
Efficient ZK Argument for Shuffle Implementation in Rust

MASTER'S THESIS

UNIVERSITY OF FREIBURG

Ahmet Ercem Bulut

Student ID: 5362638

Supervisor: Prof. Christian Schindelhauer

August, 2024

Abstract

A shuffle operation in cryptography is an operation that takes committed and anonymous series of values and returns the original serie modified with a permuted order. Which is important in many real world scenarios(e-voting, mental card games). Due to the plaintexts or data being encrypted, the correctness of a shuffle of commitments is not straight forward to verify. To overcome this problem an honest zero-knowledge verifier has been proposed to verify the correctness of a shuffle of homomorphic encryptions by Bayer and Groth(2012). This argument for correctness combines two separate arguments(Multi-exponentiation Argument, Product Argument) to produce a Shuffle Argument for correctness. The implementation of these arguments are lacking in today's literature of tools. With the use of Rust, which has enormous support from the cryptography community, and with it's efficiency in runtime, we aim to provide an extensive and easy to use zero-knowledge proof system.

Acknowledgements

acknowledgements here.

Contents

Abstract	i
Acknowledgements	ii
1 Introduction	1
1.1 Background	1
1.2 Problem Statement	1
1.3 Objectives	1
1.4 Thesis Structure	1
2 Former Literature	2
2.1 Source Material	2
2.1.1 Efficient ZK Argument for Correctness of a Shuffle	2
2.2 Related Work	2
2.3 Existing Solutions	3
2.3.1 Mental Poker	3
2.3.2 Bayer-Groth Mixnet	3
2.3.3 Practical Ad-Hoc Implementations	3
3 Methodology	4
3.1 Important Related Concepts	4
3.1.1 El Gamal Encryption	4
3.1.2 Pedersen Commitment	4
3.1.3 Homomorphic Property	4
3.2 Research Design	4
3.2.1 Ristretto Points	4

4	Implementation	6
4.1	System Design	6
4.2	Implementation Details	6
5	Results	7
6	Discussion	8
6.1	Analysis of Results	8
6.2	Implications	8
7	Conclusion and Future Work	9
7.1	Conclusion	9
7.2	Future Work	9
	References	10
A	Appendix A	11

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

background here.

1.2 Problem Statement

problem statement here.

1.3 Objectives

research objectives here.

1.4 Thesis Structure

thesis structure here.

Chapter 2

Former Literature

2.1 Source Material

2.1.1 Efficient ZK Argument for Correctness of a Shuffle

In 2012, Bayer and Groth[1] presented an algorithm with sublinear communication complexity for shuffling a deck of homomorphically encrypted values.

According to their findings, operations for an efficient sublinear size argument show linearity in group elements when they are "in the exponent".

Using this adaptation, they constructed an efficient multi-exponentiation argument that a ciphertext C is the product of a set of known ciphertexts C_1, \dots, C_N raised to a set of hidden committed values.

By reducing this bottleneck sublinearly, the argument gains significant improvement in performance ($\mathcal{O}(\sqrt{N})$).

They also provide other optimization and minor improvements over the prover computations.

The algorithms used in the Bayer-Groth paper construct the backbone of our library, with optimizations that are products of using native rust and it's efficiency.

2.2 Related Work

Your related work here.

2.3 Existing Solutions

2.3.1 Mental Poker

Mental Poker[2] is a library aimed to implement a verifiable mental poker game for research purposes. While it is also purely in Rust, the library focuses more on the implementation and the efficiency of Barnett Smart Card Protocol[3]. They also differ in their choice of cryptography primitives and go with arkworks curve points.[4].

2.3.2 Bayer-Groth Mixnet

A pure C++ implementation[5] of the protocol, for use in a messaging system. Useful for us as well since they have in detail performance metrics we can compare to. They use the same curve 25519 as ours and also give hardware specifications for the performances. They have certain drawbacks such as: for some values of the parameter m the verification fails, the row size m should always be larger than the column size n etc.. Our library works without these limitations as well.

2.3.3 Practical Ad-Hoc Implementations

While there are a few more implementations of the argument available online, they seem to either not be comprehensive enough for a comparison, or small code used as injection for other projects. Which we decided not to mention for brevity's sake.

Chapter 3

Methodology

3.1 Important Related Concepts

3.1.1 El Gamal Encryption

3.1.2 Pedersen Commitment

3.1.3 Homomorphic Property

3.2 Research Design

3.2.1 Ristretto Points

A Ristretto Point is a cryptographic construction designed to create a prime-order group from elliptic curves, enhancing the security and efficiency of cryptographic operations. In the context of mental card games, Ristretto Points enable players to prove knowledge or possession of certain cards without revealing the cards themselves.

Prime-Order Group Properties

Ristretto Points provide a prime-order group that simplifies mathematical operations and ensures predictable, secure behavior in cryptographic protocols. A prime-order group eliminates issues related to cofactor multiplication, which can complicate the implementation of secure protocols. In mental card games, this property ensures that each card, represented as a Ristretto Point, interacts securely and predictably within the proof sys-

tem.

Unique Encoding and Decoding

One of the key features of Ristretto Points is their ability to encode and decode points on an elliptic curve in a way that eliminates ambiguities. Each encoded point uniquely corresponds to a single group element, which is critical for maintaining the integrity of the cryptographic proofs. This property ensures that each card in the mental card game, when encoded as a Ristretto Point, has a unique representation, preventing issues such as duplication or misidentification.

Implementation of Choice

We have chosen Dalek Cryptography's crypto tools framework `curve25519-dalek`[6] as library of choice due to multiple reasons. As the library supports the homomorphic properties of such group points, it also reduces the dimension of operations via discarding unnecessary operations such as scalar and point multiplication(which would be cofactor multiplication).

The library is also used prominently by the Rust Cryptography community in implementations of all kinds of cryptographic proofs and arguments.(bulletproofs, r1cs etc...). This gives us the advantage of being easily implementable into already existing Ristretto Point systems.

Chapter 4

Implementation

4.1 System Design

system design here.

4.2 Implementation Details

implementation details here.

Chapter 5

Results

results here.

Chapter 6

Discussion

6.1 Analysis of Results

analysis of the results here.

6.2 Implications

discussion on implications here.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

conclusion here.

7.2 Future Work

suggestions for future work here.

Bibliography

- [1] S. Bayer and J. Groth, “Efficient zero-knowledge argument for correctness of a shuffle,” in *Advances in Cryptology – EUROCRYPT 2012*, D. Pointcheval and T. Johansson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 263–280.
- [2] N. Mohnblatt, K. Gurkan, A. Novakovic, and C. Chen. (2022) *Mental Poker. A library for mental poker (and other card games). Based on the Barnett-Smart protocol and the Bayer-Groth argument of correct shuffle*. [Online]. Available: <https://github.com/geometryxyz/mental-poker>.
- [3] A. Barnett and N. P. Smart, “Mental poker revisited,” in *IMA Conference on Cryptography and Coding*, 2003. [Online]. Available: <https://api.semanticscholar.org/CorpusID:27461759>
- [4] arkworks contributors. (2022) *arkworks zkSNARK ecosystem*. Available: <https://arkworks.rs>.
- [5] M. Fragkoulis and N. Tyagi. (2018) *Bayer Groth Mixnet Implementation in C++* [Online]. Available: <https://github.com/grnet/bg-mixnet>.
- [6] D. Cryptography. (2024) *curve25519-dalek A pure-Rust implementation of group operations on Ristretto and Curve25519*. [Online]. Available: <https://docs.rs/curve25519-dalek/latest/curve25519-dalek/>.

Appendix A

Appendix A

Your appendix content here.