A BRIEF OVERVIEW OF THE MODERN LOOKUP PROTOCOLS COMPATIBLE WITH THE KZG POLYNOMIAL COMMITMENTS *(DRAFT)*

Aleksei Vambol

November 2023

I. Introduction

The lookup protocols are of great importance for the PLONKish zk-SNARK proof systems: designed to prove that the set of the values of the rows of the table, which is constituted of certain columns, is a subset of some other (usually pre-defined) set, they facilitate implementing the in-circuit constraints for many zk-SNARK unfriendly operations.

II. Comparison of the Lookup Protocols

Since 2018, when the lookup protocols were invented, they undergone a significant evolution [SOL23]. The crucial milestones in development of the lookup protocols compatible with univariate polynomial commitments schemes are considered in Table 1. It starts with the Plookup protocol, which has been mentioned in [SOL23] as "a simplification of what is regarded as the first lookup protocol." The last row is dedicated to the cq protocol, which is the latest advance known to us in the studied field. The table considers the protocols based on the KZG polynomial commitment scheme.

Protocol	Preprocessing	Prover work	Verifier work	proof size	Hom.	Agg.
Plookup	-	$O(N\log N)\mathbb{F}+O(N)G_1$	$2P$	$9\mathbb{F},5G_1$	Yes	No
Halo2	-	$O(N\log N)\mathbb{F}+O(N)G_1$	$2P$	$5\mathbb{F},6G_1$	Yes	No
Caulk	$O(N\log N)$	$O(m\log N+m^2)\mathbb{F}+O(m)G_1$	$4P$	$4\mathbb{F},14G_1,1G_2$	Yes	No
Caulk+	$O(N\log N)$	$O(m^2)\mathbb{F}+O(m)G_1$	$3P$	$2\mathbb{F},7G_1,1G_2$	Yes	No
Flookup	$O(N\log^2 N)$	$O(m\log^2 m)\mathbb{F}+O(m)G_1$	$3P$	$4\mathbb{F},7G_1,1G_2$	No	No
Baloo	$O(N\log N)$	$O(m\log^2 m)\mathbb{F}+O(m)G_1$	$5P$	$4\mathbb{F},12G_1,1G_2$	Yes	No
cq	$O(N\log N)$	$O(m\log m)\mathbb{F}+O(m)G_1$	$5P$	$3\mathbb{F},8G_1$	Yes	Yes

Table 1. Comparison of the lookup protocols

The variables N and m stand for the numbers of elements in the lookup table and the constrained column, respectively. The prime finite field operations (for the third column) and elements (for the fifth column) are designated by "F". Respectively, "G_1"

designates the elements and operations in the elliptic curve point prime-order subgroup, which is used to define the pairing and contains the points, whose coordinates lie in a prime field. The elements of another subgroup, which is used together with the "G_1" subgroup to define the pairing, are designated as "G_2". The verification work is described by the number of pairings to compute. Some protocols have the preprocessing phase, which lies in a single-time computation of the auxiliary data repeatedly used by the prover. The sixth column indicates whether the map from lookup tables to the corresponding commitments is homomorphic. The last column specifies whether the different proofs generated using the same preprocessing output can be aggregated into a single proof.

All information in Table 1 has been taken from [SOL23] except for the Halo2 lookup protocol, for which the first five columns originate from [ZGK22] and the last two columns are the result of our analysis. Also, we argue that the proof size for the Halo2 lookup protocol can be reduced to 5F, $4G_1$ by using the KZG multi-opening or even to 5F, $3G_1$ in the case of usage of this protocol as a part of a PLONKish zk-SNARK proof system, which uses the KZG multi-opening, since the lookup protocol of Halo2 introduces only 3 additional polynomials to proof the relation between the constrained column and the corresponding lookup table [ZCT21].

Although the memory requirements were not considered in Table 1, our analysis has allowed us to conclude that Halo2 and cq provers require O(N) and O(m) RAM, respectively, while the cq preprocessing uses O(N) RAM.

III. The CQ Lookup Protocol in the Context of the Underlying Math Methods

The cq protocol incorporates such methods as the logarithmic derivative based lookups [HAB22], the univariate sumcheck for a multiplicative subgroup of a finite field [BCR19], the Feist-Khovratovich method for fast computation of KZG proofs [FK23] and the method for efficient computation of the commitments to the Lagrange basis polynomials [BGG17]. The aggregatability of the cq protocol is the consequence of the structure of the pairing equations checked by the verifier: the G_2 elements depend only on the preprocessing output [EFG22, SOL23].

Most algorithmic building blocks for the aforementioned four basic methods are typical for the PLONKish zk-SNARK proof systems based on the KZG polynomial commitment scheme: Fast Fourier Transform over prime finite fields (e.g., by means of the radix-2 Cooley-Tukey method), multi-scalar multiplication for arrays of elliptic curve points (usually done by means the Pippenger method with additional optimizations), pairings computation (based on the Miller's algorithm) and so on. However, the last two basic methods, which are used in preprocessing, rely on the following rather non-trivial algorithms:

- Fast Fourier Transform for an array of elements of the elliptic curve point prime-order subgroup. It can be described as a usual Fast Fourier Transform, where addition and multiplication are replaced with point addition and scalar multiplication, respectively, and the primitive root of unity, the powers of which the transformation matrix contains, is the corresponding element of the scalar field of the aforesaid elliptic curve point subgroup. This transform can be considered as evaluation of a polynomial, the coefficients of which are points of this subgroup, over the cyclic multiplicative group generated by the primitive root of unity. Therefore, it can be performed by means of the properly modified radix-2 Cooley-Tukey method;

- Computation of the product of a Toeplitz matrix defined over the corresponding scalar field and an array of elements of the elliptic curve point prime-order subgroup in log-linear time. This method uses the previous one as the main building block and is based on the polynomial representation of operations on circulant matrices. The paper by Feist and Khovratovich [FK23], which relies on this method, contains its explanation.

IV. The Cross-Table Lookup Approach

This approach can be described as follows:

- The first (usually small) circuit uses the lookup constraints for some columns and the non-constrained lookup table;

- The second (usually large) circuit uses the constrained lookup table to ensure the correctness of its computation;

- The RapidUp protocol is used to prove that the set of the values of the rows of the non-constrained lookup table is a subset of the set of the values of the rows of the constrained one.

The RapidUp protocol can be considered as the core of the cross-table lookup approach. This protocol has been proposed in [MBL22] and designed for proving that the specified submultiset of the values in a certain multiset is also a submultiset of the values in the other one for the multisets defined by evaluations of polynomials over different multiplicative subgroups of finite fields. It is described as a polynomial protocol operating on univariate polynomials without mentioning concrete commitment schemes, so it can be used with both the KZG and FRI-based polynomial commitments. For the first case, our analysis allowed us to conclude that prover requires O(N log N) time and O(N) RAM, respectively, where N is the number of elements in the largest multiset, and the proof size is 7F, $5G_1$ or even 7F, $4G_1$ in the case of usage of this protocol as a part of a PLONKish zk-SNARK proof system, which uses the KZG multi-opening.

Conclusion

The content of Table 1 and the amounts of the prover work and memory in use for the Halo2 [ZCT21] and cq [EFG22] lookup protocols allow us to give the following recommendations regarding using them as a part of a KZG-based PLONKish zk-SNARK proof system.

The Halo2 lookup protocol should be used in these cases:

- The number of lookup constraints is small, and m is not much smaller than N;
- The lookup table must be dynamic (generated during the proving process).

The cq lookup protocol is recommended for use if the lookup table content is known before the proving process and the case satisfies any of the following criteria:

- The number of lookup constraints is large. The cq proof aggregatability is useful in this case;
- N is much larger than m. The amounts of the computations and RAM required for the cq proving is independent on N.

A large circuit, which consists of several logical parts connected by means of dynamic lookup tables, can be splitted into several smaller circuits, which are connected using the cross-table lookups. The proofs for these circuits can be aggregated aggregated into a single proof by means of zk-SNARK recursion.

References

[BCR19] E. Ben-Sasson et al., "Aurora: Transparent Succinct Arguments for R1CS", *IACR Cryptology ePrint Archive*, 2019. https://eprint.iacr.org/2018/828

[BGG17] S. Bowe, A. Gabizon, M. Green, "A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK", *IACR Cryptology ePrint Archive*, 2017. https://eprint.iacr.org/2017/602

[EFG22] L. Eagen, D. Fiore, A. Gabizon, "cq: Cached quotients for fast lookups", *IACR Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1763

[FK23] D. Feist, D. Khovratovich, "Fast amortized KZG proofs", *IACR Cryptology ePrint Archive*, 2023. https://eprint.iacr.org/2023/033

[HAB22] Ulrich Haböck, "Multivariate lookups based on logarithmic derivatives", *IACR Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1530

[MBL22] H. Masip, J. Baylina, D. Lubarov, J. Muñoz-Tapia, "RapidUp: Multi-Domain Permutation Protocol for Lookup Tables", *IACR Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1050

[SOL23] T. Solberg, "A Brief History of Lookup Arguments", *Ingonyama*, 2023. https://github.com/ingonyama-zk/papers/blob/main/lookups.pdf

[ZCT21] Zcash Team, "The Halo2 Book", *Electric Coin Company*, 2021. https://zcash.github.io/halo2/design/proving-system/lookup.html

[ZGK22] A. Zapico, A. Gabizon, D. Khovratovich, M. Maller, Carla Ràfols, "Baloo: Nearly Optimal Lookup Arguments", *IACR Cryptology ePrint Archive*, 2022. https://eprint.iacr.org/2022/1565