Master Thesis in
Artificial Intelligence and Cybersecurity

Zero-Knowledge friendly cryptographic permutation: theory and implementation

Candidate
Stefano Trevisani

Supervisors
Prof. Alberto Policriti
Dr. Arnab Roy

Co-Supervisor
Prof. Elisabeth Oswald

Tutor
Msc. Matthias Steiner

Academic Year 2021-2022

Institute Contacts
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

Institute Contacts
Alpen-Adria-Universität Klagenfurt,
Universitätsstraße, 65–67
9020 Klagenfurt am Wörthersee — Austria
+43 463 2700
https://www.aau.at/

To <somebody_special>,
for being with me all the way through.

Acknowledgements

Sed vel lorem a arcu faucibus aliquet eu semper tortor. Aliquam dolor lacus, semper vitae ligula sed, blandit iaculis leo. Nam pharetra lobortis leo nec auctor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Fusce ac risus pulvinar, congue eros non, interdum metus. Mauris tincidunt neque et aliquam imperdiet. Aenean ac tellus id nibh pellentesque pulvinar ut eu lacus. Proin tempor facilisis tortor, et hendrerit purus commodo laoreet. Quisque sed augue id ligula consectetur adipiscing. Vestibulum libero metus, lacinia ac vestibulum eu, varius non arcu. Nam et gravida velit.

Abstract

Zero Knowledge (ZK) proof systems have been an increasingly studied subject in the last 40 years. In the last decade, the efficiency of the proposed frameworks, along with the processing power of computing devices, has improved to the point of making ZK computation feasible in real-world scenarios. One of the primary applications lies in hash-tree commitment verification, and in this past five years there has been intense research in proposing ZK-friendly cryptographic primitives. In this work, we begin by studying the history of ZK systems and reviewing the state of the art concerning ZK-friendly cryptographic permutations. We then present a novel, generic algebraic framework to design cryptographic permutations and we apply it to construct a new permutation. Finally, we implement our permutation together with the reviewed ones in the Groth16 ZK-SNARK framework and compare their efficiency for Merkle-Tree commitment verification.

Contents

1

Introduction

An important research branch of cryptography which emerged in the last fourty years is the study of *Zero Knowledge Interactive* (ZK-I) protocols, and more specifically zero knowledge proof systems (ZKP) [4]. The main idea behind ZKP systems is to have two (or, in some cases, more) parties, where one is the *prover* and the other is the *verifier*: in a classical proof system, the prover must be able to convince the verifier that a certain statement is true, when this is indeed the case, but the verifier cannot be fooled if the statement is actually false. In a ZKP system we also require that the verifier does not get any useful additional information (i.e. knowledge) other than the truth, or lack thereof, of the statement. This additional requirement is particularly interesting when dealing with statements that are notoriously (believed to be) hard to prove, so that the verifier would not be realistically able to prove them in a reasonable amount of time. As a simple example, a prover would like to show that a propositional logic formula is satisfiable (an instance of the famous SAT problem) without revealing the satisfying assignment to the verifier.

Along the years, additional interesting and useful properties have been added to extend and improve the capabilities of ZKP systems. For example, we would like to have a *Non-interactive* (ZK-NP) protocol, to minimize the amount of required communication and have it happen only at the beginning and at the end of the protocol. We could also want to relax the soundness requirement so that it is guaranteed only against computationally bounded provers: in this case, instead of 'proof' we use the term *ARgument of Knowledge*, and hence we can have ZK-IARK/ZK-NARK systems. More recently, there has been a research effort towards reducing the length of the ARK by ensuring that it is constant size or at most bounded by a logarithmic function in the length of the theorem statement: such systems are said to be *Succint*. Implementations of ZK-SNARK system, like Pinocchio [9] or Groth16 [7], represent the current state of the art (SoTA) of ZKP systems, and allow to generate proofs to verify any computation representable by means of *bounded arithmetic circuits*. A major downside of ZK-SNARK protocols is their need of a trusted third party (TTP) to setup the system, hence current research is studying *Transparent* systems (ZK-STARK) to address this issue [2].

An especially useful application of ZKP systems is proving knowledge of a preimage for a cryptographic hash function digest (a.k.a. commitment). Many data integrity systems, such as blockchains, rely on Merkle Trees [8] to ensure efficient commitment validation, especially in dynamic environments. In Merkle Trees, an hash function is applied in a bottom-up fashion: the leaves will contain the data owned by some parties, while the root will contain the tree commitment. In a non-ZK setting, a prover would send the verifier his leaf together with the co-path, the verifier would then recompute the tree commitment and compare it with the public one and be convinced whether or not the prover does actually own the leaf. On the other hand, in a ZK-SNARK setting, we first have to represent the computation through a bounded arithmetic circuit, i.e. we are allowed to use exclusively a constant number of additions and multiplications over some suitable finite field. The circuit, together with a *proving key* provided by a TTP, and some private and public data, is then used by the prover to generate a proof which is sent to the verifier, who in turn uses a *verification key*, again provided by the same TTP, to assert whether the circuit computation was performed correctly.

While the various ZK-SNARK (or ZK-STARK) frameworks differ in the details, it is intuitive to see that the complexity of generating the proof (which dominates the cost of the protocol) must depend on the size of the circuit, which in turn depends on the amount of multiplications and additions performed in the computation: in the case of Merkle Tree commitment verification, most of the computation consists in iterating the underlying hash function. Since the finite field over which ZK-SNARK frameworks works is typically a huge prime field ($\approx 2^{256}$ elements), traditional hash functions like MD5 [10] or SHA [3], which are designed to be extremely efficient on classical boolean circuits, become extremely inefficient in the ZK case.

It is no wonder then, that in the last years researchers began to study so-called ZK-friendly cryptographic permutation (ZKFCP) designs that exploit the features of large prime fields to be efficient when translated into airhtmetic circuit, fundamentally resulting in a one-to-one mapping. Being a new research topic, these designs have seen a rapid series of improvements [1, 6, 5] in the last three years: in a two-part series of papers undergoing publication, we presented an algebraic framework, called *Generalized Triangular Dynamical System* (GTDS), which allows to express many of the existing cryptographic permutation designs and eases the construction of new ones, while at the same time giving strong security guarantees, and we then applied it to devise the `Blocc` blockcipher and the `Stamp` hash function. Using the `libsnark`[1] library (an implementation of the Groth16 framework), we implemented our hash function, along with other competitor hash functions and a hash-agnostic variable-arity Merkle Tree circuit template, in a `C++` project which we then used to compare their real-world performance for same-level security gaurantees in various scenarios. This work is organized in two parts: Part I contains the background of the work, presenting all the mathematical, computational and cryprographical tools and concepts required to understand the theory, the history and the applications of ZKFCPs. In Part II, we begin with a review of state of the art ZKFCPs, we then present the GTDS algebraic framework, its instantiation in the form of the `Blocc` block cipher and the `Stamp` hash function and we conclude with an implementation analysis and experimental comparison between the current SoTA and the GTDS constructions.

[1] `https://github.com/scipr-lab/libsnark`

I
Foundations

2

Mathematical Background

3 Computational Background

3.1 Turing Machines

3.2

4

Cryptographical Background

4.1 Hash functions

4.2 Verification Trees

4.3 Zero Knowledge Proofs

4.4 ZK-SNARK systems

II

Zero Knowledge friendly permutations

5

State of the art

5.1 MiMC

5.2 Poseidon

5.3 Griffin

5.4 Other designs

6

Generalized Dynamic Triangular Systems

7

Implementations and experiments

III
Appendix

A

Titolo della prima appendice

Sed purus libero, vestibulum ut nibh vitae, mollis ultricies augue. Pellentesque velit libero, tempor sed pulvinar non, fermentum eu leo. Duis posuere eleifend nulla eget sagittis. Nam laoreet accumsan rutrum. Interdum et malesuada fames ac ante ipsum primis in faucibus. Curabitur eget libero quis leo porttitor vehicula eget nec odio. Proin euismod interdum ligula non ultricies. Maecenas sit amet accumsan sapien.

Bibliography

[1] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 191–219, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[2] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

[3] Quynh H. Dang. *Secure Hash Standard.* Jul 2015.

[4] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[5] Lorenzo Grassi, Yongling Hao, Christian Rechberger, Markus Schofnegger, Roman Walch, and Qingju Wang. A new feistel approach meets fluid-spn: Griffin for zero-knowledge applications. *IACR Cryptol. ePrint Arch.*, 2022:403, 2022.

[6] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In *USENIX Security Symposium*, 2021.

[7] Jens Groth. On the size of pairing-based non-interactive arguments. Cryptology ePrint Archive, Paper 2016/260, 2016. `https://eprint.iacr.org/2016/260`.

[8] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford University, Stanford, CA, USA, 1979. AAI8001972.

[9] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. Cryptology ePrint Archive, Paper 2013/279, 2013. `https://eprint.iacr.org/2013/279`.

[10] Ronald L. Rivest. The md5 message-digest algorithm. In *RFC*, 1990.