

# Brief Announcement: Feasibility of Cross-Chain Payment with Success Guarantees

Rob van Glabbeek  
Data61, CSIRO and  
UNSW  
rvg@unsw.edu.au

Vincent Gramoli  
University of Sydney and  
EPFL  
vincent.gramoli@sydney.edu.au

Pierre Tholoniati  
University of Sydney and  
École Polytechnique  
pierre.tholoniati@polytechnique.edu

## ABSTRACT

We consider the problem of cross-chain payment whereby customers of different escrows—implemented by a bank or a blockchain smart contract—successfully transfer digital assets without trusting each other. Prior to this work, cross-chain payment problems did not require this success, or any form of progress. We demonstrate that it is possible to solve this problem when assuming synchrony, in the sense that each message is guaranteed to arrive within a known amount of time, but impossible to solve without assuming synchrony. Yet, we solve a weaker variant of this problem, where success is conditional on the patience of the participants, without assuming synchrony, and in the presence of Byzantine failures. We also discuss the relation with the recently defined cross-chain deals.

## 1 INTRODUCTION AND RELATED WORK

With the advent of various payment protocols comes the problem of interoperability between them. A simple way for users of different protocols to interact is to do a *cross-chain payment* whereby intermediaries can help customer Alice transfer digital assets to Bob even though Alice and Bob own accounts in different banks or blockchains. To implement a payment between customers of different banks, it helps if the two banks have ways to transfer assets to each other, and moreover trust each other. The problem becomes more interesting when this is not the case. Thomas and Schwartz [4] propose two cross-chain payment protocols: (i) the universal protocol requires *synchrony* [1]; (ii) the atomic protocol merely requires *partial synchrony* [1]. Herlihy, Liskov and Shrira [3] represent a cross-chain payment as a deal matrix  $M$  where  $M_{i,j}$  characterises a transfer of some asset from participant  $i$  to participant  $j$ . They offer a timelock protocol that requires synchrony, and a certified blockchain protocol that requires partial synchrony. However, the synchronous solutions of [4] and [3] do not consider clock drift, and for their partially synchronous solutions no success guarantees are established.

In this brief announcement, we formally define the *time-bounded cross-chain payment problem*, and show that, assuming synchrony, there exists an algorithm that solves it. Our solution is the universal protocol of [4], but fine-tuned to work correctly in the presence of clock drift. We also prove that this problem cannot be solved when merely assuming partial synchrony, even if we relax the

problem statement by merely requiring eventual (instead of time-bounded) termination. Moreover, inspired by earlier work on the transaction commit problem [2], we define a weaker variant of our problem that relaxes the liveness guarantee to be solvable with partial synchrony. Contrary to the problem statements in [4] and [3], a protocol where all participants always abort is not permitted by our problem specification.

## 2 MODEL AND DEFINITIONS

We assume  $n$  banks or *escrows*  $e_0, \dots, e_{n-1}$  and  $n+1$  *customers*  $c_0, \dots, c_n$ . These  $2n+1$  processes are called *participants*. An escrow is a specific type of process that can handle values for other parties in a predefined manner. Customer  $c_0$  is Alice and  $c_n$  is Bob. The customers  $c_1, \dots, c_{n-1}$  are intermediaries in the interaction between Alice and Bob; we call them *connectors*, named Chloe $_i$ . Customers  $c_{i-1}$  and  $c_i$  have accounts at escrow  $e_{i-1}$ , and trust this escrow ( $i = 1, \dots, n$ ). We do not assume any other relations of trust.

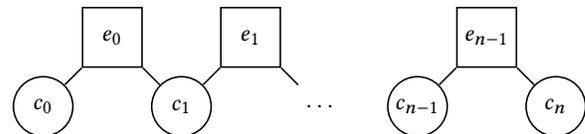


Figure 1: Customers and escrows.

We assume that value can be transferred directly only between customers of the same escrow. Moreover, any transfer between two customers of an escrow can be modelled as two transfers: one from the originating customer to the escrow, and one from the escrow to the receiving customer. Thus, the connections from Figure 1 describe both the relations of trust and the possible transfers.

Two customers may make a deal with an escrow to place value from the first customer “in escrow”, and, after a predefined period, depending on which conditions are met, either complete the transfer to the second customer, or return the value to the first one.

We suppose that the participants have already agreed upon the value they expect to transfer. As Chloe helps out transferring value from Alice to Bob, it is only reasonable that she is paid a small commission. Hence the value transferred from Alice to Chloe might be larger than the value transferred from Chloe to Bob. Additionally, these values may be expressed in different currencies, or they may be objects. Deciding which values to transfer may thus be an interesting problem. However, it is entirely orthogonal to the matter discussed here, and hence we shall not consider it any further.

We consider the classic Byzantine model with authentication.

### 3 FEASIBILITY OF CROSS-CHAIN PAYMENTS

A cross-chain payment protocol prescribes a behaviour for each of the participants in the protocol, the escrows and the customers. Let  $\chi$  be a certificate signed by Bob saying that Alice's obligation to pay him has been met.

**Definition 1.** A protocol is a *time-bounded cross-chain payment protocol* (resp. an *eventually terminating cross-chain payment protocol*) if it satisfies the following properties:

- C **Consistency.** For each participant in the protocol it is possible to abide by the protocol.
- T **Time-bounded (resp. eventual) termination.** Each customer that abides by the protocol, and either makes a payment or issues a certificate, terminates within an a priori known period (resp. terminates eventually), provided her escrows abide by the protocol.
- ES **Escrow security.** Each escrow that abides by the protocol does not lose money.
- CS **Customer security.**
  - CS1 Upon termination, if Alice and her escrow abide by the protocol, Alice has either got her money back or received the certificate  $\chi$ .
  - CS2 Upon termination, if Bob and his escrow abide by the protocol, Bob has either received the money or not issued certificate  $\chi$ .
  - CS3 Upon termination, each connector that abides by the protocol has got her money back, provided her escrows abide by the protocol.
- L **Strong liveness.** If all parties abide by the protocol, Bob is paid eventually.

Requirement C (*consistency* of the protocol) is essential. In the absence of this requirement, any protocol that prescribes an impossible task for each participant would be a correct cross-chain payment protocol (since it trivially meets T, ES, CS and L).

Requirements ES and CS (the *safety* properties) say that if a participant abides by the protocol, nothing really bad can happen to her. These requirements do not assume that any other participant abides by the protocol, and should hold no matter how malicious the other participants turn out to be. The only exception to that is that the safety properties for a customer (CS) are guaranteed only when the escrow(s) of this customer abide by the protocol.

Property L, saying that the protocol serves its intended purpose, is the only one that is contingent on *all* parties abiding by the protocol. The proofs of the following results are presented in [5].

**THEOREM 1.** *If communications and computations are synchronous, there exists a time-bounded cross-chain payment protocol.*

**THEOREM 2.** *If communications are partially synchronous, there is no eventually terminating cross-chain payment protocol.*

In view of this impossibility result we propose to weaken the liveness guarantee as indicated in Def. 2. In [5] we present a protocol in which each customer can, at any moment of their choice, lose patience and abort the transaction, without a risk of losing value. In case none of them exercises this option nor fails, a successful outcome is guaranteed. This solution involves an external *transaction manager*, that can issue an *abort* or *commit* certificate. Properties

CC and CS2 together guarantee that the commit certificate can be used by Alice as a proof that Bob has been paid. The transaction manager could be a single external party trusted by all, or a smart contract running on a permissionless blockchain shared by every customer. It can also be a collection of notaries appointed by the participants in the protocol, of which less than one-third is assumed to be unreliable. They would run a consensus algorithm for partial synchrony such as the one from Dwork, Lynch & Stockmeyer [1].

**Definition 2.** A protocol is a *cross-chain payment protocol with weak liveness guarantees* if it satisfies properties C, ES and CS3, as well as:

- CC **Certificate consistency.** An abort and a commit certificate can never be issued both.
- T **Termination.** Each customer that abides by the protocol terminates eventually, provided her escrows abide by the protocol.
- CS **Customer security.**
  - CS1 Upon termination, if Alice and her escrow abide by the protocol, Alice has either got her money back or received the *commit certificate*  $\chi_c$ .
  - CS2 Upon termination, if Bob and his escrow abide by the protocol, Bob has either received the money or the *abort certificate*  $\chi_a$ .
- L **Weak liveness.** If all parties abide by the protocol and if the customers wait sufficiently long before and after sending money, then Bob is eventually paid.

**THEOREM 3.** *There exists a cross-chain payment protocol with weak liveness guarantees.*

### 4 A TIME-BOUNDED PROTOCOL

In Figure 2 we present the protocol from Thm. 1 formalised as an Asynchronous Network of Timed Automata (ANTA), a specification formalism introduced in [5]. There is one automaton for each participant in the protocol, that is, for each escrow  $e_i$  ( $i = 0, \dots, n-1$ ) and each customer  $c_i$  ( $i = 0, \dots, n$ ). Each automaton is equipped with a unique identifier, in this case  $e_i$  and  $c_i$ . It has a finite number of *states*, depicted as circles, and *transitions* between them.

Each automaton keeps an internal clock, whose value, a real number, is stored in the variable *now*. In case a transition occurs that is labelled by an assignment  $x := \text{now}$ , the variable  $x$  will remember the point in time when the transition took place.

An automaton spends a bounded amount of time calculating in each grey (*output*) state, and leaves it by performing the action  $s(id, m)$  of sending message  $m$  to participant  $id$ . We consider three kinds of messages: (i) certificate  $\chi$ , signed by Bob, (ii) the value \$ that is transmitted from one participant to another, and (iii) promises made by escrow  $e_i$  to its customers  $c_i$  and  $c_{i+1}$ , respectively:

$G(d) :=$  "I guarantee that if I receive \$ from you at my local time  $w$ , then I will send you either \$ or  $\chi$  by my local time  $w + d$ ."

$P(a) :=$  "I promise that if I receive  $\chi$  from you at my time  $v$ , with  $v < \text{now} + a$ , then I will send you \$ by my local time  $v + \epsilon$ ."

When an automaton is in a white (*input*) state, it stays there (possibly forever) until one of its outgoing transitions becomes *enabled*; in that case that transition will be taken immediately. The

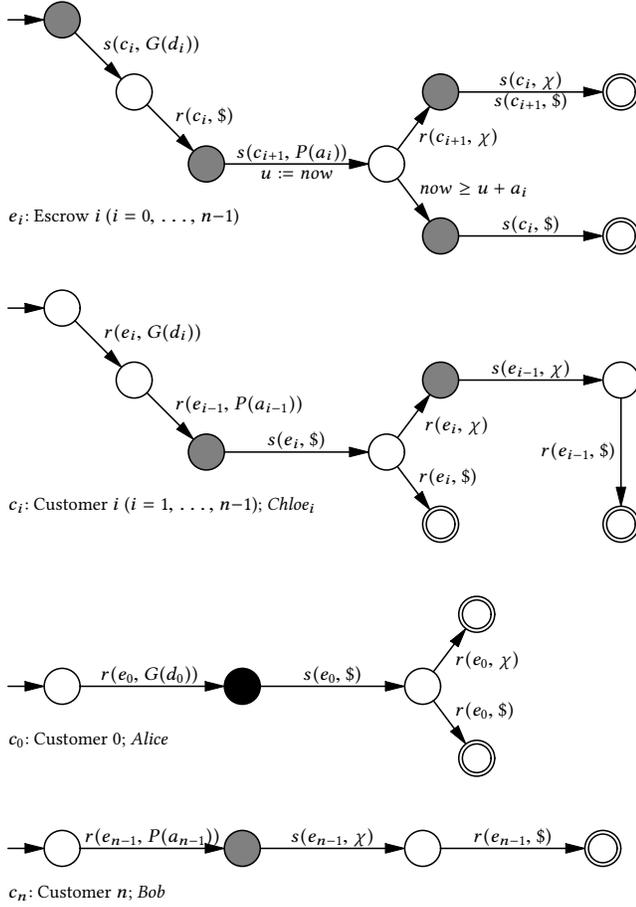


Figure 2: Automata representing escrows and customers

time-out transition  $now \geq u + a_i$  is enabled when this formula evaluates to  $\text{true}$ . An input transition  $r(id, m)$  is triggered by the receipt of message  $m$  from the automaton  $id$  in the network.

The automata of Figure 2 can be informally described as follows: An escrow  $e_i$  first sends promise  $G(d_i)$  to its (upstream) customer  $c_i$ . Here “upstream” refers to the flow of money. The precise values of  $d_i$  are calculated in [5]; here they are simply parameters in the design of the protocol. Then it awaits receipt of the money/value from customer  $c_i$ . If the money does arrive, the escrow issues promise  $P(a_i)$  to its downstream customer  $c_{i+1}$ . It remembers the time this promise was issued as  $u$ . Then it awaits receipt of the certificate  $\chi$  from customer  $c_{i+1}$ . If  $\chi$  does not arrive by time  $u + a_i$ , a time-out occurs, and the escrow refunds the money to customer  $c_i$ . If it does arrive in time, the escrow reacts by forwarding the certificate to customer  $c_i$ , and forwarding the money to customer  $c_{i+1}$ .

A connector  $Chloe_i$  starts by awaiting promises  $G(d_i)$  from her downstream escrow  $e_i$ , and  $P(a_{i-1})$  from her upstream escrow  $e_{i-1}$ . Then she proceeds by sending the money to escrow  $e_i$ . After sending the money,  $Chloe_i$  waits for escrow  $e_i$  to send her either the certificate  $\chi$  or the money back. In the latter case, her work is done; in the former, she forwards the certificate to escrow  $e_{i-1}$  and awaits for the money to be sent by escrow  $e_{i-1}$ . The automata for Alice and Bob are both simplifications of the one for  $Chloe_i$ .

## 5 RELATION WITH CROSS-CHAIN DEALS

In Herlihy, Liskov and Shrira [3], a *cross-chain deal* is given by a matrix  $M$  where  $M_{i,j}$  is listing an asset to be transferred from party  $i$  to party  $j$ . It can also be represented as a directed graph, where each vertex represents a party, and each arc a transfer; there is an arc from  $i$  to  $j$  labelled  $v$  iff  $M_{i,j} = v \neq 0$ . They present two protocols for implementing such a deal, while aiming to ensure:

- **Safety.** For every protocol execution, every compliant party ends up with an acceptable payoff.
- **Termination.**<sup>1</sup> No asset belonging to a compliant party is escrowed forever.
- **Strong liveness.** If all parties are compliant and willing to accept their proposed payoffs, then all transfers happen.

Here a payoff is *acceptable* to a party  $i$  in the deal if party  $i$  either receives all assets  $M_{j,i}$  while parting with all assets  $M_{i,j}$ , or if party  $i$  loses nothing at all; moreover, any outcome where she loses less and/or gains more than an acceptable outcome is also acceptable.

Each entry  $M_{i,j}$  contains a type of asset and a magnitude—for instance “5 bitcoins”. For each type of asset a separate blockchain is assumed that acts as escrow. The programming of these blockchains is assumed to be open source, so that all parties can convince themselves that all escrows abide by the protocol. With this in mind, their **Termination** requirement corresponds with ours, while **Safety** is the counterpart of our **Customer security**. Our requirement of **Escrow security** is left implicit in [3]; since blockchains do not possess any assets to start with, they surely cannot lose them. Finally, their **Strong liveness** property is the counterpart of ours.

Herlihy, Liskov and Shrira [3] offer a timelock commit protocol that requires synchrony, and assures all three of the above correctness properties. They also offer a certified blockchain commit protocol that requires partial synchrony and a certified blockchain, and ensures **Safety** and **Termination**; no protocol can offer **Strong liveness** in a partially synchronous environment. For both protocols the correctness is proven for so-called *well-formed* cross-chain deals: those whose associated directed graph is strongly connected.

In [5] we show that the cross-chain payment cannot be seen as a special kind of cross-chain deal, nor vice versa.

*Acknowledgements.* This research is supported by ARC Discovery Project 180104030: “Taipan: A Blockchain with Democratic Consensus and Validated Contracts” and ARC Future Fellowship 180100496: “The Red Belly Blockchain: A Scalable Blockchain for Internet of Things”.

## REFERENCES

- [1] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* 35, 2 (1988), 288–323. <https://doi.org/10.1145/42282.42283>
- [2] Vassos Hadzilacos. 1990. On the Relationship Between the Atomic Commitment and Consensus Problems. In *Proc. Asilomar Workshop on Fault-Tolerant Distributed Computing (LNCS)*. Springer, 201–208. <https://doi.org/10.1007/BFb0042336>
- [3] Maurice Herlihy, Barbara Liskov, and Liuba Shrira. 2019. Cross-Chain Deals and Adversarial Commerce. *Proceedings of the VLDB Endowment* 13, 2 (Oct. 2019), 100–113. arXiv:1905.09743v5
- [4] Stefan Thomas and Evan Schwartz. 2015. A Protocol for Interledger Payments. Available at <https://interledger.org/interledger.pdf>.
- [5] Rob van Glabbeek, Vincent Gramoli, and Pierre Tholoniat. 2019. Cross-Chain Payment Protocols with Success Guarantees. <http://arxiv.org/abs/1912.04513>

<sup>1</sup>In [3], this property is called “weak liveness”. We rename it here, to avoid confusion with our own weak liveness property, which is of a very different nature.