First Year Report

Orfeas Stefanos Thyfronitis Litos

University of Edinburgh
o.thyfronitis@ed.ac.uk

Primary Supervisor: Prof. Aggelos Kiayias
Secondary Supervisor: Prof. Kousha Etessami

Abstract. This research proposal concerns the study of economic trust in the decentralised, asynchronous setting and its applications on online commerce and blockchains. In particular, the aim of this project is twofold. The first goal is to understand whether it is possible to achieve robust online commerce with minimal risk of fraud without trusted third parties and if so at what cost. The second goal is to provide a general model of off-chain payments and explain how a relaxation of the trustless element of current solutions can facilitate the creation of a scalable blockchain. Tools from the fields of Cryptography and Game Theory will be used to achieve the aforementioned targets.

1 Research Topic

The advent of bitcoin [1] has promised to revolutionise the way online commerce takes place[2], only to meet multiple hurdles in the way[3,4]. Today there exists an entire ecosystem of cryptocurrencies, e.g. [5,6,7], each with its own benefits and drawbacks. Nevertheless, there are several fundamental questions that still remain unanswered. Some of the prominent issues are whether cryptocurrencies can become the global unit of account, whether it is possible to regulate and fairly distribute them, what are the exact benefits they provide when compared to fiat currencies, in which respects the latter outperform them and if it is possible to completely replace fiat currencies with decentralised cryptocurrencies for which the traditional measures of law enforcement are less applicable due to the global nature of blockchains.

This research proposal is crucially motivated by the following observation: The single most advertised advantage of permissionless blockchains is the lack of need for trust between the interacting individuals *and* for a mediating trusted third party [1]. Unfortunately, this benefit is limited to a certain type of digital assets that admit a cryptographically secure

proof of ownership and can be provably transferred to another party. Additionally, the overhead for setting such a system in motion is too big for the entire economy to be realised in this way.

1.1 Decentralised E-Commerce

Going in more detail, blockchains provide us with novel capabilities. It is possible to immutably and truthfully record the fact that Alice paid Bob some cryptographic coins on a blockchain; it is possible for Bob to provably and irrefutably give her a digital asset in return (e.g. an ICO [8], or a sword in a future blockchain–backed game) and record the fact on a blockchain as well; additionally, it is possible to make the exchange atomic so that no trust between the two parties is needed. Unfortunately, all the above are impossible to accomplish if the asset is physical. Alice may pay first and then Bob can refuse to give the asset; or Bob may indeed hand in the asset, but Alice may then claim that it was not as advertised and demand her money back. The possibility of fraud is exacerbated if the exhange takes place online between parties who do not know each other personally. Crucially, an external observer cannot always resolutely decide which of the parties is right. Even if she can, it seems difficult for her to prove that she is not colluding with either player and that she is completely fair and impartial, since her background, objectives and affiliations are seldom entirely transparent to both parties.

For comparison, if the payment is done in fiat currency, then there exists a mediator (usually a bank, often a court) who is charged with deciding what is the fair way to settle the dispute. Additionally, it is in the mediator's direct benefit to strive for the objectively fairest solution. Unfortunately, from a cryptographic point of view, such an assumption is too strong: it is entirely possible that the bank employee or the judge who has the final say is a friend of the fraudster. Nevertheless, it seems that currently the incentives are much better aligned when relying on the traditional legal system. As a result, transacting in fiat currency is, for practical and everyday purchases of physical goods and especially in the online setting, safer and more robust than using cryptocurrencies.

The cryptographic achievements of blockchains are currently realised only in a closed system, where everything can be proven mathematically. Nevertheless, the economy is not a closed system: New resources can be found, rules, conventions and legislations can be changed, groundbreaking inventions change the landscape of the market rapidly and the desires of consumers evolve. In order for decentralised blockchains to substitute the traditional fiat currencies, a mechanism for decentralised e-commerce with

guarantees of minimal trust is needed. We propose to attempt the creation of such a mechanism or give plausible arguments why such a mechanism is unattainable. In case we manage to create such a system, we should ideally be able to construct the backend of a decentralised ecommerce platform, similar in usage to existing platforms such as OpenBazaar [9].

In essence, such a mechanism would provide a decentralised reputation system. In current online marketplaces (e.g. ebay.com) there exists a central authority that aggregates ratings and reviews from past interactions between users and serves them to participants of future transactions with the aim of minimising fraud. We aspire to replace this central authority with a distributed protocol or prove that such an endeavour is impossible.

1.2 Off–chain payment channels

One additional impediment to the more widespread use of decentralised blockchains is the issue of scalability. Decentralised consensus protocols based on Proof-of-Work [10] or Proof-of-Stake [7] make the assumption that the majority of the hashing power or of the stake respectively is controlled by honest parties, each of which processes every single transaction. To oversimplify, one can think that such protocols function only as long as every member of a crucial portion of their users processes all transactions. A great duplication of effort takes place in order to avoid trusting a particular set of users. It seems intuitively obvious that, no matter how much the parameters are tuned, such systems can only handle a limited amount of transactions per second and thus cannot compete traditional centralised solutions such as VISA in their current state [11].

One solution to this issue is to ensure that most payments happen off–chain. A variety of mechanisms (discussed in more detail in Section 5) enable cryptocurrency owners to temporarily lock some funds in specially crafted channels with one on–chain transaction and subsequently perform any number of transactions between them with these funds without touching the blockchain. One more on–chain transaction unlocks the funds, ensuring each participant receives exactly the funds they owned in the latest state of the channel. Surprisingly, no trust is needed between the parties. The established term for this type of mechanism is “payment channels”.

As of today there exists no definitive systematisation of what payment channels can achieve and at what cost. Each proposal provides different benefits and incurs different costs; the language and formalism of each proposal is largely incompatible with the rest. Most of the proposals lack

formal security definitions and proofs. More importantly, it is unclear what are the exact inherent tradeoffs and whether an ideal balance exists. One of the targets of this research is to provide a suitable conceptual framework that encompasses the whole range of possibilities and limitations of such channels.

Two related issues that have not yet been tackled are the following. First, currently all proposals offer trustless operation as a feature. Nevertheless, it seems plausible that a relaxation of this constraint may indeed allow for more efficient, faster and lighter payment channels. This is a new direction that this research will attempt to explore, ideally incorporating it in the general framework proposed above.

Second, current blockchains do not focus on how to facilitate payment channels, instead attempt to obviate their need. An interesting topic would be to provide a treatment on what properties should a blockchain have to aid the creation of lightweight payment channels.

2 Toolset and Strategy

2.1 Simulation–based Security

The two distinct research directions will both leverage tools and methods used in Cryptoghraphy and Game Theory. In order to formally model the interactions between marketplace participants, we will employ simulation–based security [12] and more specifically the Universally Composable Security framework [13]. The general strategy of simulation–based security is as follows: First we give some security definitions that formalise the desired properties our system should have. Subsequently we give an ideal–world functionality that satisfies these properties. This functionality acts as a centralised trusted party that executes the desired operation on behalf of the participants. Afterwards we describe a protocol that hopefully realises this functionality. Finally we prove that the proposed protocol indeed realises the functionality by showing that an external observer cannot distinguish an execution of the functionality from an execution of the actual protocol.

To better capture the threat model, i.e the real-world challenge the cryptographic protocol faces), the concept of the Adversary is introduced. This party is in control of all messages that the honest participants exchange and can change, delay or completely block them at will. An additional entity is the Environment, which orchestrates the protocol or the functionality and interacts with the Adversary in arbitrary ways. A

complete security proof of the fact that a given protocol realises a particular functionality states that for every efficient Adversary (polynomial algorithm) there exists a Simulator (polynomial algorithm) such that the distribution of the execution of the functionality as viewed from the perspective of any Environment is statistically close to the respective distribution of the execution of the protocol.

The Universally Composable Security framework is a particular way of setting up and proving the security of a protocol. Achieving simulation of a protocol in the Universal Composition setting ensures that the protocol can be executed concurrently with other protocols that are secure in the Universal Composition setting. Consequently, managing to prove the security of our protocol in this setting ensures that it can coexist with other secure protocols in the same processing unit without suffering diminished security itself or damaging the security of the rest of the aforementioned protocols. Such an achievement would ensure that our protocol is more readily implementable whilst maintaining a high standard of security and would obviate the need of further security proofs for the case when it is executed in an interleaved fashion with other secure protocols.

2.2 Need for Game Theoretic Analysis

Before the advent of blockchains, most cryptographic protocols solved problems where there were clear-cut lines between the aims of honest parties and the machinations of the Adversary. For example, in the case of public key encryption [14], the honest parties should be able to exchange messages encrypted in a fashion that only those who know the private key can decrypt, even though the adversary has the ability to employ any method conceivable (within some computational limits) to read the original contents of the message. Any protocol that claims to realise secure encryption should satisfy (the formal version of) this requirement. It is intuitively obvious that it is in the benefit of the honest parties to follow such a protocol.

On the other hand, blockchains made the state of affairs much more complex. Indeed, bitcoin was a concrete solution to an not yet defined problem; several years passed before a formal definition of the problem blockchains solve was formulated [15]. Nevertheless, this definition does not necessarily coincide with the desires of the participants to the protocol. The definitions demand that a blockchain realises an immutable ledger that is constantly extended, however the goals of the participants span a great variety, such as maximising their share of coins or preventing certain transactions from entering the blockchain. In this case, assuming

that there exist only honest parties that execute the protocol exactly as described and an Adversary that employs every conceivable technique in order to break the security definitions is an oversimplification. It ignores slight but crucial deviations from the protocol (e.g. [16]) that may provide advantages not considered in the security definitions. Therefore the assumption of an honest majority can be contested.

When considering decentralised e-commerce, the problem is even more exacerbated. Formulating a simple definition for the characteristics of a healthy marketplace is highly non-trivial. The individual desires of parties should employ a central role in the analysis and not be necessarily restricted to a clear-cut protocol. Assuming that a portion of honest participants exists may be wrong, especially if other participants follow counter-strategies that make honest behaviour unprofitable or if honest players have incentive to profitably deviate from their strategy. Conversely, permitting arbitrary actions to the Adversary could complicate the analysis more than necessary, since it may be reasonable to assume that certain "unreasonable" strategies will never be followed by the Adversary.

2.3 Game Theoretic tools

This is where rational analysis and mechanism design [17] come into play. Game Theory [18] provides a series of tools and concepts that greatly support the study of multi-agent rational systems. The concept of Nash Equilibrium [19], fundamental to the study of Game Theory, will be used throughout this research. Indeed, our aim will be to describe a mechanism that resembles the real e-commerce setting as closely as possible — while abstracting away irrelevant complications — and find strategies that constitute Nash Equilibria.

General Bayesian games of incomplete information constitute another related powerful tool that will help us model the interactions between buyers and sellers. More specifically, the concepts of type, beliefs and knowledge will help us define the most beneficial action for each agent, within some computational constraints.

Fortunately, we can leverage an arsenal of cryptographic tools to help us rule out some categories of undesired behaviours and thus design a system that achieves the goals of modern e-commerce whilst minimising the trust towards third parties. The guarantees given will not be necessarily cryptographic, but rather rational; that is, participants will refrain from acting badly not because they do not have the ability, but rather because it will be detrimental to their social standing, access to goods

and services or other desirable attributes. We hope that this combination of Game Theoretic analysis and Cryptography will prove powerful enough to formalise provably secure and strategy-proof systems for use by decentralised marketplaces.

3 Progress to date

3.1 Modelling Decentralised E-Commerce

The main aim of this research is to construct a model that closely follows the dynamics of a marketplace. Following the Universal Composability framework, we define n interactive Turing Machines (ITMs) that represent the players that take part in the marketplace game. These machines execute Π_{SAT}, i.e. the satisfaction protocol. These ITMs comprise the set $\mathcal{P}$. Furthermore, we define an ITM named Environment and represented by $\mathcal{E}$.

The latter ITM is the first to start functioning. Initially, its responsibility is to allocate a utility function to each player, drawn from a distribution on legal utility functions that is common knowledge [20] to the players. We will elaborate on the nature of the utility functions later on. Additionally, $\mathcal{E}$ provides players with an initial "endowment" of assets and money. Contrary to the utility functions, these are not sent only to the players, but additionally to the $\mathcal{G}_{\text{Assets}}$ and $\mathcal{G}_{\text{Ledger}}$ global functionalities respectively. These two functionalities exist with the sole purpose of keeping track of physical and digital assets respectively. We will later provide further explanation as to why we introduce these global functionalities.

The current state of this part of the research can be found in `https://github.com/OrfeasLitos/UC-Trust/blob/master/what_is_trust.pdf`. We will now go into further detail regarding the construction and its parts.

Overall game description

After providing the utility functions and the endowments, the main part of the game begins. $\mathcal{E}$ can send a message to Alice $\in \mathcal{P}$ that contains a desire d to be satisfied. To satisfy this desire, Alice has to buy an asset that satisfies d from some other player. The first and most important task Alice faces is who to trade with. For that purpose, she consults with the functionality $\mathcal{F}_{\text{Trust}}$, which hopefully responds with a trustworthy vendor, say Bob. Then Alice asks Bob whether he is willing to satisfy her desire with a satisfying asset and, if so, at what price. Given that Bob offers a reasonable price, Alice instructs the functionality $\mathcal{F}_{\text{Trade}}$ to pay Bob the

designated price in exchange for the satisfying asset. The functionality pays Bob and crucially asks him if he wants to complete the exchange honestly or cheat. It then completes the trade as instructed by Bob by interacting with the two global functionalities appropriately and reports back to Alice the result. Finally Alice reports the result to $\mathcal{E}$ and $\mathcal{F}_{\text{Trust}}$. The latter updates the trust properties connected to Bob depending on Alice's report. Just like Bob could cheat and not deliver the satisfying asset, Alice has the ability to misreport her experience with Bob.

The fact that the buyer pays for the asset before the seller delivers it is a design choice; we could have likewise chosen that the seller should first deliver and then expect payment. We chose this ordering however to closer mimic the sequence of events in most real-world exchanges.

$\mathcal{G}_{\text{Assets}}$

This functionality keeps track of which player owns which physical assets. Only $\mathcal{E}$ can add assets to it. Alice can remove assets she owns. $\mathcal{F}_{Trade}$ can transfer an asset that belongs to Alice to another player on behalf of Alice. Both Alice and $\mathcal{E}$ (and no one else) can query the quantity of a certain asset that Alice owns.

Observe that the last rule makes it impossible for players to definitively tell whether another player owns a particular asset by just querying the functionality. This is useful, as it reflects the real-world situation where an arbiter cannot decide who of two disagreeing players is entitled to a physical asset without additional information by third parties.

$\mathcal{G}_{\text{Ledger}}$

This functionality keeps track of the digital assets, i.e. coins. For the purposes of this work, all interactions are similar to those of $\mathcal{F}_{\text{Assets}}$, except for the fact that Alice can prove ownership of some funds to an arbiter. This reflects the fact that, through the use of blockchains, Alice can prove that she owns a particular key that has the ability to spend a particular amount of coins.

This difference is fundamental. Blockchains allow for consensus on who owns which coins because the whole history is distributed and immutable. This is not the case for physical assets. Conflicting accounts of past events can happen in such a way that no third party can tell which version is true; crucial events may be hidden (on purpose or because nobody observed them) and consensus may be unreachable. Furthermore, it seems impossible to record the state of the entire physical world on

the blockchain without trusting a third party. It is our intent to capture this tension between digital and physical assets and thus we employ two separate global functionalities.

$\mathcal{F}_{\text{Trade}}$

This functionality attempts to abstract away the details of how a trade takes place once Alice has decided to buy from Bob. It checks that Alice indeed has the necessary coins and pays Bob. Then it asks Bob whether to proceed with transferring the agreed asset or cheat and acts accordingly.

This functionality does the "plumbing" of the system and is not central to the design. We have chosen to keep this functionality simple, so it does not make any important decisions; it simply executes a trade according to the wishes of the involved parties. We have provided a protocol Π_{Trade} that realises the functionality. We have not yet proved that the latter indeed realises the former, but we will certainly do so in the future.

$\mathcal{F}_{\text{Trust}}$

This last functionality is probably the most crucial — it is certainly the main missing component in order to model a functional, secure marketplace. $\mathcal{F}_{\text{Trust}}$ is responsible for choosing a suitable vendor to satisfy a particular desire. More precisely, it receives a desire and a list of possible vendors from Alice and returns a single vendor that is the most trustworthy for Alice to buy from.

There are several important things to note here. First of all, in contrast to a real trusted third party (e.g. ebay), this functionality cannot be "held accountable" if it recommends a bad vendor. No legal action can be taken against it. Contrary to a real-world enterprise, it does not have any incentives, it just follows its specification; the mechanism it uses to choose suitable vendors is common knowledge to all players, thus security through obscurity (as employed by real-world enterprises that closely guard their reputation system mechanism) is out of the question. An advantage that this functionality has is that we can choose to provide it with oracle access to the utilities of all players; such an oracle can help us describe the best conceivable trust functionality, since it can use the knowledge of the vendors' preferences and aims to always choose the best vendor for a specific trade. This oracle is however a luxury that is necessarily absent from any protocol that will attempt to realise $\mathcal{F}_{\text{Trust}}$. Last but not least, in the real world players have the option to employ a variety of fundamentally different mechanisms in order to choose a suitable

vendor, with some being word of mouth, search engines, rating websites and the reputation systems of online marketplaces. $\mathcal{F}_{\text{Trust}}$ should ideally encompass the possibilities of all these methods and more. Only then will we be able to safely assume that all coalitions of players will have incentives to keep using $\mathcal{F}_{\text{Trust}}$, instead of employing a different method.

As of now, we have built an $\mathcal{F}_{\text{Trust}}$ that, having oracle access to the utilities of all players, chooses a vendor in such a way that:

- The vendor finds it beneficiary not to cheat.
- Amongst all possible vendors that conform to the previous constraint, the additional utility the buyer will enjoy after the trade is maximised.

Utility functions properties

There is no game theoretic analysis without utility functions. In our case, each player can act upon receiving a message only if she has been given a utility function by $\mathcal{E}$. Any such utility function is defined for every moment in time, every possible asset ownership and every amount of coins owned:

$$u_{\text{Alice}} : \text{Time} \times \text{Assets} \times \text{Money} \to \mathbb{R} \ .$$

Some properties that we may assume for the utility functions if needed are quasi-concavity, continuity and non-satiation [21], at least regarding a restriction of the utility function for a particular moment in time.

Π_{SAT}

An overview of this protocol has already been provided. Here we go into further detail. Let Alice be a player that executes an instance of the protocol. First of all, we clarify that Alice will not do any operation upon receiving any messages if she has not already received her utility function from $\mathcal{E}$. Apart from message containing the utility function, there are three more types of messages that Π_{SAT} can receive.

Upon receiving `(satisfy, d, L)` from $\mathcal{E}$, Alice asks $\mathcal{F}_{\text{Trust}}$ to choose the best vendor from the list `L`. She then asks the chosen vendor if he can satisfy her desire, who responds with a suitable asset and a price if he wants to trade. If the utility function's value would increase after this trade, Alice instructs $\mathcal{F}_{\text{Trade}}$ to proceed with the agreed transaction. She then waits for the result of the trade, which can be either `satisfied` or `cheated`. In either case, Alice informs both $\mathcal{F}_{\text{Trust}}$ and $\mathcal{E}$ with a relevant message and goes idle.

If the vendor does not want to trade or if the proposed trade is not favourable for Alice, she asks $\mathcal{F}_{\text{Trust}}$ for a new vendor. This process is repeated until a suitable vendor is found or until all vendors from the list `L` are exhausted. In the latter case, Alice informs $\mathcal{E}$ that the desire remained `unsatisfied`.

As a vendor, Alice may be contacted by Bob asking her if she can satisfy a desire of his. If her utility function's value would increase after an honest trade (or after cheating Bob) she provides Bob with a specific offer. The last message Alice reacts upon is a message from $\mathcal{F}_{\text{Trade}}$, asking her whether to cheat on an already initiated trade or not. She again decides based on the amount of money she has been paid, the value of the asset to trade in and the cost to her reputation. Note that it is as of yet unclear how Alice measures the cost that specific choices will have upon her reputation. It seems to be inextricably tied to the specific way Π_{Trust} is specified. We should also clarify that every player can act both as a buyer and a vendor interchangeably.

The ideal market functionality: $\mathcal{F}_{\text{SAT}}$

Following the simulation-based security paradigm and in order to help us conceptualise how an ideal market should function, we have additionally designed a satisfaction functionality $\mathcal{F}_{\text{SAT}}$. In the world where this functionality exists, the protocol players are dummy, i.e. they simply forward all messages they receive from $\mathcal{E}$ to $\mathcal{F}_{\text{SAT}}$ and vice versa. Therefore, $\mathcal{F}_{\text{SAT}}$ knows the utilities of all players and can complete optimal trades without ever having to show trust to other actors.

More precisely, the current version of $\mathcal{F}_{\text{SAT}}$ behaves as follows: upon receiving a desire from Alice, it creates a list of vendors, assets and prices that constitute possible trades, each of which would be beneficial both to Alice and the implicated vendor. It then sends this list to the Adversary $\mathcal{A}$; the Adversary sends back the trade that will take place and $\mathcal{F}_{\text{SAT}}$ executes it. If the Adversary does not respond with a valid message, then $\mathcal{F}_{\text{SAT}}$ chooses the best trade for Alice and executes that. Note that this is the first time we have implicated the Adversary in any way.

Putting it all together

Our current approach is as follows: In the real world, each player may follow any protocol. At an arbitrary moment in time $\mathcal{E}$ stops the game and evaluates the utility functions of all the players. This way we do not restrict the players to optimization protocols. We will attempt to find

specific protocols that achieve desirable properties. More specifically, our main concern is to find a protocol that realises $\mathcal{F}_{\mathcal{SAT}}$. It may make sense to leverage previous work [22] to argue on the extent to which such a protocol constitutes a Nash Equilibrium, but the correct approach in this respect is not yet clear. We will further discuss future work in Section 4.

Results

The following result is a sanity check that a bad $\mathcal{F}_{\text{Trust}}$ does not allow for a realisation of $\mathcal{F}_{\text{SAT}}$ by any Π_{SAT}. Let $\mathcal{F}'_{\text{Trust}}$ be an alternative trust functionality that simply chooses a vendor at random. We have proved that there exists an Environment that can provide suitable utility functions and initiate a trade such that it always distinguishes whether it is interacting with the real-world protocol Π_{SAT} or with the ideal-world functionality $\mathcal{F}_{\text{SAT}}$. This proof confirmed our intuition that the trust functionality is of crucial importance if we wish to realise $\mathcal{F}_{\text{SAT}}$.

Security Definitions

Apart from defining ideal functionalities and real-world protocols, it is useful to give some security definitions. Until now we have given only one definition.

Definition 1 (Security against Cheating). *We say that a protocol is secure against cheating if* $\forall$ PPT $\mathcal{E}, \Pr[\mathcal{E}$ receiving $(\texttt{cheated}, \_, \_)]$ *is negligible.*

Here a function f is negligible if $\forall c \in \mathbb{R}, \exists n_0 \in \mathbb{N} : \forall n \geq n_0, f(n) \leq \frac{1}{n^c}$.

Since $\mathcal{F}_{\text{SAT}}$ never cheats, it obviously is secure against cheating. More definitions will be needed however, because it is very simple to create a corresponding Π_{SAT} protocol that is secure against cheating: simply never trade anything. Also this security definition does not protect against maliciously sent $(\texttt{cheated}, \_, \_)$ messages.

3.2 Research on Payment Channels

In this part of the research we have mainly focused on understanding the general framework within which the various realisations of payment channels can be classified, compared and evaluated. To achieve this, we reviewed the relevant literature, which has rapidly grown to be quite

extensive and contains concrete systems [23,24,25,26,27,28,29,30], applications on top of channels [31], ideas towards inter-blockchain transactions [32] and attempts for the formalisation of payment channels [33]. Subsection 5.2 provides a summary of the field.

We will give here an informal description of the simplest use case of a payment channel. In this scenario, Alice and Bob are about to enter a financial relationship that will last for some time and during which Bob will have to pay Alice small sums very often. Performing each transaction on-chain would be unwieldy both because of the per-transaction fees lost to miners/minters and because of the need for Alice to wait for confirmation after each transaction.

The solution is for Bob to initially publish on-chain a transaction with e.g. 10,000 coins that can be spent only by an "update" transaction u_0 that would deposit the whole sum to an address controlled by Bob. He privately sends u_0 to Alice. Note that Alice cannot make u_0 herself, because it is signed by Bob. If Bob wants to pay Alice e.g 1 coin, he creates and privately sends her a new update transaction u_1 which, if published, would give 9,999 coins to Bob and 1 coin to Alice. He also sends her a "revocation" transaction r_0 that would invalidate u_0 if it was to be published. Therefore, Alice is sure that Bob cannot take this coin back as long as she watches the blockchain. The same idea can be applied more times, until all 10,000 coins are handed over to Alice or either party decides to publish the latest u_i and unlock both parties' funds.

This simplified description has several shortcomings, but it provides intuition on how many transactions can happen off-chain and still require no trust between the parties; both sides can unilaterally publish the latest update and get on-chain exactly the funds they are entitled to.

As we will see in more detail in the literature review, there are much more capabilities to payment channels. Bidirectional channels, transfer of funds from one channel to another, privacy, implementation of arbitrary smart contracts and even exhange between different blockchains can happen off-chain. It is indeed very interesting to see how far these ideas can go and if they provide the much-needed solution to the blockchain scalability problem.

The inquiry into the nature of payment channels amounted to the following (possibly incomplete) list of characteristics any payment channel has:

- Number of participants in the channel
- Directionality of the channel
- Relevant on-chain transactions (that keep funds locked in the channel)

- Available actions, e.g. open, update to new state, execute rules on chain, close
- Who needs to sign for each action, who is notified, how many rounds of communication does this amount to?
- How much time does a party have to wait between actions or between the various steps of each action?
- How often do parties need to check the blockchain for changes to the channel?
- What information leaks to the blockchain?
- What can be told on the automatic routing of payments? To what extent can it be non-blocking and private?
- Under what circumstances an operation cannot complete? (e.g. concurrency issues, veto power with misaligned incentives)
- Which participants know the identity of which participants?
- Is there an upper bound to the amount of updates? How is this number decided?
- What are the necessary tools the blockchain should provide?
- Can a participant unilaterally commit on-chain?
- Up to how much money can a participant unilaterally obtain?
- What can a malicious party do? If it corrupts more participants it can do more?
- Can a malicious/honest-but-curious party that is a participant learn who is transacting with who, especially in payments she is not involved?
- How much slower is the process in case of a malicious party?
- Can settling on-chain happen faster in case everybody is honest?
- How faster/cheaper can the system be if we assume some form of trust?
- Can arbitrary smart contracts be implemented off-chain? At what cost?
- How expensive are the actions? (CPU, memory, storage)
- How expensive are interactions with the blockchain? (fees, time, etc.)

An additional interesting question is how the design of the blockchain itself can facilitate the creation of such payment channels and whether such design choices would degrade the quality of the blockchain itself.

Model for Payment Channels

We have been considering the following simple but general model for payment channels. Stricter definitions, implications and realisation of existing solutions on this model will follow as further work.

A payment channel is a tuple $PC \in \mathcal{PC}$ such that

$$PC = (\{(P_1, c_1), \ldots, (P_n, c_n)\}, \{(e_1, b_1), \ldots, (e_m, b_m)\}, f : \mathcal{A}^n \to \mathcal{PC})$$

where $\sum_{i=1}^{n} c_i \leq \sum_{i=1}^{m} b_i$.

(P_i, c_i) represents the i-th player and her available funds on settling.

(e_j, b_j) represents the j-th on-chain endpoint and the corresponding funds that will be released for use in the blockchain if this endpoint is settled.

f is a function from player actions to a new payment channel. The new payment channel must have at most as much funds as the old.

We understand that the current formalisation of payment channels contains a cyclic definition that stems from the way the function f is used and we will strive to correct this problem before using this definition. We however want to maintain the intuition that each payment channel is accompanied by a transition function that defines the result of the various possible actions as new payment channels.

4 Directions for further work and plan of action

In this section we lay out our plan for further development of our models, known and possible hurdles we will have to overcome and a strategy to reroute our research to new, related topics in case the current approach does not bear fruit. We give a rough schedule of our projected progress at each time frame, along with the incremental results that we hope to lead to relevant publications.

We will treat separately the e-commerce and the payment channels settings, starting from the simpler issues and gradually making our way to the most complex (and most interesting) ones for each setting.

4.1 E-Commerce

Our model for reputation in e-commerce is still incomplete. As already mentioned, we have to prove that Π_{Trade} UC-realises $\mathcal{F}_{\text{Trade}}$. This will not be hard to prove, but we have chosen to defer this proof to a later point in time because it is still likely that the details of both the protocol and the functionality will change.

Security definitions and the role of the Adversary

A more fundamental and pressing issue is the exact formulation of security definitions related to the model. In particular, it is unclear whether Definition 1 is too strong or needs further refinements. Currently nothing stops Alice from deviating and falsely reporting that she has been `cheated` after an honest trade, apart from speculated game-theoretic incentives. This means that the current version of Π_{SAT} is not secure under this definition.

On the other hand, we have seen that $\mathcal{F}_{\mathrm{SAT}}$ is cheating–secure. This is proof that Π_{SAT} does not UC-realise $\mathcal{F}_{\mathrm{SAT}}$. This is another important problem with the current formulation of the model. If we consider static corruptions, the Adversary can very easily break this kind of security by telling to $\mathcal{E}$ (truthfully or not) that she has been `cheated`, or by cheating on a trade and causing another player to report the cheat. Even if we ignore the Adversary, our intuition is that every cheating–secure protocol is either of little use — consider a protocol where no trades ever take place, or a protocol where a buyer trades only with the single most trustworthy vendor — or may actually achieve security by hiding the fact that cheats happen, e.g. in case the protocol never reports any cheats that took place. This obviously defeats the purpose of this security definition. We will consider two approaches to remedy this issue.

The first is to follow the paradigm set forth in [22], according to which we can assign a utility function to the Adversary to model the cost of corruptions and her benefit when security is broken. We can then argue that a specific Π_{SAT} protocol is not secure, but is *optimal* with respect to some (or ideally all) utility functions of the Adversary — meaning that there exists no protocol with which the Adversary may obtain higher utility. This approach gives us greater flexibility and allows us to attain useful results even if we do not achieve security.

Secondly, we can add more security definitions and try to create a protocol that achieves them. For example, it is worthwhile to look for a security definition that, if achieved, ensures that no false reports of cheated trades take place, or that no unresolvable conflicts take place.

A related issue that is still not tackled in its entirety is the role of the Adversary. The current setting poses a significant departure from the usual cryptographic one in the sense that the participants' are not as clear-cut as e.g. in the case of modelling encryption. In our case, it may be to the participants' advantage to commit subtle variations from the protocol but at the same time still want to maintain its security guarantees. In this sense, arguing with respect to every polynomial algorithm may be

an overkill. Most probably though, a combination of both a conventional Adversary and rational actors will be needed. The former could model a state-level actor determined to undermine the entire scheme at all costs (e.g. a government ready to spend large amounts in order to break the security guarantees), whereas the latter would represent small-scale players that want to enjoy the security properties of the system while maximising their own gain, possibly creating a "tragedy of the commons" effect [34] in the process (c.f. selfish mining [16] in bitcoin, where each miner has incentive to deviate from the protocol, but if everybody deviates security breaks down). Combining the two actors in one system is something that has not been accomplished yet to a satisfactory degree to our knowledge and indeed seems too daunting an undertaking. Alternative approaches such as [22] must be explored and a sensible reach for the capacities of the Adversary must be established.

We would like to have solved the issues of security definitions, define the capabilities of the Adversary and, if suitable, to employ the rational protocol design approach to a satisfactory degree within the next five to six months, i.e. until December 2018. A possible publishable outcome of this stage would be to prove that, under a particular utility function for the Adversary and with respect to some plausible security definitions, a certain protocol is optimal in the sense explained previously.

Trust functionality and protocol

An additional question that remains as of yet unanswered is the way in which Π_{Trust} will be designed. Apart from placeholder and easily attackable solutions (e.g. choose the vendor with the most reported honest trades), the only real candidate is a suitable wrapper around "Trust is Risk" [35]. For reference, "Trust is Risk" is a reputation system in which Alice chooses to "directly trust" some funds to other players, publicly acknowledging that they can easily steal those funds from her. This creates a trust network, which Alice can leverage to calculate her "indirect trust" towards a potentially unknown player Bob as the maximum flow [36] from herself to Bob. This number is a measure of her trust towards him, built in a decentralised manner.

We speculate that we can effectively use this system to build a Π_{Trust} with desirable properties. The construction will roughly be as follows: Upon receiving a desire and a list of potential vendors from Alice, Π_{Trust} returns the vendor that Alice indirectly trusts the most. Upon receiving a report that Alice has been `cheated` by Bob for a loss of p_1 coins, Π_{Trust} decreases Alice's direct trusts in a way that her indirect trust towards

Bob is reduced by p_1. Upon receiving a report that Alice has successfully completed a trade with Bob, during which she was exposed to a risk of losing p_2 to Bob, Π_{Trust} increases Alice's direct trust towards Bob by p_2.

This rough protocol is intuitively appealing because we have not come up with obvious attacks and it contains a minimal number of hand-tuned parameters. Furthermore, it seems to align correctly the incentives of the various players, be it buyers, vendors or middlemen. A buyer wants to increase her trust towards a vendor of honourable past and penalise a dishonest vendor, thus a vendor is incentivised to refrain from cheating. What is more, from the point of view of a middleman, Alice would like to only directly trust players with "good taste", so that she could build a name for "good taste" herself and acquire direct (and thus indirect) trusts from others that share the same taste. She can even charge a fee for those leveraging her service in providing trustworthy references. Thus she is incentivised to choose wisely who she directly trusts based both on how good vendors they are and on who they in turn directly trust.

Last but not least, calculating indirect trust based on the maximum flow has a sociological justification [37], as people tend to lend money more commonly to those towards which their maximum flow is the highest, as calculated on a network where edges represent the amount of time two neighbouring parties have spent together.

It should be noted that such a trust protocol requires access to Alice's funds in order to reorganise the direct trusts, thus Π_{SAT} should be adapted accordingly.

Our target for the following three months is to formally describe the above sketch, decide upon its details and argue whether it satisfies our security definitions and, if so, under which conditions.

Connecting with UC modules and increasing robustness

The following issues are considered refinements to better approximate the real world and will not be resolved until we first provide a functional prototype of the simplified system that we have been discussing up to now.

Further inflection on the exact nature of the global functionalities is needed. More specifically, the interface with $\mathcal{G}_{\text{Ledger}}$ that we currently use is not aligned with that of the UC formalisation of the functionality [38].

As far as robustness goes, we would like to expand the current model to additionally accomodate for digital assets that can be placed on a blockchain — thus making it possible to achieve consensus on who is the

rightful owner. It seems natural to treat this kind of assets differently, since less trust needs to be implicated for their handling.

Additionally, in our current model there is no provision for a vendor who delivers an asset of inferior quality or a different asset from the one promised; we just account for trades that either complete successfully or do not complete at all. On a similar note, there is no provision for bargaining, currently the buyer either accepts the vendor's offer or rejects the vendor altogether for the satisfaction of the current desire. Since these kinds of behaviour are common and very usual sources of disputes, our model should be able to handle this situation in a satisfactory manner.

Furthermore, we would like to present a formal treatment on whether it is possible to realise $\mathcal{G}_{\text{Assets}}$ or not and if so, how. The point of the matter is that currently $\mathcal{G}_{\text{Assets}}$ keeps track of who owns which asset, but we would like to avoid relying on such a centralised repository of titles in the real world. To make matters worse, since physical assets cannot be reliably put on a blockchain, the real-world equivalent of this functionality is highly non-trivial and certainly needs some form of trust between players.

Further considerations

To conclude this subsection, we would like to lay out some important — if rather philosophical — observations that have arisen through inflection on the topic at hand and should be kept in mind to guide the further development of the project.

We have already alluded to the fact that there exist scenarios in which consensus is not achievable. In contrast to systems that exist entirely on blockchains, our model aspires to facilitate trades that include physical assets. There exist several fundamental differences between those and on-chain assets. Physical assets cannot be created out of nothing because of physical, immutable laws (e.g. the conservation of mass); this does not hold for digital assets, which may indeed be cloneable: there exist smart contracts where it is legal for new assets to emerge from nothing [39]. Given the flexibility of the digital world, we can create protocols where consensus on who owns which asset and exactly how history transpired can be reached, based on modest (if disputable) assumptions such as honest majority. The physical world does not possess such infrastructure, so simple and complex scenarios where consensus is unreachable abound. For example, Alice may hand over an asset to Bob in a private room and later ask it back, only to have Bob claim that she gave it to him as a

present. Any external arbiter will never be able to conclusively decide who is right, thus consensus is unreachable.

The simplistic approach of putting everything on the blockchain is not sufficient for two reasons: Firstly, it would be prohibitively expensive to put videos of everything on-chain. Secondly, even if we somehow managed to scale blockchains to accommodate for such needs, just access control rules to such material are not clear at all and would probably constitute a science in and of itself. Last but not least, interpreting this information in case of dispute is no trivial task. Current consensus protocols accommodate a strictly defined set of assertions and do not permit contradictory facts to be recorded in the first place. In the physical world however, disagreements, disputes and inconsistencies are definitely possible and we have to constantly work around them. A standard approach taken is escalation to a third party chosen to be as impartial as possible, commonly taking the form of a court of law. When however accounts mismatch, it often devolves to a "my word against yours" situation where the apparently most trustworthy account is deemed true. This is where trust comes unavoidably in action. This is the kind of trust we are trying to harness with the current work.

The ultimate target of this research is to create a plausible framework within which we can argue on the trustworthiness of institutions, people and statements based on one's experience, knowledge and interconnections in order to decide which course of future action will maximise some particular objective. Having such a framework in place, we could then recognise general tendencies and properties of the system as a whole when each individual agent tries to attain their objective. Ideally we will be able to set up the game in a particular fashion so as to achieve desirable properties for the system, such as maximisation of social welfare.

4.2 Payment Channels

Application and refinement of the model

Moving on to future plans for the research on payment channels, first and foremost we want to experiment with the model presented in Subsection 3.2. In particular, it is of great importance to ascertain whether the various existing concrete designs can be expressed in terms of our model. We will start by transcribing the simplest payment channel design, the Lightning Network [23], and gradually make our way to more complex systems. We will alter and improve our model so as to accommodate for as many systems as possible. If serious hurdles are encountered in the

process, we will attempt to formally argue on the reasons and produce relevant impossibility results. We may need to create two or more incompatible models to account for various incompatible situations.

A direction that has not been taken as of yet by any of the proposed systems (to our knowledge) is to make some trust assumptions in order to build a more robust, cheaper system. Even though trustless off-chain payments are certainly a useful feature, the lack of trust can be seen as a constraint. For example, it is plausible to consider channels where in case of unilateral settlement, an external, pre-agreed arbiter could have the final say on who receives which funds. This could obviate the need for long timeouts before the funds are unlocked for on-chain use but would entail some limited trust to the arbiter.

An example of a payment channel where trust is involved in a simple (and easily exploitable) manner is the Lightning Network construction but with a 1-of-2 multisignature [40] funding transaction instead of a normal 2-of-2 transaction (more on this in Section 5). Our model should be capable of expressing such naive systems as well.

We hope to have completed developing and fine-tuning our model in the span of the following 3-4 months, i.e. until the end of September 2018.

Security definitions and design

After having established our model, a series of security definitions should be given to account for the various desirable properties of specific payment channels. Similarly to the previous step, these definitions will be inspired by the concrete payment channel constructions. Furthermore, we will attempt to prove (or disprove) the security properties of individual designs. This process will gradually begin when our model starts taking a more definitive form and will hopefully be completed in the next 5 to 7 months.

We will then move on to the most important stage of this research, namely to attempt to create one or more concrete payment channel systems that attain specific goals in an optimal manner. A complementary goal is to identify incompatible properties and prove the exact reasons why it is impossible to create a system that enjoys all of them.

4.3 Contigency Plan

There are several things that can go wrong with the aforementioned stages of further development. The most obvious is the danger of overgeneralisation. It may be the case that the problems we have embarked to solve prove untractable. In this case, we will have to consider more

specific simpler cases in which arguing will be easier. For example, it is possible that the evidently general way in which we attempt to model payment channels has too little structure, thus reducing our ability to express or prove interesting properties. In this case, we will focus on special cases under which we will be able to effectively argue with respect to these properties.

Another possible issue in this rapidly evolving area is that our problems are solved by other researchers before we can provide a solution. We hope that this will not be a serious impediment, as our exploration of the field, in combination with said results from third parties, will provide us with material to reroute our research to other relevant open questions.

We would like to note that having two different, albeit related problems at hand makes it easier to backtrack and devote more time to one of the two in case we cannot keep pursuing the other for any reason; we have effectively reduced the danger of a "single point of failure". Nevertheless, we hope to eventually manage to attain a unified thesis that will stem from the research of these two related issues.

5 Literature Review

5.1 E-Commerce and Reputation Systems

In this subsection we will give a brief overview of related work in reputation and trust systems, as well as relevant game theoretic results.

Trust and Reputation

Webs-of-trust can be used as a basis for trust as shown by Caronni [41]. PGP [42] implements one and Pathfinder [43] explores its transitive closure. Freenet [44] implements a transitive web-of-trust for fighting spam. Mui et al. [45] and Jøsang et al. [46] propose ways of calculating trust towards distant nodes. Vişan et al. [47] calculate trust in a hierarchical way. CA- and Byzantine-based [48] PKIs [49] and Bazaar [50] require central trusted third parties or at least authenticated membership. FIRE [51], CORE [52], Grünert et al. [53] and Repantis et al. [54] do not prove any Sybil resilience. All these systems define trust in a non-financial manner.

We agree with Gollmann [55] in that the meaning of trust should not be extrapolated. When formally defining trust, we will keep in mind their advice.

Beaver [56] includes a trust model that, to discourage Sybil attacks, relies on fees, something we would like to avoid. Our motivating application for exploring trust in a decentralized setting is OpenBazaar, where

transitive financial trust has previously been explored by Zindros [57]. That work however does not define trust as a monetary value. We are strongly inspired by Karlan et al. [58] who give a perceptive sociological approach to the connection between trust and risk. TrustDavis [59] proposes a financial trust system with transitivity and in which trust is defined as lines-of-credit.

The work of Fugger [60], applied in Ripple [61] and Stellar [62], leverages the concept of money-as-debt. This approach is complementary to that of Trust is Risk [35], where a functionally similar construction defines trust-as-risk. We are currently experimenting with using the latter work as a concrete instantiation of $\mathcal{F}_{\text{Trust}}$.

Game Theory and Mechanism Design

There are several concepts relevant to reputation and trust in the realm of Game Theory. To begin with, the Vickrey-Clarkes-Grove mechanism [17] is strategy-proof mechanism to select a particular outcome out of many in a way that maximises utilitarian social welfare and incentivises players to be honest about their preferences. Furthermore, there exist positive results on Nash Equilibria in games of fair resource allocation [63]. Such results may be leveraged to employ trustworthy mechanisms with envy-free outcomes — two caveats are the finite horizon of these mechanisms and the possibly exponential resources needed to calculate outcomes. Trust to unknown agents is also related to the belief system used in bayesian games [64]. These games are better suited to our case, since they can have an infinite horizon and the limits of each player's knowledge about the rest are more clearly defined.

The study of economic trust rests heavily upon results regarding the existence of market equilibria [21] and the non-existence of fair and free elections under general circumstances [65]. These results show us the importance of a common understanding of the concept of value amongst players in order to achieve fair outcomes, a concept that to some extent can be identified with that of money.

5.2 Payment and State Channels

There exists already a considerable body of work related to payment channels. Here we will explain relatively thoroughly the Lightning Network [23] in order to make clear the basic mechanism. Then we will explain other systems at a higher level and compare the different properties, advantages and drawbacks each system has.

Lightning Network

This construction is the first to achieve a functional, efficient model for payment channels. It is designed for bitcoin [1] and requires some new opcodes and removing the malleability of transactions to function properly [23].

– **Simple two-party channel**

The basic construction is as follows. Suppose that *Alice* and *Bob* want to create a payment channel that contains 1 BTC consisting of 0.5 BTC from each party. To achieve this, they follow these steps (see also section 3.1.2 and Figure 4 in 3.3.2 in [23]):

1. Either party (say *Alice*) creates a "Funding" transaction (F) with an input of 0.5 BTC from her and 0.5 BTC from *Bob*, and a 2-of-$\{Alice, Bob\}$ multisig as output; she then sends F to *Bob*. This transaction is not yet signed nor broadcast. F needs to be signed by both parties to be valid.
2. *Alice* creates, signs and sends to *Bob* a "Commitment" transaction ($C1b$) that spends F and has the following outputs:
   (a) 0.5 BTC that can be spent by *Alice* immediately when $C1b$ is broadcast.
   (b) 0.5 BTC that can be spent by either party, but *Bob* can spend it only after a specified amount of blocks (say n) have been mined on top of $C1b$, whereas *Alice* can spend it only if *Bob* provides her with a "Breach Remedy" transaction (explained later) signed by him. This output is called "Revocable Sequence Maturity Contract" (RSMC).

   Furthermore, *Alice* creates, signs and sends a "Revocable Delivery" transaction ($RD1b$) that pays the first of the two outputs of $C1b$ to *Bob*, but will be accepted by the network if it is in the mempool only after n blocks have been mined on top of $C1b$.
   Bob similarly creates, signs and sends $C1a$ and $RD1a$ to *Alice*.
3. After *Alice* receives the signed $C1a$ and $RD1a$ from *Bob*, she verifies that they are both valid and correctly spend F. Given that everything works out right, she signs F and sends it to *Bob*.
   Bob similarly verifies that $C1b$ and $RD1b$ have the correct structure, along with *Alice*'s signature on F. He then signs F and broadcasts it. Note that he does not have to trust *Alice* in any way.
   The fact that *Alice* holds $C1a$ and $RD1a$, already signed by *Bob*, ensures her that her 0.5 BTC cannot be locked in the 2-of-2 multisig of F in case *Bob* stops cooperating. If she decides that *Bob*

stopped cooperating, she can broadcast $C1a$, wait for it to be confirmed n times and broadcast $RD1a$ to get her money back. Thus *Alice* need not trust *Bob* either.

Observe that if *Bob* refuses to cooperate in signing F, then the blockchain has not been changed and no funds are at risk. In such case, to ensure that *Bob* cannot lock her funds in the future, she should immediately transfer her funds to a new address or periodically check the blockchain for F and broadcast $C1a$ and $RD1a$ in case she finds F on the ledger.

After initially setting up the channel, *Alice* and *Bob* can update it as follows (see also section 3.3.4 and Figures 7, 8 in [23]):

1. Both *Alice* and *Bob* follow exactly the same steps as before to create $C2a$, $C2b$, $RD2a$ and $RD2b$; the only difference these transactions have to their counterparts from the previous state of the channel is that, instead of 0.5 BTC for each player, they contain the new agreed balance of the channel (e.g. 0.4 BTC for *Alice* and 0.6 BTC for *Bob*).
2. *Alice* creates, signs and sends to *Bob* a so-called "Breach Remedy" transaction ($BR1a$). This transaction lets *Bob* redeem the RSMC output of $C1a$ as soon as $C1a$ is broadcast. *Bob* similarly creates, signs and sends $BR1b$ to *Alice*.

Note that this effectively disincentivises *Alice* from ever broadcasting $C1a$, since in such case *Bob* will have a window of n blocks during which he can claim the entire sum in $C1a$, 1 BTC, for himself. *Alice* had better purge $C1a$ after $BR1a$ is sent to *Bob*. Similarly *Bob* is incentivised to refrain from ever broadcasting $C1b$.

This arrangement creates a situation where both players can be confident that the state of the channel is the one expressed by $C2a$, $C2b$, $RD2a$ and $RD2b$, thus they can assume that *Alice* has just paid *Bob* 0.1 BTC. No trust between the two players was needed all along. There are only two caveats: First, both players must periodically check the blockchain to ensure that the other party has not broadcast an old Commitment transaction. Second, in case of an uncooperative counterparty, one has to wait a prespecified amount of time before releasing their funds, which may be undesirable.

Thus, the necessary number of blocks mined on top of a Confirmation transaction for a subsequent Revocable Delivery to be valid (previously called n) must be carefully chosen in a way that does not lock up the funds for a long time in case of a dispute and at the same time does not require that the parties check the blockchain too of-

ten for a malicious broadcast of an already invalidated Commitment transaction.

Alice can outsource the task of the periodic check to a dedicated service by sending it all the previous Breach Remedy transactions. To incentivise the service to cooperate, *Alice* can pay a fee to it as an output of these transactions. Note that *Alice* does not need to trust the service, since the only thing it can do is to broadcast a Branch Remedy transaction that was created by *Alice*; she never discloses any of her private keys to it.

Finally, the parties can cooperatively close the channel without having to wait n blocks as follows: When both parties have agreed to closing the channel, *Alice* creates, signs and sends to *Bob* an "Exercise Settlement" transaction (ES) that spends the Funding transaction and has two simple outputs, each paying to the respective party the sum of the last agreed Commitment transaction. Following the previous example, this transaction would pay 0.4 BTC to *Alice* and 0.6 BTC to *Bob*. *Bob* can then also sign and broadcast the transaction to close the channel.

Once *Alice* has sent ES, she considers the channel as closed. If *Bob* does not broadcast ES, we have a dispute and she has to broadcast the latest Commitment transaction and wait for her funds to be unlocked.

– **Payments depending on preimage knowledge (HTLC)**

Multi-hop payments can take place between players (e.g. *Alice* and *Dave*) who do not share a simple channel (i.e. an on-chain Funding transaction), but share simple channels with intermediate nodes (e.g. *Alice* with *Bob*, *Bob* with *Carol* and *Carol* with *Dave*).

To enable the creation of multi-hop channels, so-called "Hashed Timelock Contracts" (HTLC) are used. An HTLC is an additional output in a Commitment transaction which can be redeemed by either *Alice* or *Bob*; *Alice* can redeem it after a specified number of additional blocks, say m, have been mined after the creation (*not* the broadcast) of the Commitment transaction, whereas *Bob* can redeem it at any time, but only if he produces the preimage R of a hash specified in the HTLC output (see also section 4.2 and Figure 12 in [23]).

More specifically, consider $C2a, C2b$ where, contrary to the example in the previous subsection, *Alice* has paid the 0.1 BTC to an HTLC instead of directly to *Bob*. *Bob* should be able to redeem the 0.1 BTC only if he knows the preimage R before the m blocks have been mined.

In addition to $RD2a$ and $RD2b$, six additional transactions have to be signed and exchanged.

1. *Alice* signs and sends an "HTLC Execution Delivery" transaction ($HED1a$) to *Bob*. $HED1a$ pays the HTLC output of $C2a$ to *Bob*, only if he knows the required preimage R. Only *Bob* can broadcast the transaction.
2. *Bob* signs and sends a so-called "HTLC Timeout Delivery" transaction ($HTD1b$) to *Alice*. $HTD1b$ pays the HTLC output of $C2b$ to *Alice*, only after m blocks have been mined from the time $C2b$ was created. Only *Alice* can broadcast this transaction.
3. *Alice* signs and sends an "HTLC Execution" transaction ($HE1b$) to *Bob*. $HE1b$ pays the HTLC output of $C2b$ to *Bob*, only if he knows the required preimage R. Only *Bob* can broadcast this transaction. Its single output is an RSMC with duration n, spendable by *Bob*.
4. *Alice* signs and sends an "HTLC Execution Revocable Delivery" transaction ($HERD1b$) to *Bob*. This transaction spends the RSMC output of $HE1b$. *Bob* can broadcast this transaction after n blocks have been mined on top of $HE1b$.
5. *Bob* signs and sends an "HTLC Timeout" transaction ($HT1a$) to *Alice*. $HT1a$ pays the HTLC output of $C2a$ to *Alice*, only after m blocks have been mined from the time $HT1a$ was created. Its single output is an RSMC with duration n, spendable by *Alice*.
6. *Bob* signs and sends an "HTLC Timeout Revocable Delivery" transaction ($HTRD1b$) to *Alice*. This transaction spends the RSMC output of $HT1b$. *Alice* can broadcast this transaction after n blocks have been mined on top of $HE1b$.

Note that once again, no trust is necessary in the process described above. The RSMC outputs of $HT1a$ and $HE1b$ are necessary for future invalidation according to the "Breach Remedy" method. More details can be found in Figure 14 of section 4.3. In case of common desire to close the channel, they can be cooperatively closed using the "Exercise Settlement" method.

– **Multi-hop channels**

With the use of HTLC outputs, it is possible to execute multi-hop payments as follows. Suppose *Alice* wants to pay *Dave* 0.001 BTC and they find out that they are connected through the preexisting channels *Alice* $\Leftrightarrow$ *Bob*, *Bob* $\Leftrightarrow$ *Carol* and *Carol* $\Leftrightarrow$ *Dave*. This payment can be completed with the following steps:

1. *Dave* generates a random number R and sends $hash\,(R)$ to *Alice*, *Bob* and *Carol*.
2. *Alice* and *Bob* update their channel with an e.g. 300-block HTLC that transfers 0.001 BTC from *Alice* to *Bob*.
3. *Bob* and *Carol* update their channel with an e.g. 200-block HTLC that transfers 0.001 BTC from *Bob* to *Carol*.
4. *Carol* and *Dave* update their channel with an e.g. 100-block HTLC that transfers 0.001 BTC from *Carol* to *Dave*.
5. *Dave* discloses R to *Carol*; he obtains 0.001 BTC from the 100-block HTLC transaction.
6. *Carol* discloses R to *Bob*; she obtains 0.001 BTC from the 200-block HTLC transaction.
7. *Bob* discloses R to *Alice*; he obtains 0.001 BTC from the 300-block HTLC transaction.

Thus *Alice* has paid *Dave* 0.001. No party can be defrauded: For example, *Carol* will pay 0.001 BTC to *Dave* if he shows her R within 100 blocks but then she can take the 0.001 BTC back by disclosing R to *Bob*; she has at least 100 more blocks to do so. In case *Dave* does not disclose R, all parties can take their funds back by settling on-chain.

On the other hand, assume that *Bob* does not cooperate after the establishment of the HTLC transactions, but keeps R hidden. In this case *Bob* will lose his 0.001 BTC to *Carol* and no other player will be negatively affected; *Carol* and *Dave* can fulfill their part without *Bob*'s cooperation, albeit *Carol* will have to wait for her channel with *Bob* to expire, since she has to settle on-chain. Likewise *Alice* can take back her 0.001 after the 300-block HTLC lock has expired. Thus no trust between parties is needed.

One can note three things: Firstly, there is no such thing as a persistent multi-hop channel. The whole procedure must be repeated for each subsequent multi-hop payment and the successful completion of one such payment does not facilitate the creation of future payments along the same route as far as the techniques described above are concerned. Nevertheless, previous cooperation between players can obviate the need of exploring the network anew for a connecting series of preexisting channels.

Secondly, merely the existence of a channel is not enough to ensure that multi-hop payments can be achieved through it. It must be the case that the correct player holds at least as much funds as the desired payment, which can only be verified by asking the players of the channel, since the latest state is not public. Thus, in the previous example,

Bob must own at least 0.001 BTC in the $Bob \Leftrightarrow Charlie$ channel in order for the payment to be possible. *Alice* (or *Dave*) must ask *Bob* and *Charlie* whether this is the case before initiating the multi-hop payment process.
Finally, all intermediate players have to actively engage for a multi-hop payment to go through. This means that a multi-hop payment's latency increases linearly with the length of the chain, as well as the waiting time if on-chain settlement is needed (given that the same margin of security is desired irrespective of the payment length). This reduces the scalability of the design and fosters the creation of centralized, heavily connected players that ensure that short chains are available instead of distributed, loosely connected players that exchange funds through long chains.

Perun

Perun [24] is a type of trustless payment channel designed for Turing-complete smart contract scripting languages. It has been implemented for Ethereum. Its main contribution is *multistate channels* that allow the dynamic deployment of virtual contracts, known as *nanocontracts*. Contracts of this type do not have to exist in the blockchain if all parties are cooperative and are only broadcast to the blockchain in case of a dispute.

The paper describes specifically the use of such multistate channels for creating virtual payment channels between parties that do not have a basic payment channel between them, but both have basic multistate channels with an intermediary. Then the intermediary could substitute for the blockchain and thus a virtual payment channel on top of the two basic multistate channels can be created. The parties need the intermediary only for setting up the channel and to close it fast. If the intermediary refuses to close the channel, they can always fall back to the blockchain in order to close it.

Perun is more robust and feature-rich than the Lightning Network. As already mentioned, arbitrary contracts can be implemented off-chain in the former, whereas the latter does not offer such functionality. Furthermore, persistent multi-hop channels can be built in Perun. Thus updating the state of long chains is independent of the length of the channel, in expense of the cost of setting up and closing the channel in case of dispute.

One minor drawback in Perun is that a player that broadcasts an earlier state of the contract does not risk losing her rightful share of the

funds (according to the latest state). This incentivizes players to broadcast earlier states of a contract more easily.

Payment Network with Duplex Micropayment Channels

This work [26] is historically the first payment channel solution. It serves as proof-of-concept for the viability of payment channels, but has some drawbacks. First of all, it requires a rather high amount of local storage for each payment channel. Secondly, the number of possible updates throughout the existence of a channel is limited. Nevertheless, this proposal gives valuable insight on the inherent tradeoffs in payment channels design.

Sprites

Sprites [66] constitutes an improvement upon Lightning regarding the worst-case time needed to settle in case of a dispute. This construction needs general-purpose smart contracts to be implemented. Consider a channel of l hops, where Δ is the time given to each participant to publish their state after a counterparty has unilaterally broadcast theirs. The worst-case time to settle in the case of Lightning is $\Theta(l\Delta)$, whereas in Sprites it is $\Theta(l + \Delta)$.

To achieve this, this system uses a smart contract called Preimage Manager, which is used only in case of dispute and keeps track of published preimages. A player that wants to settle on-chain has to broadcast the preimage instead of just sending it to the next hop of the multi-hop channel and thus all subsequent players can use this preimage to retrieve their funds, thus obviating the need of a $\Theta(\Delta)$ timeout for each hop.

Bolt

This construction [30] offers an alternative approach that provides stronger privacy guarantees. In particular, the proposal consists of two-party unidirectional or bidirectional channels where one end of the channel (say Alice) cannot distinguish between payments made from or to the various players with which Alice is connected. Thus two such bidirectional channels can be combined if Bob wants to pay Charlie through Alice in such a way that the transactions do not touch the blockchain and Alice does not learn who paid whom. Ideas from e-cash and blind signatures [67] are applied in this work.

Concurrency and Privacy with Payment-Channel Networks

This work [29] focuses on issues arising when considering routing of payments through existing networks of payment channels. In contrast to other applications that assume the path that a multi-hop payment will follow is known in advance, this paper proposes a way to decide which path to use for payments given a particular network topology. In particular it proposes two intimately related routing mechanisms, Fulgor and Rayo. The first provides better privacy guarantees but is prone to deadlocks while routing, in which case time and mining fees are wasted. On the other hand, Rayo enforces non-blocking progress at the expense of lower privacy guarantees. Importantly, an impossibility result is provided, ruling out a system that provides both privacy and non-blocking progress. This system can be deployed on bitcoin, since it uses simple HTLC contracts.

Teechan

Teechan [25] achieves off-chain payments with superior speed, lower latency with fewer messages per transaction compared to other systems and only two on-chain transactions for the entire channel lifetime. It achieves such improvements by leveraging Intel SGX, which is a trusted execution enclave present in modern Intel processors [68]. This is a stronger but reasonable assumption that permits two parties to securely and efficiently execute the desired code in a remote processor, trusting only that Intel has correctly implemented SGX.

Such a class of assumptions may prove versatile and efficient enough to reliably enable realistic and scalable monetary transactions in the future, with Teechan providing an influential blueprint, but further researh upon such enclaves is needed [69,70].

Blindly signed contracts

Last but not least, this construction [31] extends solutions like Lightning [23] and Duplex Micropayment Channels [26] to add anonymity using blindly signed contracts. Their technique is mainly developed for on-chain mixing purposes, but is extended to function in the off-chain setting as well.

Payment channels literature: Conclusion

The variety of payment channel instantiations and the breadth of their aims underlines the abstraction level our model should achieve. Indeed,

Perun [24] offers arbitrary off-chain contracts, Bolt [30] provides strong privacy guarantees and "Concurrency and Privacy with Payment-Channel Networks" [29] highlights central issues in multi-hop payment routing. Ideally our model should be able to capture all these dimensions of payment channels and provide a unified vocabulary to facilitate the comparison of existing as well as future instantiations of such systems.

References

1. Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
2. Bitcoin and cryptocurrencies – what digital money really means for our future. https://www.theguardian.com/technology/2018/jan/29/cryptocurrencies-bitcoin-blockchain-what-they-really-mean-for-our-future: accessed: 24/5/2018
3. Bitcoin Scaling Problem, Explained. https://cointelegraph.com/explained/bitcoin-scaling-problem-explained: accessed: 24/5/2018
4. Bitcoin and cryptocurrencies – what digital money really means for our future. https://hackernoon.com/the-crypto-civil-war-40ee1ee9314f: accessed: 24/5/2018
5. Wood G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 151, 1-32 (2014)
6. Zhong M.: A faster single-term divisible electronic cash: ZCash. Electronic Commerce Research and Applications: vol. 3–4(1), pp. 331–338 (2002)
7. Kiayias A., Russell A., David B., Oliynykov R.: Ouroboros: A provably secure proof-of-stake blockchain protocol. pp. 357–388: Springer, Cham: Annual International Cryptology Conference (2017)
8. block.one: EOS.IO Technical White Paper v2. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md (2018)
9. OpenBazaar.org. https://openbazaar.org
10. Back A.: Hashcash - A Denial of Service Counter-Measure. http://www.hashcash.org/papers/hashcash.pdf (2002)
11. Vermeulen J.: Bitcoin and Ethereum vs Visa and PayPal – Transactions per second. https://mybroadband.co.za/news/banking/206742-bitcoin-and-ethereum-vs-visa-and-paypal-transactions-per-second.html (2017)
12. Lindell Y.: How To Simulate It – A Tutorial on the Simulation Proof Technique. Tutorials on the Foundations of Cryptography: pp. 277–346 (2017)
13. Canetti R.: Universally composable security: A new paradigm for cryptographic protocols. In Foundations of Computer Science: pp. 136–145: IEEE (2001)
14. Diffie W., Hellman M.: New Directions in Cryptography. IEEE transactions on Information Theory: vol. 22(6), pp. 644–654 (1976)
15. Garay J., Kiayias A., Leonardos N.: The Bitcoin Backbone Protocol: Analysis and Applications. In Annual International Conference on the Theory and Applications of Cryptographic Techniques: pp. 281–310: Springer (2015)
16. Eyal I., Sirer E. G.: Majority is not enough: Bitcoin mining is vulnerable. In International conference on Financial Cryptography and Data Security: pp. 436–454: Springer (2014)

17. Nisan N., Roughgarden T., Tardos E., Vazirani V. V.: Algorithmic game theory: vol. 1. Cambridge University Press Cambridge (2007)
18. Leyton-Brown K., Shoham Y.: Essentials of game theory: A concise multidisciplinary introduction. Synthesis Lectures on Artificial Intelligence and Machine Learning: vol. 2(1), pp. 1–88 (2008)
19. Daskalakis C., Goldberg P. W., Papadimitriou C. H.: The complexity of computing a Nash equilibrium. SIAM Journal on Computing: vol. 1(39), pp. 195–259 (2009)
20. Fagin R., Halpern J. Y., Moses Y., Vardi M.: Reasoning about knowledge. MIT Press (2004)
21. Arrow K. J., Debreu G.: Existence of an equilibrium for a competitive economy. Econometrica: Journal of the Econometric Society: pp. 265–290 (1954)
22. Garay J., Katz J., Maurer U., Tackmann B., Zikas V.: Rational protocol design: Cryptography against incentive-driven adversaries. In Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on: pp. 648–657: IEEE (2013)
23. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments
24. Dziembowski S., Eckey L., Faust S., Malinowski D.: PERUN: Virtual Payment Channels over Cryptographic Currencies. IACR: Cryptology ePrint Archive (2017)
25. Lind J., Eyal I., Pietzuch P., Sirer E. G.: Teechan: Payment channels using trusted execution environments. arXiv preprint arXiv:1612.07766 (2016)
26. Decker C., Wattenhofer R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In Symposium on Self-Stabilizing Systems: pp. 3–18: Springer (2015)
27. Chiesa A., Green M., Liu J., Miao P., Miers I., Mishra P.: Decentralized anonymous micropayments. In Annual International Conference on the Theory and Applications of Cryptographic Techniques: pp. 609–642: Springer (2017)
28. Miller A., Bentov I., Kumaresan R., Cordi C., McCorry P.: Sprites and State Channels: Payment Networks that Go Faster than Lightning. arXiv preprint arXiv:1702.05812 (2017)
29. Malavolta G., Moreno-Sanchez P., Kate A., Maffei M., Ravi S.: Concurrency and privacy with payment-channel networks. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security: pp. 455–471: ACM (2017)
30. Green M., Miers I.: Bolt: Anonymous payment channels for decentralized currencies. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security: pp. 473–489: ACM (2017)
31. Heilman E., Baldimtsi F., Goldberg S.: Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In International Conference on Financial Cryptography and Data Security: pp. 43–60: Springer (2016)
32. Herlihy M.: Atomic cross-chain swaps. arXiv preprint arXiv:1801.09515 (2018)
33. Dziembowski S., Faust S., Hostáková K.: Foundations of State Channel Networks. https://eprint.iacr.org/2018/320 (2018)
34. Hardin G.: The Tragedy of the Commons. Science: vol. 162(3859), pp. 1243–1248: ISSN 0036-8075: doi:10.1126/science.162.3859.1243: URL http://science.sciencemag.org/content/162/3859/1243 (1968)
35. Thyfronitis Litos O. S., Zindros D.: Trust Is Risk: A Decentralized Financial Trust Platform. IACR: Cryptology ePrint Archive (2017)
36. Orlin J. B.: Max Flows in O(nm) Time, or Better. In STOC '13 Proceedings of the forty-fifth annual ACM symposium on Theory of computing: pp. 765–774: ACM, New York: doi:10.1145/2488608.2488705 (2013)
37. Johnson-George C., Swap W. C.: Measurement of specific interpersonal trust: Construction and validation of a scale to assess trust in a specific other. Journal of personality and social psychology: vol. 43(6), p. 1306 (1982)

38. Badertscher C., Maurer U., Tschudi D., Zikas V.: Bitcoin as a transaction ledger: a composable treatment. In Annual International Cryptology Conference: pp. 324–356: Springer (2017)
39. Cryptokitties. https://www.cryptokitties.co/: accessed: 29/5/2018
40. Multisignature. https://en.bitcoin.it/wiki/Multisignature: accessed: 15/5/2018
41. Caronni G.: Walking the web of trust. In Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE 9th International Workshops: pp. 153–158 (2000)
42. Zimmermann P.: PGP Source Code and Internals. The MIT Press (1995)
43. Penning H. P.: PGP pathfinder. pgp.cs.uu.nl
44. Clarke I., Sandberg O., Wiley B., Hong T. W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In H. Federrath (editor), Designing Privacy Enhancing Technologies: pp. 46–66: Springer-Verlag Berlin Heidelberg, Berkeley, USA (2001)
45. Mui L., Mohtashemi M., Halberstadt A.: A Computational Model of Trust and Reputation. In Proceedings of the 35th Annual Hawaii International Conference on System Sciences: pp. 2431–2439: IEEE (2002)
46. Jøsang A., Ismail R.: The Beta Reputation System. In Proceedings of the 15th Bled Electronic Commerce Conference (2002)
47. Vişan A., Pop F., Cristea V.: Decentralized Trust Management in Peer-to-Peer Systems. In 10th International Symposium on Parallel and Distributed Computing: pp. 232–239 (2011)
48. Lamport L., Shostak R., Pease M.: The Byzantine Generals Problem. In ACM Transaction on Programming Languages and Systems: vol. 4.3: pp. 382–401 (1982)
49. Adams C., Lloyd S.: Understanding PKI: concepts, standards, and deployment considerations. Addison-Wesley Professional (2003)
50. Post A., Shah V., Mislove A.: Bazaar: Strengthening User Reputations in Online Marketplaces. In Proceedings of NSDI'11: 8th USENIX Symposium on Networked Systems Design and Implementation: p. 183 (2011)
51. Huynh T. D., Jennings N. R., Shadbolt N. R.: An Integrated Trust and Reputation Model for Open Multi-Agent Systems. Autonomous Agents and Multi-Agent Systems: vol. 13(2), pp. 119–154 (2006)
52. Michiardi P., Molva R.: Core: a Collaborative Reputation Mechanism to Enforece Node Cooperation in Mobile Ad-hoc Networks. In Advanced Communications and Multimedia Security: pp. 107–121: Springer US (2002)
53. Grünert A., Hudert S., Köning S., Kaffille S., Wirtz G.: Decentralized Reputation Management for Cooperating Software Agents in Open Multi-Agent Systems. ITSSA: vol. 1(4), pp. 363–368 (2006)
54. Repantis T., Kalogeraki V.: Decentralized Trust Management for Ad-hoc Peer-to-Peer Networks. In Proceedings of the 4th International Workshop of Middleware for Pervasive and Ad-hoc Computing: p. 6: MPAC: ACM (2006)
55. Gollmann D.: Why trust is bad for security. Electronic notes in theoretical computer science: vol. 157(3), pp. 3–9 (2016)
56. Soska K., Kwon A., Christin N., Devadas S.: Beaver: A Decentralized Anonymous Marketplace with Secure Reputation (2016)
57. Zindros D.: Trust in Decentralized Anonymous Marketplaces (2015)
58. Karlan D., Mobius M., Rosenblat T., Szeidl A.: Trust and social collateral. The Quarterly Journal of Economics: pp. 1307–1361 (2009)
59. DeFigueiredo D. D. B., Barr E. T.: TrustDavis: A Non-Exploitable Online Reputation System. CEC: vol. 5, pp. 274–283 (2005)

60. Fugger R.: Money as IOUs in Social Trust Networks & A Proposal for a Decentralized Currency Network Protocol. http://archive.ripple-project.org/decentralizedcurrency.pdf (2004)
61. Schartz D., Youngs N., Britto A.: The Ripple protocol consensus algorithm. Ripple Labs Inc White Paper: vol. 5 (2014)
62. Mazieres D.: The stellar consensus protocol: A federated model for internet-level consensus. Stellar Development Foundation (2015)
63. Brânzei S., Miltersen P. B.: Equilibrium analysis in cake cutting. In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems: pp. 327–334: International Foundation for Autonomous Agents and Multiagent Systems (2013)
64. Harsanyi J. C.: Games with incomplete information played by âĂIJBayesianâĂİ players, I–III Part I. The basic model. Management science: vol. 14(3), pp. 159–182 (1967)
65. Arrow K. J.: A difficulty in the concept of social welfare. Journal of political economy: vol. 58(4), pp. 328–346 (1950)
66. Miller A., Bentov I., Kumaresan R., Cordi C., McCorry P.: Sprites and State Channels: Payment Networks that Go Faster than Lightning. ArXiv preprint arXiv:1702.05812 (2017)
67. Chaum D.: Blind signatures for untraceable payments. In Advances in cryptology: pp. 199–203: Springer (1983)
68. Costan V., Devadas S.: Intel SGX Explained. IACR Cryptology ePrint Archive: vol. 2016, p. 86 (2016)
69. Swami Y.: Intel SGX Remote Attestation is not sufficient (2017)
70. Spectre attack against SGX enclave. https://github.com/lsds/spectre-attack-sgx: accessed: 19/5/2018