(first)

1. Esteemed Chairperson, Distinguished Experts, Dear Colleagues: I am delighted to have the opportunity to present my research findings at this prestigious academic conference. My name is Chenhao Shi, and I represent Shanghai Jiao Tong University. The objective of this presentation is to tackle the obstacles associated with R1CS paradigm generation and introduce a groundbreaking algorithm designed to surmount these hurdles.
2. During my presentation at the academic conference, I will address the following key 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.
3. First and foremost, I will commence my presentation by delivering an insightful introduction to the research background, focusing on the importance and complexities surrounding R1CS paradigm generation. This initial discussion will effectively establish the context and relevance of the subsequent topics to be explored.
4. Zero Knowledge Proofs (ZKPs) offer a powerful means to substantiate a fact or statement without divulging any ancillary information. They enable a prover to demonstrate possession of specific knowledge or authority to a verifier through a series of calculations, all while safeguarding sensitive details. In our increasingly digitized world, personal information has evolved into a highly valuable asset vulnerable to commercial exploitation. With technological advancements, the risks of information leakage have escalated. Consequently, the cryptography community is actively engaged in addressing pivotal challenges, notably privacy and scalability, in the realm of blockchain. Blockchain technology, with its wide-ranging applications in Web3 and cryptocurrencies, necessitates novel approaches to fortify privacy and security.()However, applying ZKPs directly to real-world computational problems is not straightforward. It entails a complex transformation process. Initially, the underlying computational problems must be converted into arithmetic circuits. These circuits are subsequently transmuted into Rank-1 Constraint Systems (R1CS) and eventually into Quadratic Arithmetic Programs (QAPs). The final stage involves constructing actual ZKPs for the QAPs. By harnessing this transformation process, ZKPs can be effectively leveraged to enhance privacy and security across diverse applications, including the domain of blockchain technology.
5. The conversion process from circuit languages to R1CS constraints in the underlying toolchain for zero-knowledge proofs encounters several limitations that warrant attention. While R1CS serves as a prevalent means to depict the execution of high-level programming language statements in numerous zero-knowledge proof applications, the absence of a standardized representation method poses challenges in terms of mergeability. When merging Code Segment A and Code Segment B, the resulting R1CS may lack formal association with the R1CS independently generated by each segment. This discrepancy impedes the ability to verify the accuracy of the resulting R1CS. Additionally, the flexibility inherent in R1CS representations introduces another constraint. Given the same program semantics, multiple equivalent R1CS constraints can be generated. Furthermore, the act of merging and splitting constraints within the R1CS constraint set can alter the structure of the R1CS. Thus, it becomes imperative to propose a paradigmatic approach for constructing equivalent R1CS constraint groups. Such an approach would facilitate the determination of equivalence and correctness among different R1CS constraints. Moreover, it would greatly enhance the verification of program equivalence and correctness while enabling further research on the mergeability aspect of R1CS.

(second)

1. In the forthcoming section, I will elucidate the algorithm flow design. This segment aims to provide a comprehensive overview of the specific flow of the algorithm proposed in this thesis.
2. To accomplish this, I will showcase the intermediate outputs of a pair of equivalent R1CS constraint sets at various stages of the algorithm. These R1CS constraint sets are derived from the same Circom circuits, as documented in the blog authored by Vitalik Buterin, the esteemed founder of Ethereum. By meticulously analyzing these intermediate outputs, we can acquire a profound understanding of how the algorithm progresses and orchestrates transformations within the R1CS constraint sets throughout its execution. This step-by-step presentation endeavors to facilitate a lucid comprehension of the inner workings of the algorithm and its profound implications on the R1CS constraint sets.
3. The algorithm commences by converting the equations associated with each R1CS constraint into arithmetic equations. These converted equations are then consolidated and merged to construct an arithmetic tree, wherein common sub-equations are stored as a Directed Acyclic Graph. Throughout the execution of the algorithm, the constraints undergo systematic transformations in a sequential manner. For each processed constraint, new nodes are introduced to accommodate the variables specific to that constraint. Simultaneously, previously encountered nodes are reused, fostering computational efficiency and minimizing redundancy. This systematic approach facilitates the effective representation and organization of arithmetic equations, thereby enabling a streamlined handling of constraints within the R1CS framework.
4. The RNode data structure is employed to store essential information related to R1CS variables. An RNode can be treated as both a simple variable and the result of an arithmetic subtree, with the RNode serving as the root node. This design choice is aligned with the inherent nature of R1CS, especially when dealing with the splitting and merging of constraints. Consequently, it helps minimize disparities that may arise from the decomposition and merging of constraints within the R1CS framework. ()When merging and splitting constraints in R1CS, several aspects are influenced, including the number and selection of intermediate variables, the quantity and form of constraints, as well as the variable mapping in specific R1CS compilers. However, upon comparing the two equations, it is observed that these changes are not overtly apparent or significant. Nevertheless, during this step, there remains a discrepancy between the arithmetic trees generated by the equivalent R1CS instances. This discrepancy arises from the concatenation of equations within the R1CS constraint set. Currently, our algorithm lacks sufficient information to determine the specific order of variable additions within the concatenated equations. Consequently, it traverses the input R1CS without making such determinations. Therefore, further abstraction is required.
5. 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)

1. The process of tile selection is illustrated as follows. During the selection of each tile, we identify a node within the current data flow graph that does not possess any successor nodes, designating it as the root node. Subsequently, we choose the tiles that satisfy our predetermined criteria, utilizing this root node as a reference point. Following the selection of the new tile, we eliminate the isolated nodes that emerge as a result of this selection from the original graph.
2. In the third step, we undertake abstraction on the Linear tiles to generate an abstracted data-flow graph that corresponds to the equivalent R1CS. As discussed earlier in the graph construction step, the disparity observed between the data-flow graphs generated by the equivalent R1CS constraint groups stems from the order in which nodes are summed within 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 summation order during the tile selection process merely reflects the order in which nodes are added to the linear tile. ()To address this situation, we abstract linear tiles into a single large abstract node, effectively concealing the specific internal structure of the tile from external nodes. Consequently, we eliminate the variations 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 exhibits a simpler structure compared to the pre-abstraction graph. Several types of edges exist within this graph:

* 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.

1. Subsequently, we utilize 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. In contrast to 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 employing 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. However, 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. Such symmetry can pose challenges for subsequent constraint-related tasks and sorting processes. To address this issue, we leverage the weighted PageRank algorithm, which assigns distinct weights to different nodes. This approach enhances the asymmetry of the graph, minimizing the occurrence of identical scores as much as possible.

(forth)

1. In the final step of our algorithm, we address the sorting of variables that exclusively appear in linear constraints. The process of determining constraint divisions in the R1CS paradigm occurs 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. Therefore, the positions of variables that solely appear in linear constraints remain to be determined until the final R1CS paradigm is generated.()This situation arises due to the abstraction step in the data flow graph, which eliminates the influence of the internal specific structure of linear variables on the overall data flow graph structure. Consequently, the preceding steps do not provide sufficient information to facilitate the sorting of these variables. To address this, we propose 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. This formulation is 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, as their coefficients in other constraints are all zero. ()By introducing this weight sorting method, we can effectively handle the sorting of variables that exclusively appear in linear constraints, completing the generation of the final R1CS paradigm.
2. 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.
3. Currently, the field lacks a comprehensive test set due to limited research in this area. To address this gap, we have generated our own test set for the research by conducting extensive algorithm testing. Through this testing, we have identified patterns that have guided us in carefully generalizing and dividing the generation of equivalent R1CS constraint sets. ()Using these derived patterns, we have designed a more comprehensive test set known as the "Separated Generation" test set. This test set consists of multiple categories, with each category containing two to three underlying R1CS constraint groups. ()To ensure a thorough assessment of 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. This approach allows us to verify that the algorithm consistently produces outputs that align with the R1CS paradigm definition when handling different equivalent constraint groups.
4. The last section is conclusion. it mainly introduces the main contribution of this paper and future work.
5. This paper presents a novel data-flow-based algorithm for constructing the normal form of semantically equivalent R1CS, a programming language widely used for Zero-Knowledge Proofs. Furthermore, we have created a dedicated dataset for equivalent R1CS to support our research efforts.
6. In future work, our objective is to establish comprehensive rules for merging constraints and conduct an extensive benchmarking analysis. This will involve conducting thorough research and exploration of the generation rules of R1CS, with the aim of further enhancing the algorithm's performance and capabilities.
7. We sincerely appreciate your attention and active participation. Through this presentation, we hope to deepen our understanding of R1CS and provide valuable insights into validating the correctness and mergeability of R1CS in future endeavors. Thank you all for your attention and support. If you have any questions or comments, I am available for further discussion. Thank you.