Report

v. 1.0

Customer

Zcoin Foundation

Cryptography Audit

Lelantus Protocol

20th August 2020

Report prepared by

ABDK Consulting

Protocol Connectivity Problem: Printing Money when Secure Components are used Insecurely

Dmitry Khovratovich

Ilya Kizhvatov

Aram Jivanyan

ABDK Consulting and Zcoin Inc.

August 2020

Abstract

Several cryptocurrencies claim to preserve privacy of senders and recipients by obscuring the link between receiving and spending coins with Zcash, Zcoin, Monero being the famous examples. Lelantus is a recent and popular protocol where sender proves that he can spend one out of a big set of unspent coins. We show that although it comes with a security proof and relies on two secure subprotocols, the latter are connected insecurely. As a result, a spender can create coins of much bigger face value than required. We had privately disclosed the issue to the designers and collaborated with them on a fix.

1 Lelantus and Its Features

Most cryptocurrencies that aim to keep senders and recipients private, do so by hiding the transaction origins among many possible unspent coins (in the UTXO model) or among all possible accounts. Lelantus [Jiv19] is a protocol for the former setting. A coin is represented as a homomorphic commitment to its face value and its secret, so that one can prove the ownership of a coin by proving the knowledge of the commitment opening. In order to hide the particular coin one wants to spend, Lelantus exploits a modified *one-out-of-many* protocol [GK15; BCC+15]. A great advantage of Lelantus over other privacy-oriented protocols is its reasonable performance (logarithmic proof size and proving time within 2 seconds for 2^{16}-size anonymity set) and no need of trusted setup as in cryptocurrencies such as Zcash [Zca] which have to use dedicated hash functions for this.

Lelantus has been published in 2019 and has gathered quite some attention in the community. Several[1] cryptocurrency protocols[2] announced a transition to Lelantus to increase privacy. However, the critical bug that we have found seems to have eluded public review.

In Lelantus, all coins are Pedersen multicommitments of form $g^{\text{secret}} f^{\text{randomness}} h^{\text{value}}$ where the product of the first two components can be seen as owner's public key Q. An owner can Spend coins $I_1, I_2, \ldots, I_s$ by creating output coins $O_1, O_2, \ldots, O_t$ of the same form accompanied by the following (the details are important for the attack):

- *Encryption* of committed values on recipient public key and creation of range proofs on the face value of output coins.
- *Ownership proof*: specify the coin anonymity set $\mathcal{C}$ which should include the input coins among many others. For each input coin I_j with secret s_j construct a proof π_j that if we homomorphically subtract g^{s_j} from all coins in $\mathcal{C}$ then the resulting set contains a commitment to zero secret, thus

[1] https://zcoin.io/lelantus-zcoin/
[2] https://beam.mw/

proving that we own some coin in $\mathcal{C}$. The proof is a modification of [BCC+15] with the main addition that the transcript additionally contains certain values $\{G_k\}$ and $\{H_k\}$, about which we prove that they multiply to commitments to 0.

- *Balance proof*: construct a proof $\pi_{balance}$ that the combined value of output coins equals the value of input coins. For this prove that the product of output coins divided by the product of G_k is a valid coin with zero face value. For a single output and single input coin pair O_1, I_1, which suffices for us, the equation looks as follows:

$$\frac{O_1^{x^m}}{\prod_k G_k^{x^k}} \stackrel{?}{=} Q^\alpha f^\beta$$

where x is the Fiat-Shamir challenge from the ownership proof, Q is the recipient public key (equivalent to a zero-value coin) and α, β are discrete logarithms we prove knowledge of.

The protocol designers provide two security proofs for the protocol:

- That the coin ownership proof is sound, complete, and zero-knowledge.
- That the balance proof is complete, sound, and zero-knowledge.

This (among with properties of other proofs) implies the correctness of the Spend operation.

2 Our attack in brief

The problem of the protocol described above is that the security of the balance proof is proven assuming that G_k used by the prover are formed correctly. **However, this condition is not actually proven in the coin ownership proof!** Indeed, the latter implies the knowledge of a coin secret but does not automatically imply the correctness of the transcript. We have been able to exploit this and apparently create coins out of thin air.

We can explain the concept of the attack in a few formulas.Concretely, suppose we spend a coin I of value v, then in the ownership proof we create modified G_0, H_0 which we denote by $\widetilde{G_0}, \widetilde{H_0}$:

$$\widetilde{G_0} = G_0 \cdot h^\xi \tag{1}$$

$$\widetilde{H_0} = H_0 \cdot h^{-\xi} \tag{2}$$

so that their product does not change.

Prover then generates some Fiat-Shamir challenge x, which depends on $\widetilde{G}_0$. Then, instead of coin O with face value v we create an output coin $\widetilde{O} = O \cdot h^{\frac{\xi}{x^m}}$ with value $v + \frac{\xi}{x^m}$. As a result, the balance correctness proof holds:

$$\frac{\widetilde{O}^{x^m}}{\prod_k \widetilde{G}_k^{x^k}} = \frac{O^{x^m} h^{\frac{x^m \xi}{x^m}}}{h^\xi \prod_k G_k^{x^k}} = \frac{O^{x^m}}{\prod_k G_k^{x^k}} = Q^\alpha f^\beta.$$

Thus we have created a coin that has extra $\frac{\xi}{x^m}$ value.

There are several other details that make the attack possible, but we skip them for brevity.

3 Why it works

There can be several explanations why the attack works. The most interesting, however, should be the question *Why does it work if there is a proof of security?* The answer comprises several parts:

- Both coin ownership and transaction balance protocols are correct under certain assumptions on their inputs.
- The proof of the transaction balance protocol assumes something that *has not been proven* in the coin ownership protocol. Concretely, the fact that *the ownership protocol is a proof of knowledge does not imply that its transcript provably follows an honest execution.*
- The knowledge of the challenge value x by the spender had a critical importance for creating malicious output coins. The attack may still have failed if output coins have to be created prior to the ownership proof. However, this was not asserted.

4 Fix

We have communicated our findings to the protocol authors solely and early enough to prevent production deployment of the new protocol.

After the communication a rigorous soundness proof was designed and provided for the balance property of the Spend protocol. The proof uses witness-extended emulation as is defined in [Lin01] and used in [BBB+18] for example. It formally shows that whenever the spender produces an proof argument which satisfies the verifier with some probability, then there exists an emulator producing an identically distributed argument with the same probability, but also the witness of the Spend protocol. The Spend protocol witness is comprised of all transaction input coin values and their blinding factors, the random factors used to generate the G_k values which play crucial role in the balance proof construction, and also all output coin values and their blinding factors.

Although the provided proof looks correct, a more accomplished approach would be the following: to describe the Spend transaction as an multi-round interactive protocol between the spender and the verifier, which unites the range, ownership, and balance proofs all in turn discussed as interactive protocols. Then this interactive protocol could be compiled into a non-interactive protocol using the standard FS approach. In the new construction, it would be more transparent to discuss how the verifier challenges should be generated at each step of the interaction to have the attack fixed. Apparently this results in a minor modification to the full SPEND operation, concretely in rearranging steps of the proof construction and making the FS challenges dependent on all the input and output coins.

There could be other possible fixes to the protocol. An obvious one is to augment the ownership proof to ensure the correct form of G_k. However, it increases the protocol communication and computational complexity.

5 Lessons Learned

We think that the following lessons drawn from this attack are important and are often neglected by protocol designers:

- Security of subprotocols does not imply automatically that they are (or can be) securely composed into a bigger protocol.
- Secure proof-of-knowledge protocols may not have a proof of transcript correctness. Another example of this kind is a recent Zcash flaw[3] where the trusted setup transcript contained enough information to recover its secret.
- Third-party review is an important pre-requisite for a production-grade protocol. Such a review needs much more resources than a conference review, so a publication of a protocol at a conference, even a top one, does not imply a thorough check of details.

References

[BBB+18] B. Bünz, J. Bootle, D. Boneh, et al. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 315–334 (cit. on p. 4).

[BCC+15] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, et al. "Short Accountable Ring Signatures Based on DDH". In: *ESORICS (1)*. Vol. 9326. Lecture Notes in Computer Science. Springer, 2015, pp. 243–265 (cit. on pp. 2, 3).

[GK15] Jens Groth and Markulf Kohlweiss. "One-Out-of-Many Proofs: Or How to Leak a Secret and Spend a Coin". In: *EUROCRYPT (2)*. Vol. 9057. Lecture Notes in Computer Science. Springer, 2015, pp. 253–280 (cit. on p. 2).

[Jiv19] Aram Jivanyan. "Lelantus: Towards Confidentiality and Anonymity of Blockchain Transactions from Standard Assumptions". In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 373 (cit. on p. 2).

[3] https://www.coindesk.com/zcash-team-reveals-it-fixed-a-catastrophic-coin-counterfeiting-bug

[Lin01] Yehuda Lindell. *Parallel Coin-Tossing and Constant-Round Secure Two-Party Computation*. Cryptology ePrint Archive, Report 2001/107. https://eprint.iacr.org/2001/107. 2001 (cit. on p. 4).

[Zca] *ZCash protocol specification*. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf. 2020 (cit. on p. 2).