Lelantus Cryptographic Audit

Dmitry Khovratovich and Ilya Kizhvatov

ABDK.Consulting

September 2020

1 Introduction

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.

The Zcoin team has asked us to review a cryptographic paper that formally describes Lelantus and provides proofs of its security. We have identified one major problem that arises from incorrect usage of the modified one-out-of-many protocol. Concretely, it has not been proven that part of the transcript has special form, and this can be exploited by an adversary to generate fake coins. We have suggested a possible solution in the report. We have suggested several ways to repair the security proof.

We have also identified a few issues in the security proof that need clarification but do not seem to be security-critical.

The critical issue and a few less severe ones were fixed in the next version of the paper that we have reviewed. Below we provide a detailed description of the bug and other issues.

2 Fixed Critical Issue

2.1 Protocol Details

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 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.

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

- *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.

2.2 The Bug

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.

To do the above, the malicious prover executes step 6 of the **Spend** algorithm before step 3. This is possible because step 6 does not depend on any previous steps.

This indicates a problem in the protocol security theorem (Appendix A.B, pages 15-16). Apparently it assumes that if the transaction balance property does not hold then the transaction balance proof also fails. This reasoning is flawed as it implicitly assumes that G_k are formed correctly, which we show might not hold.

A proof of representation for all G_k values would not be enough because $\widetilde{G_k}$ also passes it:

$$G_k = h^{\rho_k} f^{\gamma_k + \tau_k} \tag{3}$$

$$\widetilde{G_k} = h^{\rho_k - v} f^{\gamma_k + \tau_k} \tag{4}$$

$$\tag{5}$$

We note that the G_k and Q_k values can be modified in many other ways, the only condition is that their product remains the same.

2.3 The Fix

The authors of Lelantus modified the proof by adding the output coins to the protocol transcript used when generating the x challenge, and repaired the other parts of the proof. We have found no issues in the new proof.

2.4 Another possible fix

It might be better to describe in an interactive form first, where the range, ownership, and balance subprotocols are part of the big one. Then prove the soundness of the interactive version using the rewinding technique. Finally compile the protocol into a non-interactive one applying the Fiat-Shamir transformation sufficiently many times.

3 Fixed Moderate Issues

- Appendix A.A, Page 14, line 75: it is not formally proven that C has this form. Probably it can be derived from the proof of soundness for the SPEND transaction. The same for line 79. *Explanation was added.*
- Appendix A.A, Page 14, lines 85-105: The proof is not rigorous. A regular proof of this kind would demonstrate that a successful adversary can produce S, x, or s, or we can extract these values from such an adversary. *Proof was modified.*
- Condition 4: the balance inequality does not directly imply the non-zero exponent in A/B. There must be a soundness proof of the balance protocol which would imply that. *Proof was modified.*
- Appendix A.B, Page 15, lines 6-8: It should be explained why the alleged output coin parameters are considered part of witness. Probably it follows from the structure of the range proof; otherwise it is not straightforward to prove that output coins have certain format.

A few more typos were spotted and fixed.

4 Remaining Minor Issues

Original paper:

- Page 7: In the **Balance** property paragraph it is unclear how exactly the amount of coins sent from one address to another is defined. Do we associate with each Spend an explicit list of input and output addresses? How does it deal with stealth addresses? How do we guarantee that all coins accepted by the ledger have clearly identifiable addresses?
- Page 7: In the **Ledger Indistinguishability** paragraph, it would be better to provide an explicit form of transactions that are sent to either ledger, as currently the description is not rigorous.
- Page 8, Setup algorithm: $\text{Setup}_{\dots}$ are not defined; not clear what bp and rp are (bulletproof and range proof?).
- Page 9, in ooom proof text descrition: $\sigma_{l,i}$ notation confusing with signatures, consider using $\delta_{l,i}$ instead.
- Page 9, line 59: g_k^{γ} should be f^{γ_k}
- Page 9, Figure 2:
  - The protocol does not clearly specify the statement that it is proven;
  - In the middle, the iteration on j is not aligned to the left.
  - All Z_q shoud be Z_p.
- Page 10, confusion between Q_i's from the private address and Q_k's in Fig. 2; consider renaming Q_k's in Fig. 2 as they are anyway used only internally in the ooom proof.
- Page 10, line 17: π_{range} here has 3 parameters instead of 4 (range not passed)
- Page 11: the transaction balance protocol description mixes the completeness and soundness proofs inside. Instead, one should provide an algorithm and then explicitly prove completeness, and then refer to Appendix C for soundness.

- Page 11,line 43: denominators should be g^{sn_k}
- Page 11, line 65: h should be f.
- Page 13, first paragraph of the section: Z_P should be Z_p.
- Page 13, second paragraph third line: missing figure number.
- Page 14, end of first paragraph: it is not guaranteed that serial numbers are unique. Probably the scheme is complete assuming so.
- Page 15, lines 53-56. It is not straightforward why these 5 conditions imply the inequality. Is it possible to associate a part of the inequation with each condition?
- Page 16, line 38-41: it is probably enough to show that an adversary who breaks condition 5 can produce (s, v, r), then argue that the spend transaction by honest parties do not reveal those due to zero-knowledge property of ooom-proof.
- Page 18: instead of G_k there should be $(G_k \cdot Q_k)^{-1}$, and then the last line of the protocol is rewritten as

$$c_l^{x^m_{(e)}} \cdot \prod_{k=0}^{m-1} \left((G_k \cdot Q_k)^{-1}\right)^{x^k_{(e)}} = Comm(0, z^e_v, z^e_R)$$

Updated paper:

- Page 19: w.r.t recovery of $v^o_1, \ldots, v^o_{new}$, perhaps the witness should include not these individual value but just their sum?
- Page 19: eq. (4) and following equations: R^I_t shoudl be R^i_t
- Page 20: It might be possible to extract the separate coin values if the range prover is also covered by simulator.

References

[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 p. 1).

[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. 1).

[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. 1).

[Zca] *ZCash protocol specification*. https://github.com/zcash/zips/blob/master/protocol/protocol.pdf. 2020 (cit. on p. 1).