The Reusability of zk-SNARKs in the Zcash Protocol

Cordian A. Daniluk
Laboratoire Jean Kuntzmann
Grenoble, France
cordian.daniluk@grenoble-inp.org

Supervised by: Aude Maignan

Abstract

A brief introduction to the internship's topic is presented. This is followed by a summary of the progress made so far and an outlook of what remains to be done.

1 Introduction

Cryptocurrencies such as Bitcoin [Nakamoto, 2008] were designed with the goal to decentralize money transfer. There should not be a central trusted party such as a bank, through which passes every transaction to remove the need for trust. However, without any trusted party, transactions are not checked for validity such as the fact that a spender must really be the owner of the spent money and that a spender cannot spend money twice.

A commonly used solution by cryptocurrencies to enforce these rules and yet others is a public ledger called the blockchain [Nakamoto, 2008]. The public nature of this data structure enables any node in the cryptocurrency's network to verify rules such as ownership of spent money or double spending money. However, without appropriate measures a public ledger can lead to compromised privacy by revealing information such as the identities of transaction participants. In the special case of Bitcoin, a spent amount of money is associated with a destination address, effectively a user's pseudonym. A user may employ an arbitrary number of pseudonyms to hide their true identity and additionally use methods similar to money laundering, but even these fail to guarantee anonymity [Reid and Harrigan, 2011]. Hence, people have tried improving anonymity in new cryptocurrencies such as Monero [van Saberhagen, 2013] or Zcash [Hopwood *et al.*, 2023]. Zcash is the one this internship is focusing on.

Zcash still uses a blockchain, but to make stronger anonymity guarantees than Bitcoin, Zcash offers transparent transactions inherited from Bitcoin as well as shielded transactions. While the former leak data such as the pseudonyms of senders and receivers, the latter hide it by either encrypting sensitive information on the blockchain in an anonymity-preserving way or not including it at all. This lack of inclusion needs additional care, however, since classical Bitcoin nodes need some public information such as a sender's pseudonym to enforce the blockchain's validity. To this end, a cryptographic construct called zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) is used. A zk-SNARK has primary and auxiliary inputs. The former are publicly known, the latter are only known to the sender. The sender adds all sensitive information as the auxiliary input and some other as the primary input, creates a zk-SNARK from these, and adds it to a shielded transaction. The zk-SNARK expresses the statement that the sender knows primary and auxiliary inputs such that the transaction is valid, e.g., that the sender rightfully claims ownership of a given amount of money. Upon inclusion in the blockchain, anyone can verify the zk-SNARK to enforce the transaction's validity. Hence, nodes do not do this on public data directly included in the blockchain as in Bitcoin, but on a zk-SNARK, which enables validation of transactions without revealing its auxiliary inputs.

The various properties of zk-SNARKs render them useful in Zcash: they are zero-knowledge, meaning that they prove statements on auxiliary inputs without revealing them. Their succinctness makes for proofs of small size. Since they are non-interactive, unlike proposed by, for example, the foundational paper of zero-knowledge proofs [Goldwasser *et al.*, 1985], they can be included in the blockchain without further communication between sender and other parties. Finally, they are arguments, meaning that a malicious, computationally bounded sender cannot fake proofs to mislead nodes in the network. More specifically, they are arguments of knowledge, which means that not only do auxiliary inputs exist that satisfy the proven statement, but also that the prover knows them.

Beyond its theoretical definition, zk-SNARKs steadily evolve to make them even more practical. In Zcash alone, three different zk-SNARKs [Ben-Sasson *et al.*, 2014b] [Groth, 2016] [Hopwood *et al.*, 2023] have been used throughout the protocol's history. While there are many other details in the Zcash protocol to increase anonymity than merely zk-SNARKs, they are the main mechanism.

There might be situations, where the creation of zk-SNARKs does not have to be done from scratch: the network might have rejected a transaction for various reasons (insufficient transaction fee, chain reorganization, unfulfilled consensus rules, etc.) and the user wants to resend the transaction. This internship will investigate under what circumstances and to what degree the zk-SNARKs in the resent transaction have to be entirely recalculated, modified, or can be left as is.

2 Current Work

Zcash has gone through several network updates (Sprout, Sapling, and Orchard are the most relevant ones) and is based on a theoretic cryptocurrency called Zerocash.

The Zcash protocol [Hopwood *et al.*, 2023] and the Zerocash paper [Ben-Sasson *et al.*, 2014a] are studied intensively to understand the basic principles on one hand and get a good understanding of the real protocol on the other.

During the first few weeks, Zerocash was looked at primarily. It extends a base currency providing transparent transactions such as Bitcoin by two operations for shielded ones, *Mint* and *Pour*. The former serves to "mint" shielded coins from transparent ones in order to bring shielded money into circulation, essentially a transaction with transparent inputs and shielded outputs. The latter "pours" shielded coins into new ones, which is a transaction with shielded inputs and shielded outputs. Omitting some details, the acquired understanding of these operations will be explained now.

To perform a fully shielded transaction, a user must first obtain shielded money. For that, he issues a *Mint* transactions: given a transparent input that provably belongs to him, he commits to the input's amount of money v, the minted money's serial ID ρ, and the destination address a_{pk} using a commitment scheme $\mathtt{COMM}$. Nobody learns a_{pk} because of $\mathtt{COMM}$'s hiding property. The user cannot change any of the data commmited to without changing the commitment itself because of $\mathtt{COMM}$'s binding property. Finally, the commitment is put into the blockchain.

Now, the user can perform a fully shielded transaction. Given two commitments c_1 and c_2 to shielded money, he creates a *Pour* operation, which contains two output commitments d_1 and d_2, a zk-SNARK π and, for each of c_1 and c_2, a value $\hat{\rho}$. π proves that

- c_1 and c_2 exist on the blockchain
- for each of c_1 and c_2, a_{pk} belongs to the user; it is a public key derived from a secret key a_{sk}, which the user proves to know
- for each of c_1 and c_2, the revealed value $\hat{\rho}$ is the same as $f(\rho, a_{sk})$ where f is a pseudo-random function
- c_1, c_2, d_1, and d_2 are commitments to shielded money as described above
- c_1 and c_2 add up to the same amount as d_1 and d_2; no money is lost

Some data that is put onto the blockchain as part of a *Pour* operation is encrypted with an asymmetric encryption scheme. Only the owner of d_1 and d_2, respectively, can decrypt these data. Additionally, the *Pour* operation is signed with a one-time signature scheme.

Once the *Pour* operation is sent, nodes can verify this proof, but never learn source and destination of the money nor the amount spent. Since $\hat{\rho} = f(\rho, a_{sk})$ is public, every node can verify that c_1 and c_2 have been spent and thus cannot be double-spent by the user because ρ uniquely identifies a coin.

While Zerocash is conceptually the simplest, it is also the farthest away from reality. It has never been deployed as a true cryptocurrency. Furthermore, more recent versions of Zcash use more complex constructs to instantiate the cryptographic primitives such as commitments and zk-SNARKs. For example, while Zerocash uses SHA256 to instantiate $\mathtt{COMM}$, Sapling uses a variation of Pedersen commitments over elliptic curves. Similarly, Zcash replaces *Mint* and *Pour* by other, very similar operations that are more complicated, but enable more efficient proofs or provide other advantages.

The Zcash source code is online [zca, b] and is consulted when testing on a Zcash node or in case of doubts about Zcash's implementation. Most of the time is spent on reading relevant papers, however.

Over time, some code has been written to test out Zcash's functionality. Zcash nodes offer an RPC interface, which makes it easy to test commands such as address generation, transaction creation, or transaction transfer. To interact with the Zcash network, a Python script was written to interact with a Zcash node via `getblock.io`, which provides an API to send RPC commands to a node. To locally test more than one Zcash node, scripts were written to start and interact with any number of nodes simultaneously. All code and other resources related to this internship can be found at `https://gricad-gitlab.univ-grenoble-alpes.fr/danilukc/stage`. A `README.md` file describes which code does what.

3 Future Work

Finding out to what extent proofs need to be recalculated will involve examining in which situations what inputs to the zk-SNARK change and what effect changing the inputs has on other blocks and transactions. Once this high-level framework has been established, it will have to be understood how changing inputs changes the zk-SNARK's creation. This includes both research on how the encoding of the statement proved by the zk-SNARK is changed (a "rank-one constraint system" [Ben-Sasson *et al.*, 2013] for Sprout and Sapling and UltraPLONK [zca, a] for Orchard) and how this change affects the value of the zk-SNARK itself (whose calculation is detailed in [Ben-Sasson *et al.*, 2014b] for Sprout, in [Groth, 2016] for Sapling, and in [zca, a] for Orchard).

References

[Ben-Sasson *et al.*, 2013] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 90–108, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Ben-Sasson *et al.*, 2014a] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash:

Decentralized anonymous payments from bitcoin. Cryptology ePrint Archive, Paper 2014/349, 2014. `https://eprint.iacr.org/2014/349`.

[Ben-Sasson *et al.*, 2014b] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *Proceedings of the 23rd USENIX Conference on Security Symposium*, SEC'14, page 781–796, USA, 2014. USENIX Association.

[Goldwasser *et al.*, 1985] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[Groth, 2016] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[Hopwood *et al.*, 2023] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. `https://zips.z.cash/protocol/protocol.pdf`, 2023. Accessed: 2023-04-07.

[Nakamoto, 2008] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008. Accessed: 2023-04-07.

[Reid and Harrigan, 2011] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 1318–1326, 2011.

[van Saberhagen, 2013] Nicolas van Saberhagen. Cryptonote v 2.0. `https://www.getmonero.org/resources/research-lab/pubs/whitepaper_annotated.pdf`, 2013. Accessed: 2023-04-07.

[zca, a] The halo2 Book. `https://zcash.github.io/halo2/`. Accessed: 2023-04-07.

[zca, b] Zcash 5.4.2. `https://github.com/zcash/zcash`. Accessed: 2023-04-07.