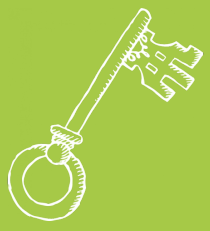

High-Speed Implementation of POSEIDON

Advisor: **Markus Schofnegger**

Motivation

Modern cryptographic protocols cover many more use cases than just pure encryption or hashing. Indeed, it is often important that the primitive being used (e.g., a block cipher) has some specific advantageous properties. For example, in multi-party computation (MPC) and fully-homomorphic encryption (FHE) scenarios, it is important that the underlying construcion has a small total number of multiplications or a small multiplicative depth.

Another popular use case includes the possibility to prove that a leaf exists in a Merkle tree in zero knowledge, i.e., without revealing the leaf itself. Strategies like rank-1 constraint systems (R1CS) provide a solution to this problem, and they are especially efficient if the total number of multiplications in a construction (e.g., a hash function) is small. This is due to the fact that a lower number of multiplications reduces the required number of constraints.

However, besides the application of a specific hash function in this use case, the same hash function is often also needed for plain/ordinary hashing. For this reason, the construction being used should also provide acceptable plain performance.

In this thesis, the task is to implement a high-speed version of the hash function POSEIDON in software.

Goals and Tasks

- Get familiar with the POSEIDON hash function
- Get familiar with the Rust programming language
- Evaluate different optimization techniques
- Implement a fast version of POSEIDON

Literature

- L. Grassi et al.
  Poseidon: A New Hash Function for Zero-Knowledge Proof Systems
  IACR Cryptol. ePrint Arch. 2019

Courses & Deliverables

- ☑ **Introduction to Scientific Working**
  Short report on background
  Short presentation
- ☑ **Bachelor Project**
  Project code and documentation
- ☑ **Bachelor's Thesis**
  Project code
  Thesis
  Final presentation

Recommended if you're studying

☑ CS ☑ ICE ☑ SEM

Prerequisites

- Interest in the topic area
- Programming (C/C++, Rust, Python)

Advisor / Contact

markus.schofnegger@iaik.tugraz.at