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

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 full paper will be available on e-print.

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 have improved and expanded the narrative of the main body adding suitable information from the appendix as well as responding to the comments of the committee. Short sections on Future Work and Conclusion are also added.

Comments for author

- p. 4: You mention the generic composition theorem. However, many
applications do need joint state theorems. Is this the case here?
* The GUC (the model we use) is more general than JUC (UC with joint state); in particular, the ledger is modelled as a global functionality and the environment has direct access to its state. Thus, other arbitrary protocol instances can operate within the environment and freely interact with the ledger (as indeed anticipated in the real world where lightning is deployed over the bitcoin ledger).

- 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 instance that runs that code. Therefore, Alice and Bob are different ITIs, executing the same ITM. We refer to https://eprint.iacr.org/2000/067 for more details.

* *

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.
* In the new version, 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 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.
* This is a great point. 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”. Nevertheless, we believe that a full formally verified proof would be a great next step (and we hope that our work would motivate it). We now describe this in the future work section where we cite “Ran Canetti, Alley Stoughton, Mayank Varia: EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security. CSF 2019: 167-183” as the natural candidate to base this future work on along with our paper.

The paper's only reusable result or method appears to be the "no
instant finality" result.
* While reusability is probably something that is somewhat subjective, we would hope that the Combined Signatures primitive that is defined in our paper is also reusable in the same sense the reviewer uses the term above. It captures an interesting set of properties in terms of controlling message integrity and authentication between two parties and abstractly expressing it in the way we did, contributes to both the

primitive's cryptographic agility as well as its potential use for modularising security in other protocols. 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" so any future work can more readily relate to 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?
* Yes indeed. 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. If a protocol is to implement the paynet functionality it should take the necessary steps so that it doesn't halt.

(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). 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.
* Yes indeed. We mention now in more detail that synchronisation between players is handled entirely by the environment (as opposed to using the global clock) and that we leverage the liveness and persistence properties of the ledger functionality to achieve our goals. Another modelling decision that we now expand more on in the main body is modularization of intermediate cryptographic primitives such as combined signatures.

[...]

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?
* This is a great point. We expanded more on this in Future Work. We believe that combining our results with recent work of "Ran Canetti, Alley Stoughton, Mayank Varia: EasyUC: Using EasyCrypt to Mechanize Proofs of Universally Composable Security. CSF 2019: 167-183" would be an interesting direction towards this goal for future work.

Are there local correctness proofs (and local specifications of correctness) for some of the components of the lightning protocol?
* We did modularise lightning where it was possible. In particular, the characterisation of the relevant protocol components in lightning as equivalent to using identity based signatures, PRFs and our newly defined primitive of Combined Signatures is all novel to our work. We believe that this modularisation not only helps in the analysis but also contributes to the cryptographic agility of the protocol enabling the substitution of such primitives with others.

* *

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 any functionality beyond G_ledger. The existence of other functionalities such as G_clock and their interaction with G_ledger is in accordance with the GUC framework.

- 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?
* I think our choice is the simplest and most general possible. We “delegated” polling to the environment and expressed syntactically the conditions under which the paynet functionality will offer its guarantees. Using now our machinery, one can deploy the paynet functionality within an environment that does probabilistic polling according to some distribution and then prove a probabilistic result along the way suggested by the reviewer. However, from our point of view this would restrict us to specific environments, complicating the composability of our result and restricting its generality.

-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. For example, 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.
* We expanded with more details about the problems of the abstractions used in previous works including [13,16]. In more detail, [16] explicitly states that it uses a synchronous communication model. Also its Ledger updates balances directly upon receiving a relevant command so the issue of instant finality applies. Regarding [13], indeed a delay is allowed but the problem with their abstraction lies elsewhere - we have added more details on why the model of [13] is non-satisfactory in "Related Work" (p.3 right column).

- 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 and analysis 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 the security analysis given the lack of fees. We expand on this point further in the end of Section 3 as well as in Future Work, where we explain how it is possible to extend our paynet functionality formalism to adhere to the path specified by the payer. This (slightly stronger) functionality would express the enhanced security guarantees that lightning will offer after the upgrade to the PTLC mechanism (https://lists.linuxfoundation.org/pipermail/lightning-dev/2019-December/002375.html, "Pointlocked Timelocked Contracts" section) that prevent attacks like the wormhole attack. Fully incorporating fees and analyse lightning with the PTLC mechanism will be an interesting step for follow up work.