Lelantus Spark Audit Final Report

Linfeng(Daniel) Zhou

Jan 2022

1 Introduction

Lelantus Spark is a protocol which modified Lelantus protocol to provide recipient privacy, improved security features and additional usability features. I started auditing the paper while Firo team keeps updating the Lelantus Spark protocol. This report audits the version which has been published in IACR eprint - https://eprint.iacr.org/2021/1173. I have digged into details of the paper. There is no critical security issues found but I do find several issues which are listed in this report below.

2 Findings

2.1 Missing definitions for completeness, special soundness and special honest-verifier zero knowledge of proving systems

Lelantus Spark protocol requires using representation proving system, modified Chaum-Pedersen proving system, parallel one-out-of-many proving system and all these proving systems should satisfy completeness, special honest-verifier zero knowledge and special soundness properties. However, this paper lacks formal security definitions of these important properties. Since their definitions are important, especially in security proofs as described in Appendix, I would suggest adding formal definition of them. For example,

1. *Completeness*. If $(x, w) \in \mathcal{R}$ then all honest 3-move transcripts for (x, w) are accepting.
2. *Special Soundness*. There exists an efficient algorithm Ext that, on input x and a collision for x, outputs a witness w such that $(x, w) \in \mathcal{R}$.
3. *Special Honest Verifier Zero Knowledge (SHVZK)*. There exists a PPT simulator Sim that takes as input $x \in L_{\mathcal{R}}$, security parameter 1^{λ} and $c \in \{0,1\}^{\ell}$ and outputs an

accepting transcript for x where c is the challenge. Moreover, for all ℓ-bit strings c, the distribution of the output of the simulator on input (x, c) is computationally indistinguishable from the distribution of the honest transcript obtained when $\mathcal{V}$ sends c as challenge and $\mathcal{P}$ runs on common input x and any private input w such that $(x, w) \in \mathcal{R}$. We say that Π is perfect when the two distributions are identical.

Note that the above is just an example, the paper may need to add a slightly different definition to meet the requirement.

2.2 Mismatch between the output elements and the defined output in CreateCoin

In section 4.4, the **CreateCoin** algorithm is defined to output a serial number commitment S, a recovery key K, value commitment C, value commitment range proof Π_{rp} (if $b = 0$, encrypted value $\overline{v}$ (if $b = 0$) or value v (if $b = 1$), encrypted memo $\overline{m}$. However, in step 9, the algorithm outputs the tuple $(S, K, C, \Pi_{rp}, \Pi_{rec}, \overline{v}, \overline{m})$ if $b = 0$, or output $(S, K, C, \Pi_{rec}, v, \overline{m})$ if $b = 1$. We need to either get rid of Π_{rec} in the output because it seems not used in other algorithms or re-define the output of **CreateCoin**.

As the feedback from the Firo team, the proof Π_{rec} is used in **Identify** algorithm and this issue has been fixed in the following versions.

2.3 Π_{chaum} should be an independent proof in the output of Spend transaction

In the **Spend** algorithm, it outputs a spend transaction

$$\mathsf{tx}_{spend} = (\mathsf{InCoins}, \mathsf{OutCoins}, f, \{S'_u, C'_u, T_u, (\Pi_{par})_u, \Pi_{chaum}\}_{u=0}^{w-1})$$

but Π_{chaum} is a proof independently computed so it should be an independent element in the output instead of inside of each tuple.

As the feedback from the Firo team, this issue has been fixed in the next version.

2.4 Receiver should not know $d_{\beta,k}$ and $e_{\beta,k}$ in Section 6.2, PreCompute

In section 6.2, **PreCompute** algorithm, the step 2 describes that on receipt of a vector L_β from another player β, check that $d_{\beta,k}G \neq 0$ and $e_{\beta,k}G \neq 0$ for all k. However, $d_{\beta,k}$ and $e_{\beta,k}$ should be private values of party β and party α is not able to know them. Also, since $d_{\beta,k}G$ and $e_{\beta,k}G$ are group values, we cannot check $\neq 0$. Thus, the best way to describe this is that the party α first parse each $L_{\alpha,k} = (D_{\beta,k}, E_{\beta,k})$ and check $D_{\beta,k}$ and $E_{\beta,k}$ are not group generator.

I discussed with the Firo team and basically it's a notation issue, which may mislead readers misunderstanding the paper and we agreed to make clearer notations in the next version.

2.5 Adding a comparison table comparing Lelantus Spark with other similar works

In section 7 the paper presents tables to show the efficiency of Lelantus Spark. In order to show efficiency advantages of Lelantus Spark, it would be better to add a table to list all similar works and compare corresponding value sizes in each column. If we can show Lelantus Spark provides more security features and also provide good efficiency as well, this would be a good signal and this is beneficial for this paper to be accepted.

The Firo team agreed to adding such a comparison table in the next version.

2.6 Misuse the commitment function Com in Appendix B

In Appendix B, the paper uses $\mathsf{Com}(\cdot,\cdot)$ incorrectly. For example, $A = \mathsf{Com}(\{a_{j,i}\}_{j,i=0}^{m-1,n-1}, r_A)$. However, Com only takes two values as inputs, one is a committed value, one is the randomness, this is confusing to readers, so need to change to $\mathsf{Com}(a_{j,i}, r_A)$ for all $0 \leq j \leq m-1$ and $0 \leq i \leq n-1$. There are also many other such places that misuse it in this section, so need significant change.

2.7 Lack description of the modification to payment system security definition

In Appendix C, the paper mentions that "we formally prove Spark's security within a related (but modified) security model; proofs follow somewhat similarly to that of [3]. " However, it's unclear what's the modification. We need to add context to highlight what's the modified security model and what security properties it should achieve.

We discussed this problem and agreed we have defined a security model, but it lacks considering CreateCoin, Identify, Recover queries in the security model. The Firo team will add more details about it.

2.8 Mismatch in definitions of DAP system

In section 3, when the paper introduces the definition of DAP system, it contains algorithms

$\mathbf{Setup}, \mathbf{CreateKeys}, \mathbf{CreateAddress}, \mathbf{CreateCoin}, \mathbf{Mint}, \mathbf{Identify}, \mathbf{Recover}, \mathbf{Spend}, \mathbf{Verify}$

but in appendix C it is defined as

$$\Pi = (\mathbf{Setup}, \mathbf{CreateAddress}, \mathbf{Mint}, \mathbf{Mint}, \mathbf{Spend}, \mathbf{Recover}, \mathbf{Verify})$$

There must be some differences between Lelantus Spark protocol and the original DAP system. We should either define our own DAP system (maybe Lelantus Spark achieves a stronger DAP system) and prove that all algorithms together achieve the payment system security.

The Firo team will fix this issue in the next version.

3 Minor Comments and Typos

- In section 2.5, since IND-CCA2 security implies IND-CPA security, maybe we don't need to mention IND-CPA.
- Step 3 of **Recover** algorithm misses input λ and pp, should be

$$\mathsf{Identify}(\lambda, pp, \mathsf{addr}_{in}, \mathsf{Coin}, \mathsf{AddrTable})$$

- In Section 5 **CreateKeys** algorithm, $C_{\beta,j}$ means the jth element of C_β received from player β but there is no explaination about the notation. Need to add some sentences to explain the notation.
- Since we use parallel one-out-of-many proving systems as an important building block, maybe worthwhile considering using the many-out-of-many proving system https://eprint.iacr.org/2020/293 and study if it has more advantages.
- In Appendix B, add a citation to Kronecker delta function and describe what it is.

The Firo team agreed to fix these in the next version.