We would like to thank the reviewers for their valuable comments and suggestions. We respond inline to some of the comments. Our responses are prefixed with an asterisk (\*). All "comments for authors" that do not have an answer have been addressed to the fullest extent possible in the new version.

Review #11A

=====================================================

Overall merit

-------------

3. Weak accept

Reviewer expertise

------------------

3. Knowledgeable

Weaknesses

----------

To me the main problem of this paper is its presentation. Given the

volume of the work this is admittedly very challenging, but the

current version does not contain any formal definition in the body of

the paper (not even parts of the ideal network payment

functionality). It contains several lemmas and theorems but they all

rely heavily on the appendices.

\* We have tried to incorporate many parts of the formalism in the main body, while striving to avoid a dry text without any intuition or insights. We now moved more definitions from the Appendix to the main body, e.g., the syntax and correctness definitions of the combined signature primitive. Unfortunately the main definitions (functionality, protocol) are far too lengthy to be part of the main body.

The proof sketch of the main theorem basically only explains the

structure of the proof, and on its own is not very informative. Such a

proof structure is valuable as a preliminary to the proof, but not as

a substitute, I believe. I would have preferred that the authors

explain in detail at least some part of the definitions to give the

reader some more insights.

\* We expanded the "Security proof overview" section to provide additional insight into how the actual proof works by describing the proof of one of the lemmas.

More generally, are there some general lessons learned by performing

this security analysis. One such lesson that is mentioned is the

necessity to have a less idealized ledger functionality. It would have

been interesting to illustrate more concretely the kind of properties

that could be wrongly shown secure on such an idealized ledger, but

not with your one.

\* We now mention that part of the motivation of layer-2 solutions is the high latency of the base layer. If such instant finality ledgers were realistic, the practical utility of layer-2 solutions would be much lower. Our approach highlights the considerable latency improvement that the lightning protocol offers as opposed to the direct use of the ledger.

Even though the paper is 50 pages long the body does not use the full 12

pages (only 10.5). The additional available space should be used to

add a conclusion and future work (currently missing), and give a bit

more formal details.

\* We now use all the space adding the syntax, correctness and security definitions of the Combined Signature primitive and sections for Future Work and Conclusion.

Comments for author

-------------------

- p. 4: You mention the generic composition theorem. However, many

applications do need joint state theorems. Is this the case here?

\* GUC (the model we use) is a more general case of JUC, so it seems redundant to mention the latter.

- p. 8: What are ITIs? Do you mean ITMs?

\* In UC, the term ITM is used for "code" (e.g. the protocol), whereas ITI refers to a concrete instantiation of that code. Therefore Alice and Bob are different ITIs, executing the same ITM.

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

Review #11B

=====================================================

Overall merit

-------------

2. Weak reject

Reviewer expertise

------------------

2. Some familiarity

Weaknesses

----------

Of the strengths just mentioned, only the last [perfect ledger

unrealisability] is explained in a self-contained way in the paper

itself. The functionality, the lightning mechanisms, and the

argument for the latter, are all mainly contained in the appendices.

The summary in the paper is not understandable on its own.

\* We strived for presenting a self-contained main body. The new version has numerous clarifications and new explanations, which will hopefully make the summary understandable. Furthermore, we included some more formal definitions in the main body, e.g., the syntax and correctness definitions of the Combined Signature primitive. The definitions of the functionality and protocol however are far too lengthy to include in the main body.

Assurance is an open question. The functionality is certainly not

simple, and the protocol is certainly complex. The proof does break

down into a succession of plausible partial steps. But if some small

details were a little bit wrong, would anyone be sure to notice?

Which means that it's possible something "interesting" is also wrong.

\* As alluded to in response to another comment below, there currently exists no proof checking tool for the Universal Composition framework. Creating such a tool would be an interesting direction for future work. We believe that our approach to the proof, namely following the “execution token” as it flows from one ITI to the next and analysing each possible execution path separately, to be especially effective in minimising the probability that we missed something “interesting”.

The paper's only reusable result or method appears to be the "no

instant finality" result.

\* The Combined Signatures primitive that is defined in our paper is reusable. We also now mention that our approach to modelling payment networks is as generic as possible and we delineate the changes necessary to adapt the functionality to other networks in the first bullet of "Our results".

Comments for author

-------------------

Section 5: In the (POLL) event, so you really want to allow the

functionality to halt in case of a maliciously closed channel? Or is

the point that the implementation will prevent this from happening?

\* As explained at the end of Section 5, the functionality doesn't know the low-level mechanism of LN, therefore it can only be certain that things went as planned only by checking the blockchain after channel closure.

(RESOLVEPAYS ...): "expiry values in block heights": You mean,

expiry values that are expressed as a number of blocks to be added?

\* the two expiries are expressed in absolute block height, like the cltv\_expiry field of the update\_add\_htlc message of BOLT ([https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md#adding-an-htlc-update\_add\_htlc](https://github.com/lightningnetwork/lightning-rfc/blob/master/02-peer-protocol.md" \l "adding-an-htlc-update_add_htlc)). We rephrased for clarity.

----

What modeling decisions allowed you to specify lightning

accurately and prove your results rigorously? We know

that UC is one. And rejecting "instant finality" is

another. But there were doubtless many other choices

that needed to be made.

\* We mention that synchronisation between players is handled entirely by the environment (as opposed to using a global clock) and that we leverage the liveness and persistence properties of the ledger functionality to achieve our goals.

[…]

This is of course a core issue that motivates work using

mechanized reasoning for program correctness,

programming language implementations, etc. Could a

mechanized Hoare logic (or relational Hoare logic, or

choose some other suitable formalism) help you make sure

that the secure implementation relation really holds in

detail?

\* We discuss this in Future Work. We mention that automatic proof checkers would add credibility to our approach, however such tools do not exist for the case of UC. It would indeed be worth exploring different frameworks that would allow automatic proof checking.

Are there local correctness proofs (and local

specifications of correctness) for some of the

components of the lightning protocol?

\* The "latest" state cannot be independent of G\_Ledger, or of the other parts of the protocol, as our multiple attempts to modularise better all ran into a dead end. For example, the existence of HTLCs would complicate such formulations, as coins in an HTLC are not yet in any state, but will be in the future. We would like to avoid "leaky" abstractions. However, we did modularise where possible, contributing to cryptographic agility. In particular, the characterisation of the relevant protocol parts as equivalent to

using IBS, Combined Signatures and PRFs eases the separate analysis of those components.

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

Review #11C

=====================================================

Overall merit

-------------

4. Accept

Reviewer expertise

------------------

3. Knowledgeable

Comments for author

-------------------

[...] why is it better (or ok) to give the protocol access to the two

functionalities as separate ones?

\* We state in the 3rd paragraph of "Hybrid functionalities used" that our protocol does not directly use the G\_Ledger hybrids.

- why is the \*number\* of times that the parties query the status of the

chain important? maybe there can be a simpler mechanism whereby a party

can possibly catch a cheater with some probability?

\* This could be achieved by assigning some distribution to the POLL messages sent by the Environment. This would however restrict us to specific environments, therefore precluding the composability of our result.

-how would "slow down" attacks (ie attacks where the adversary sufficiently

slows down one of the parties so that it cannot check the gossip network or

ledger in time) be captured in your analysis? are they allowed? or ruled

out by some ideal functionality? (and then, how do you justify this ruling

out?)

\* We explicitly model such situations already. We explain in the Introduction that it is possible for parties to be negligent if they don't check the ledger often and state exactly when a party is negligent in the 2nd bullet of section 5, items 1,2 and 3.

- the need to check the ledger is not symmetric (ie, only one of the

parties needs to). can one decide which side would be the one that needs

to check?

\* The exact polling requirements are presented in Section 4, 1st and 2nd bullets. Both counterparties have to poll often otherwise they risk losing funds. In particular, if there exist two past channel states F, G such that Alice is better off in F than in the latest state and Bob is better off in G than in the latest state, then both have to poll regularly.

- The paper claims that the previous work assumed that the ledger has

instant finality ("finality" is the time needed for a transaction to appear

on the ledger). This does not appear to be correct, in particular in

[13,16] messages are assumed to arrive to the ledger functionality in time

at most some parameter delta . Please address how your work differs.

\* [16] explicitly states that it uses a synchronous communication model. Also its Ledger updates balances directly upon receiving a relevant command. We have added a justification on why we find the model of [13] non-satisfactory in "Related Work".

- The paper doesnt appear to model the fees, which are an essential part

of the protocol, and where interesting attacks have been recently mounted

(see https://eprint.iacr.org/2018/472.pdf) How will the introduction of

fees change the security modeling? How do you justify abstracting the fees

out?

\* This is a great point. Our model is already quite extensive and we had to limit the scope to a level that the modelling and analysis would still be feasibly done in one paper. Adding fees would only increase the complexity of all parts (protocol, functionality, simulator, proof) and would make the balance security guarantees less clear. It is worth noting that the attack mentioned is already captured by our model, as an adversary that controls two non-neighbouring nodes on a payment path can skip the intermediate nodes. Nevertheless, such an attack is inconsequential in our analysis given the lack of fees. As we now mention in Future Work, it is possible to extend our formalism to a stronger functionality which enforces the utilisation of the path specified by the payer. This functionality would express the security guarantees that lightning will offer after the introduction of the PTLC mechanism (<https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-December/002375.html>, “Pointlocked Timelocked Contracts” section).