ABCDE ZK Hacker Camp Jul 22, 2023

Lecture 1 — Efficient ZKP: Theory and Practice

Instructor: Jiaheng Zhang *Scribes: Leo Fang*

1 Overview

This lecture was given by Dr. Jiaheng Zhang, an assistant professor at NUS and the Chief Scientist of Polyhedra. He started this lecture with the paradox of verification and privacy, introduced the basic definition and properties of zero-knowledge proofs (ZKPs), and elaborated on the design of ZKP protocols from both theoretical and practical perspectives. Finally, he discussed the prospects of hardware acceleration and compilation optimization for ZKP, as well as the intersection with machine learning.

2 Introduction to Zero-Knowledge Proofs

Verification is a critical process in ensuring the security of various systems such as bank and website log-in. It involves checking the validity of a claim or statement to ensure it is true. However, verification and privacy are two crucial concepts that often come into conflict. Verification requires access to information to prove a statement's validity, while privacy requires that personal information be kept confidential. This paradox can be resolved using Zero Knowledge Proofs (ZKPs), which allow for verification without revealing any information about the statement's validity.

ZKPs are mathematical constructions that allow one party to prove to another that a statement is true, without revealing any information about the statement beyond its truth. A ZKP consists of a witness w and a function $C(x, w)$ that maps an input x to an output y and a proof π. The proof must leak no information about w. ZKPs have three fundamental properties:

- Completeness: If the statement is true, then the verifier will be convinced of its truth by the end of the protocol. Specifically, if $C(w) = y$, then verification is guaranteed to succeed.
- Soundness: If the statement is false, then the verifier will be convinced of its falsity by the end of the protocol. Specifically, if $C(w) \neq y$, then verification is guaranteed to fail.
- Zero Knowledge: A proof that satisfies the completeness and soundness properties should leak no information about the witness w. This means that even if an attacker intercepts the proof, they will not be able to learn anything about w.

Taken together, these three properties make ZKPs a powerful tool for ensuring the security and privacy of various systems, especially blockchain and cryptocurrency. Blockchain technology has become increasingly popular due to its potential to provide secure and decentralized systems. However, it faces several challenges, including privacy and scalability. Specifically, blockchain's

transparent data on-chain poses privacy concerns, while its limited transaction processing speed (e.g., VISA: 24,000 TPS) presents scalability challenges.

ZKP can be used to address these problems. For example, privacy concerns can be addressed through the use of ZKP-based cryptocurrencies like Zcash, which provide privacy-preserving transactions while still allowing for verification of their validity. Meanwhile, zk-rollup can be used to improve scalability by directly placing proofs of batch transactions on the blockchain, rather than the entire transaction process.

In addition to blockchain, ZKPs can also be applied to machine learning. For example, if you want to construct a market for ML models, merchants in this market are supposed to prove the permformance of their model as well as the intergrity of the whole inference process, without leaking model parameters through ZKPs.

So here comes a critical question: How do we evaluate a good enough ZKP? In general, we hope that a good ZKP has following properties:

- For the verifier, the verification time should be faster than directly execute the computation.
- For the prover, the proof size should be succinct, with sublinear terms in $|w|$ and $|C|$.

Currently, the bottleneck of ZKPs is prover time, especially for larger computations and more complex functions. An example is given of a ZKP generated by Groth16, the protocol used in Zcash. In this example, we try to generate the ZKP of matrix multiplication:

$$C(w) = w \cdot w, \ \ w \in \mathrm{M}_{256\times 256}$$

The proof size is 192 bytes, the verification time is 3ms, but the prover time is over 1000s, which is unacceptable for industrial-grade applications. To address this issue, Jiaheng Zhang has conducted a series of works in both theory and practice, focusing on designing optimal provers for any function and tailored protocols for specific applications. In the following, we will introduce three of his works: **Libra**, **ZK Bridge**, and **ZK Decision Tree**.

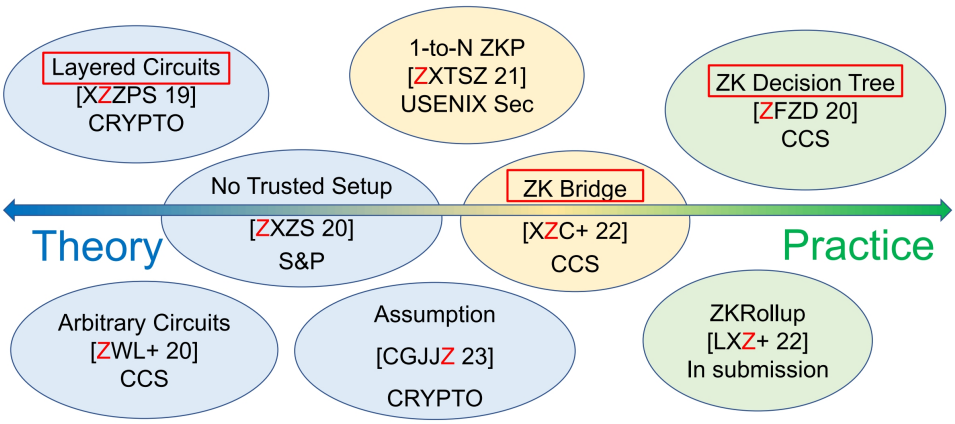

Figure 1: Jiaheng Zhang's Academic Works

3 Libra: Optimal Prover Time

Libra is a good example of a ZKP that meets our standards, as it achieves linear prover time, succinct proof, and fast verifier. To introduce the implementation details of Libra, we first need to discuss the building blocks: Sum-check protocol and GKR protocol.

3.1 Sum-check Protocol

Sum-check protocl uses the idea of "equality testing by comparing on a random point". Suppose we have a n-variate polynomial g over a finite field $\mathbb{F}$, we can generate a proof of the following sum with $O(\log n)$ proof size and $O(\log n)$ verification time:

$$H := \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} g(b_1, b_2, \dots, b_n)$$

The detailed implementation of sum-check protocol will be covered in Justin Thaler's lecture note. It should be noted that sum-check protocol isn't a ZKP.

3.2 GKR Protocol

GKR protocol is developed by Goldwasser, Kalai, and Rothblum. The computation is modeled as a layered arithmetic circuit C with a depth of d, which means wires only connect gates in adjacent layers.

Letting V_0 denot the output layer and $V_1, V_2 \dots, V_{d-1}$ is the intermediate layer, and the V_d is the input layer, with $w_1, w_2 \dots w_n$, n variables. The protocol then works its way in iterations towards the input layer, then we can use sum-check protocol to simplify it.

The advantage of GKR is that it implements light operations (only + and $\times$), but the disadvantage is that it does not have practical prover time ($O(|C|^3)$) and have no zero-knowledge property.

3.3 Implementation of Libra

Many cryptographic protocols have improved upon GKR, but they rely on assumptions about the arithmetic circuit (such as regular circuits or circuits with N different copies). Libra achieves linear prover time of $O(|C|)$ for general circuits, which is optimal for the prover (as at least one traversal of the program is required) and achieves zero-knowledge efficiently.

Firstly, we will achieve optimal complexity of $O(n)$ by applying the sum-check protocol to two variables simultaneously:

Phase 1: Define $g(b)$ as the sum over $b' \in [n]$ of $f(V_{i+1}(b), V_{i+1}(b'))$. Initialize $g(0), \dots, g(n-1)$ in $O(n)$ time. Perform sumcheck on the sum $\sum_{b \in [n]} g(b)$ in $O(n)$ time.

Phase 2: Define $h(b')$ as $f(V_{i+1}(r_{i+1}), V_{i+1}(b'))$. Initialize $h(0), \dots, h(n-1)$ in $O(n)$ time. Perform

sumcheck on the sum $\sum_{b' \in [n]} h(b')$ in $O(n)$ time.

Next, we will achieve zero-knowledge property. There are several solutions for achieving this, such as homomorphic encryption, which is not feasible due to its slow computation speed; adding a random polynomial directly to the computation, which increases prover overhead and nearly doubles computation cost. We finally add a mask polynomial to the original one, which only increases proof size by $O(\log n)$.

The protocol design of Libra provides a great example of how to design a feasible zero-knowledge proof system using simple math building blocks, as well as how to balance the trade-off between prover time and proof size. In addition to theoretical design, experimental verification is also needed to validate performance. Not only does Libra have excellent asymptotic performance, but its prover also outperforms all other ZKP systems in practice, while verification time and proof size are also very competitive.

[XZZ+19] [Tha22]

4 ZK Bridge

Cross-chain bridges enable the transfer of assets and information between different blockchain networks, They are considered the "highways" of the Web3 world, facilitating the interoperability and connectivity of different blockchain networks. However, the current solutions for cross-chain bridges often rely on a commitee-based method, which requires a trust assumption of the honesty majority. This can lead to poor performance or centralization, undermining the decentralization and security of the blockchain ecosystem.

We first note that the circuits employed by cross-chain bridges are data-parallel, consisting of multiple copies of a smaller sub-circuit. For example, the circuit for verifying N digital signatures duplicates the signature verification sub-circuit N times. To harness this data-parallelism, we propose **deVirgo**, a novel distributed zero-knowledge proof system based on **Virgo**. By distributing the proof generation process across T machines, deVirgo can reduce the prover time by a factor of T.

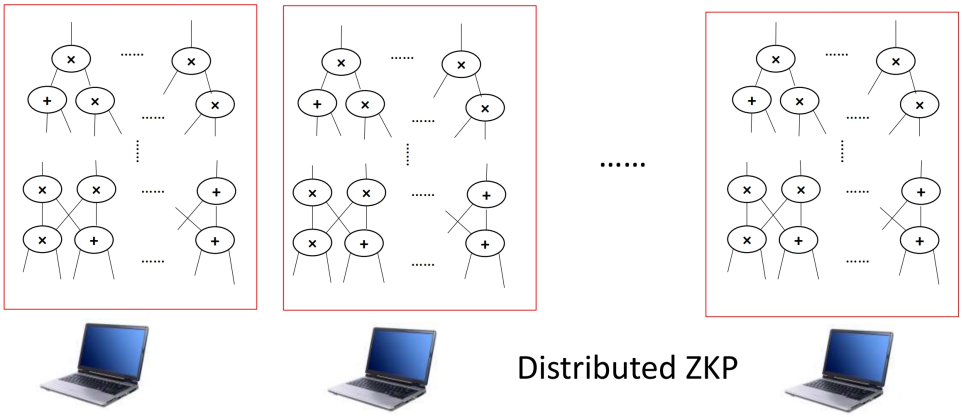

Figure 2: Data Parallel Circuits of zkBridge

Verifying deVirgo proofs on chain, particularly for the billion-gate circuits in zkBridge, can be computationally expensive for smart contracts with limited resources. To address this, we use **Groth16** to recursively prove the correctness of a deVirgo proof, resulting in constant-size proofs that can be quickly verified by a smart contract on an EVM blockchain.

[XZC+22]

5 ZK Decision Tree

Machine learning has a wide range of applications, but there are integrity issues that arise in terms of model validity and reproducibility. To address this, ZKP can be used to generate a proof for ML integrity without revealing the model parameters and input data.

One example of this is the ZK decision tree, where decision tree prediction is relatively simple, only requiring comparisons between values, and does not create an overflow issue. However, developing if-else statements in circuits can result in high overhead . To address this technical challenge, we validate the prediction instead of running the program. For example, the original input is $\mathbf{a}$, and $\bar{\mathbf{a}}$ is ordered by prediction path:

$$\mathbf{a} = [(1, \mathbf{a}[1]), (2, \mathbf{a}[2]), \ldots, (d, \mathbf{a}[d])]$$
$$\bar{\mathbf{a}} = [(i_1, \bar{\mathbf{a}}[1]), (i_2, \bar{\mathbf{a}}[2]), \ldots, (i_d, \bar{\mathbf{a}}[d])]$$

Then we assume the prediction path $\text{Path}_{\mathbf{a}} = v_1, v_2, ...v_h$ (v_i is the node of the decision tree). We simply compare $\mathbf{a}[i_j]$ with v_j one-by-one.The whole circuit consists of three parts:

- validate the prediction algorithm of the decision tree.
- check the permutation between $\mathbf{a}$ and $\bar{\mathbf{a}}$.
- check the validity of the prediction path in the committed decision tree.

[ZFZS20]

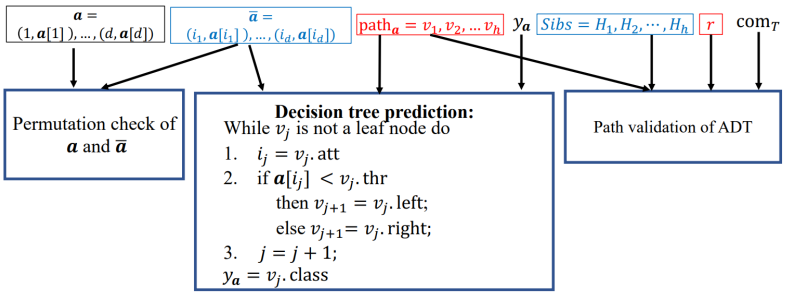

Figure 3: Process Diagram of ZK Decision Tree Prediction

By using this protocol instead of running the whole decision tree program, we can achieve zero-knowledge property and the optimal size $O(d+h)$ where d is attributes of the input and h is height of the tree.

In the future, there will be more applications of ZKML, which will eventually achieve off-chain computing and on-chain proof, combining AI and web3. With the use of ZKP, the integrity issues of machine learning can be addressed, ensuring that models are valid and reproducible, and contributing to the development of a more secure and trustworthy machine learning ecosystem.

6 Future Directions

Drawing an analogy to the development process of AI, the future of ZKP relies on the co-design of hardware and software, including ZKP-friendly hardware and hardware-friendly ZKP, as well as system-level optimization such as adaptive resource allocation. In addition, as ZKP becomes more widespread, there will be more intersections with other subfields of computer science, such as zkdatabase and zkLLM.

The big picture is to achieve optimal provers for *any* model (theory) and efficient ZKP for *any* application (practice). To achieve this goal, it is crucial for researchers in academia and engineers in the industry to work together.

References

[Tha22] Justin Thaler. *Proofs, Arguments, and Zero Knowledge*. 2022.

[XZC+22] Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3003–3017, 2022.

[XZZ+19] Tiacheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 733–764, Cham, 2019. Springer International Publishing.

[ZFZS20] Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 2039–2053, 2020.