(first)

Honorable Chairperson, Distinguished Experts, Dear Colleagues: It is an honor to present my research findings at this esteemed academic conference. I am Chenhao Shi from Shanghai Jiao Tong University. The purpose of this talk is to address the challenges associated with R1CS paradigm generation and propose a novel algorithm to overcome these challenges.

During my presentation at the academic conference, I will cover the following aspects of my research: Introduction to the relevant research background, Elaboration of the specific steps of the proposed algorithm, including the intermediate outputs of equivalent R1CS pairs, Presentation of the datasets used in the experiments and the corresponding experimental results, Summary of the unique contributions of my research and discussion of the limitations of the current study, and Exploration of future work and potential directions for further research.

First, I will begin by providing an introduction to the research background, highlighting the significance and challenges associated with R1CS paradigm generation. This will set the stage for the subsequent discussions.

Zero Knowledge Proofs (ZKPs) provide a method to prove a fact or statement without revealing any additional related information. It enables a prover to demonstrate possession of specific knowledge or authority to a verifier through a series of calculations, without disclosing specific details. In today's digital era, personal information has become a valuable commodity that is susceptible to commercialization and exploitation. With advancing technology, the risk of information leakage has increased. Within the cryptography community, there is a growing focus on addressing crucial challenges in blockchain, such as privacy and scalability. Blockchain technology finds widespread application in areas like Web3 and cryptocurrencies。However, applying ZKPs directly to real-world computational problems is not feasible. A complex transformation process is required. First, the real computational problems need to be transformed into arithmetic circuits. Then, these circuits are converted into Rank-1 Constraint Systems (R1CS) and subsequently into Quadratic Arithmetic Programs (QAPs). Finally, actual ZKPs are constructed for the QAPs. By leveraging this transformation process, ZKPs can be effectively applied to enhance privacy and security within various applications, including blockchain technology.

The conversion process from circuit languages to R1CS constraints in the underlying toolchain for zero-knowledge proofs faces several limitations. While R1CS is widely used to describe the execution of high-level programming language statements in many zero-knowledge proof applications, there is currently no standardized representation method, leading to challenges in mergeability. When merging Code Segment A and Code Segment B, the resulting R1CS may not be formally related to the R1CS generated independently by each segment. This makes it difficult to verify the correctness of the generated R1CS. Another limitation is raised by the flexibility in R1CS representations. Due to the same program semantics, multiple equivalent R1CS constraints can be generated. Moreover, merging and splitting constraints within the R1CS constraint set can alter the form of the R1CS. Consequently, it is crucial to propose a paradigmatic approach for constructing equivalent R1CS constraint groups. This would facilitate determining equivalence and correctness among different R1CS constraints. Such an approach would greatly benefit the verification of program equivalence and correctness, as well as enable further research on the mergeability aspect of R1CS.

(second)

In the next section, I will present the algorithm flow design. This section will outline the specific flow of the algorithm proposed in this thesis. I will achieve this by showcasing the intermediate outputs of a pair of equivalent R1CS constraint sets at different stages of the algorithm. These R1CS constraint sets are based on the same Circom circuits, as introduced in the blog of Vitalik Buterin, the founder of Ethereum. By analyzing the intermediate outputs, we can gain a deeper understanding of how the algorithm progresses and transforms the R1CS constraint sets throughout its execution. This step-by-step presentation will facilitate a clearer comprehension of the algorithm's inner workings and its impact on the R1CS constraint sets.

The algorithm starts by converting the equations associated with each R1CS constraint into arithmetic equations. These transformed equations are subsequently merged to form an arithmetic tree, where common sub-equations are stored as a Directed Acyclic Graph (DAG). During the algorithm's execution, the constraints are systematically transformed in a sequential manner. As each constraint is processed, new nodes are introduced to accommodate the variables specific to that constraint. At the same time, nodes that have been previously encountered are reused, promoting computational efficiency and minimizing redundancy. This process allows for the effective representation and organization of the arithmetic equations, enabling a streamlined approach to handle the constraints within the R1CS framework.

In this step, we employ the RNode data structure to store information regarding R1CS variables. An RNode can be treated both as a simple variable and as the outcome of an arithmetic subtree, with the RNode serving as the root node. This design choice aligns with the nature of R1CS, particularly when constraints are split and merged. Consequently, it minimizes the discrepancies arising from the decomposition and merging of constraints within R1CS.When merging and splitting constraints in R1CS, various aspects are affected, including the number and selection of intermediate variables, the quantity and form of constraints, and the variable mapping in specific R1CS compilers. However, upon comparing the two equations, we observe that these changes are not overtly apparent or significant. Nonetheless, during this step, there remains a disparity between the arithmetic trees generated by the equivalent R1CS. This discrepancy stems from the concatenation of equations within the R1CS constraint set. At present, our algorithm lacks sufficient information to determine the specific order of variable additions within the concatenated equations. As a result, it simply traverses the input R1CS without making such determinations. Therefore, further abstraction is necessary.

The next step involves the selection of tiles, where we divide the data flow graph generated in the previous step into customized tile types. In our design, we have specifically crafted three types of tiles: Quadratic, MulLinear, and AddLinear. We have chosen to focus on these basic tile types for several reasons. Firstly, we have temporarily deferred the inter-constraint merge step until subsequent steps acquire more information from the tree. This approach allows us to gather additional insights and make more informed decisions during the merge process. Secondly, by concentrating on the simplest tile types, we have streamlined the implementation of the tile selection step. This simplification enhances the algorithm's efficiency and reduces complexity in its execution. Lastly, by establishing the merge step based on the unmerged paradigm, we have further simplified the algorithm's design. This decision enables a more straightforward implementation and improves the overall coherence of the algorithm.

(third)

The process of tile selection is shown below. In the process of selecting each tile, we choose a node in the current data flow graph that does not have any successor nodes as the root node. We then select the tiles that meet our criteria based on this root node, and remove the isolated nodes generated by the selection of the new tile from the original graph.

In the third step, we perform abstraction on the Linear tiles to generate an abstracted data-flow graph that corresponds to the equivalent R1CS. As previously discussed in the graph construction step, the disparity between the data-flow graphs generated by the equivalent R1CS constraint groups lies in the order of summation between nodes when dealing with Linear tiles. However, if we consider the summed nodes within the selected linear tiles as a set, they become identical once again. This implies that the difference in the order of summation during the tile selection process merely reflects the order in which nodes are added to the linear tile. To deal with this situation，we abstract linear tiles into a single large abstract node, which shields the specific internal structure of the tile from external nodes. As a result, we eliminate the differences in specific addition orders within linear tiles, and establish connections from external nodes to the specific tile to which a node belongs. This process ensures the generation of a consistent data flow graph. The abstracted data flow graph is structurally simpler compared to the pre-abstraction graph. There are several types of edges:

* Non-linear tile abstract node to non-linear tile abstract node: These connections already exist in the pre-abstraction data flow graph and remain consistent.
* Non-linear tile abstract node to linear tile abstract node: This connection exists only when the abstract node represents a linear tile that contains non-abstract nodes.
* Linear tile abstract node to linear tile abstract node: This connection exists only when the two abstract nodes represent linear tiles that share non-abstract nodes.

Then we employ a weighted PageRank algorithm to compute the weights of nodes within the abstracted data flow graph. These weights serve as the foundation for determining the weights of the corresponding constraints associated with the tiles. Unlike the traditional PageRank algorithm, our approach assigns weights to each edge in the graph and adjusts the iterative formula for node weights. The primary objective of utilizing the weighted PageRank algorithm is to mitigate the symmetry present within the abstracted data flow graph. As a result of simplifying the linear tiles in the previous step, the structure of the data flow graph has undergone substantial simplification. This simplification has introduced symmetry in certain graph structures. If a generic algorithm were used to calculate the weights of nodes within the graph, symmetrically positioned nodes would tend to yield identical weights. This can pose challenges for subsequent constraint-related tasks and sorting processes. To address this issue, we leverage the weighted PageRank algorithm to assign distinct weights to different nodes. This approach enhances the asymmetry of the graph, thereby minimizing the occurrence of identical scores as much as possible.

(forth)

In the last step of our algorithm, variables that exclusively appear in linear constraints are sorted. The process of determining constraint divisions in the R1CS paradigm takes place during the tile division step, while the ordering of constraints in the R1CS paradigm is established during the tile weight computation step. Additionally, the previous step determines the position of variables that appear in secondary constraints within the variable mapping. Consequently, only the positions of variables that solely appear in linear constraints remain to be determined until the final R1CS paradigm is generated. This is expected since the abstraction step of the data flow graph eliminates the influence of the internal specific structure of linear variables on the overall data flow graph structure. As a result, insufficient information is available in the preceding steps to facilitate sorting of these variables. To address this, this paper introduces a novel weight sorting method for these variables. In each newly added linear tile, the weight assigned to each newly introduced variable is determined as the sum of the absolute values of the products derived from multiplying the coefficients of that variable in the linear tiles, excluding its own linear tile, with the respective tile weights. The formulation of this equation is primarily based on representing the importance of variables within the entire constraint group by considering their occurrences in individual constraints. Simultaneously, for variables that solely appear in a single linear constraint, their weights calculated using this equation are zero. This implies that the sorting of such variables has minimal impact on the overall constraint group since their coefficients in other constraints are all zero.

The next section is EVALUATION, this section is going to introduce the test set designed in this thesis and show the effect of testing the implemented algorithm on data.

Currently, the field lacks a comprehensive test set due to limited research in this area. To generate the data set of our research, we have conducted extensive algorithm testing and identified patterns that guided us in carefully generalizing and dividing the generation of equivalent R1CS constraint sets. Based on these derived patterns, we have designed a more comprehensive test set. Each category within the "Separated Generation" test set comprises two to three underlying R1CS constraint groups. To thoroughly assess the algorithm's robustness and correctness, we have generated five to six equivalent R1CS constraint groups for each constraint group, depending on the specific category. These equivalent constraint groups are then inputted in pairs into the algorithm for testing, ensuring that the algorithm consistently produces outputs that align with the R1CS paradigm definition when handling different equivalent constraint groups.

The last section is conclusion. it mainly introduces the main contribution of this paper and future work.

This paper introduces a novel data-flow-based algorithm for constructing the normal form of semantically equivalent R1CS, a programming language widely used for Zero-Knowledge Proofs . Additionally, we have developed a dedicated dataset for equivalent R1CS to facilitate our research.

In future work, we aim to establish comprehensive rules for merging constraints and conduct an extensive benchmarking analysis. This will involve conducting in-depth research and exploration of the generation rules of R1CS to further enhance the algorithm's performance and capabilities.

Thank you for your attention and active participation. Through this presentation, we hope to deepen our understanding of R1CS and provide valuable insights for validating the correctness and mergeability of R1CS in future endeavors. We sincerely appreciate your attention and support. If you have any questions or comments, I would be delighted to engage in further discussion. Thank you all.