Improved Lightweight Formal Verification of Halo2 Proof Systems

Primary Contact:	Jan Gorzny
Amount requested:	$20,000 USD (upon completion of grant)
Grant Description:	[Introduction & Goals] Zero-knowledge (ZK) proof systems introduce new sources of risk to software by translating software into complex mathematical representations. These representations are often impossible to check by hand and require a custom compiler or translator. To mitigate this additional risk, formal methods can be used to check the outputs automatically. We propose to develop tooling for the Halo2 ZK proof system written in Rust. Our proposal aims to leverage lightweight formal methods to catch large classes of bugs and errors that can present themselves during development with the Halo2 library. Such methods can help detect unused gates, unconstrained cells, unused columns, and under-constrained circuits, among other potential vulnerabilities. Our work presents a pioneering approach to PLONKish arithmetization and Halo2 circuit analysis, combining abstract interpretation, bounded-model checking, and more. We have developed (https://github.com/quantstamp/halo2-analyzer) a proof-of-concept tool that checks under-constrained circuits for

	Halo2 using the CVC5 SMT solver. This represents a significant step forward in enhancing the security of these crucial systems. However, additional work is necessary, outlined below, before this tool is ready for mainstream adoption. Building on our previous work, we propose investigating several innovative research directions: (1) Halo2 integration: Directly integrating our analyzer with the mock-prover in the Halo2 library. This will enhance our tool's functionality and developer ergonomics and make it readily available to library end-users. (2) Halo2 benchmarks: Our approach was not evaluated on real-world circuits as no standard benchmark circuits are available. Taking examples from the Ethereum Foundation's Privacy & Scaling Explorations (PSE) team, we can construct an open-source collection of real circuits. Similar examples will be extracted from other open-source projects building on Halo2. (3) Reduce false-negative rate: Reducing the likelihood of potential false negatives. This will enhance and simplify the reliability of our verification process further. (4) Enhancing solver performance. Multiple approaches may improve the solver's run time. First, the underlying finite field SMT solver may be made more efficient through symmetry breaking or constraint propagation within the CVC5 model. Another approach is to change how various gates are modelled within the SMT solver. For example, the approach of Pilspector could be adapted, or the representation of lookup arguments can be improved. Second, the abstract interpretation may be modified to use additional domain-specific knowledge and techniques. For example, some functions may be computed during the analysis to reduce a reliance on lookup tables. The following optional directions can also be explored: (5) Support for other PLONKish arithmetization schemes: Extending our work to other related PLONKish arithmetization schemes. This will broaden the range of systems our methods can verify. (6) Support for text-based PLONKish representations: Adding support for standard text-based PLONKish representations such as PLAF. This will increase the applicability of our tool.

	(7) Intra-circuit type systems: Exploring intra-circuit type systems, potentially in Chiquito a DSL for Halo2. This will expand the scope of our verification methods. Two major classes of bugs in circuit design are arithmetic overflows and type mismatches. An arithmetic overflow happens when the circuit designer's assumption or intent is to treat some cell values as numbers in a specific range. A constraint over those cells may only be satisfied because there is no range check to ensure that the assumption is true. The core problem is the constraints do not completely capture the circuit designer's intent. Sometimes circuit designers avoid making their assumptions explicit in the constraints as they believe the input to the circuit always has a specific type or structure. Needless to say, the motivation is to reduce the number of constraints and optimize the circuit size. The problem often arises when gadgets or circuits get used by others as a black box without satisfying the assumptions. This direction would introduce an intra-circuit type system which allows circuit designers to explicitly express their assumptions about the type and structure of values in certain cells. The constraints system can dynamically decide when to make these assumptions explicit in constraints or avoid doing that because other constraints already take care of it. A similar approach is a common practice in many typed programming languages; it makes a familiar experience for circuit developers and helps avoid large classes of bugs. We can use SMTs to identify when extra constraints are necessary to enforce types and when we can skip them. (8) Automated constraint augmentation: Investigating the automatic forcing of naturally under-constrained circuits to be fully constrained and assessing the overhead associated with this process. This aims to improve accuracy and performance. Not all circuits are naturally fully constrained. For instance, when dealing with NP statements with multiple accepting witnesses, the circuit implementing the naive NP verifier would be underconstrained. For example, imagine a circuit that aims to prove a given public input graph is Hamiltonian by verifying a Hamiltonian path in the graph as a private witness. The naive implementation of such a circuit would be naturally under-constrained, as potentially, there exist many different Hamiltonian paths in a Hamiltonian graph, hence the fact that the circuit is under-constrained is not a meaningful signal. However, sometimes we can add more constraints to the circuit to force the witness to be unique for each set of public inputs. In the example above, we can force the circuit only to accept the lexicographically smallest path. The goal of this research direction is to find a solution to automatically make circuits fully-constrained by augmenting constraints. This may

	result in larger circuits but would prevent under-constrained circuits. This gives developers the desirable guarantee that the circuit is sound, given that it is complete. [Timeline] We anticipate that the core research directions will require the following amounts of time for a suitable investigation: (1) Halo2 integration: 4 man-weeks. This direction is primarily an engineering effort. This requires integrating our tooling with the most recent version of Halo2 and adding tests and documentation. (2) Halo2 benchmarks: 6 man-weeks. This direction is also an engineering effort. We will collect benchmarks for the tool in order to evaluate its performance and limitations. Benchmarks will need to be collected, documented, and possibly modified to work with any changes necessary for our tool. (3) Reduce false-negative rate: 8 man-weeks. This research direction is open-ended, and will take time to explore potential approaches. Successful approaches will be integrated into the analyzer, and the documentation for the analyzer will be updated. The solver will be evaluated on the circuits collected as part of the second research direction. (4) Enhancing solver performance: 8 man-weeks. This research direction is also open-ended, and will take time to explore potential approaches. Successful approaches will be integrated into the analyzer, and the documentation for the analyzer will be updated. The solver will be evaluated on the circuits collected as part of the second research direction. Please note that these estimates are rough and more time may be required, depending on staffing requirements. Quantstamp will match the funding for this project.
Team:	The team at Quantstamp consists of many talented researchers, many of which have earned PhDs in computer science. Beyond these accomplishments, Quantstamp was awarded three Layer 2 Ecosystem grants from the Ethereum Foundation. The exact team appointed will depend on availability at the time the grant is

	awarded, and will be a mix of the original analyzer team and other researchers. Dr. Martin Derka, Head of New Initiatives (https://www.linkedin.com/in/mderka/) Martin holds a Ph.D in Computer Science from the University of Waterloo. He has also studied and taught at Brock University, McMaster University, and Masaryk University in the Czech Republic. He is a former Vanier Scholar and NSERC postdoctoral fellow at Carleton University. Before Quantstamp, Martin was CTO of CAR MEDIA 2.0, and also developed APIs for Google Cloud as a software engineer at Google. Dr. Jan Gorzny, Head of L2 Scaling (https://www.linkedin.com/in/jangorzny/) Jan is an experienced researcher in algorithm design and lightweight formal methods. At Quantstamp, Jan is Head of L2 Scaling. He is interested in all things rollups, plasma, and beyond. He pays special attention to new protocol designs, interesting zero-knowledge developments, and making blockchains scale -- securely. Jan has worked on projects that reduce the size of SMT generated proofs and used SAT solvers to solve problems in a range of domains. Jan received his PhD in Computer Science from the University of Waterloo in 2022. Jan worked on the original analyzer code and paper. Dr. Ed Zulkoski, Head of Research (https://www.linkedin.com/in/ed-zulkoski-7a002b91/) Ed holds a Ph.D in Computer Science from the University of Waterloo. His research there was primarily in SAT/SMT solvers and formal verification technologies, with a focus on understanding what makes SAT formulas hard or easy for solvers. Before joining Quantstamp, he worked at Microsoft Research. Dr. Mohsen Ahmadvand (https://www.linkedin.com/in/mahmadvand/)

	Mohsen holds a PhD in Cyber Security from the Technical University of Munich, Germany. His doctoral research concentrated on securing applications in adversarial settings and compiler-based protection, with a particular focus on Low Level Virtual Machine (LLVM). At Quantstamp he specializes in smart contract security audits, researching rollup security, and zero knowledge systems. Prior to this, Mohsen worked as a cloud security expert at SAP and Brainloop, where he focused on Kubernetes security, applied cryptography, and runtime security. Mohsen worked on the original analyzer code and paper. Jeffrey Kam, Research Engineer (https://www.linkedin.com/in/jeffreyhykam) Jeffrey is a research engineer at Quantstamp with experience in audits and tooling research. He holds a bachelor's degree in computer science and mathematics from the University of Waterloo. During his studies, he did research in various areas, including combinatorial reconfiguration, computer algebra, coding theory, and software analysis. Before joining Quantstamp, he interned at Google working on optimizing the performance of their ads infrastructure. Jeffrey worked on the original analyzer code and paper. Fatemeh Heidari (https://www.linkedin.com/in/fateme-heidari/) As a Ph.D. candidate at Polytechnique Montréal University, Fatemeh studies blockchain security and access control, formal verification methods, and zero-knowledge proofs. Fatemeh is also a Research Engineer at Quantstamp and prioritizes blockchain security and zero-knowledge development. Having previously designed a robust and flexible access control service architecture for smart contracts, she is investigating zero-knowledge libraries. Her current research aims to improve ZK protocols' security by formal methods to detect vulnerabilities of Halo2 circuits. Before starting her research journey, Fatemeh gained years of experience as a software engineer and system analyst in enterprise systems. Fatemeh worked on the original analyzer code and paper.