Inter-Channel Payments

a.k.a., "Impulse"

v2.0.1, January 17, 2015

Objective

Instant, secure transactions, as trustless as possible.

Overview

The solution presented models a wallet as a set of N always-open payment channels ("paychan"). A user makes a payment by gathering together active paychans sufficient to cover the value of the purchase. For example, a 1 BTC purchase may require a paychan of value 0.5 BTC and a paychan of 0.8 BTC, used together.

The payment is made across N payment channels, executed instantly. Those paychans are closed immediately (in parallel with payment). A change paychan may be reopened, in parallel with these operations. Individual payment channels may be thought of as unspent outputs, to be gathered for a transaction.

This will increase the number of transactions on the blockchain per payment (N inputs x 2 TX per paychan) and associated TX fees. Consolidation of UTXOs will be important to minimize fees.

Dramatis Personae

Party A: Payment channel payer (the end user).
Party B: Operates payment channel server (BitPay)
Party C: Optionally contributes funds to payment channel open
Party D+: One or more recipients of payment channel payments

Payment channel basic order of operations

Party A signs first:

1. Party A (somehow) receives pay-to information.
2. Party A constructs/mutates a 2of2 TX with pay-to outputs.

3. Party A signs 2of2 TX, passes sig to Party B.
4. Party B signs 2of2 TX. TX now valid (assuming locktime past etc.)
5. Go to step #1.

Background: Bitcoin protocol rules for transaction “final” state

A bitcoin transaction may only be mined into a block if it is “final.” A transaction is considered final according to the following rules (evaluated in the order shown):

1. Lock time is zero.
2. Lock time is in the past. Lock time may be a Unix time, or block height.
3. All input sequence numbers are 0xffffffff

If the transaction is not final, it will not be relayed across the network. Storage of non-final transactions such as refund transactions is an open issue.

Opening a payment channel

1. Party A requests public key B’K2 from server B.
2. Party A receives B’K2, generates key A’K1.
3. P2SH 2-of-2 multisig address A1 generated from p2sh(sort(A’K1, B’K2))
4. Party A builds, signs, **but does not reveal** funding transaction TF1.
    a.

Inputs	Outputs
One or more signed inputs from party A.	99.99% of funds to A1 (2of2)
	(implicit) Network mining fee

5. Party A builds refund transaction TR2, and transmits to party B.
    a. Locktime set to LT2.
    b.

Inputs	Outputs
TF1 (requires 2of2 sigs)	99.99% of funds to party A. Typically A’K1.
	(implicit) Network mining fee

6. Party B validates TR2, signs, and returns signature. TR2 may now be signed by Party A at any time, resulting in a valid transaction.
7. Party A publishes TF1 to B and blockchain. Funds are now committed, once confirmed. Payment channel is open and funded.

Analysis

- Refund transaction TR2 must be stored somewhere, for use at a later date.
- If A or B disappears before TF1 published, no funds committed, no funds lost.
- If A disappears after TF1 published, the money is in limbo until A reappears. B may wish to be helpful, and publish the refund transaction.
- If B disappears after TF1 published, A will wait until locktime expires, and publish the refund TR2.
- TODO: Some TX malleability issues exist. Resolved with OP_CHECKLOCKTIMEVERIFY.

Making a payment, within a single channel

Preparation: Party A creates a payment transaction TP3, to be used as a template,

- Locktime set to LT1
- Input sequence numbers set to zero
-

Inputs	Outputs
TF1 (requires 2of2 sigs)	99.99% of funds to party A (A'K1).
	(implicit) Network mining fee

1. Party A decides to make one or more payments. Party A obtains, perhaps via BIP70 payment protocol,
   a. List of (output, value) pairs to pay (Party D+)
   b. Some indication of the ability to support instant+secure payments
2. Party A mutates the transaction template, reducing the self-paid funds, and adding/adjusting transaction outputs,
   a. Input sequence number increased
   b.

Inputs	Outputs
TF1 (requires 2of2 sigs)	96.99% of funds to party A (A'K1).

	1% of funds to party D1
	2% of funds to party D2
	(implicit) Network mining fee

3. Party A signs updated TP3, and transmits to party B. If party B already knows the desired outputs, then party A need only transmit their signature and an amount of funds to transfer (versus the full tx).
4. Party B validates the updated contract terms (outputs) and signature. Party B may optionally attach its signature and publish the transaction at any time after LT1.

Analysis

- If party A disappears or ceases to sign transaction updates, party B may publish the last revision of TP3 any time after LT1. "Payment state" is frozen at the time of party A's disappearance.
- If party B disappears, party A may publish the refund transaction any time after LT2. "Payment state" is reset to zero, and goods/services will have been rendered for zero payment.

Closing a payment channel

A payment channel may be closed immediately by mutual agreement.

1. Party A updates TP3, sets input sequence number(s) to 0xffffffff
2. Party A signs TP3, transmits to Party B.
3. Party B signs TP3. The 2of2 is now valid.
4. Party B transmits TP3 to Party A and bitcoin network.

Analysis

- If party A disappears, party B publishes last signed version of TP3 any time after LT1.
- If party B disappears, party A may publish TP3 any time after LT1, or refund TR2 any time after LT2.

Re-opening a payment channel

If the goal is to constantly maintain open payment channels, it is necessary close a payment channel, and open a new payment channel, as rapidly as possible. The following illustrates channel re-open using the same funds.

1. We begin with an open payment channel, whose TP3 reflects many mutations,
    a. Input sequence number equals 300
    b.

Inputs	Outputs
TF1 (requires 2of2 sigs)	89.99% of funds to party A (A’K1).
	10% of funds to party D1
	(implicit) Network mining fee

2. Party A indicates the desire to reopen the payment channel (perhaps spurred by a Party B recommendation), and makes a final update to TP3 (we rechristen it “TF5”), **does not sign**,
    a. Input sequence number equals 0xffffffff
    b.

Inputs	Outputs
TF1 (requires 2of2 sigs)	89.99% of funds to A1 (2of2)
	10% of funds to party D1
	(implicit) Network mining fee

3. Party A signs TF5, **does not reveal** until later.
4. Party A builds and signs refund transaction TR4, and transmits to party B.
    a. Locktime set to LT4.
    b.

Inputs	Outputs
TF5 (requires 2of2 sigs)	99.99% of funds to party A. Typically A’K1.
	(implicit) Network mining fee

5. Party B signs and returns TR4.
6. Party A passes sig to Party B.
7. Party B signs TF5, publishes to Party A and the bitcoin network.

Analysis

- Party A may attempt to use a refund transaction (TRx) from a prior payment channel generation.
- If Party A disappears, payment channel remains at last-signed state (TP3).
- If Party B disappears, likewise, payment channel remains at last-signed state (TP3).

Multi-channel payments

Consider the case of a single 1.0 BTC payment, composed of an aggregation of multiple 0.25 BTC payment channels. Multiple payment channels implies multiple transactions on the blockchain. This cannot be atomic. We can nonetheless usefully align incentives to approach that goal.

We make use of the property that Party A provides signatures first.

1. Party A (somehow) receives pay-to information.
2. Party A mutates and re-signs N transactions for N payment channels.
3. Party B gathers transactions submitted across N payment channels.
4. When required payment amount is reached, Party B may sign all transactions at once.

Analysis

- If Party A disappears or takes no action, payment channels remain in their most-recently-signed state. Party B may sign and publish each after their respective LT1's.
- If Party A takes partial action (makes payments on some payment channels, but not the entire set), Party B signs nothing new, channels remain in their last-signed state.
- If Party B disappears or takes no action, payment channels remain in their last-signed state.
- If Party B takes partial action, there is potential for confusion or abuse. The multi-channel payment will result in a partial payment, at the whim of what Party B chose to sign and publish. It is unlikely Party B would ever do this intentionally due to incentives; it may occur due to software bug or exceptional condition.
- Careful attention must be paid to LT1 and LT2 across all channels in set, to avoid races and mischief where one channel's LT2 approaches another channel's LT1.

Funding payment channel with third party

Party A wishes to open a payment channel with Party B, using funds sent by Party C.

1. Party A requests public key B’K2 from server B.
2. Party A receives B’K2, generates key A’K1.
3. P2SH 2-of-2 multisig address A1 generated from p2sh(sort(A’K1, B’K2))
4. Party C builds funding transaction TF1.
    a.

Inputs	Outputs
One or more signed inputs from party C.	99.99% of funds to A1 (2of2)
	(implicit) Network mining fee

5. Party C signs TF1, **does not reveal** it.
6. Party A and B receive and approve **unsigned** TF1 + signed TF1 hash.
7. Party A creates refund transaction TR2,
    a. Locktime set to LT2.
    b.

Inputs	Outputs
TF1 (requires 2of2 sigs)	99.99% of funds to party A. Typically A’K1.
	(implicit) Network mining fee

8. Party B validates TR2, signs, and returns signature. TR2 may now be signed by Party A at any time, resulting in a valid transaction.
9. Party A indicates to Party C that TF1 may be signed and published to the blockchain, locking in funds.

Third party transaction proofs

We are interested in the use case where a Party D+ would trust Party B’s digital authority, as an assertion that payment has been executed in a manner Party B deems secure.

Consider a mid-stream payment channel transaction TP3.

1. Parties A or B supply Party D+ with a fully signed TP3. This TX is spendable when LT1 expires.
2. Party D+ obtains Party B’s public key associated with the payment channel, via some digital chain of trust (https query?).

3. Party D+ verifies TP3 is signed and spendable (>= LT1), and the parent (TF1) is confirmed in the blockchain.
4. Party D+ verifies TP3 is signed by Party B.
5. Party D+ verifies TP3 contains one or more outputs where they are the payee.

Analysis

- Crucially, a future revision of TP3 may have 100% different outputs from a prior version. A proof for TP3 version X says nothing about version X+1.
- As such, the proof is defined as “Party D+ was at one time provably paid”
- As such, the security of this scheme requires trusting 1-of-2 parties to remain honest, never presenting a tx proof-of-payment then disappearing that payment.

Locktime notes

Careful selection of LT1 and LT2 is required. For example, 6 hours difference can be considered “low & risky.” Ideal is days (72 hours as a starting point guideline, perhaps).

Edge cases

Incentives typically prevent these cases from ever occurring. However, it remains a possibility that some transactions may not be confirmed in the proper order due to software malfunction or malware etc.

Most notably, refund transactions may be mined or payment channels may expire if proper attention is not paid to renewing payment channels and publishing transactions on a timely basis.

Payment channels requires that transactions be stored until they are valid, which leaves open a window for that stored transaction to fail to make it into the blockchain,

- Party A must store refund TR2 until locktime
- Party B must store payment TP3 until locktime
- Party A and Party D+ also have incentives to store TP3 until locktime.

High Level Analysis

Party A

- Party A delivers payment to Party D+ with the approval of Party B.

- If Party A disappears, the payment channels remain in last-signed state. Either party may choose to publish TP3 after LT1, or TR2 after LT2.
- It is presumed that Party B and Party D+ have incentives to publish TP3, and prevent TR2 from being mined by Party A.
- Party A also has an incentive to publish TP3 (versus TR2) in many business cases.

Party B

- Party B validates and signs 2of2 payments already signed by Party A.
- Presumably, there is some out-of-band (BIP70) protocol by which Party A, B and D+ agree on who must be paid how much. Party B must validate Party A's proposed payees.
- If Party B disappears, Party A may choose to publish TP3 (after LT1) or TR2 (after LT2).
- If Party B disappears, Party D+ cannot be guaranteed payment unless they also have a copy of TP3.

Party D

- Provides payment to Party A & B via OOB mechanism (shopping cart -> BIP70?)
- Optionally receives copies of TP3 via OOB mech
- Depends on Party A or B -- probably B -- to provide TP3 copy or publish on blockchain (req. for payment)
- Depends on Party A or B -- probably B -- to provide provable key + chain of trust, to prove TP3 is valid and provides Party D+-spendable outputs.
- Thus, likely relies on Party B to ensure payment.