1. Honorable Chairperson, Distinguished Experts, Dear Colleagues: Good morning/afternoon/evening to you all. I am very honored to present my research results to you at this important academic conference. I am Chenhao Shi, from Shanghai Jiaotong University,. The goal of this talk is to explore the problems related to R1CS paradigm generation and to propose a new algorithm to cope with the problem of paradigm generation.

2. First, I will introduce the relevant research background. Then, I will elaborate the specific steps of the proposed algorithm by showing the intermediate outputs of a pair of equivalent R1CS during the algorithm. Then, I will present the datasets used in the experiments as well as the experimental results. Finally, I will summarize the unique contributions of my research and discuss the limitations of my current research as well as FUTURE WORK.

3. First, let me introduce the research background of the thesis

4. (Introducing Zero Knowledge Proofs) Zero Knowledge Proofs are a method by which a fact or statement can be proved without having to reveal any other information related to that fact or statement. Its basic principle is to allow the prover to prove to the verifier that he or she possesses a certain knowledge or authority through a series of calculations, without revealing specific information. (Primary Application) In this digital age, personal information has become a commodity that can be commercialized and exploited, and as technology advances, the risk of leakage of this information increases. Increasingly, the cryptography community is attempting to address some of blockchain's most important challenges: privacy and scalability. In addition to this, blockchain is also widely used in Web3, cryptocurrencies and other important areas. The blockchain is also widely used in important areas such as Web3, cryptocurrencies, etc. (complex transformation process). However, zero-knowledge proofs cannot be directly applied to any real-world computational problems. We need to transform the real computational problems into arithmetic circuits first, then transform the arithmetic circuits into R1CS, then into QAPs, and finally create the actual zero-knowledge proofs for the QAPs.

5. There are many limitations in this step of conversion from circuit languages to R1CS constraints in the underlying toolchain for zero-knowledge proofs. R1CS describes the execution of statements written in a high-level programming language and is used by many zero-knowledge proof applications, but there is currently no standardized representation. (Limited Mergeability) This results in poor mergeability of R1CS. The R1CS generated by merging Code Segment A and Code Segment B may be formally unrelated to the two R1CSs generated independently by Code Segment A and Code Segment B, which makes it difficult to verify the correctness of the generated R1CSs. (Flexible R1CS Representations) In addition to the limitations of the R1CS itself, the fundamental reason is that a program with the same semantics can generate multiple equivalent R1CS constraints, and the merging and splitting of constraints in the R1CS constraint set can also lead to changes in the form of the R1CS. (Research Implications) Therefore, we need to propose a paradigmatic way of constructing R1CS constraint groups in equivalent R1CS constraint groups, which makes it easier for us to determine the equivalence and correctness for different R1CS constraints. This will be of great benefit to us in verifying the equivalence as well as the correctness of the program, including the subsequent more in-depth study of the mergeability aspect of R1CS.

6. The next section is the algorithm flow design. This section describes the specific flow of the algorithm proposed in this thesis.

7. In this section, I will present the exact flow of the algorithm FOR BETTER UNDERSTANDING through the intermediate outputs of a pair of equivalent R1CS constraint sets at various stages in the algorithm. R1CS was introduced in V-God's blog, corresponding to the same Circom circuits

8. (Main Step) transforms the equations that need to be satisfied by each constraint R1CS constraint into arithmetic equations, and then combines these transformed equations together to obtain an arithmetic tree with common subequations stored as a DAG. In the algorithm, we transform the constraints in turn, creating new nodes for the newly appearing variables in them and reusing nodes that have already appeared in the previous constraints.

9. (LEFT Black text above) In algorithms, we use a special data structure, the RNode, to store information about R1CS variables. Any RNode can be treated both as a simple variable and as the result of an arithmetic subtree with it as the root node. This design is consistent with the characteristics of R1CS when constraints are split and merged, and thus also minimizes the differences caused by the decomposition and merging of constraints in R1CS. (left gray) In R1CS, the merging and splitting of constraints have many effects on R1CS, such as the number and choice of intermediate variables, the number and form of constraints, and variable mapping in particular R1CS compiler, but according to the comparison of the two equations, we can find that these changes are not very obvious in the equations. are not very obvious. (right) In this step, there is still a difference between the arithmetic trees generated by the equivalent R1CS, which comes from the concatenation of equations in the R1CS constraint set. In this step, our algorithm does not have enough information to determine the order of addition of the variables in the concatenated equations, but simply traverses the input R1CS. therefore, further abstraction is needed.

10. The second step is the tile selection process. In this step, we split the data flow graph generated in the previous step into customized types of tiles. We designed three tile types: Quadratic, MulLinear and AddLinear.We only designed the simplest tiles for three consideration.First, we firstly put the inter-constraint merge step on hold until the subsequent steps get more information from the tree, secondly, the algorithmic implementation of the tile selection step is simpler, and finally, the merge step is built on the unmerged paradigm. algorithm is simpler to implement, and finally, the design of the algorithm for this step is simplified by basing the merged step on the unmerged paradigm.

11. The process of tile selection is shown below.

12. In the third step, we abstract the Linear tiles to generate the same abstracted data-flow graph for the equivalent R1CS.As mentioned in the graph-constructing step, the difference between the data-flow graphs generated by the equivalent R1CS constraint groups is that the order of summation between the nodes when dealing with the Linear tiles is different. However, if the summed nodes in the selected linear tiles are considered as a set, they are the same again. In other words, the difference in the order of summation in the tile selection process simply means a difference in the order in which the nodes are added to the linear tile.

In this step, we abstract the linear tile into a large abstract node, and shield the concrete internal structure from external nodes that are not contained in that tile. In this way, we mask the differences in the order of summation within the linear tile, so that the connection from an external node to a specific node in the linear tile becomes a connection to the tile to which the node specifically belongs. This results in a consistent data flow graph.

The data flow graph after abstraction has a simpler structure compared to the pre-abstraction one. There are several types of edges:

\* Nonlinear Tile Abstract Node to Nonlinear Tile Abstract Node: both vertices are present in the dataflow graph before abstraction. Consistent with the pre-abstraction dataflow graph.

\* Non-Linear Tile Abstract Node to Linear Tile Abstract Node: exists if and only if there is a non-abstract node in the linear tile represented by the abstract node.

\* Linear Tile Abstract Node to Linear Tile Abstract Node: exists when and only when there is a public non-abstract node for the linear tile represented by the two abstract nodes.

13. In this step, a weighted PageRank algorithm is used to compute the weights of the nodes in the abstracted data flow graph, which in turn is used as a basis to compute the weights of the constraints corresponding to the tiles. As opposed to the traditional PageRank algorithm, this algorithm assigns weights to each edge in the graph and adjusts the iterative formula for node weights.

The main purpose of this algorithm using weighted PageRank is to reduce the symmetry of the abstracted data flow graph. Due to the simplification of the linear tiles in the previous step, the structure of the data flow graph has been simplified substantially, and symmetry exists in some of the structures in the graph. If a general algorithm is used to calculate the weights of the nodes in the graph, the nodes that exist symmetrically in the structure of the graph are prone to calculate the same weights, which can cause trouble for the subsequent problems on the constraints as well as the sorting, so the weighted PageRank algorithm is utilized to calculate weights for the different nodes to increase the asymmetry of the graph, thus avoiding the appearance of the same scores as much as possible.

14. (Adjustment of Scope) Variables that have only appeared in linear constraints are sorted in this step.

In the tile division step, the division of constraints in the R1CS paradigm is determined; in the step of tile weight computation, the ordering of constraints in the R1CS paradigm is determined; and in the previous step, the position in the variable mapping of variables that have appeared in secondary constraints is determined. Thus, only the positions in the variable mappings of those variables that have appeared only in the linear constraints remain to be determined until the final R1CS paradigm is generated. This is normal because in the abstraction step of the data flow map, the influence of the internal specific structure of the linear variables on the overall data flow map structure is eliminated. So there is not enough information to sort the variables in it in the previous steps.

(Sorting Criterion) For this type of variables, this paper proposes a new weight sorting method, the formula is shown in Figure

That is, in each newly added linear tile, for each newly introduced variable, the weight is the sum of the absolute values of the products of the coefficients of that variable in the linear tiles other than its own linear tile and the weight of that tile.

15. 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.

16. (benchmark ck) Due to the lack of related research, there is no very complete test set in this field at present. So we generalized and divided the generation of equivalent R1CS constraint sets more carefully through some laws found in the process of algorithm testing, and designed a more complete test set based on the derived laws.

Each category in the (Separated Generation) test set contains two to three underlying R1CS constraint groups. In order to test the robustness and correctness of the algorithm more comprehensively, five to six equivalent R1CS constraint groups were generated for each constraint group, depending on the reason corresponding to the category. The equivalent constraint groups for each constraint group were input into the algorithm in pairs for testing to verify that the algorithm was able to generate outputs that were consistent and in accordance with the definition of the aforementioned R1CS paradigm when dealing with different equivalent constraint groups.

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

18. This paper proposes a data-flow-based algorithm for constructing the normal form of a group of semantically equivalent R1CS, a high-level programming language used for ZKP. And design a dataset for equivalent R1CS

19. Future work includes establishing rules for merging constraints and conducting a more comprehensive benchmark. This requires us to conduct more in- depth research and exploration of the generation rules of R1CS to improve the algorithm. depth research and exploration of the generation rules of R1CS to improve the algorithm.

20. Thank you for listening and participating. Through this debriefing, we hope to increase our understanding of R1CS and provide valuable information for future validation of R1CS correctness as well as mergeability. Thank you for your attention and support. If you have any questions or comments, I would be happy to discuss them with you further. Thank you all!