Risks of Using New Cryptographic Schemes in ZKFlow

February 2021

1 Introduction

Zero-knowledge proofs (ZKP) became a prominent solution to address privacy concerns in DLT systems in the last couple of years. Since their introduction as a theoretical concept in the 1980s, significant progress made in the research and development of ZKPs, and many practical schemes are proposed. The adoption of ZKPs also brought another field of research and development effort which is called ZKP-friendly primitives that are cryptographic primitives designed to work efficiently with the arithmetic structure of ZKP systems which is different than classical cryptographic systems.

Despite the increasing interest in ZKPs, enterprises have several concerns about their adoption at the production level due to the immaturity of the existing schemes. The research on ZKP is ongoing, and existing schemes can be outdated with newer proposals at a fast pace. The possible vulnerabilities of the proposed schemes and their implementation have not been deeply investigated. In this document, we explain what kind of vulnerabilities are possible for ZKPs and ZKP-friendly schemes and the consequences of such vulnerabilities by focusing on specific primitives that we use in the ZKFLow product.

2 Risks around ZKP schemes

In our project, we use zkSNARKs [12] as the zero-knowledge proof scheme. zk-SNARKs enable non-interactive proving, where any verifier that has access to the verifying keys can perform verification. An important advantage of zk-SNARKs over other proof systems is their succinctness, which provides efficiency in computations by short proof size and constant verification time. The practicality of zkSNARKs made them gain attention from a large community and provided a broader independent analysis for its security and efficiency improvements. However, there still exist several concerns around zk-SNARKs, which might impact their wider adoption as a privacy-preserving solution. In the following, we summarize these risks.

Security of the underlying hardness assumption: One concern around zkSNARKs is their underlying security assumption, the knowledge of exponent assumption [8], which is not a standard security assumption. Most of the cryptographic primitives in use today base their security on computationally hard assumptions such as integer factorization or discrete logarithm. These assumptions are considered standard security assumptions since they are widely investigated and their security guarantees are well-understood. A cryptographic mechanism that is built using one of these assumptions can be proven secure by reducing its security to the hardness of breaking the underlying assumption. A non-standard assumption, on the other hand, is more recent compared to the standard assumptions and its security is not investigated thoroughly. The risk with these assumptions is that if a non-standard assumption is later proven to be wrong, then the provable security of a cryptographic protocol based on this assumption is also in question. In a series of articles [13, 14, 15], Koblitz and Menezes have discussed the importance of thorough analysis in the design of new hardness assumptions and in proving the security of cryptographic protocols with reductions to such assumptions.

Another criticism towards the knowledge of exponent assumption is on its non-falsifiable nature. The standard hardness assumptions in cryptography are falsifiable, such that if the assumption is false, it is easy to show a counterexample that proves it. However, for non-falsifiable assumptions, it is not possible to show that the assumption is false by showing a counterexample. This situation leads to a certain bias against these kinds of assumptions and might limit their wider investigation by the cryptographic community.

Toxic waste Performing a non-interactive ZKP requires generating a common reference string (CRS) that contains the keys for proving and verifying and can be accessed by the parties. Generation of CRS results in toxic waste that should be destroyed after CRS generation, otherwise, a malicious party can generate fake proofs using the toxic waste. In theory, ZKP protocols assume these parameters are generated in a trusted setup with a trusted party, and toxic waste is destroyed by this party. However, delegating the CRS generation to one party in real-life is problematic. This issue can be overcome by running a multiparty computation ceremony for the setup such that if there is at least one trusted party involved in the ceremony, the destruction of the toxic waste and security of the ZKP parameters are guaranteed [7].

Quantum Resistance Another risk in the adoption of zkSNARKs is they are not quantum resistant. The number-theoretic primitives of zkSNARKs (elliptic curves, pairings, etc.) are known to be broken by quantum computers. Therefore, they do not provide post-quantum security, which limits their adoption for the long term. However, there are alternative ZKP schemes that can provide quantum resistance, such as zk-STARKs[5], which is based on hash functions, or lattice-based zk-SNARKS [9].

Implementation and Protocol Vulnerabilities Despite their broader investigation, it is still possible to find vulnerabilities in the design or implementation of zkSNARKs. ZCash team reported an example of such vulnerability in 2019 [17]. After finding a mistake in their former proof scheme [6] that allows an attacker to mint an infinite amount of counterfeit ZCash, the team switched to the proof system of Groth [12], which has better security analysis.

3 Risks around ZKP-friendly primitives

ZKP-friendly cryptographic primitives are designed due to the need for schemes that can perform efficiently in arithmetic ZKP circuits. There is a broad area of research that aims to design lightweight encryption, hashing, and signing mechanisms for arithmetic circuits [10, 1, 11]. In our protocol, we use such primitives to perform hash operations to eliminate the expensive computation cost of Corda's default hash scheme SHA256. Specifically, we use Blake [2] and Pedersen [4] hash functions to perform Merkle tree computations.

Blake hash function was one of the final candidates for the SHA3. Therefore, it has a broader investigation into its security. The function is resistant to preimage and collision failures for arbitrary size messages. It is also resistant against length extension attacks, which provides an advantage on SHA2. To this date, the attacks on Blake were successful only in modified versions with a lower number of rounds [2]. In our protocol, we use a variant of Blake, Blake2s [3], which provides better performance compared to the original Blake with a fewer number of rounds and parameters. Despite lower numbers, the parameters of Blake2s are still large enough to resist known attacks on the hash function [2]. Regarding its quantum resistance, the known quantum attacks are not practical yet to break Blake hash function and its variants [16].

Pedersen hash is an algebraic hash function whose security is based on the hardness of discrete logarithm problems on elliptic curves. Its algebraic structure provides the hash function better efficiency in arithmetic circuits. However, the Pedersen hash does not guarantee unconditional collision resistance. The hash function is collision-resistant only for fixed-length messages or for the messages that are prefixed with their lengths. We analyze the collision-resistance of Pedersen hash in ZKFlow protocol white paper [18]. Furthermore, the number-theoretic basis of the hash function is known to be broken by existing quantum attacks [16].

4 The effect of a security failure on ZKFlow

After reviewing possible threats to the security of the chosen cryptographic primitives, in this section, we discuss how a security failure in one of the primitives can affect the ZKFlow protocol. The protocol design of ZKFlow does not explicitly depend on any of the chosen cryptographic schemes. These schemes are chosen due to their superior security and efficiency guarantees compared to

other existing cryptographic schemes. However, should they become susceptible to an attack, they can be replaced with an alternative without changing the main flow of the protocol.

If zkSNARKs become vulnerable because of the one of the aforementioned risks, a malicious adversary will be able to generate fake ZKPs. Generating a fake proof in the case of ZKFlow might enable an adversary to convince counterparties and the notary for the ownership of an asset that is owned by some other party. However, to be able to claim the ownership of the asset, the attacker should also gain access to the signing keys of the real owner. Our protocol does not weaken the key management of Corda ledger on the protection of signing keys. Therefore, in ZKFlow, the risk of signature forging is the same with the original Corda protocol. Only if an attacker manages to steal the signing keys and generate fake proofs, can they break the integrity of the ledger. This combination of requirements greatly reduces the likelihood of a successful attack. When such an attack is detected, an integrity recovery on the ledger is required to maintain operations. Since the transaction data is not stored on the ledger but only the ZKPs are stored, recovering the integrity of the ledger after such a failure is not trivial. Our current implementation does not address this problem yet. However, we are planning to extend our protocol with a patent pending solution that guarantees the recovery of ledger to the latest valid state after such a failure.

A failure in the chosen hash functions means an adversary can perform collision and preimage attacks. A collision attack enables an attacker to replace the valid transaction data with arbitrary data whose hash is the same as the original data. A preimage attack helps the attacker to find the preimage of a hash input, which might be sensitive transaction data. Designed as a candidate for SHA3, Blake provides better security compared to SHA256 (the default hash function of the Corda ledger) by providing security against length extension attacks on top of other security properties. The likelihood of a failure in Blake hash is equivalent to (or less than) SHA256, thus, our protocol does not degrade the security guarantees of the original Corda protocol.

The usage of Pedersen hash requires more care. The hash function is known to provide collision resistance only for fixed-length messages. In ZKFlow, the Pedersen hash is used only in the computation of Merkle tree nodes that do not require randomness. To avoid collisions, each message that is hashed with Pedersen is prefixed with its message length[1]. This usage is, therefore, safe.

References

[1] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *ASIACRYPT (1)*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.

[1] https://forum.zcashcommunity.com/t/pedersen-hash-collision-resistance/33586/2

[2] Jean-Philippe Aumasson, Willi Meier, Raphael C.-W. Phan, and Luca Henzen. *The Hash Function BLAKE*. Information Security and Cryptography. Springer, 2014.

[3] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: simpler, smaller, fast as MD5. *IACR Cryptol. ePrint Arch.*, 2013:322, 2013.

[4] Jordi Baylina and Marta Belles. 4-bit window pedersen hash on the baby jubjub elliptic curve. *iden3*, n.a.

[5] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptology ePrint Archive*, 2018:46, 2018.

[6] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium*, pages 781–796. USENIX Association, 2014.

[7] Sean Bowe, Ariel Gabizon, and Matthew D. Green. A multi-party protocol for constructing the public parameters of the pinocchio zk-snark. In *Financial Cryptography Workshops*, volume 10958 of *Lecture Notes in Computer Science*, pages 64–77. Springer, 2018.

[8] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.

[9] Rosario Gennaro, Michele Minelli, Anca Nitulescu, and Michele Orrù. Lattice-based zk-snarks from square span programs. In *ACM Conference on Computer and Communications Security*, pages 556–573. ACM, 2018.

[10] Lorenzo Grassi, Daniel Kales, Dmitry Khovratovich, Arnab Roy, Christian Rechberger, and Markus Schofnegger. Starkad and poseidon: New hash functions for zero knowledge proof systems. *IACR Cryptol. ePrint Arch.*, 2019:458, 2019.

[11] Lorenzo Grassi, Reinhard Lüftenegger, Christian Rechberger, Dragos Rotaru, and Markus Schofnegger. On a generalization of substitution-permutation networks: The HADES design strategy. In *EUROCRYPT (2)*, volume 12106 of *Lecture Notes in Computer Science*, pages 674–704. Springer, 2020.

[12] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[13] Neal Koblitz and Alfred Menezes. Another look at "provable security". *J. Cryptol.*, 20(1):3–37, 2007.

[14] Neal Koblitz and Alfred Menezes. The brave new world of bodacious assumptions in cryptography. *Notices of the American Mathematical Society*, 57(3):357–365, 2010.

[15] Neal Koblitz and Alfred Menezes. Critical perspectives on provable security: Fifteen years of "another look" papers. *Adv. Math. Commun.*, 13(4):517–558, 2019.

[16] National Academies of Sciences Engineering, Medicine, et al. *Quantum computing: progress and prospects*. National Academies Press, 2019.

[17] Josh Swihart, Benjamin Winston, and Sean Bowe. Zcash counterfeiting vulnerability successfully remediated, 2019.

[18] Matthijs van den Bos, Victor Ermolaev, Scott King, Alexey Koren, and Gamze Tillem. ZKFlow: Private transactions in corda with zkp. *ING Bank*, 2021.