes:** Daniel Benarroch and Eduardo Moraes

General Notes

Overview of the comments by NIST (attached):

- How to bring all the comments into the Reference document?
- Want larger audience and contributors to help

R1CS vs Circuits

1. Could be useful to have more clear description of r1cs representation and how it is mapped to and from circuits, as well as advantages over other representations.
2. Better linkage on R1CS
3. Follow comments in PDF by NIST

Comment on Taxonomy: QAP vs linear-PCP

1. Succinctness and efficient verification is an important topic, which is only achieved by QAP (within all linear-PCPs)
2. The Reference document should give QAP as an instantiation and not as an abstraction
3. Not clear that any general linear PCP has specific properties that are useful for some constructions
4. "Would like to see a comment that this is only efficient example"

Gadgets

1. Not all gadgets are about SNARKs
2. There are many gadgets (Proof for Shuffle) that are not necessarily related to SNARKs / NIZK
3. Should we add a table about gadgets that are represented for non-generic proving systems
4. Also, there are ways to optimize the gadgets, but hard to minimize the size of a circuit
5. Instead, there is out-of-the box functionality
6. Gadgets table is more about

1. Canonical use-cases for creating circuits
2. These are "functions" that you can call in programming language"
3. Can be seen from the point of view of composability (this is a concer, there is also many primitives that allow for composability)
4. This table is about standardization
5. Can also be seen as the specific protocols
6. Used for building specific applications

Composability: who's responsibility is it?

1. Implementation who implements the primitives,
2. Applications since they compose them into circuit application
3. Security since they need to define the actual security

Comment on audience of document and specific documentation (who is the reader of this?)

1. Handling witnesses and private info - using legacy technology
2. Building use-cases (companies and biz people not easy to read)

Branch out general constructions?

1. Approaches for general functionality
2. For some gadgets can use specialized primitives

Suggested Contributions

Name
Email
Specific Contribution / Action Point

Name	Contribution / Action point
Mariana Raykova	QAP vs Linear PCP efficiency and instantiation
Luis Brandao	Specific comments integration from NIST
Eduardo Morais	Motivation and reference of the applications (NIST Comment C12, C16)
Eduardo Morais	Gadgets improvement table (NIST comment C13)
Armando Fac	NIST C20
Daniel Benarroch	Describe better gadget purpose / functionality (start discussion of scope)
Daniel, Angela	Applications need to be better explained and worked on, as well as the predicates
Angelo de Caro	How to integrate legacy technology to applications in ZK (lego SNARK?)

Sorry if I mispelled any name…(?)

Post #3: Luis — June 4, 2019, 1:27am #3

In early April 2019 the NIST-PEC team ("crypto-privacy at nist dot gov") produced a PDF document with "NIST comments on the initial ZKProof documentation". These comments were briefly presented during the "ZKProof Proceedings and Community Reference" session at the 2nd ZKProof workshop, and have since then been available online (find link in the initial post in this thread).

This post enumerates the titles of the comments (C1,…,C22 and D1,…,D7) and associates with each item a corresponding short note indicating needed or/and volunteered contributors.

Please consider volunteering to address the several items that indicate "needs contributors" — post a comment mentioning which item(s) you propose to contribute to, and/or send an email to "editors at zkproof dot org" with any question or comment.

The notes below may be updated as more contributors are identified.

=== EDITORIAL:

- C1. REFERENCE DOCUMENT: the initial draft is already available; the editors will integrate the new contributions.
- C2. RECOMMENDATIONS AND REQUIREMENTS: the editors will identify and highlight proposed "recommendations" and "requirements" already contained in the initial draft; other contributors are welcome to suggest additional "recommendations" and "requirements".
- C3. SCOPE OF THE CREATIVE COMMONS LICENSE: the "CC-BY-4.0" mention is already in the cover of the initial draft (version 0.1); the editors will promote clarity about the license in the new version of the reference document.
- C4. GLOSSARY: the editors will complement the current glossary with terms already described in the reference document; other contributors are welcome to suggest new entries.
- C5. EXECUTIVE SUMMARY: the PEC team volunteers as contributor.
- C6. EXAMPLES: needs contributors.

=== TRACK "SECURITY/THEORY":

- C7. PROOFS OF KNOWLEDGE: the PEC team proposes adding some text; other contributors are welcome.
- C8. CONCURRENCY: needs contributors.
- C9. TRANSFERABILITY: some contributors are identified in the "interactive ZK" session.
- C10. CIRCUITS VS. R1CS: needs contributors.
- C11. COMMON VS. PUBLIC: the PEC team proposes adding some clarification text; other contributors are welcome.

=== TRACK "APPLICATIONS"

- C12. MOTIVATION: one contributor is identified in the "Community Reference" session (see post above).
- C13. GADGETS: one contributor is identified in the "Community Reference" session (see post above); more contributors are welcome.
- C14. INTERACTIVITY VS. TRANSFERABILITY: some contributors are identified in the "interactive ZK" session.
- C15. IMPLICIT SCOPE OF USE-CASES: needs contributors.
- C16. REFERENCES: one contributor is identified in the "Community Reference" session (see post above); more contributors are welcome.

=== Track "IMPLEMENTATION":

- C17. BACKEND CHOICE NIZK-R1CS: needs contributors.
- C18. COMPUTATIONAL SECURITY PARAMETER: the PEC team proposes to contribute with some text (likely to suggest changing 120 to 128).
- C19. STATISTICAL SECURITY: some contributors are identified in the "interactive ZK"

session.

- C20. SIDE-CHANNELS: one contributor is identified in the "Community Reference" session (see post above);
- C21. VALIDATION: needs contributors.
- C22. INTELLECTUAL PROPERTY: the PEC team proposes to contribute with some text.

=== OTHERS (TOWARDS A REFERENCE DOCUMENT):

- D1. TITLE; - D2. PURPOSE; - D3. AIM; - D4. SCOPE; - D5. FORMAT: the editors will consider integrating the proposed elements in the preamble of the reference document.
- D6. EDITORIAL METHODOLOGY: a proposal was presented during the "ZKProof Proceedings: Review & Process" session at the 2nd ZKProof workshop; the editors will write a corresponding description and submit it for review by the community.
- D7. INITIAL COMPILATION: already done (draft version "0.1") and made available.

3.2 Breakout 2: Recursive Composition of SNARKs

Time: Thursday April 11, 2019, 12:05–12:45

Moderators: Izaak Meckler & Eran Tromer

Abstract: Preprocessing SNARKs have the major drawback that the size of the computation to be verified must be fixed before performing a setup. Recursive composition of SNARKs is a powerful technique for creating SNARKs for a priori unknown length computations. Its applications include merging proofs, blockchain compression, and decentralized private computation a la ZEXE. This workshop will cover both the theory and the practice of recursive composition. We will discuss

- The basic construction and its extensions
- Techniques for improving efficiency in practice
- The underlying cryptographic primitives and how their efficiency may be improved

We can also discuss recursive composition of IOP-based proof systems if there is time and interest.

Informal notes from the ZKProof community forum:

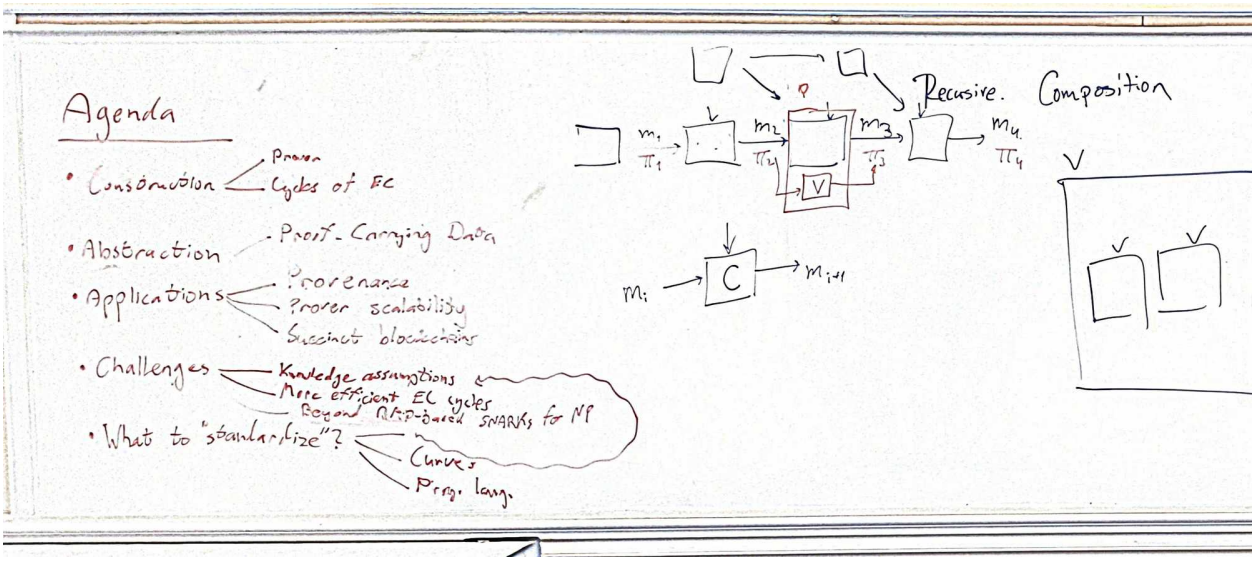
[[Omitted posts with abstract, question and empty form.]]

Post #4: Tromer — April 11, 2019, 9:58pm

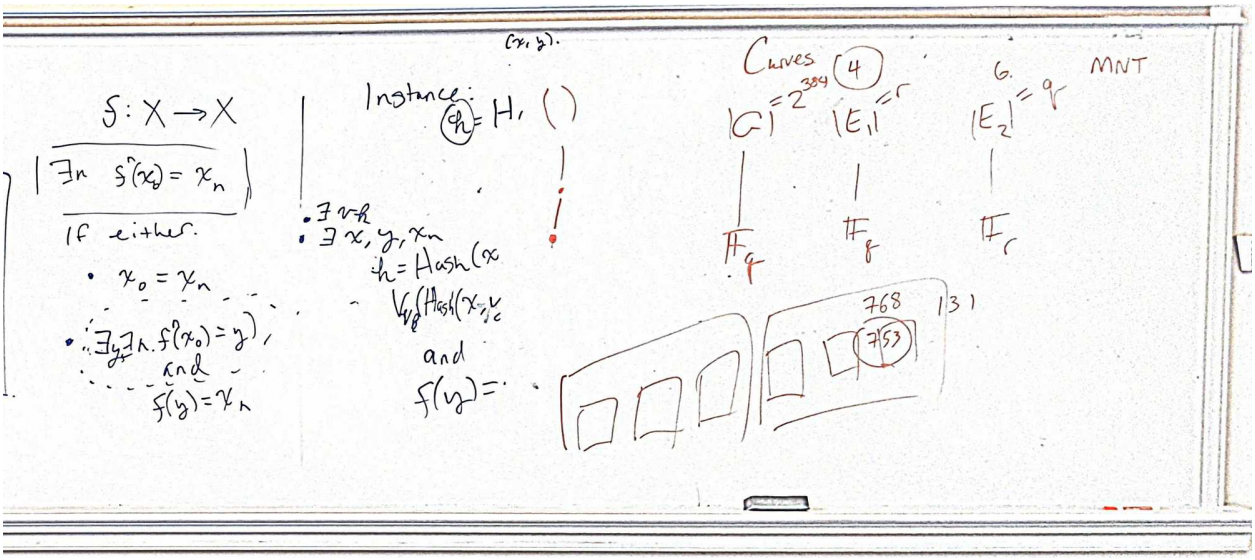
Session whiteboard: [[See Figure 3 and Figure 4; file size was reduced for inclusion in this document.]]

Post #5: Tromer — April 11, 2019, 10:37pm

Some of the people interested in recursive composition of SNARKs (session photo): [[See Figure 5.]]



(a) left side of the board



(b) right side of the board

Figure 3: Photo in breakout 2 — photo of whiteboard

Post #6: dlubarov — April 12, 2019, 6:50am

Here are my notes from the workshop. Apologies if I missed or misheard some parts of the discussion.

Izaak: A SNARK can include a SNARK verifier; this is called proof composition. Why would you want that? Let me give a motivation.

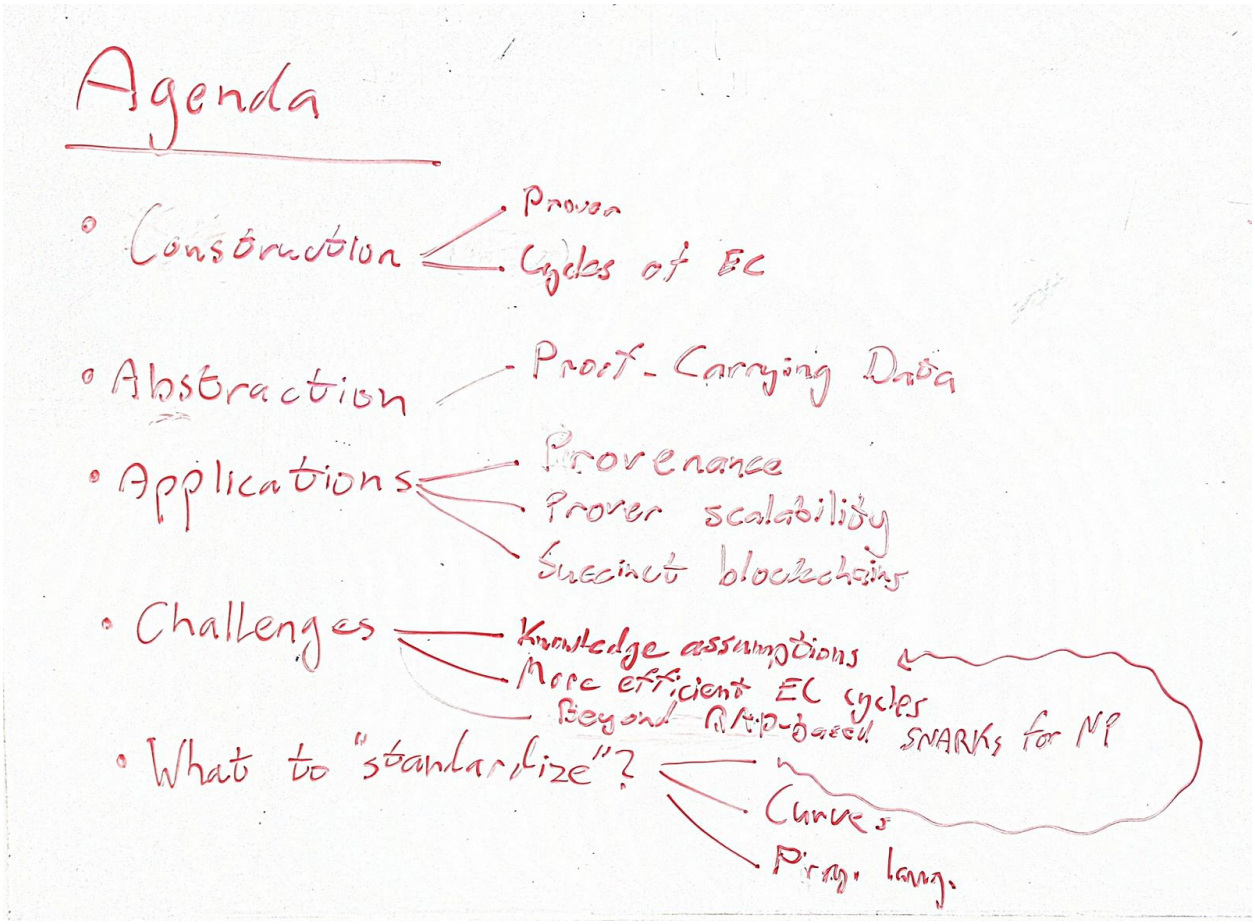


Figure 4: Photo in breakout 2 — Agenda

Figure 5: Photo in breakout 2 — discussion session

Say you have a map $f : X \to X$. You want to prove $\exists n. f^n(x_0) = x_n$. If you use pairing based SNARKs, you need to write down a circuit, so you need a bound on n.

The statement can be proved inductively. It's true if either $x_0 = x_n$, or $\exists n, y$ such that $f^n(x_0) = yandf(y) = x_n$. So instead of proving the statement directly, we can say that $\exists y$ such that $\text{verify}_{\text{me}}(x, y) = \top$ and $f(y) = x_n$.

How do we pass the verification key into the SNARK? We have two ways of binding variables in SNARKs: the instance and the witness. You can try putting the verification key in the instance, but that doesn't work, because the size of the verification key depends on the size of the instance.

We need to make sure that the inner proof was verified with the right verification key, not just a key corresponding to zero constraints. So we include $\text{hash}(x, y, vk)$ in the instance. In the circuit, we require that $\exists vk, x, y$ such that $\text{hash}(x, y, vk) = h$ and $\text{verifier}_{vk}(\text{hash}(x, y, vk)) = \top$.

Eran: There is a useful abstraction called proof carrying data (PCD). You have various parties that communicate data to one another. The first party sends message m_1 to second party, who sends m_2 to the third party, etc. The messages could be blocks in blockchain.

We want to show that all messages are compliant with some predicate. We attach a proof to each message, with two parts, one showing that the message is valid and another showing that the previous message had a valid proof. By verifying the final proof, you can verify compliance of the entire chain. This is implemented in libSNARK. We can write the message compliance predicate in R1CS.

A simple example is provenance. We might want to verify that some image file, text file, etc. evolves in a permissable way. We start with authentic images, then require that edits must abide by certain rules. Maybe the only permissible modifications are cropping, adjusting brightness, etc.

Another application is scalable SNARKs. If our computation involves many steps, we might run out of memory if we try to create one huge SNARK. Using PCD, we can break up the computation into chunks, where each chunk verifies one clock cycle of a larger computation.

A third application is succinctly verifiable blockchains, like CODA.

Izaak: Succinct blockchains use the same idea as PCD. The message predicate verifies VRFs etc. to ensure that a block is correct.

Discussion

???: Some computations don't have a recursive structure. **Izaak:** The scheme is flexible; you can stick any verifiers inside verifiers. Can merge proofs.

???: Why are you restricting proofs to be non-interactive? **Eran:** A naive answer is that the person verifying one message can't look at previous interactions. There might be some possible schemes though. **Alessandro:** Even with public coin arguments, you have to be there to witness the randomness. **Alessandro:** Could work with designated verifier proofs.

???: If you want to merge multiple proofs, you don't know how many there will be, so you aggregate them in a tree. Is it more efficient to have one circuit verify several nested proofs? **Eran:** You can do either. Depends on the constants.

???: How does composition affect security? **Eran:** You can recursively extract the whole initial witness, but parameters deteriorate with recursion depth. You need strong, non-falsifiable assumptions about the efficiency of a knowledge extractor. That said, no attack is known.

???: Knowledge extractor may not run in polynomial time. Can I still compose arguments? **Alessandro:** We can look at it in two ways: attacks or analysis. From the perspective of attacks,

a fresh proof looks the same as a recursive one. I wouldn't know how to attack it. From an analysis perspective, we have a spectrum of knowledge assumptions one can make. The weakest would only allow constant recursion depth; others would allow logarithmic depth; others would allow arbitrary polynomial depth. I would feel comfortable with constant or logarithmic depth.

???: It would be nice to get an idea of the proving time, verification time, field sizes, etc. **Izaak:** Only known, decently efficient construction involves cycles of elliptic curves. You have two pairing-friendly curves: $E_1(F_q)$ with order r, and $E_2(F_r)$ with order q. The problem is that the only known cycles have low embedding degrees. One has 4 and the other has 6. With 768 bit fields, we get a claimed 131 bits of security. Need high 2-adicity. If you don't care about 2-adicity, you can generate these cycles on your laptop.

???: You're talking about a cycle of two MNT curves, but we could also use a cycle of curves from different families. **Alessandro:** Would love to see more cycles of any reasonable length. It's an open question. Want a minimum embedding degree of 10-12. Can you demonstrate or rule out the existence of such cycles? These are only type within MNT. We can rule out certain combinations. All have to be prime order. Izaak: And besides cycles, we could also have a graph of related curves. The graph could have a small curve, say 2^{384}, for leaf SNARKs, then a cycle of curves with low embedding degree for composition.

???: CODA is using recursive SNARKs to get a constant size blockchain, which doesn't need zero knowledge. What about applications which require zero knowledge? **Izaak:** We could add private transactions to CODA. **Eran:** Also image authenticity – you might not want to reveal which regions were cropped out.

???: I might have process loops that run an arbitrary number of times. Are there ways to stop a proof from continuing forward without affecting soundness? E.g. how many times a car has a maintenance cycle. **Eran:** You can always recurse, but with security implications.

???: Are there other argument systems that have this property? **Eran:** We need succinctness, non-interactivity, etc. so any such system would be a SNARK. **Izaak:** Systems other than pairing based SNARKs? **Eran:** Composition has only been implemented for QAP based SNARKs so far, but we could use STARKs, etc. Any argument system with sublinear verification could work. **Eran:** One could also ask, do we need a SNARK for NP? The answer is no—we just need a language for a SNARK verifier, plus whatever you put in your compliance predicate.

???: Are cycles of curves used only for performance? **Eran:** Yes.

???: Can we compose Fiat-Shamir based non-interactive arguments? **Alessandro:** Formally, all SNARKs compose. Concretely, we have to pay a cost for Fiat-Shamir based arguments.

???: Don't need SNARK at top level. Could be non-succinct. **Eran:** SHARK is an example [of composing a succinct proof system with a non-succinct one].

???: We often need to verify signatures; does the cycles scheme imply what signature scheme we should be using? **Eran:** We want the signatures to be SNARK-friendly, i.e. involving linear constraints over the SNARK's field.

Eran: What should go into the community reference? Proposals for the next iteration? **???:**

What curves to use. **???:** Collaboration with programming languages people. **???:** Guidelines for how to combine argument systems without breaking security.

Post #7: Youssef — April 13, 2019, 11:47am

Papers that have been briefly mentioned during the session:

[BCCT12] https://eprint.iacr.org/2012/095.pdf [BCTV15] https://eprint.iacr.org/2014/595.pdf

New users are resteicted to 2 links per comment so I'm putting the last paper about pairing-friendly EC cycles by Chiesa et al. in the next comment.

Post #8: Youssef — April 13, 2019, 11:48am

[CCW18] https://arxiv.org/pdf/1803.02067.pdf

3.3 Breakout 3: Structure Reference String Generation

Time: Thursday April 11, 2019, 12:05–12:45

Moderators: Sean Bowe & Mary Maller

Abstract: Non-Interactive Zero-knowledge proofs and arguments require a common reference string known to all parties. Common reference strings with structure are termed structured reference strings. Currently we are unaware of methods to generate structured reference strings without the use of trusted third parties. In this session we shall discuss best practice for distributing the generation of structured reference strings. We shall discuss both MPC and updatable techniques.

Informal notes from the ZKProof community forum:

[[Omitted post with abstract]]

Post #2: mmaller — April 26, 2019, 4:35pm

2nd ZKProof Workshop Notes

Scribes: Ariel Gabizon and Aviv Zohar

Objectives/thoughts:

Call it decentralized setup instead of "trustless setup"?

Size of the SRS (try to keep it small because users need to download it) In Zcash - having crs small enough for passing around in DVD.

Can we have an update process where updates can be done in parallel?

Important e.g. if several people try to update same state in similar time

- Is there a point in more than 128 parties for 128 bit security if each person is honest w.p 1/2
- Experiment: distribute Ether amongst 40 parties to see practical capability of attacking many parties simultaneously - though perhaps hard to simulate an experiment with similar reward.
- Model: not all participants are online at once. Stronger than having everyone online for multiple rounds.
- Will people stop participating at some point?
- Will there be a DOS attack when too many people try to update at once
- Making sure the person requesting to participate is not forever delayed by the organizer (is the organizer subverting the process by selecting participants)?
- Focus on building a public SRS that would be publicly used. How do we build this? Build for different curves, circuit sizes, etc.
- How do we handle Sybils in such a process (there should be posts to twitter, attestations of contributions by reputable individuals, etc.)
- Which MPC we use, what are the update proofs, don't suggest best practices for filtering people.
- Diversity of hardware – have some record of what is used (software versions, hardware, etc.)
- Can the updatable SRS generated suffice for smaller circuits. Proofs of well formed updates in Sonic can be read only for monomials that are relevant for the circuit size.
- Updatable and non-updatable SRS generation.
- Who are the participants? what is the attack model? What are security assumptions? What's the ideal functionality?
- Ideal functionalities vs. "Code based games" approach for security. For sapling: Showing that if an attacker corrupted the SRS, he would have also succeeded in doing something similar in a trusted setup SRS.

Post #4: danib31 — May 16, 2019, 5:50am

Hi @mmaller and @arielg were there specific contributions or action items that derived from this discussion? I could suggest some:

- looking at the reference document I would say it is missing a proper explanation of updateability / universality, its implications and potentially a description on how to achieve this (maybe the latter should go in the security section)
- A more specific explanation on how to proceed with an MPC (without getting into details of the construction), but detailing the practical needs to make it happen.
- probably more stuff

3.4 Breakout 4: Security Assumptions Underpinning Zero-Knowledge Proofs

Time: Thursday April 11, 2019, 16:45–18:00

Moderators: Jens Groth & Yuval Ishai

Abstract: The security of zero-knowledge proofs rely on an expectation that certain tasks are

infeasible for attackers, for instance, that they are unlikely to find a collision for a hash function. To maximize efficiency, succinct zero-knowledge proofs often rely on "strong" assumptions such as knowledge-of-exponent assumptions or idealized models such as the random oracle model. In this discussion session we seek guidelines for which cryptographic assumptions to rely on in deployed zero-knowledge proofs and how different assumptions should be compared against each other.

Informal notes from the Source: ZKProof community forum:

Post #1: arielg — April 19, 2019, 10:07pm

Relevant to that breakout, here is a blog post I wrote about the relation between the algebraic and generic group model

Medium – 24 Aug 18 Moving SNARKs from the generic to algebraic group model The most efficient zk-SNARK construction known, due to Jens Groth, has a security proof written in what's called the generic group model. Reading time: 3 min read

Post #2: danib31 — May 16, 2019, 9:55am

Thanks Ariel, this is a great post! Here is the abstract for the breakout session, as per the moderators.

Title: Security assumptions underpinning zero-knowledge proofs

[[. . .]]

Intro: Classification of assumptions, categorization of assumptions (e.g. falsifiable vs non-falsifiable), idealized models (e.g. ROM or generic group model), portfolio of assumptions (e.g. use of both KoE in one proof system and ROM in another means your full system relies on both), physical assumptions

Discussion topics: portfolio diversity vs specificity, efficiency vs strength, is there a role for standardization (e.g. terminology to distinguish variations of assumptions, definitions of assumptions to make comparisons easier, recommended use of assumptions to optimize global assumption portfolio)?

Discussion plan: soliciting concerns about the existing practices, discussing proposals for standardization.

Post #3: danib31 — May 16, 2019, 9:56am Anyone know who the scribe was?

EMAIL: === **Scribe:** Muthu Venkitasubramaniam shared the notes by email to the editors on June 12, 2019:

Think about assumptions

1. Which assumptions are ok to use? it would be great to have unconditional assumptions. generic group models are hard

Assumptions in the plain model and assumptions in the generic model/ random oracle model.

Amit: Generic models are under appreciated. No construction proved in the generic model has been shown to be broken. In practice, random oracle model is what we use. Ripe for automation.

Yuval: Are there discrepancies between ideal models and real models, eg: bilinear generic group model? Anyone know about such instances? People working on groups, are there examples for which they are different from ideal model.

Using assumptions are easier, working in the ideal model is subtle and tricky. Proofs are here often wrong and not well done.

There are no two papers that define DDH the same way. Do we have fully specified assumption with quantifiers?

Ariel Gabizon: All the papers of KOE have this auxiliary information that is benign. They never say what is benign. This is because of counter examples due to obfuscation results. Suggestion: Auxiliary input is uniform output of a low degree polynomial. (This is because SNARKs typically have a few elements from the CRS). Response by Nir: If we have obfuscation has in low degree then it can still contradict.

Action Item: Define benign.

Jens: There is a generator that generates a group.

Yuval: Use what SSL does. Then use random oracle. Heuristic leap-of-faith. Xiao et al -> Random oracle instantiations are not correlation robust.

Yuval: Is it ok to hypothesize, whatever is good for IBE is it good for ZKsnark? Then we can use whatever is used for IBE or BLS-signatures.

Prove security in theory, then we believe in security. Fiat-Shamir:

Yuval: Identifying toy versions. Breaking snarks might be a beast. Is there a toy version that we can find attacks.

Can we make heuristic guidelines? eg, if good for IBE or signatures then good for snark. (1) Analyze in Gen Group model and then instantiate, then (2) assume instantiating is ok.

Is there a KA that is good for all bilinear applications. Mary Maller: Algebraic group model. Between standard and ideal. Dont know what this is. Adversary is algebraic - when it gives a new group element, it gives a relation with elements that have been seen in the past.

Ariel: If you prove generic group model security with low degree CRS then it is secure against algebraic adversary.

action item: algebraic group model.

Suggestion: Use other standards.

Can someone suggest a natural condition in the GGM that can be cryptanalyzed. Quantitative question: GGM broken in $2^{\sqrt{(n)}}$ but instantiations are $2^{n^{1/3}}$. Simple snark (maybe signature), toy version, toy version security => snark security.

Standard should propose that anyone proposing groups show give toy challenge problems.

Break soundness of a toy version mimicking the security of Snark.

Yuval: Shoudl we allow people to propose groups. Benchmarking is an issue.

Yuval: Elgamal is only additively homoorphic, targeted non-malleability. linear-only. win-win: either FHE or linear-only.

Hada-Tanaka: was broken. (If there are three quantifiers then it is wrong)

Ariel:

2. Which models are ok?

3. Which precise assumption? (Discussed above) Any good pointers to precise formaulation? Daniel: It is already assumed there is an extractor. Mismatch.

4. Assumption portfolio?

snark friendly hash functions, zk-friendly primitives - needed or hash functions

5. All assumptions - setup, timing? Human ignorance assumption

3.5 Breakout 5: Formal Verification for ZK Systems

Time: Thursday April 11, 2019, 16:45–18:00

Moderators: Izaak Meckler & David Archer

Abstract:

When you program SNARKs, you write a program that generates a constraint system that you hope enforces some high-level property that you have in mind. But how do you know for sure that the constraint system you wrote down does indeed enforce that property? There is an almost exactly analogous problem which occurs in programming in general: how does one know that the program one has written conforms to a particular specification?

Formal verification (FV) refers to a set of techniques for writing formal proofs of adherence of a program to a specification. FV is still a highly specialized skill, and can reasonably handle only moderately complex protocols and code, but interest in these techniques is growing substantially.

Today, commercial and government organizations actively employ FV on critical areas of protocols, and aim to expand that coverage to formally verify entire protocol libraries.

In this breakout session, we introduce the basic notions of FV. We then lead an exploratory discussion to identify the diverse abstractions that may be useful in formally proving correctness of SNARK programs.

See you there! David and Izaak

Informal notes from the ZKProof community forum:

[[Omitted post with abstract]]

3.6 Breakout 6: Domain-Specific Languages for ZK

Time: Friday April 12, 2019, 11:25–12:45

Moderators: Stefan Demi & Ahmed Kosba

Abstract:

The increasing interest in zero knowledge proofs in both academia and industry has lead to the development of several libraries, domain-specific languages and compilers for making primitives like zk-SNARKs more accessible. These tools differ in various aspects including expressiveness, efficiency, usability and others. In this session, we will briefly review the existing tools, and discuss the current challenges in the field. In addition, we will debate which features the audience would like to see in the next iterations of such domain-specific tools and how to leverage them in real-world applications.

Looking forward to your attendance. Ahmed and Stefan

Informal notes from the ZKProof community forum:

[[Omitted posts with abstract and question]]

Post #3 sanchopansa — May 1, 2019, 5:35pm

Hello everyone,

You can find the notes for the session here. Please feel free to comment with suggestions directly in the file.

The notes were compiled by the scribe, Valentin Ganev, with further amendments from Stefan Deml and Ahmed Kosba.

[[Begin transcription of the downloaded notes]]

Domain-specific languages for ZK protocols

- Can we have a well-defined set of metrics or a framework to evaluate and compare DSLs/compilers?
  - Possible metrics: Low vs high-level, usability, reliability, optimizations, compatibility with different back ends, .. etc.
- DSLs should not only be for one type of SNARKs; would it be useful/possible to have an intermediate representation produced by DSL compilers, which works for R1CS, AIR, etc?
- How easy is it to add support for new curves to ZoKrates (or any other DSLs)?
  - @Stefan: This should be easy as ZoKrates has a clear interface to the backend proving system(libsnark, bellman) and the curves need to be implemented there as well
- How to ensure that optimizations preserve correctness of the circuit?
  - A relevant work is PinnochioQ: certified compiler, which proves that a QAP is equivalent to the original program.
- Reusing optimizations between projects
  - Could it be interesting to have optimizations separate from compilation? (@Jordi Baylina - Iden3)
    - The optimizers can be re-used between different languages
    - Optimizations could be easier to verify
    - Optimizer might need access to the logic of the circuit in some cases
    - Low-level vs high-level optimizations
      - High-level optimizations are done at the DSL/Compiler level
      - Low-level optimizations are done at the circuit level
      - @Ahmed: Low-level optimizations are easier to be separate, but as observed from xjsnark, higher-level optimizations on the language level usually lead to more savings when translating high-level code. Compilers can take better decisions when the context and the types of the variables are known. Looking at circuits/r1cs directly might not enable as many optimizations.
      - @Jordi: Can we have a common intermediate format that provides useful information to the optimizers about context, ..etc?
  - Another possible venue could be to re-use LLVM & other existing toolkits for optimizations & analysis (@Harry Roberts - EthSnarks)
    - LLVM is unstable between releases
    - LLVM might change semantics (@Henry de Valence - Zcash Foundation)
      - E.g. Rust & Swift developers had issues with LLVM as the semantics were coupled with C
    - Graal – fully open-source compiler, a large and well-funded project (@Mike Hearn - Corda)
      - More stable and higher-level semantics compared to LLVM
      - Corda looked into this for converting JVM bytecode into R1CS
      - It would be beneficial if it would be possible to convert from between a traditional compiler representation (e.g. SSA graph and a R1CS)
- Feature requests / Experience by the community
  - For blockchain/asset specific applications you could design a scoped DSL
    - Interstellar is creating a custom DSL for mobile payments (@Cathie Yun - Interstellar)
      - Only merge, join, split operations supported by the language
    - Drawback: You could have an explosion of languages
      - Limited resources in the community
      - Few people working per project
- Standardization of gadgets

- Re-use gadgets between different libraries
  - Overlap with the interoperability discussion
- Requires standardization of the elliptic curves
  - Hash to point, base points, etc.
- Most gadgets keep having little improvements, makes it more difficult
- What is the difference between standardization and specification for gadgets? (@Sean Bowe - ELECTRIC COIN COMPANY)
  - If we want to have the same implementation of gadgets in every language, it should be a specification
    - Is this necessary/beneficial?
- Optimal implementation of gadgets can be input dependent (@Jack Grigg - ELECTRIC COIN COMPANY)
  - Need a format for describing a set of transformation based on input data
  - Once again overlap with interoperability discussion

[[Omitted post]]

Post #5 danib31 — May 2, 2019, 9:16am

Hi @sanchopansa, thanks for posting the notes, they look great! I skimmed through them but could not identify specific action points or contributions. Do you know / have a list of these?

At this point, DSLs are not a bit part of the reference document, and I believe it would be amazing to have a more refined section about it.

Post #6 sanchopansa — May 31, 2019, 2:32pm

I just saw the comment now @danib31. Let me have a chat with Ahmed and Stefan and we'll see if we can identify some concrete action items/contributions next week.

Post #7 danib31 — June 6, 2019, 1:38pm

Thank you @sanchopansa! Be on the lookout for a formal email about contributions and feel free to reply there as well.

3.7 Breakout 7: Interactive Zero Knowledge

Time: Friday April 12, 2019, 16:45–18:00

Moderators: Yael Kalai & Justin Thaler

Abstract: This breakout session will identify advantages and disadvantages of interactive zero-knowledge proofs relative to non-interactive ones, and attempt to identify scenarios and applications where interactive protocols are particularly suitable or relevant.

This forum serves as a place for people to comment and keep the conversation alive. A partial list of advantages and relevant applications has been compiled, and will be discussed at the breakout session. Participants are encouraged to add to the list before, during, and after the session.

Informal notes from the ZKProof community forum:

[[Omitted posts with abstract and schedule]]

Post #4: Sean — April 14, 2019, 4:27pm

[[A subsequently edited version appears in the next post]]

[[...]]

Please let me know if you'd like the raw OneNote file - Sean Coughlin

Moderators: Justin Thaler, Yael Kalai

Scribe: Sean Coughlin

Notes deliverable owner: Sean Coughlin

Participants:

Suggested contributions:

- Most of the focus has been on non-interactive
- Surely settings where interactive is valuable
- Goals: identify advantages and disadvantages of interactive protocols relative to non-interactive
- Advantages
  - Makes protocols easier to design (more efficient) if interactive
    - GKR's PCP (non-interactive) requires all possible conditions
    - IOPs allow for interaction, Fiat-Shamir allows for non-interactive
      - Fiat-Shamir requires raised computability, since has security loss
    - Generally, more efficient and not easier to design
  - Space is smaller (Muthu)
    - Can prove layer-by-layer
    - DIZK necessary for huge size parameters
  - If public coin, make non-interactive via Fiat-Shamir
    - Fiat-Shamir is not memory efficient (Muthu): not really
    - Have to have fixed rounds (Ivan)
    - More degrees of freedom so harder
    - Public coins can just be made into non-interactive (Kalai)
    - Fiat-Shamir requires honest verifying (Muthu): perhaps (Kalai, Ivan)
    - Random Oracle (Riad)
  - Have security with rewinding (since can make into protocol)
  - Transferability, plausible deniability (Luis)
  - In literature: in public verifiable case, scheme where proofs are transferrable must be non-interactive: false (Luis)

    - Combining ZK with oblivious transfer
        - Receive of oblivious transfer … only if sender satisfies NP relation (Muthu)
        - Certified oblivious transfer can become interactive
    - Why is there so little interest in applications/cases of interactive? (Kalai)
        - Where need deniable
        - Why should we start with assumption that interactive is OK
            - Gives us freedom and can then go NI (Kalai)
        - Are transparent snarks dependent on interactive? (Ynpeng)
            - Huge advantage of size
        - Removing trusted setup (Ynpeng)
    - If you assume RO then you assume transparent setup…
        - It depends on you assumption
    - Tradeoffs are important
        - In particular setting you can gain efficiency with interactive (Luis)
        - With concurrent cases it would require more steps
        - Depends on if need rewind
        - So transparency, transferability…
    - In setting where communication time is a problem, can do things like CRS (Ivan)
        - CRS isn't necessary (Muthu)
    - Deniability necessary (Kalai)
        - Want to be able to prove statement but prevent transferability from verifier to others (Luis)
        - VDF can prove either you know statement or next block, so hard to transfer (Muthu)
        - Can prove I know witness or I know you secret key
    - Want interactive design that's not transferrable but where messages are signed, prover signs own messages, now is transferrable (Luis)
        - Non-cryptographers should know about this
- Disadvantages
    - Concurrent security concerns
    - Network latency
    - Some applications require interactivity (Kalai)
- Soundness
    - Statistical security: can convince verifier with certain probability
        - When noninteractive there must be higher probability
    - 40 bits in interactive protocol is good enough
        - There is only 1 try interactively
        - Asynchronous NI must be higher security
    - Soundness is not ZK (Riad)
    - Time to live is only limited by time of protocol
- Trusted setup or assumptions is not removed but simplified (Ynpeng)
    - RO is lower security
- Action item: inaccuracy in documents since no discussion of Fiat-Shamir loss of security (Ynpeng)
    - Will get relevant references (Ynpeng)
        - Is inaccurate about hash function/RO
- Disadvantage
    - Interactive protocols take longer (Kalai)
    - Interactive have more exposure to side-channel and timing attacks (Riad)

      - Verifier can track time of responses (Luis)
      - Public coin can have prover do timing attack
      - Being careful with implementation is not good enough since side-channels keep innovating (Luis)
      - Verifier computation should be the same regardless of the size of the witness (Ivan)
      - Bad programming is a universal solvent; not guaranteed full compliance (Riad)
      - Could use weaker definition of timing attack: for any 2 witnesses the timing must be the same (Muthu)
        - Theoreticians don't care about constant time (Kalai)
        - Define ideal functionality where timing is t_0 to t_d where clock must reset (Luis)
- PCP have constant length; public coins need small proof size
  - Difference between public coins and private/enterprise (Daniel)
    - Regulators will demand auditing
    - Easiest private blockchain is database (Kalai)
      - Enterprise is diverse (Daniel)
      - Has different trust model (Muthu)
      - DB not immutable (Riad)
      - Amazon AWS has DB structured like blockchain (black jacket)
        * Removes consensus (Daniel)
  - If there are interactive proofs that have super-constant/linear with short interactive proofs
    - How does this compare with longer proofs since both are possible (Luis)
    - Interactive proofs are always longer (Kalai)
    - With interactive proofs we can't get as small size as NI (Luis)
    - Why is that? Do interactive version with Fiat-Shamir and use SNARG which is short and no Fiat-Shamir (Kalai)
    - SNARG has RO in circuit (Ynpeng)
    - Not ZK just correctness (Kalai)
    - Why not use random string? (Riad)
    - This is a general principle to shorten proof (Muthu)
    - Need so many ROs for SHA256 (Ynpeng)
    - Use Pedersen since it's faster (Muthu)
- Is there another way to remove interactivity other than Fiat-Shamir (Daniel)
  - Yes, but less efficient (Kalai)
  - Can lose soundness but not terrible (Muthu)
  - Public verifiability with pairings (Kalai)
    - Costs efficiency of proving (Ynpeng)
- Disadvantages
  - Shortest known NI are shorter than interactive
- Calls for action points
  - Last time didn't cover interactive at all
  - Need paragraph from each volunteer
  - List advantages/disadvantages
  - Not discuss inaccurate document
  - Daniel: applications
  - Luis: statistical security
  - Ynpeng: efficiency and trusted setup

    - Ivan: deniability
- Public verifiability of interactive proofs
    - Can use external structures

Post #5: justin — May 6, 2019, 11:20am

The notes from the breakout session on Interactive Zero Knowledge have been edited. They can be found in docx and pdf format at the following links (.docx, .pdf). Below is also the text of the notes, though some formatting has been lost.

2nd ZKProof Workshop Notes

Interactive Zero Knowledge

https://community.zkproof.org/t/breakout-session-interactive-zero-knowledge/

https://community.zkproof.org/c/breakout-sessions

LaTeX: https://drive.google.com/drive/u/2/folders/1HWZYMH-6Mx8wcX8geium506L0KRxcgPe

Moderators: Yael Kalai and Justin Thaler

Scribes: Sean Coughlin

Notes Delivery Owner (post notes by April 19th):

[[. . .]]

General Notes

The goal of the session was to identify advantages and disadvantages of interactive zero-knowledge protocols, and, upon weighing them, identify settings or applications where interactive protocols may be particularly suitable or relevant.

Some relevant background: Zero-knowledge protocols are often constructed in a two-step process. First, an information-theoretically secure protocol is constructed (e.g., an interactive proof, an interactive oracle proof, a PCP, a linear PCP, etc. As their names suggest, interactive proofs and interactive oracle proofs are interactive, while PCPs and linear PCPs are not). Second, the information-theoretically secure protocol is "compiled" via cryptographic techniques into a zero-knowledge argument, which may or may not be interactive. See Yuval Ishai's talk at this workshop for more information on this two-step approach to constructing zero-knowledge protocols.

Below is a list of advantages (or non-disadvantages) discussed:

- Efficiency, Simplicity, and Setup Assumptions: Interactive protocols can often be simpler than non-interactive ones, and may be more efficient as a result. They also may rely on simpler or weaker trusted setup assumptions (i.e., they typically do not require a structured reference string (SRS), the way many linear PCP-based arguments do.). Yet, if an interactive protocol is public coin, it can be rendered non-interactive in most settings via the Fiat-Shamir

heuristic (secure in the Random Oracle Model), often with little loss in efficiency. This means that protocol designers have the freedom to leverage interactivity as a "resource" to simplify protocol design, improve efficiency, or weaken or remove trusted setup, and still have the option of rendering the protocol non-interactive via Fiat-Shamir.

A concrete example of how interactive protocols can be simpler and accordingly more efficient than non-interactive variants: researchers attempted to render PCPs practical (e.g., BCGT, STOC 2013), but the resulting PCPs were still complicated, and fell short of practicality. More recent work (SCI, STARKs, Aurora, etc.) turned to interactive oracle proofs rather than PCPs, leveraging interaction to achieve simpler and more efficient protocols.

Space efficiency of the prover was also discussed. PCP- and linear-PCP (and even interactive oracle proof based arguments) often have large space requirements and require the prover to perform FFTs on big vectors (see efforts in the DIZK work that Howard talked about to distribute the prover due to these issues). Interactive proof based protocols currently seem more space efficient (e.g., GKR protocol works one circuit layer at a time, no FFTs), easier to distribute.

Note: in some specialized settings, the Fiat-Shamir heuristic is not known to be able to render a public-coin protocol totally non-interactive (see, e.g., https://eprint.iacr.org/2019/188 1)

- Interactive protocols can be based on weaker cryptographic assumptions than non-interactive ones (e.g., interactive proofs are information-theoretically secure, interactive oracle proofs can be compiled into interactive arguments based only on string commitments). In comparison, zkSNARKs are often based on knowledge assumptions.
- Many applications are inherently interactive (e.g., real-world networking protocols involve multiple messages just to initiate a connection). If an application is inherently interactive, why not leverage interaction as a resource to make a protocol simpler, more efficient, remove trusted setup, or use weaker crypto assumptions?
- Interactive protocols can potentially be run with fewer bits of security and hence be more efficient (adversaries may only have one try to break things in an interactive setting). Can also impose a time limit for the protocol to terminate, limiting the runtime of attackers, and thereby get away with fewer bits of security.
- Interactive protocols may be non-transferrable/deniable: the verifier cannot turn around and convince someone else of the validity of the statement. This can be essential in many applications. However, subtleties may arise if messages are signed by the prover (having the prover sign messages of an interactive protocol can make it transferrable).
- Zero-knowledge protocols are often combined with other cryptographic primitives in applications (e.g., oblivious transfer). If the other primitives are interactive, then the final cryptographic protocol will be interactive regardless of whether the zero-knowledge protocol is non-interactive.

Below is a list of disadvantages discussed:

- Currently, the zero-knowledge protocols with the shortest known proofs are based on linear PCPs, which are non-interactive. These proofs are just a few group elements. While (public-coin) zero-knowledge protocols based on interactive proofs or interactive oracle proofs can be rendered non-interactive with the Fiat-Shamir heuristic, they currently produce longer proofs (log or polylog field elements). The longer proofs may render these protocols unsuit-

able for some applications (e.g., public blockchain), but they may still be suitable for other applications (even related ones, like enterprise blockchain applications).
- Applying the Fiat-Shamir heuristic to an interactive protocol may increase soundness error.
- Network latency can make interactive protocols slow.
- Interactive protocols must occur online, i.e., the proof cannot simply be published or posted and checked later at the verifier's convenience.
- Also discussed was whether interactive protocols are more vulnerable to concurrency attacks on zero-knowledge (i.e., multiple malicious verifiers may interactive with a single prover and coordinate and interleave their messages to try to learn information from the prover).
- Many applications require non-interactivity.
- Because interactive protocols require the prover to send multiple messages, there may be more vulnerability to side channel or timing attacks (timing attacks will only affect zero-knowledge, not soundness, for public-coin protocols, since the verifier's messages are just random coins and timing attacks shouldn't leak information to the prover in this case. In private coin protocols, both zero-knowledge and soundness may be affected by these attacks).

Other topics that came up:

- Luis raised a possible error in the documents produced from the last workshop, regarding whether in public verifiable case, schemes where proofs are transferrable must be non-interactive. Luis will look into this.
- There was brief discussion about techniques other than Fiat-Shamir to remove interaction. Some exist but are not necessarily practical (e.g., Kalai, Paneth, Yang 2019).
- Verifiable Delay Functions

Suggested Contributions

Name	Email	Specific Contribution / Action Point
Ivan Visconti		Deniability
Luís T. A. N. Brandão		Statistical Security
Yupeng Zhang		Efficiency and trusted setup
Yael and Justin		Intro

Post #6: danib31 — May 16, 2019, 8:29am

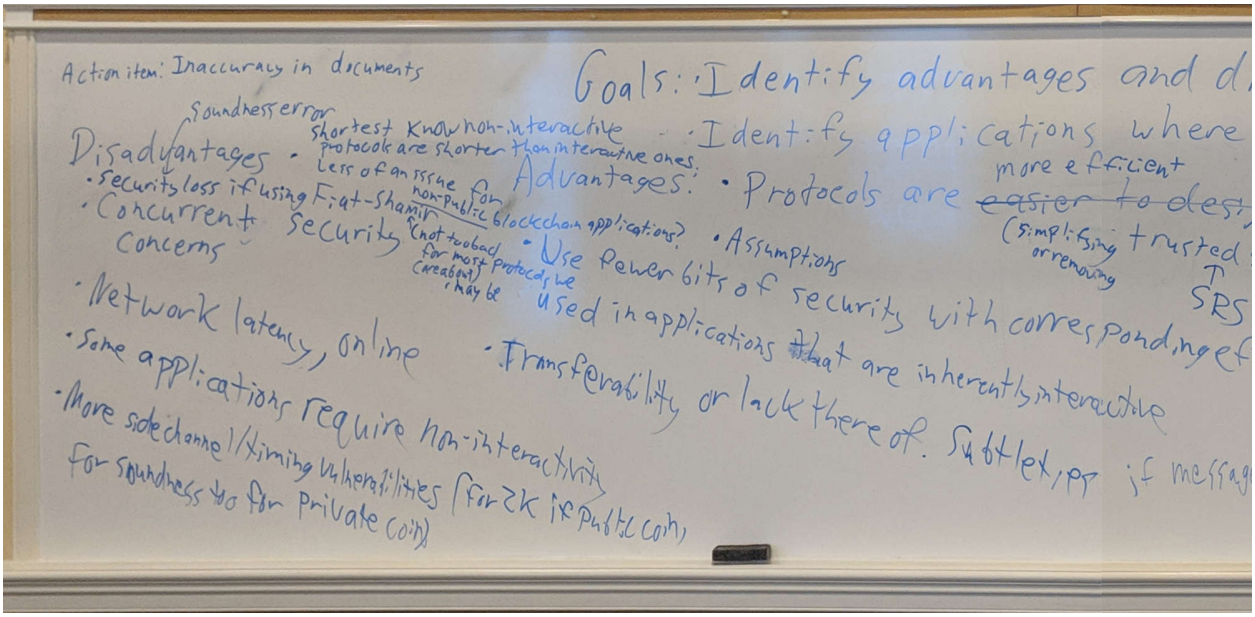
These are great notes! Thank you @Sean and @justin, it seems to me that the notes can (with minor editing and formatting) be included in the Reference Document as they are.

Two things I would like to add:

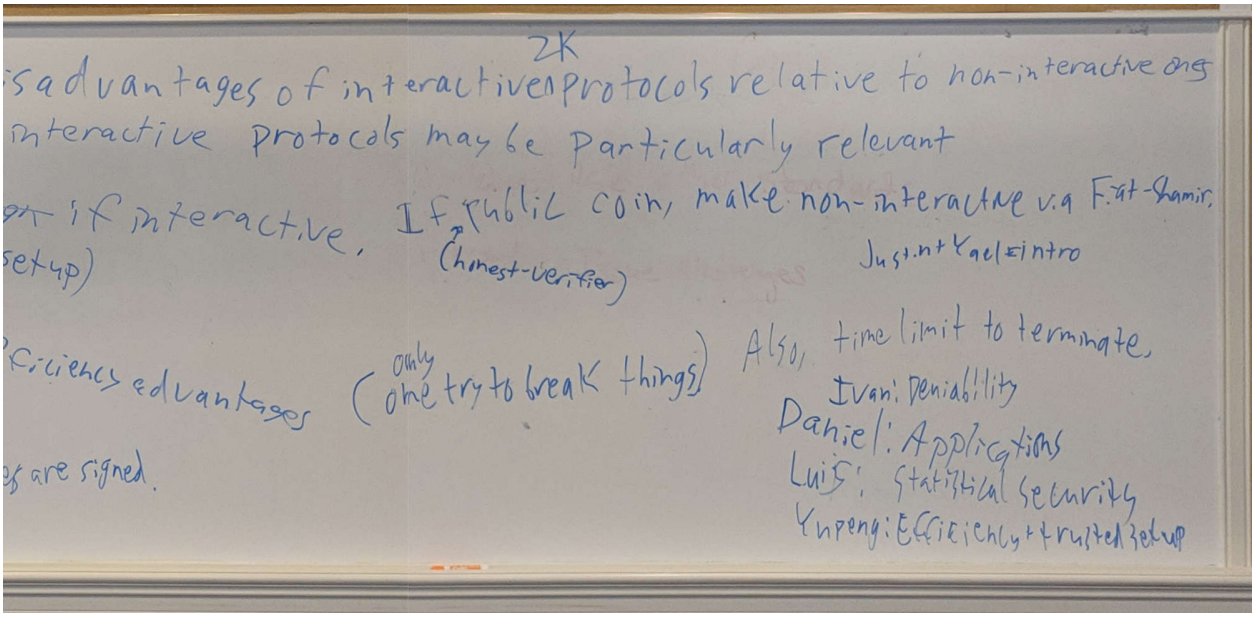
- I volunteered as well to help with the contributions, specifically about applications of interactive ZK. As we mentioned, there are use-cases where non-transferability of proofs is needed (see this quote):

> Interactive protocols may be non-transferrable/deniable: the verifier cannot turn around and convince someone else of the validity of the statement. This can be essential in many applications. However, subtleties may arise if messages are signed by the prover (having the prover sign messages of an interactive protocol can make it transferrable).

- I just wanted to leave these two pics here (from the board during the session) [[See Figure 6.]]



(a) left side of the board



(b) right side of the board

Figure 6: Photo in breakout 7 — photo of whiteboard (merge of 3 photos)

3.8 Breakout 8: ZK Protocol Security Analysis & Proofs of Correctness

Time: Friday April 12, 2019, 16:45–18:00

Moderators: Ariel Gabizon & Ran Canetti

Abstract: —

Informal notes from the ZKProof community forum:

Post #1: arielg — April 22, 2019, 2:26pm

Summary by Michele Orrù [[See Figure 7.]]

Summary by Vivek Arte:

NOTES:

How important are sanity checks?

- check for unused variables
- check for inconsistencies (between spec and code)
- prove security in GGM? (more for pairing based SNARKs)

How hard is it to build automated proofs for these analyses?

- CryptoVerif exists
- but we can manage with something simpler for these types of proofs?
- maybe just via information flow, etc?

for proofs, have multiple people check it independently (and from scratch)! - would help avoid simple errors

Think from the point of view of a regulator?

- even things related to composability?
- what NIST did for PQC
- defined different security levels - IND-CCA bits of security : level 1,3,5
- the assumptions used don't make any difference - the people have to put forward how they meet these levels
- have a competition - more people will look closely at the papers then.

proVerif has a GGM analyzer but it doesn't satisfy the win condition needed for SNARKs?

- is it something to improve? Who maintains the code?

composability issues??

how do you incentivize people to find bugs? bug bounties don't work.

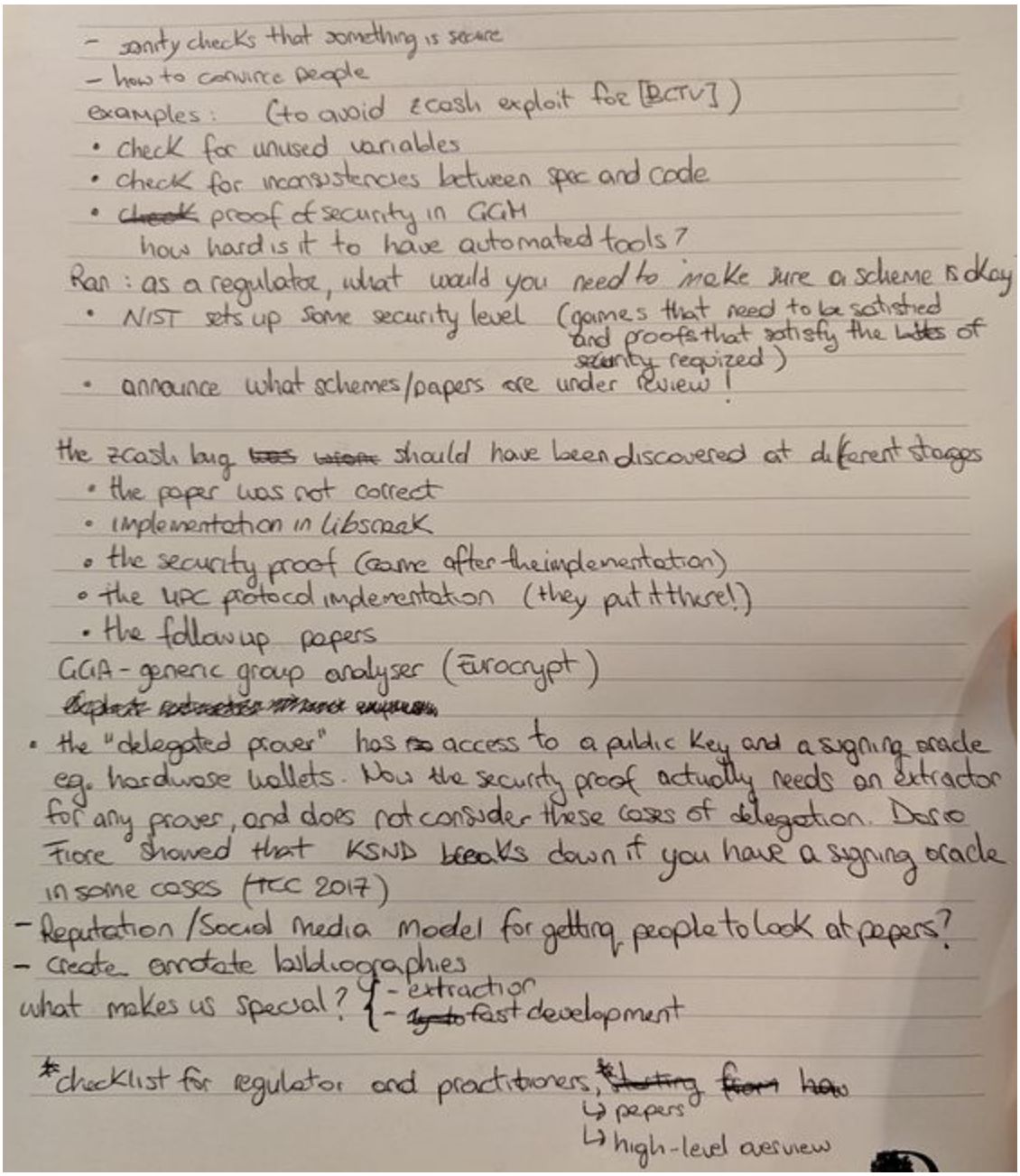


Figure 7: Photo in breakout 8 — Handwritten notes

- maybe something like a stackoverflow/wiki to explain papers?
- something like complexityzoo?

have dual implementations? as a fallback if the less tested portion fails?

Hydra? - 3 different ways, reject if they don't match up?

come up with a checklist for regulators and implementers?

go from what could go wrong to prescriptive guidelines for what should be done.

4 Proposals

This section transcribes the "call for proposals" (from the workshop website) and the scribed notes (posted in the community forum and some sent by email) from the discussions in the three "proposal" sessions. The Proposals Committee reviewed several proposals submitted prior to the workshop, and accepted four different ones to be presented at the workshop. While the workshop website included a call for "community standards", the editorial process introduced in the workshop redefined the terminology to "community proposals" instead. A compiled PDF version of each accepted proposal has since then been made available in the ZKProof GitHub repository. Two proposals — jR1CS and zkInterface — on the topic of interoperability of zero knowledge systems were combined into one session. The other two were presented separately: one was on the topic of commit-and-prove functionality; and another another was on the topic of generation of elliptic curves for ZK systems.

4.1 Call for community [standards] proposals

[[The following is transcribed from https://zkproof.org/workshop2/standards_proposals.html]]

Submit your Community Standard proposal by sending us a PDF to contact@zkproof.org

Community Standards. We are now accepting proposals for Community Standards on topics related to the security, implementation and applications of zero-knowledge proofs. Community Standards serve as guidelines agreed upon by the community, that promote correct usage and interoperability of zero-knowledge proofs. (Further work, to be defined in collaboration with standard bodies, will be required to gain official status as a standard.)

Submission Requirements. Submissions must be prepared in LaTeX in 11-point font; there is no page limit. We encourage submission of every proposal, even if not finalized, because even discussing in-progress proposals is valuable to the community. To submit your proposal send an email with the PDF document at contact@zkproof.org with every author in CC.

Submissions must have the following structure:

- Title: proposal title, author names, and affiliations.
- Scope: a section specifying the scope of the standard, highlighting what is being standardized and what is not.
- Motivation: a section describing at least one concrete application motivating the proposed standard, including an explanation of why the community will benefit from such a standard.
- Background: a section introducing the problem, including definitions, references to previous work and other background details.

And should address the following points:

- Terminology: for consistency across documents, adopt throughout the proposal, terminology and definitions used in the ZKProof proceedings, with pointers to the relevant sections.
- Challenges: for motivating the discussions, highlight the main challenges in creating such a standard, as well as any open or unresolved questions.

- Security: if relevant, provide a proof of security in the description.
- Implementation: if relevant, submit an open source prototype implementation (by including a reference to the repository with the code).
- Intellectual Property: we aim to ensure that proposals can be freely implemented. Thus, proposals should disclose the existence of any known patents (awarded or pending) which may restrict free implementation. This may affect the decision process, and a detailed policy is being developed.

Initial Timeline.

Each community standard proposal will be reviewed by the **Proposals Committee (PC)** and, if deemed appropriate, will be subsequently discussed in the second workshop. After the workshop, a second draft that includes the suggestions and comments from the discussions must be submitted. **Authors of accepted proposals must attend the workshop and must submit each draft in a timely manner**, according to the timeline below. After the initial submission, failure to submit the further drafts may result in a rejection of the proposal. Alternatively the **Steering Committee (SC)** may assign others to continue leading the proposal.

March 1st, 2019, 23:59 (UTC): submission of the first draft of proposals are due

March 20th, 2019: decisions by the PC are communicated to the authors

April 10th, 2019: community discussions start at the workshop, moderated by the PC

May 15th, 2019, 23:59 (UTC): submission of the second draft of proposals are due

Online Discussions. After the workshop, shepherds will be assigned to each working draft in order to moderate online discussions that will take place in an open forum (to be determined). The shepherds will attempt to reach consensus by the community on different topics and will publish a "Last Call" and due date for final comments and suggestions. Once these final changes have been made, the shepherds will review the final drafts and submit them to the SC for approval as a Community Standard.

Topics.

Proposals on any topic related to zero zero-knowledge proofs are welcome, including:

- **Terminology**
  - Motivation: the first thing to be standardized should be terminology, language and definitions. The field of zero knowledge is packed with terms and concepts that require careful definitions, which vary across the literature (see Security and Implementation tracks).
  - Proposal example: provide a unified glossary encompassing all (implicit and explicit) terminology in the proceeding documents.
- **Benchmarks**
  - Motivation: benchmarks are important for efficiency trade-offs. Today, there is no clear preferable construction since there are many trade-offs to consider.
  - Proposal example: a specific implementation of a functionality (e.g., an agreed-upon arithmetic circuit for SHA256) or by the functionality itself (leaving freedom to choose

the "most friendly" realization thereof).

- **Interoperability**
  - Motivation: many constructions have the ability to use Rank-1 Constraint Systems and it would be useful to have APIs and file formats standardized for interoperability. There is currently a mailing list for discussing interoperability of zkp systems and implementations.
  - Proposal example: here is an outline of topics that derived from the first workshop, and here is an initial proposal that was written.
- **Constructions**
  - Motivation: there are many constructions out there, yet some are seeing a lot of adoption and we want to encourage proper usage make sure that people are using them correctly. Having a standardized specification and test vectors will be beneficial to the industry.
  - Proposal example: see zkp.science for examples of scheme constructions and respective implementations.
- **Domain specific languages**
  - Motivation: one of the biggest bottlenecks for using zero knowledge systems is the difficulty in writing secure and robust constraint systems, for which a DSL would allow more adoption and usability.
  - Proposal example: see zkp.science for examples of DSL's for constraint system generation.
- **Protocols and Applications**
  - Motivation: ensuring that a zkp based system uses a secure protocol can be tricky, especially since each one can have different privacy requirements or caveats that are not easy to detect.
  - Proposal example: as is outlined in the Applications Track proceeding, some (if not most) use-cases share basic requirements (confidentiality, anonymity, etc...). Once can use Pedersen Hashes or SHA256 for committing to data and use RSA accumulators or Merkle TRees for set membership.

For any further questions, please email contact@zkproof.org

4.2 Proposals 1 and 2: Interoperability of Zero-Knowledge Systems

Source: Combines two threads (this and this) from the ZKProof community forum, corresponding to two separate proposals: "J-R1CS — a JSON LINES file format for R1CS" and "zkInterface, a standard tool for zero-knowledge interoperability".

Time: Friday April 12, 2019, 11:25–12:45

Moderators: Sean Bowe & Abhi Shelat

Hyperlinks: slide deck (Interoperability of Zero-Knowledge Systems)

Abstract for "J-R1CS — a JSON LINES file format for R1CS": Most ZK proof applications use R1CS to some extent, either as a producer (frontend) or as a consumer (backend). Having a standardized format for expressing R1CS will allow much better interaction and collaboration with-in the ZKProof eco-system. In response to the call for community standard, we have defined

a JSON lines format for describing a R1CS. You can find the proposal here.

Abstract for "zkInterface, a standard tool for zero-knowledge interoperability": [[TBA]]

Scribe: Daniel Benarroch

Informal notes from the ZKProof community forum

Post #1: aurel — April 30, 2019, 8:16am

We're starting to see more and better software that implements various zero-knowledge techniques, which is great.

There are different types of code out there:

1. Proving systems that generate and verify proofs.
2. Frameworks that let us express the statements to prove.
3. Collections of helper routines and gadgets.
4. Applications that implement specific statements and are distributed to end-users.

There's a parallel with classical software engineering to make:

1. Machine architectures, 2. Languages and compilers, 3. Functions and libraries, 4. Executable programs.

Today, these layers tend to be tightly coupled, and we find multiple stacks, written in different languages, with their own approach to each issue. This does make sense for now, as a team of experts can produce multiple layers together and deliver a specific product.

But zero-knowledge proving has potential for a variety of applications; not just in specialised products, but as a new technique engineers can use in larger systems, as they do today with encryption, signatures, etc. Meanwhile, complexity and requirements will increase, and there will be more teams throughout the world working on zero-knowledge software. We might even see some kind of equivalent to JavaScript, included in many products to execute on many clients. If any of this is to happen, the ecosystem will exhibit more and more software engineering practices. A particular impact may come from thinking in terms of systems, components, and interfaces, which is generally beneficial in multiple ways:

- Less effort required to build and review complex applications if it is made of components.
- The ability to coordinate the efforts of multiple teams who agree on the interfaces between components.
- More effective systems, as the most appropriate component can be chosen for each layer.
- Quicker deployment of innovations, when a new technique can be implemented in one layer and integrated with existing tools.

- Familiarity of concepts for non-specialists.

Some elements of system engineering do appear naturally in individual projects. Libsnark separates the proving systems from the gadgets through C++ interfaces. The implementation of Zcash Sapling looks a lot like a set of layers and components. In ZoKrates, the developer is exposed to the familiar concepts of programs being compiled and functions returning outputs from inputs; this independently of third-party proving systems and gadget libraries the compiler might support.

In this spirit, we put a lot of thoughts into how interoperability across today's and tomorrow's projects could work. We present a step in this direction as a proposal in the 2nd ZKProof standards workshop, which you can find at https://github.com/QED-it/zkinterface. Many thanks to all who did or will review and comment this proposal. Feel free to discuss here.

Post #2: danib31 — May 2, 2019, 9:30am

The above is one of the proposals, the second one, on JSON R1CS format, can be found here

Here are the notes of the discussion - feel free to comment and use them.

Here are the specific action points that were discussed (in this case to be part of improving the respective proposals and generate further discussion and a summary of the discussion for the reference document.

- @aurel and authors will update their proposal to include comments on dependencies, types, wasm comments.
- @Guillaume is there any specific update to the proposal you aim to do?
- @str4d, Sean Bowe, @jbaylina and myself will write up a summary for the reference document on the following topics:

1. Platform issue (browser vs native)
2. Calling convention issue (need protocol for calling gadgets between each other)
3. Interactivity and semantics
4. Dependency of libraries / languages - not try to reinvent (talk to people who have thought about this)

Please point out if there is anything I am missing or something more you think we should do / add.

Shoot the comments.

[[Below is the copy-paste of the above mentioned notes]]

zkInterface - https://community.zkproof.org/t/zkproof-proposal-interoperability-of-zero-knowledge-tools/86

J-R1CS - https://community.zkproof.org/t/zkproof-proposal-j-r1cs-a-json-lines-file-format-for-r1cs/117

LaTeX: https://drive.google.com/drive/u/2/folders/1HWZYMH-6Mx8wcX8geium506L0KRxcgPe

Moderators: Sean Bowe and Abhi Shelat

Scribes: Daniel Benarroch

Notes Delivery Owner: Daniel Benarroch

[[. . .]]

General Notes

1. Two presentations
    a. JSON for readability
        i. Flat file, then witness produced
        ii. Parsers available everywhere.
        iii. **Arbel:**
            1. could include comments generated by frontend
            2. Specify what lines refer to, good for extensibility
    b. zkInterface for gadget composability (like an API)
        i. Calling convention (need protocol for calling gadgets between each other; by making this protocol language-agnostic, can fulfill many deployment and interoperability scenarios)
        ii. Motivation for complexity of recursiveness?
            1. Special case is the flat format (can be json / binary and compact)
            2. Solve two use-cases (interop between frontends and composability of gadgets)
        iii. One issue could be at level of needing to use every library
            1. Can solve to call messaging to wasm
        iv. What about the versioning of the gadgets themselves?
            1. Include tag for the version of the gadgets
2. **Sean:** How about computing / representing -1 in constraints?
    a. In general how to encode field elements, powers of two, negative numbers in these formats?
3. On dependencies and versioning
    a. **Barry:** these two formats can create dependencies
    b. **Jack:** Message passing model also solves dependencies by outputting specific message to be used without import
    c. **Kostas:** what about gadget versioning in dependencies (Barry: can have gadget version tags)
4. On Uniformity
    a. **Abhi:** How to express this efficiently in zkInterface and others?

i. **Aurel:** Can add metadata, another field, but not enough experience in the industry
ii. **Eran:** the natural way is to introduce new messages, or vector type of the current ones [with multiple] witness assignments
iii. **Jack:**
    1. when understand high level structure of the statement, then can get conversion at the execution level.
    2. Enumerating variables needs to take care of connecting them properly.
iv. **Abhi:** but API should be extensible to allow for uniformity
v. **Jack:** Need extensibility mechanism within the API (but are we waiting to design the perfect API from the get go?

5. On composability
    a. Question: could you use Json files for the composability part and make it into a C program?
    b. Difficulty at using different programming languages
    c. **Jordi:** sometimes can optimize the composability of the gadgets, hence why use wasm natively. Would use flat file and then enumerate files with symbols.
    d. **Eran:** flat / monolithic vs composable format
        i. Many people would like to have textual representations for debugging, but no more functionality from readable format
        ii. Back end only needs low level representation for parsing
        iii. Binary compact format from ZKProof 1 https://github.com/str4d/zk
6. On Wasm
    a. Some use-case want to avoid complexity of deploying software, compiling dependencies
    b. Standard way of packaging witness generator in wasm
    c. Out of scope for the discussion, but the wasm programs could speak the standard protocol for interoperability between each other.
    d. Not all use-cases are covered by a wasm format. The standard protocol can be used in many other ways.
    e. There are not good enough tools for implementing in wasm natively all the things that people want to use
7. Mentioned that binary can be good and have files for variable names
    a. But backends only need to know the compact representation (so no readability)
    b. Some use-cases favor efficiency.
    c. The binary representation can be read using a tool (flatbuffers->json pretty print)
8. Javascript frontend?
    a. In enterprise models, cannot bring the code from outside
9. Mapping frontends to interface vs wasm (messaging vs language)

a. Linear scaling problem in interface definition when dealing with witness assignments, but not really.
b. Security conscious environments that include semantics

10. Calling conventions do not exist (usually have wrappers)
    a. Let's move away from semantics
11. Semantics vs variable assignment vs code that does assignment outside of circuit
12. Language to R1CS to circuit

Open questions

- What is the standard going to look like when extending to non-R1CS formats?
- What will a deployment of witness generation on the web look like?
- Start a wasm packaging effort?

Suggested Contributions

Name	Email	Specific Contribution / Action Point
ALL?		Online meeting?
Aurel Nicolas		Web assembly backing for zkInterface
Daniel, Jack, Sean, Jordi		1. Platform issue (browser vs native) 2. Calling convention issue (need protocol for calling gadgets between each other) 3. Interactivity and semantics 4. Dependency of libraries / languages - not try to reinvent (talk to people who have thought about this)

4.3 Proposal 3: Commit-and-Prove Functionality

Time: Friday April 12, 2019, 11:25–12:45

Moderators: Jens Groth, Yael Kalai, Mariana Raykova & Muthu Venkitasubramaniam

Hyperlinks: slide deck

Abstract: In a Commit-and-Prove system I can convince you that I know an opening u to a public commitment c and that this opening satisfies a certain property (e.g. it is a number in a certain range). One advantage of such systems (among others) is that I can publish a committed version of my data ahead of time and then later prove facts about them. Such commitment can

be compressing (i.e. very small compared to the original data). In order to allow interoperability among different applications (as well as other reasons), it may be worthy to start a discussion on a standard for this type of proof systems.

You can find a full draft of the proposal here.

Scribe: Carmit Hazay

Informal notes from the ZKProof community forum:

[[Omitted posts with abstract and link to slides]]

Email with notes, received by the editors on June 13, 2019: Notes

Matteo: Problem description: A commitment for the witness and a proof related to the commitment, the prover holds decommitment information.

Matteo: Need to have a single notion for CPZK. Two notions (1) where either the commitment (and commitment key) is generated at the setup phase and depends on the relation R, (2) or only after running the key generation algorithm.

Muthu: there is a single security notion of CP due to CLOS.

Jens: does not agree, there are complications we need to handle.

Yael: why should the commitment depend on R? A commitment key should be generated based only on a security parameter.

Mariana: in SRS constructions the commitment depends on R.

Jens: R is kind of the language and includes the field as well and vice versa.

Ivan: why do we need to have trusted parameters. Not always, we discuss generic notions.

Matteo: We should standardize notion B. All proofs share the opening stage. Commitments are the interoperability bottleneck. Agree on verification flavour of opening.

Angela: what is the point of opening the data? It is a ZK proof that the prover can open the data.

Ivan: for generality why shouldn't we allow interaction in the commit phase? Also holds for blockchain application.

Yael: can also use partial information of u.The additional functionality of Merkle trees is not captured by this notion. We may want to open a function of the data.

Daniel: This can be handled using Merkle trees and a more general notion.

Louis: There are two functions. One of the function we want to prove and the other function is about the committed data. This framework has two proofs, one regarding the SNARK and one

regarding the commitment. What is the focus here? Whether the opening relies on revealing the randomness of the commitment. The proof might not be a ZK.

Matteo: the syntax captures this case as well. o value is an output of the commitment phase. The second part of the proof is related to the integrity of the overall proof. Dario: programming language type problems when typing relations, should be verified at the time of verification.

Matteo: CP and relation representation, Intersects with other problems. Beyond the scope of CP.

END OF PRESENTATION

Mariant: useful to follow the three different tracks specified in the document. Start with what functionality definition we want to add as well as security definition.

Muthu: many variants of CP (as well as for ZK), couple of the principals are to see where are the applications. Practitioners should be able to frame the problem. Guiding principle is to be inclusive but check where are the practical contributions are (like with non-interactive ZK).

Dario: the motivation for this proposal. Current of implementations have different syntax. For instance, verification that takes the witness or precomputation of the witness. Following a common syntax will help.

Mariana: one phrase in CP model, allows easier composition. For what applications we need CP.

Louis: interesting use case, proving something regarding credentials. Useful CP functionality should support when the proven predicate is unknown yet. Transferability, prove something to another party in a non-transferable way. There may be cases where the opening phase is interactive.

Aviv: accumulated proof is different than CP due to non-membership proofs.

Muthu: Boils down to the difference between a single use and multiple usages.

Yael: what can we open to? In accumulators we can open an element or to a function of the data. Can try to understand applications.

Dario: accumulators CP for set-membership. MK is a commitment.

Daniele: one way to open a commitment to any function, to open to x and then compute f. Does not achieve privacy nor succinctness.

Dario: when modeling security relation is being sampled in an honest way.

Angelo: already deployed anonymous credentials, will not touch the system again. Need a legacy system and connect then to the new part. The function should be on the proof not on the commitment. Care much about verification time.

We should recommend which commitments should be used.

Louis: not good distinction between the commit and open phase. May have different properties. May have a simple commitment and then want to prove that the commitment value satisfies some properties. For compsability the commitment should be equivocal and extractable.

Daniele: enable prove equality of different commitments of different schemes. Could help in combining different commitments.

Dario: Legosnark follows this problem. Just another CP.

Muthu: don't have a concrete application. Only committing over the BC.

Daniel: want to use the actual reference of the commitment and need to keep track of it. In Zcash protocol, commiting to the output transaction.

Muthu: using SHA no need to worry about trusted setup.

Angelo: how to connect the two systems with snarks, cannot move to another scheme. Proofs of space are expensive. Need to prove correctness of such proofs.

Daniel: in Qedit take a big circuit, decompose into a smaller circuit and link the two circuits.

Action item: writing the full application of anonymous credentials. Angelo will organize that.

Eran: different applications rely on different properties of the commitments, binding, hiding. Definitions need to work for all cases. Recursive composition needs two properties.

Hitarshi: the BC commitment is important for filtering messages. A time stamp proof for an event is highly important.

Mariana: releasing statistics for DP. Can prove that released data is related to committed data. Moving to constructions.

Dario: using Pederson's commitment for compression is widely used and efficient and support relations. SHA256 is another solution. Have different properties.

Muthu: need to standardize commitments for that.

Louis: applications for additively homomorphic commitments.

Yael: need a feature of locally opening. In cases of using only part of the data.

Louis: can define what functionalities we need such equality, succinctness. Can be part of the appendix.

Dario: need to define the efficiency properties.

Action item: Dario and Matteo will be in charge the deinitional part.

Dario: must agree on definitions first before moving to API.

To which extends the composition of different commitment schemes will compose and not violate properties such as hiding.

Yael: should include constructions.

Mariana: constructions will be included in the appendix.

Angelo: need to make sure legacy systems can be securely combined with the proofs.

Muthu: can give examples.

Yael: regarding the definitions, need to go by option B. Need to focus on non-interactive and mention non-transferability.

Muthu: should be focused on non-interactive and add a note regarding interaction.

4.4 Proposal 4: Deterministic Generation of Elliptic Curves for ZK Systems

Time: Friday April 12, 2019, 16:45–18:00

Moderators: Sean Bowe & Alessandro Chiesa

Hyperlinks: slide deck (Generation of Elliptic Curves for Circuit Use)

Abstract: The standard aims to standarise the construction of elliptic curves for circuits based on different elliptic curve families. We are still working on the general standard proposal and the current draft only contains an example of a generated elliptic curve motivated by zkSNARK. In particular, given a prime p, we searched for a safe (meaning, satisfying SafeCurves critera) Montgomery curve defined over F_p using a deterministic algorithm to avoid speculation of trapdoors. The draft can be found here: https://github.com/bellesmarta/CallForProposals.

Scribe: ?

Informal notes from the ZKProof community forum:

[[Omitted slide with abstract]]

Post #2: Youssef — May 31, 2019, 3:32pm

The proposal addresses the generation of an embedded elliptic curve (of Edwards shape) over a given pairing-friendly elliptic curve (e.g. BN128 or BLS12-381). We can add to the proposal:

- The generation of an embedded pairing-friendly elliptic curve (of Weierstrass shape) if one wants to do in-circuit pairing based computations (e.g. BLS signatures).
- The generation of pairing-friendly EC cycles (e.g. MNT4 and MNT6 as used in Coda) if one wants to do recursive zkSNARKs.
- The generation of pairing-friendly EC chains (e.g. BLS12-377 and Cocks-pinch as used in Zexe) if one wants to do bounded recursive zkSNARKs or aggregate many proofs into one.

The second and third point require the generation of at least two EC where pairing-friendliness and high 2-adicity w.r.t. both the subgroup order and the field size are into consideration.