Payment Channels with Proofs

Chad E. Brown
Czech Technical University in Prague, Czech Institute of Informatics, Robotics and Cybernetics, Prague, Czech Republic

Cezary Kaliszyk
University of Melbourne and University of Innsbruck

Josef Urban
Czech Technical University in Prague

Abstract

The fundamental building blocks of the Bitcoin lightning network are bidirectional payment channels. We describe an extension of payment channels in the Proofgold network which allow the two parties to bet on whether a proposition will be proven by a certain time. These provide the foundation for a Proofgold lightning network that would allow parties to request proofs (by betting there will be no proof by a certain time) and other parties to provide proofs (and be rewarded by betting there will be a proof). The bets may also provide a way to approximate the probability that a certain proposition is provable (in the given amount of time). We describe the implementation of payment channels with proofs in Proofgold and discuss a potential lightning network that could be built as a result. One application of such lightning network would be a large decentralized infrastructure for fast collaborative formalization projects.

2012 ACM Subject Classification Networks → Network architectures; Theory of computation → Logic and verification

Keywords and phrases Blockchain, payment channels, proofs, lightning network

Digital Object Identifier 10.4230/OASIcs.CVIT.2016.23

Supplementary Material The repository version of ProofGold Lava that includes an implementation of payment channes with proofs: `https://github.com/ckaliszyk/proofgold-lava`

Acknowledgements We want to thank Thibault Gauthier who provided some of the important ideas ultimately leading to this work. The results were supported by the Ministry of Education, Youth and Sports within the dedicated program ERC CZ under the project POSTMAN no. LL1902 and the ERC PoC grant *FormalWeb3* no. 101156734.

1 Introduction

The Bitcoin lightning network [3] provides a Layer 2 solution for securely and quickly sending off-chain payments. The same ideas can be used to create a lightning network for payments in similar cryptocurrency networks, e.g., Proofgold [1]. The ideas underlying the Bitcoin lightning network can be extended in Proofgold to allow users on the network to also pay for proofs of theorems by betting on whether a theorem will be proven by a given deadline. We describe such an extension in this paper. The bets could also be used as a prediction market, to estimate the probability that a proof of a proposition will be public by a certain deadline.

Proofgold is a cryptocurrency supporting formal logic and mathematics launched in 2020. In Proofgold users can publish theories (primitives and axioms) within a framework of intuitionistic higher-order logic. These theories can be developed as users publish documents, making new definitions and proving new theorems within the theory. As of early 2025, six theories have been created in Proofgold, with the most developed one being a higher-order set theory [2]. In addition, over 2000 definitions and over 50000 proofs have been published into the Proofgold blockchain. Proofgold users can also place bounties on conjectures they would

like to see proven. As of early 2025, over 13000 bounties have been placed on conjectures and almost 3000 have been collected (as the result of someone proving the conjecture or its negation).

The usual method of publishing Proofgold documents onto the chain is time consuming. Proofgold has a block time approximating one block per hour. Furthermore, a commit-and-reveal scheme is used to protect document authors from plagiarism. As a consequence, it takes about a day to publish a document into the Proofgold blockchain. An appropriate off-chain lightning network would provide a faster alternative to placing a bounty on a conjecture (on chain) and waiting for someone to publish a document (on chain) with a corresponding proof of the conjecture in order to collect the bounty. Since the lightning network would work off-chain, we will also need an alternative to the commit-and-reveal scheme Proofgold uses (on chain) to ensure authorship of proofs. The alternative we consider is paying agents who bet there will be a proof by a certain deadline, regardless of who authored the proof. Those agents will be paid by other agents who bet there will not be a proof (presumably because they want to pay for a proof to appear).

2 Payment Channels and Lightning

The fundamental building blocks of the lightning network (for Bitcoin) are bidirectional payment channels (or, simply "payment channels"). We give a simple explanation here, and leave the curious reader to study [3] for more details.

A payment channel involves two agents, e.g., Alice and Bob. To open a payment channel, Alice and Bob combine some bitcoin into a single asset (bitcoin utxo). For example, suppose Alice and Bob both have 1 bitcoin and they together build a transaction spending each of their bitcoins to a new address with 2 bitcoins. The new address is a 2-of-2 multisig address, so it can only be spent if Alice and Bob agree on how to spend it. (That is, they must *both* sign any transaction spending from the address.)

Of course, if they simply do this, then Bob can hold Alice's bitcoin hostage – only agreeing to spend the 2 bitcoins if he receives, e.g., 1.9 bitcoins and Alice receives only 0.1. Likewise, Alice can hold Bob's bitcoin hostage for the same reason. So, before sending the bitcoins to the multisig address, Alice and Bob create three unsigned transactions: a "funding transaction" sending Alice and Bob's bitcoin to the combined multisig address and two initial "commitment transactions." Before Alice or Bob sign and publish the funding transactions, each must sign one of the commitment transactions. The commitment transactions reflect the current balance of the payment channel. We will oversimplify and say both commitment transactions spend from the (currently hypothetical) utxo from the funding transaction with two outputs: 1 bitcoin for Alice and 1 bitcoin for Bob. Alice creates this commitment transaction, signs it, and sends it to Bob. Bob does the same. Now both can sign and publish the funding transaction, so that the 2 bitcoins at the multisig address is an actual asset (utxo) with 2 bitcoins. These 2 bitcoins cannot be held hostage by either Alice or Bob, since either can sign and publish their commitment transaction (which was already signed by the other party).

The procedure described above does allow Alice and Bob to create an asset (essentially "opening a channel") and spend the asset to recover their balance (essentially "closing a channel"), but lacks the ability to *update* the balance. Without the ability to update the balance, payments are not yet possible. In order to allow for payments, the commitment transactions use hash timelock contracts (htlcs).

Described simply, an htlc is a certain kind of script that can be spent in one of two ways:

by one party after enough confirmations or by another party (immediately) with a "secret." Extending the example above, suppose the two initial commitment transactions Alice and Bob did are as follows:

- Alice's initial commitment transaction spends the multisig utxo with 1 bitcoin to Bob and 1 bitcoin to an htlc address. The htlc address is spendable by Alice after 3 days but is also spendable by Bob immediately if Bob knows Alice's secret. Assume only Alice knows Alice's secret initially.
- Bob's initial commitment transaction is symmetric, spending the multisig utxo with 1 bitcoin to Alice and 1 bitcoin to an htlc address. The htlc address is spendable by Bob after 3 days or by Alice immediately with Bob's secret. Again, assume only Bob knows Bob's secret initially.

Assume Alice signs Bob's initial commitment transaction and Bob signs Alice's initial commitment transaction before they both sign the funding transaction. After the funding transaction has been confirmed, the payment channel is open. Alice can close the channel by signing and publishing her initial commitment transaction. This would give Bob access to his 1 bitcoin immediately and give Alice access to her 1 bitcoin after 3 days. Assuming Bob does not know Alice's secret, her 1 bitcoin will be unspent and available after those 3 days. Likewise Bob could close the channel by signing and publishing his initial commitment transaction, causing him to wait 3 days for his 1 bitcoin to be available.

The use of htlc's and secrets makes the current pair commitment transactions *revokable*. This is what allows Alice and Bob to update their balance. For example, suppose Alice wishes to send 0.1 bitcoin to Bob over the channel. Assuming the two of them agree, they create a new pair of commitment transactions (with new secrets) sending 0.9 to Alice and 1.1 to Bob. They then reveal the previous secrets to each other. Now that Alice and Bob know the previous secrets, neither can close the channel using the old commitment transactions without losing their entire balance in the channel. Thus they can still close the channel at any time, but can only do so by using their latest commitment transaction.

It will be convenient to have a notation for htlc scripts. Let h be the (SHA256) hash of a secret s (a 256 bit number), N be a number of blocks and α and β be addresses (the hash of the public key of a private key) and let $\mathsf{h}(h, \alpha, N, \beta)$ denote the htlc script that can be spent in two ways (where which way is chosen by the spender):

1. A secret s hashing to h is given and the transaction is signed by α. This behavior uses cryptographic operations common to both the Bitcoin and Proofgold scripting languages.
2. At least N blocks have passed and the transaction is signed by α. Checking if N blocks have passed is enforced by `OP_CSV` (check sequence verify), an operation common to both the Bitcoin and Proofgold scripting languages.

Suppose α is Alice's address and β is Bob's address. In practice the N would be chosen to be a specific number. For example, in Bitcoin taking $N = 288$ would require 288 confirmations (roughly 2 days) before $\mathsf{h}(h, \alpha, N, \beta)$ is spendable by Bob. This would give Alice roughly two days to notice if Bob publishes an outdated commitment transaction. Each of Alice's commitment transactions sends Alice's balance to an address controlled by the script $\mathsf{h}(h^A_m, \beta, N, \alpha)$ and Bob's balance to β. Here h^A_m is the hash of Alice's m^{th} secret s^A_m. Each of Bob's commitment transactions sends Alice's balance to α and Bob's balance to $\mathsf{h}(h^B_m, \alpha, N, \beta)$. Here h^B_m is the hash of Bob's m^{th} secret s^B_m. If m is the most recent pair of commitment transactions, then both parties should know s^A_i and s^B_i for each $i < m$, but only Alice should know s^A_m and only Bob should know s^B_m.

Once there are multiple agents with open payment channels, it is clear that one can implement a network of such channels. As a simple example, suppose Alice and Bob have

an open payment channel and Bob and Charlie have an open payment channel. Suppose Alice has a balance of at least 1 bitcoin in her payment channel with Bob, and that Bob has a balance of at least 1 bitcoin in his payment channel with Charlie. In this case Alice can send Charlie 1 bitcoin by sending Bob 1 bitcoin via the first payment channel while Bob (atomically) sends 1 bitcoin to Charlie via the second payment channel. Note that Alice's balance has decreased by 1 bitcoin and Charlie's balance has increased by 1 bitcoin. Bob's total balance has not changed: his balance in the first payment channel has increased by 1 bitcoin and his balance in the second payment channel has decreased by 1 bitcoin. In order to incentivize Bob to play this intermediate role, he may charge Alice a small fee. For example, Bob could require Alice to send Bob 1 bitcoin and 1 satoshi on the first channel while Bob still sends only 1 bitcoin to Charlie on the second channel.

The lightning network is essentially a network of payment channels where payments can be routed through multiple payment channels. There are a number of details (such as finding routes with sufficient liquidity) which are beyond the scope of this simple explanation.

3 Payment Channels with Proofs in Proofgold

Since Proofgold supports multisig and htlc addresses, bidirectional payment channels as described for Bitcoin can also be opened, updated, and closed in Proofgold. A lightning network is arguably more important on Proofgold, as Proofgold's Layer 1 is slower than Bitcoin's; Bitcoin has 1 block every 10 minutes on average while Proofgold has 1 block an hour on average. Suppose Alice and Bob have an open payment channel with a 2-of-2 multisig utxo with 200 Proofgold bars and a current balance of 100 bars for Alice and 100 bars for Bob. The current balance is reflected by the latest pair of commitment transactions that Alice and Bob can use to close the channel.

Suppose Alice wants a proof of proposition P. In Proofgold (Layer 1), she could put a bounty on the proposition P. Whoever proves P (or its negation) would be able to collect the bounty. Proofgold uses a commitment scheme so that the author of a proof can safely publish a proof without having it "stolen" during the publication process. The author first publishes a commitment and (roughly 12 hours) later publishes the document with the proof. If someone else wanted to "steal" the proof, they would first see it when the document is being published. They would then need to publish their own commitment and wait another (roughly) 12 hours before they could publish their version of the document (essentially replacing the original author with themselves). Presumably, the original document with the original author would have been published during this waiting period. The first author to publish the proof of a proposition P becomes the "owner" of P. The owner of P can collect any bounties on P.

Lifting the bounty mechanism to a Layer 2 lightning network seems impossible. One cannot use the commitment scheme since that relies on Layer 1, which is slow. This conflicts with the desire for Layer 2 to be fast and not need to wait for blocks on Layer 1. Layer 1 should only be used to open and close channels.

Since Alice and Bob have an open channel, she could simply announce to Bob that she will pay 50 bars for a proof of P. Suppose Bob proves P. If he sends her the proof, she has no reason to update the balance to reflect paying Bob 50 bars, and Bob has no recourse. If he claims to have the proof but will only reveal it after Alice sends the 50 bars, she has no recourse if she sends the 50 bars but Bob never reveals a proof. Ultimately, Layer 1 should be the "recourse" if one party is uncooperative. That is, if one party is uncooperative, the other party should be able to close the channel in a way that reflects the expected balance.

Suppose Alice and Bob update the balance using new commitment transactions with three

outputs instead of two. One output would send 50 bars to Alice and the second output would send 100 bars to Bob. One of these first two outputs would be an htlc address (depending on which of the pair of commitment transactions it is). The third output would send 50 bars to a third address. The intention of this third address is that the 50 bars go to Bob if Bob has a proof of P and the 50 bars go to Alice otherwise. Assuming we can do this, then Bob can safely reveal the proof of P to Alice privately. Alice should then update the balance of the channel to reflect that she has 50 bars and Bob has 150 bars, without the need for a third output. Otherwise, Alice is uncooperative and Bob can publish the proof of P on the blockchain and close the channel by publishing the commitment transaction with the third output. Since P will have been proven on Layer 1, Bob should be able to spend the third output with 50 bars to himself. On the other hand, if Bob tries to close the channel without P being proven, Alice should be able to collect the third output (after an appropriate timeout, assuming Alice has requested a proof of P by a certain time).

Note that in this high-level description it is not important that Bob actually be the one to prove P. If anyone else publishes a proof of P on Layer 1, then Bob can still use this to collect the 50 bars. As a consequence, the 50 bars can be seen as a "bet" by Alice that Bob will not know a proof of P by some timeout and a "bet" by Bob that he will know a proof of P by the timeout (even if the proof was given by someone else). With the current setup, however, it is not quite a "bet" since Alice makes no reward if no one provides a proof by Alice's timeout. Also, since the bet is specific to paying Bob, but not specific to Bob proving P, it is natural to question why Alice would make this bet with Bob instead of someone else with whom she has a payment channel.

Bob could encourage Alice to make the bet with him instead of another partner by adding 10 bars of his own to the bet. After making such a bet, the commitment transactions would send 50 bars to Alice, 90 bars to Bob and 60 bars to the third output. In this scenario, Alice will be rewarded 10 bars for "winning" the bet because no one proved P in time. We will now assume this modification, so that the third output is worth 60 bars.

As a first approximation, the third output should be spendable by a script that lets Bob spend the third output if P has been proven and lets Alice spend the third output after a certain time has passed. The key element of Proofgold's scripting language that allows for such an output is an operation `OP_PROVEN` which checks if a given proposition (identified by its hash root) has been proven in the Proofgold blockchain. Let us call such a script a *ptlc (proposition time lock contract)* and denote it by $\mathsf{p}(P, \beta, T, \alpha)$ for a proposition P, an address β, a time (either in block height or unix timestamp) T and an address α. In the discussion above, β is Bob's address and α is Alice's address. The script $\mathsf{p}(P, \beta, T, \alpha)$ states that the output can be spent in two different ways (which way is determined by spender):

1. The proposition P has been proven (on chain) and the transaction is signed by the private key address for β. This is where `OP_PROVEN` is used, in addition to the usual operations for signature checking.
2. Time T has passed and the transaction is signed by the private key for α. This uses the usual operations for signature checking and `OP_CLTV` (check locktime verify). `OP_CLTV` is an operation common to both Bitcoin and Proofgold and checks if a certain time has passed (either judged by block height or by the unix timestamps of blocks).

Using the ptlc $\mathsf{p}(P, \beta, T, \alpha)$ does work to enforce the bet as described above, but has the drawback that it is not revokable. Suppose Bob does prove P (off chain) and Alice acknowledges this by updating the payment channel to reflect Alice's balance of 50 bars and Bob's balance of 150 bars. Updating the payment channel requires Alice to reveal the secret she used for the htlc of her previous commitment transaction (with the third output

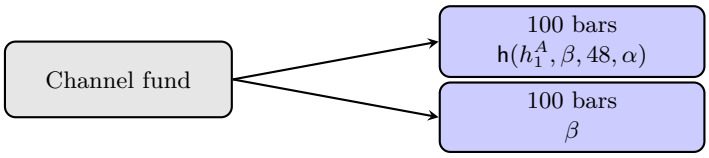

Figure 1 Alice's initial commitment transaction τ_1^A

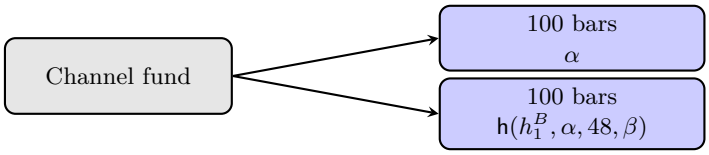

Figure 2 Bob's initial commitment transaction τ_1^B

corresponding to the bet). Alice could, nevertheless, wait until time T has passed and publish her revoked commitment transaction with the third output. If she does this, Bob can take the first two outputs, using his signature and Alice's revealed secret. This allows Bob to take 140 bars. However, he cannot take the third output – with 60 bars – until he publishes the proof of P on chain. Before he does so, Alice can spend the third output (since time T has passed) and obtain 60 bars. In effect, Alice has been able to use a revoked commitment transaction to withdraw more than her intended balance.

We can remedy this situation by combining the htlc behavior with the ptlc behavior. That is, we also need the third output to be controlled by the counterparty after a secret has been revealed. We simply denote this as the composition $\mathsf{h}(h, \delta, N, \mathsf{p}(P, \beta, T, \alpha))$ where h is the hash of a secret, N is a number of blocks, P is a proposition, T is a time and δ, β and α are addresses. The script $\mathsf{h}(h, \delta, N, \mathsf{p}(P, \beta, T, \alpha))$ states that the output can be spent in two different ways (chosen by the spender):

1. A secret s hashing to h is given and the transaction is signed by δ.
2. At least N blocks have passed and inputs for the script $\mathsf{p}(P, \beta, T, \alpha)$ are given.

Note that there are actually three ways of spending the output, since the script $\mathsf{p}(P, \beta, T, \alpha)$ can be satisfied in two ways. Note also that though there are three addresses α, β and δ, the intention is that δ is either α or β (whichever is the counterparty of the commitment transaction).

Let us reconsider the example above. We will assume the value of N is 48, so that the htlc behavior requires a user to wait for 48 confirmations (roughly 2 days in Proofgold). Alice and Bob both make initial commitment transactions before the funding transaction is signed, transmitted and confirmed. Alice's initial commitment transaction τ_1^A will use secret s_1^A with hash h_1^A and will have two outputs: 100 bars to $\mathsf{h}(h_1^A, \beta, 48, \alpha)$ and 100 bars to β, as shown in Figure 1. For Bob's initial commitment transaction τ_1^B, the roles of α and β are reversed, with an output of 100 bars to α and 100 bars to $\mathsf{h}(h_1^B, \alpha, 48, \beta)$, as shown in Figure 2. After these initial commitment transactions are signed by the counterparty and the funding transaction has confirmed, the channel is open.

Suppose Alice and Bob agree to bet on whether or not Bob will have (access to) a proof of P at some block height T in the future, with Alice contributing 50 bars and Bob contributing 10 bars. Since the htlc behavior will lead to a delay of 48 blocks, we use the block height $T + 48$ in the ptlc below. Alice's new commitment transaction τ_2^A will have three outputs: 50 bars to $\mathsf{h}(h_2^A, \beta, 48, \alpha)$, 90 bars to β and 60 bars to $\mathsf{h}(h_2^A, \beta, 48, \mathsf{p}(P, \beta, T + 48, \alpha))$, as shown

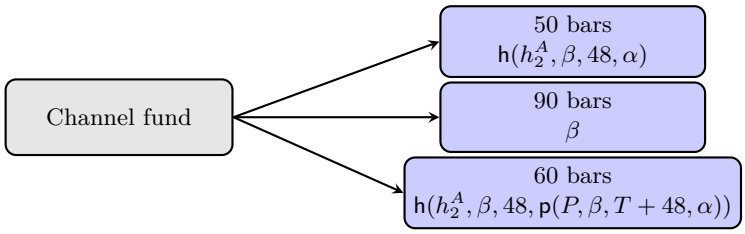

Figure 3 Alice's commitment transaction τ_2^A with the bet that Bob will not have a proof of P before block height T

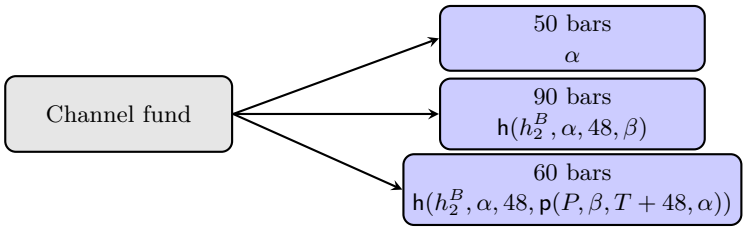

Figure 4 Bob's commitment transaction τ_2^B with the bet that Bob will have a proof of P before block height T

in Figure 3. Bob's new commitment transaction τ_2^B will have three outputs: 50 bars to α 90 bars to $\mathsf{h}(h_2^B, \alpha, 48, \beta)$, and 60 bars to $\mathsf{h}(h_2^B, \alpha, 48, \mathsf{p}(P, \beta, T + 48, \alpha))$, as shown in Figure 4. After the counterparty has signed the new commitment transactions, Alice and Bob revoke the previous commitment transactions by sharing the secrets s_1^A and s_1^B.

Suppose Bob has a proof of P before block height T and shares this proof with Alice. Alice should agree he has won the bet and update the channel to reflect a balance of 50 bars for Alice and 150 bars for Bob. Otherwise, Bob should publish τ_2^B and publish the proof of P in a document on the Proofgold blockchain. After 48 blocks (which is presumed to still be before block height $T + 48$), Bob can spend the outputs controlled by $\mathsf{h}(h_2^B, \alpha, 48, \beta)$ and $\mathsf{h}(h_2^B, \alpha, 48, \mathsf{p}(P, \beta, T + 48, \alpha))$, giving him 150 bars, as desired. Note that if Bob delays and τ_2^B is confirmed at block height T or after, then Alice will also be able to spend the 60 bars controlled by $\mathsf{h}(h_2^B, \alpha, \mathsf{p}(P, \beta, T + 48, \alpha))$ and it will essentially be a competition to spend the output first. Thus Bob should really supply the proof of P at most a few blocks before height T to prevent this possibility.

Next suppose block height T has arrived and Bob has not shared a proof of P with Alice. Bob should agree she has won the bet and update the channel to reflect a balance of 110 bars for Alice and 90 bars for Bob. Otherwise, Alice should publish τ_2^A at block height T. After 48 blocks, the block height will be beyond $T + 48$ and Alice can spend the outputs controlled by $\mathsf{h}(h_2^A, \beta, 48, \alpha)$ and $\mathsf{h}(h_2^A, \beta, \mathsf{p}(P, \beta, T + 48, \alpha))$, giving her 110 bars, as desired. However, Bob still has the possibility of finding and publishing a proof of P on chain before the 48 confirmations have passed. In this case, after 48 confirmations both Alice and Bob will race to spend the 60 bars controlled by $\mathsf{h}(h_2^A, \beta, \mathsf{p}(P, \beta, T + 48, \alpha))$.

Given this behavior, it is more accurate to say Bob is betting he will have a proof of P at the latest a few blocks before T, and Alice is betting Bob will not have a proof of P by block $T + N$ (e.g., $T + 48$ in our example). If Bob obtains a proof during the uncertain period from T to $T + N$, either could be considered the winner of the bet. In such a case,

it would arguably be in Bob's interest to withhold sharing the proof and make a new bet before revealing it – as he would be certain of winning the new bet and have only a partial chance of winning the old bet.

The appropriate commands to support payment channels with proofs have been implemented in the Proofgold Lava client [1]. These commands would be the backbone for a user friendly interface for creating and updating channels, including making bets as described above.

4 A Lightning Network with Proofs

Similar to the way payment channels can be combined to form the lightning network, it seems possible to do the same with these "bets" about the existence of proofs by a time limit. In simple terms, Alice could bet Bob as above (50+10=60 bars) that there will not be a proof of P by Block T. Bob could later hedge by betting Charlie 50+10=60 bars there will not be a proof of P by Block T. In this case, if no one proves P by Block T, Alice will gain 10 bars, Bob's total balance will not change (losing 10 bars on the first channel but gaining 10 bars on the second channel) and Charlie will lose 10 bars. On the other hand, if someone proves P by Block T, Alice will lose 50 bars, Bob's total balance will not change (gaining 50 bars on the first channel but losing 50 bars on the second channel) and Charlie will gain 50 bars. Effectively, Bob will have sold his side of the bet to Charlie.

In the case of three actors, it is useful to consider the motivations of each actor. If Charlie finds a proof of P, he is motivated to share the proof with Bob to win one bet and then Bob is motivated to share the proof with Alice to win the other bet. In general, there is a motivation for proofs to propagate from those betting there will be a proof to those betting there will not be a proof. On the other hand, if Alice finds a proof of P, she is not motivated to share it at all. As long as Charlie does not have access to a proof of P by the time limit, he has technically lost the bet (even if Alice has obtained a proof of P). This suggests a third possibility that both Alice and Charlie lose the bet. Suppose Bob finds a proof of P. He is motivated to share the proof with Alice, but not to share the proof with Charlie so that he can potentially win both bets. If Alice wishes to protect against such a possibility, then she could agree that whenever her channel counterparty (Bob) sends her a proof of P, she makes that proof publicly available immediately. In this case, Charlie will have access to the proof as well. Alice will lose nothing by making the proof public. Alice will also lose nothing by allowing Bob (who was not intended to be a party to the bet) to win both bets, so the appropriate behavior when a middleman finds the proof is admittedly questionable.

5 Prediction Market

Suppose again that Alice is willing to bet 50 bars Bob will have no proof of P by Block T and Bob is willing to bet 10 bars he will have a proof of P by Block T. This can be interpreted as the two parties agreeing that the probability that Bob will have a proof of P by Block T is 0.1667 (1 chance out of 6). Alice is incentivized to take the highest counteroffer (since she will then maximize her payout if no proof appears in time). If Charlie is willing to bet Alice 50 bars he will have a proof of P by Block T, then Alice will prefer to make the bet with Charlie than Bob. The bet with Charlie can be interpreted as the probability that Charlie will have a proof of P by Block T is 0.5. In an extreme case, before Alice has accepted a counteroffer, Daria may secretly find a proof of P. It is then in Daria's interest to bet Alice as much as possible (since Daria is sure she will win the bet). Thus when someone

(other than Alice) has a proof, the bets will reflect that by making the apparent probability approach 1. If Alice has a proof, she will presumably no longer offer to make such bets.

On the other hand, suppose Alice believes P is not provable (e.g., she has privately proven its negation). She can freely make arbitrary bets that P will not be proven by a deadline as she will win every bet (unless, of course, the relevant theory turns out to be inconsistent). Over time, it seems likely there will be fewer counterparties willing to bet Alice, even with very low contributions to the bet. This would push the apparent probability towards 0.

It is likely the odds of the bet will change over time based on the perceived likelihood that P will be proven by Block T. This change would be reflected in multiple participants making new bets which would give the lightning network with proofs many properties of a prediction market. A similar idea for a prediction market regarding the potential truth of mathematical statements was described in [4].

6 Conclusion

We have described an extension of payment channels in Proofgold to allow counterparties to bet on whether a certain proposition will be proven by a certain time. The payment channels with proofs have been implemented in the Proofgold Lava repository version (as linked from the supplementary material) and have been tested[1].

These payment channels can be combined into a lightning network allowing parties to bet on propositions via combining bets through a route consisting individual channels. The bets can be interpreted as indicating a probability, opening up the use of the network as a prediction market for the provability of mathematical statements by a given deadline.

References

1 Chad E. Brown, Cezary Kaliszyk, Thibault Gauthier, and Josef Urban. Proofgold: Blockchain for formal methods. In Zaynah Dargaye and Clara Schneidewind, editors, *4th International Workshop on Formal Methods for Blockchains, FMBC@CAV 2022, August 11, 2022, Haifa, Israel*, volume 105 of *OASIcs*, pages 4:1–4:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. URL: `https://doi.org/10.4230/OASIcs.FMBC.2022.4`, `doi:10.4230/OASICS.FMBC.2022.4`.

2 Chad E. Brown and Karol Pąk. A tale of two set theories. In Cezary Kaliszyk, Edwin C. Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen, editors, *Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings*, volume 11617 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2019.

3 Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016. https://lightning.network/lightning-network-paper.pdf.

4 Borching Su. Mathcoin: A blockchain proposal that helps verify mathematical theorems in public. *IACR Cryptol. ePrint Arch.*, 2018:271, 2018.

[1] `https://github.com/ckaliszyk/proofgold-lava/blob/69fc329ab1175ab01e69385abb88402a4ee447e4/doc/paymentchannels-pfg.md`