

Zero Knowledge Compilers

John T. McCall
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
mcca0798@morris.umn.edu

ABSTRACT

<Insert Abstract Here>

General Terms

Need to figure this out yet

Keywords

Zero Knowledge Protocols, Compilers

1. INTRODUCTION

I will focus on Zero-Knowledge Compilers which are compilers that automatically generate Zero-Knowledge proofs. This is how I plan to use the following sources:

- I expect [2, 3, 5] to be my core sources, depending on how relevant [5] turns out to be I'll replace it with a better source.
- I will use [4, 6, 7] for background information and examples of Zero-Knowledge Protocols.
- I will need to find some papers for background information on compilers.

As stated above I need to find some sources about compilers. I probably will need to find more papers dealing with ZK-Compilers as well.

1.1 Key Points

What main problems(s) or questions(s) does the research address?

The main problem that the research address is how to create reliable zero knowledge protocols. They can be difficult to define and even harder to verify. Zero knowledge compilers help because they can efficiently generate zero knowledge protocols, and because of how they are constructed the user can trust that they will work.

What are the key contributions of each of your main sources?

This work is licensed under the Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

UMM CSci Senior Seminar Conference, December 2013 Morris, MN.

Source [2] provides a great deal of information about their zero knowledge compiler, ZKCrypt. They go into detail about zero knowledge protocols, how their compiler produces them, and they give a proof verifying that their protocols are valid. They also talk about a few applications of their compiler.

Source [5] talks in depth about ZKPD, which is a language they created for writing zero knowledge protocols. They also created an interpreter for this language, which performs optimizations to lower computational and space overhead. This paper also provides an example dealing with electronic cash.

Source [3] uses Σ -Protocols in a compiler to automatically generate sound and efficient zero knowledge proofs of knowledge. The compiler automatically generates the implementation of the protocol in Java, or it can output a description of the protocol in \LaTeX .

How are the main sources related to each other?

The main sources all use compilers to generate zero knowledge protocols, but the ways they are implemented are all different so there is some room for comparison. All the compilers are also based off Σ -Protocols, or variations of Σ -Protocols.

What is the state of the research?

The current state is that the compilers have been implemented and tested. They all provided enough data to back up their research. Most of the work they are doing now will extend the applications of their compilers to support other proof types.

What background material will you need to present in order for your audience to understand the research?

I will need to provide background information on zero knowledge protocols and compilers. It's probably more important that I focus on zero knowledge protocols and only give basic compiler background.

2. ZERO KNOWLEDGE PROTOCOLS

Zero knowledge protocols, also referred to as zero knowledge proofs, are a type of protocol in which one party, called the *prover*, tries to convince the other party, called the *verifier*, that a given statement is true.

2.1 Definition

Formally, a zero knowledge proof is a type of interactive proof. Mohr gives a concise definition in [6].

DEFINITION 1. *An interactive proof system for a set S is a two-party game between a verifier executing a probabilis-*

tic polynomial-time strategy and a prover which executes a computationally unbound strategy satisfying:

- *Completeness:* For every $x \in S$, the verifier always accepts after interacting with the prover on common input x
- *Soundness:* For some polynomial p , it holds that for every $x \notin S$ and every potential strategy P^* , the verifier rejects with probability at least $\frac{1}{p(|x|)}$ after interacting with P^* on common input x .

To summarize: if an honest verifier is always convinced after interacting with an honest prover, then the proof is complete. A proof is sound if a cheating prover can only convince an honest verifier with some small probability.

For an interactive proof to be a zero knowledge proof it must, unsurprisingly, satisfy the condition of *Zero Knowledge*.

DEFINITION 1. A proof is zero knowledge on (inputs from) the set S if, for every feasible strategy B^* there exists a feasible computation C^* so that the following two probability ensembles are computationally indistinguishable:

- the output of B^* after interacting with A on common input $x \in S$
- the output of C^* on input $x \in S$

In other words, any information, in this case B^* , that can be learned by interacting with A can also be learned without interacting with A . [6]

In certain cases the ability to prove a statement requires some secret knowledge by the prover. A *zero knowledge proof of knowledge* is a specific type of zero knowledge protocol in which the prover must convince the verifier of the validity of the statement without revealing that secret knowledge. [1]

2.2 Examples

3. COMPILERS

This section will provide some basic background information about compilers.

3.1 Background

4. ZERO KNOWLEDGE COMPILERS

This section will be the main section. Here I will talk about my core sources and how they are using their compilers.

4.1 Sigma-Protocols

Here I'll talk about Sigma-protocols and how they are used in the following compilers.

Note to myself: Look into the Fiat-Shamir heuristic

4.2 ZKCrypt

Here I'll talk about ZKCrypt, its implementation and verification steps.

4.3 ZKPDL

Similar to ZKCrypt section.

5. APPLICATIONS

Here I'll talk about how those compilers are used in the real world.

5.1 Electronic Cash

5.2 Another Application

6. CONCLUSION

This is where I'll neatly wrap everything up.

7. REFERENCES

- [1] Zero-knowledge proof. *Wikipedia, The Free Encyclopedia*, October 2013. *Wikipedia is great for background information and definitions.*
- [2] J. B. Almeida, M. Barbosa, E. Bangerter, G. Barthe, S. Krenn, and S. Z. Beguelin. Full proof cryptography: verifiable compilation of efficient zero-knowledge protocols. In *Proceedings of the 2012 ACM conference on Computer and communications security, CCS '12*, pages 488–500, New York, NY, USA, 2012. ACM. *This paper is one of the Core papers. It deals heavily with ZK-Protocols and on ZK-Compilers.*
- [3] E. Bangerter, T. Briner, W. Hancecka, S. Krenn, S. Ahmad-Reza, and T. Schneider. Automatic generation of sigma-protocols. In *Proceedings of the 6th European conference on Public key infrastructures, services and applications, EuroPKI'09*, pages 67–82, Berlin, Germany, 2009. Springer-Verlag. *This paper is a core paper and focuses on compilers that automatically generate sound and efficient Zero knowledge proofs of knowledge based on sigma-protocols.*
- [4] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Advances in Cryptology Proceedings, CRYPTO'89*, pages 218–229, New York, NY, USA, 1987. ACM. *This paper is referenced by most papers that deal with zero knowledge protocols, but it's old. As such it will be used for background information and maybe an example.*
- [5] S. Meiklejohn, C. C. Erway, A. Kupcu, T. Hinkle, and A. Lysyanskaya. Zkpd: a language-based system for efficient zero-knowledge proofs and electronic cash. In *USENIX Security'10 Proceedings of the 19th USENIX conference on Security, Security '10*, Berkeley, CA, USA, 2010. USENIX Association. *This is a core paper, it delves into efficient ZK-Proofs and an application dealing with electronic cash.*
- [6] A. Mohr. A survey of zero-knowledge proofs with applications to cryptography. *This article is great background information and has several great examples I can use.*
- [7] J.-J. Quisquater, L. Guillou, M. Annick, and T. Berson. How to explain zero-knowledge protocols to your children. In *Proceedings on Advances in cryptology, CRYPTO '89*, pages 628–631, New York, NY, USA, 1989. Springer-Verlag New York, Inc.