Notebook - Bitcoin-lightning-network-paper-DRAFT-0.5

Page 1 | Highlight

> The payment network Visa is believed to do 45,000 peak transactions per second on its network during holidays, and hundreds of millions average per day. Currently, Bitcoin supports around 7 transactions per second with a 1 megabyte block limit. If we use an average of 300 bytes per bitcoin transaction and assume unlimited block sizes, an equivalent capacity to peak Visa transaction volume of 45,000/tps would be nearly 8 gigabytes per Bitcoin block, every ten minutes on average. Continuously, that would be over 400 petabytes per year. Clearly, that isn’t feasible today. Today’s personal computers cannot operate with that kind of bandwidth and storage. If Bitcoin is to replace all electronic payments in the future, not just Visa, as currently implemented it can only achieve a small portion of that, or at best, scale with extreme centralization of a few capital-intensive Bitcoin nodes and miners.

Note:

Trasaction yavaşlığı bahsedilmiş ve
bu neden olacak verebilirsin
↳ neden lightning daha iyi bitcoin yerine

Page 4 | Highlight

> payments in micropayment channels are real bitcoin communicated and exchanged off-chain.

Note:

Lightning network'de offchain varmı
bakmak lazım

Page 5 | Note

Hashlock kısmında süre geçtikten sonra
bir istek atılıp işlemi geri alınıyor yoksa
otomatik mi gerçekleşiyor

Page 6 | Highlight

> An HTLC is opened by creating a transaction output which only the final recipient can redeem.

Page 6 | Highlight

> The recipient first generates random data R, and hashes R using hash(R) to produce H. This information is provided directly from receiver to sender of funds, along with the recipient's bitcoin address. The sender routes their payment to the receiver. When the recipient has received an updated transaction in a micropayment channel, the recipient may elect to redeem the transaction by disclosing the random data R, which will ultimately pull funds from the sender.

Page 7 | Highlight

```
OP_DEPTH 3 OP_EQUAL OP_IF OP_HASH160 <hash160(R)> OP_EQUALVERIFY OP_0 2
<AlicePubkey1> <BobPubkey1> 2 OP_CHECKMULTISIG OP_ELSE OP_0 2 <AlicePubkey2>
<BobPubkey2> 2 OP_CHECKMULTISIG OP_END
```

Page 8 | Highlight

> The second path is the Timeout Transaction (T), consuming the same txout as the Settlement Transaction (S). The Timeout Transaction has a 2-of-2 output and is consumed by the Timeout Refund (TR) with a refund back to the original party and cancels the clearing process. The extra timelocked step from a Timeout Transaction to a Timeout Refund Transaction is necessary to invalidate the Timeout Transaction. Per our example, only Bob sends the signature for T and TR to Alice.

Page 10 | Highlight

> The counterparty may immediately redeem the funds if the Funding Transaction, as well as the Commitment Transaction (which spends from the Funding Transaction) is included in the blockchain. The party which broadcast the transaction must wait until a pre-agreed time before they may redeem their transaction which is enforced by a 2-of-2 output -- for our examples, we will presume a 40-day locktime. This locktime is a 2-of-2 spend, a Commitment Close Transaction, from the Commitment Transaction. For instance, if Alice broadcasts the Funding Transaction and the Commitment Transaction, Bob will be able to receive a refund for whatever he put into the funding transaction using the Commitment Refund Transaction (CR1a), while Alice must wait 40 days before broadcasting her Commitment Close Transaction (CC1a) which includes an output to Alice. Bob does not have to wait because Alice was the party who broadcast the Commitment

Page 10 | Highlight Continued

> Transaction, so Alice is attesting that Bob should receive at least a certain amount, which is automatically sent to Bob, whereas some timelocked proof is necessary to show Alice’s refund is correct.

Page 12 | Highlight

> output from a Commitment Transaction must use a unique pubkey and address.

Page 12 | Highlight

> Reusing pubkeys will likely result in coin theft.

Page 20 | Highlight

> It is possible construct a cryptographically provable “Delivery Versus Payment” contract, or pay-to-contract[10] , as proof of payment. This proof can be established as knowledge of the input R from hash(R) as payment of a certain value. By embedding the contract between the buyer and seller that knowing R is proof of funds sent, the recipient of funds has no incentive to disclose R unless they have certainty that they will receive payment. When the funds eventually get pulled from the buyer by their counterparty in their micropayment channel, R is disclosed as part of that pull of funds. One can design paper legal documents that specify knowledge or disclosure of R implies fulfillment of payment. The sender can arrange a cryptographically signed contract for knowledge of inputs for hashes as fulfillment of contracts before payment.

Note:

Burada daha ödeme gelmeden, ödeme yapıldı bilgisi alındığı belirtiliyor. Bu bilgi ile süreç aktif olduğunda mal kontrolu yapılabilir.

> When Alice sends payment to Dave through Bob and Carol, she requests from Dave hash(R) to use for this payment. Alice then counts the amount of hops until the recipient and uses that as the HTLC expiry. In this case, she sets the HTLC expiry at 3 days. Bob then creates an HTLC with Carol with an expiry of 2 days, and Carol does the same with Dave with an expiry of 1 day. Dave is now free to disclose R to Carol, and both parties will likely agree to immediate settlement via novation with a replacement Commitment Transaction. This then occurs step-by-step back to Alice. Note that this occurs off-chain, and nothing is broadcast to the blockchain when all parties are cooperative.

Note:

Birden fazla party olunca expire güzel ayarlanmış ama iade nasıl oluyor bakmak lazım.

Page 22 | Highlight

> Decrementing timelocks are used so that all parties along the path know that the disclosure of R will be able to pull funds, since they will at worst be pulling funds after the date whereby they must receive R. If Dave does not produce R within 1 day to Carol, then Carol will be able to close out the HTLC. If Dave broadcasts R after 1 day, then he will not be able to pull funds from Carol. Carol's responsibility to Bob occurs on day 2, so Carol will never be responsible for payment to Dave without an ability to pull funds from Bob provided that she updates her transaction with Dave to the blockchain or via novation.

Note:

Bir önceki notta yazdığım durumdan bahset vs.

Page 23 | Highlight

> It is necessary to use a small payment per HTLC. One should not use an extremely high payment, in case the payment does not fully route to its destination. If the payment does not reach its destination and one of the participants along the path is uncooperative, it is possible that the sender must wait until the expiry before receiving a refund. Since transactions don't hit the blockchain with cooperative channel counterparties, it is recommended to use as small of a payment as possible. A tradeoff exists between locking up transaction fees on each hop versus the desire to use as small transaction fee as possible. Smaller transfers with more intermediaries imply a higher percentage paid as Lightning Network fees to the intermediaries.

Page 28 | Highlight

> As parties must be online and using private keys to sign, there is a possibility that if one's computer is compromised, that coins will be stolen by the counterparty. While there may be methods to mitigate the threat for the sender and the receiver, the intermediary nodes must be online and will likely be processing the transaction automatically.

Note:

Nodes hackbar-nur online.