Promise: Leveraging Future Gains for Collateral Reduction

Abstract. Collateral employed in cryptoeconomic protocols protects against the misbehaviour of economically rational agents, compensating honest users for damages and punishing misbehaving parties. The introduction of collateral, however, carries two disadvantages: (i) requiring actors to lock up substantial amount of collateral can be an entry barrier, limiting the set of candidates to wealthy agents; and (ii) affected agents incur ongoing opportunity costs as the collateral cannot be utilized elsewhere.

We present Promise, a mechanism to decrease the initial capital requirements of economically rational service providers in crypt economic protocols. The mechanism leverages future income (e.g. service fees) prepaid by users to reduce the collateral actively locked up by service providers, while sustaining secure operation of the protocol. Promise is applicable in the context of multiple service providers competing for users. We provide a model for evaluating its effectiveness and argue its security. Demonstrating Promise's applicability, we discuss how Promise can be integrated into a cross-chain interoperability protocol, XCLAIM, and a second-layer scaling protocol, NOCUST. Last, we present an implementation of the protocol on Ethereum showing that all functions of the protocol can be implemented in constant time complexity and function execution costs at most USD 0.02.

1 Introduction

Since their creation, arguably the most significant property of blockchains is their facilitation of trustless exchange between entities with *weak identities* [7] [1]. Yet the trustless nature of the systems means not only that parties *may* transact without trusting each other, but also that they *should not trust* each other. This creates a design challenge for interactions which would typically involve such trust. In this paper, we focus on blockchain protocols which, at least in part, encode *trust* by monetary *collateral*. Here, collateral is value escrowed by an agent Alice to guarantee another agent Bob that regardless of the behavior of Alice, Bob cannot lose funds. In particular, payment, cross-chain, and generic computation protocols can be designed such that Bob is guaranteed to receive at least the same amount of funds from Alice that are at risk. Protocols involving collateral include cross-chain communication [20], scalable off-chain payments [14], state channels [9], watchtowers [15,3,4], and outsourcing of computation and verification games [19]

[1] Agents with *weak* identities are able to suddenly leave a network; agents with *strong* identities cannot.

Problem. Relying on collateral as trust is itself associated with a set of challenges. Collateralization requires the provision of a substantial amount of funds upon protocol initialization, limiting the set of participants to a selected few. Leaving participation to a small set of agents can lead to phenomenons like "rich are getting richer" through wealth compounding [10]. While it is not possible to grant less wealthy agents proportionally higher rewards due to Sybil identities, we can lower the entry barrier for agents to join a protocol. Finally, locked funds result in opportunity costs for the agent who could use their collateral for participating in other protocols [13].

This work. We present Promise, a simple but effective mechanism to lower entry barriers for intermediaries in protocols relying on collateral for secure operation. Instead of locking up a significant amount of funds as collateral, Promise allows intermediaries to stake their future income, "promised" for correctly providing the service (i.e., fees), instead. Similar to online orders in e-commerce, users can choose to pay fees upfront ("forward payments") – for a some pre-agreed service period. However, instead of transferring these payments directly to the intermediary, users lock pre-paid fees in a escrow smart contact, preventing theft by either party. The intermediary needs to provide the service honestly for the entire period set by the user. The benefit of this scheme is twofold: (i) the intermediary is incentivized to act honestly while enjoying a lower initial collateral, and (ii) the user can reduce his transaction cost and only pays if the service was provided honestly over his defined period. As long as the expected future revenue from correct operation exceeds potential gains by the intermediary, users have the option to leave the protocol, and misbehavior can be proved to the smart contract, Promise incentivizes correct behavior.

Application. We discuss how Promise can be applied to XCLAIM and NOCUST. Both protocols are suitable candidates for Promise, as in both protocols "service providers" are a necessary part. XCLAIM is a cross-chain protocol that allows creation of Cryptocurrency-backed Assets (CbA) on an issuing blockchain enabled by a collateralized third-party called a *vault* [20]. Vaults provide collateral on the issuing blockchain to ensure that it is not economically rational for them to steal the locked cryptocurrency on the backing blockchain. NOCUST is a commit-chain protocol that allows to send cryptocurrency payments off-chain facilitated by so-called *operators* [14]. Operators are service-providing agents that (i) collect fees for operating the off-chain payment network, and (ii) provide a certain amount of collateral to insure finalization of payments. Both, vaults and operators are service providers from the perspective of Promise: (i) any agent can become a service provider by locking a certain amount of collateral, and (ii) agents can earn fees by providing their services. We show that with an increasing number of service providers, the amount of collateral each service provider has to lock, can be *lowered*. Moreover, we show that the implementation of Promise in an Ethereum Solidity smart contract only adds USD 0.01 for the user and USD 0.10 cost for the service provider.

Outline. We introduce the system model and assumptions in Section 2, followed by a description of Promise in Section 3. Next, we discuss the security of Promise in Section 4 and argue in which cases Promise can provide benefits to users and intermediaries in Section 5. Also, we present how Promise can be applied to existing systems in Section 6. We discuss related work in Section 7 and conclude in Section 8.

2 System Model

In Promise, a user Bob engages an intermediary Alice to fulfill a task valued at V_B on his behalf. Bob pays Alice p each period t for performing the task. Given the absence of *strong identities*, the total value of the task to Bob (V_B) needs to be fully collateralized, via a deposit D, such that $D \geq V_B$. For example, if a particular task involves Alice offering a service and Bob having a $100 exposure—in the form of counter-party risk—to Alice, Alice will need to post at least $100 as collateral to *insure* offset the exposure, such that Bob does not stand to lose funds if Alice behaves maliciously.

Formally, we adopt the definitions of agreements in cryptoeconomic protocols from [13]. The service providing agent Alice A and the receiving agent Bob B participate in an agreement encoded by:

$$\mathcal{A} = \langle \Phi, p, D \rangle \tag{1}$$

In such an agreement, Alice needs to fulfil the specification Φ and provide the collateral D in advance. When Alice fulfils the specification, the payment p held in escrow is released to Alice.

2.1 Specifications

The specification ϕ describes the *task* that Alice needs to provide and the *proof* that serves as evidence that the task has been provided. There are several approaches to encode the specification.

In the BitML calculus, a specification consists of (i) a model describing a contract and agent choices symbolically and (ii) a model encoding a sequence of transactions that form a smart contract in Bitcoin computationally. Essentially, one can encode a contract statement in the BitML calculus, use the BitML compiler to create a sequence of Bitcoin transactions that encode the contract, and utilize the transactions to prove the outcome of the contract [5]. For example, Alice can deliver a digital good to Bob in return for a payment. The contract would then specify that if Bob receives the good, payment to Alice is being made. In this simplified example, very little guarantees are being made between Alice and Bob. However, the BitML calculus would allow one to create more elaborate schemes including partial payments, a trusted mediator, and time constraints.

Specifications are also useful when exchanging digital goods with the FairSwap protocol [8]. In FairSwap, simply put, Alice sends a digital good to Bob

and proofs (partial) sending by providing a witness (hashes of the transferred data) to a smart contract as proof. The specification in FairSwap is encoded as a boolean circuit that evaluates whether the provided witness (the hash) satisfies the specification. In case the circuit evaluates to true, Alice is paid for delivering the data.

Expressing the specification abstractly gives us the freedom in Promise to leave the encoding and implementation up to the protocol that integrates with Promise.

2.2 Roles

Promise adopts the BAR model of rational agents [1] including private preferences of agents as proposed in [13]. We define the following roles.

- **Alice**, the Intermediary: Alice is economically rational and entrusted with executing a task. She provides a deposit D into the escrow before executing the task and receives a payment p upon successful completion.
- **Bob**, the User: Bob represents the user requesting execution of a task by Alice. A user provides payments $\{p_0, ..., p_m\}$ into the escrow.
- **Escrow**: The escrow is a smart contract responsible for holding deposits by Alice and payments by Bob.
- **Verifier**: The verifier detects malicious behavior of Alice. In practice, this role is fulfilled by a smart contract, a dedicated third party, or the user.

2.3 Protocols

Promise is a mechanism to reduce initial collateral locking. However, Promise is not meant as a stand-alone protocol, but rather, serves as a "plug-in" to existing cryptoeconomic protocols. Given a generic cryptoeconomic protocol π, we can apply Promise and write the protocol as π_{P}. We note that the agreement $\mathcal{A}$ is given by the generic protocol π. We assume that Alice and Bob into the agreement $\mathcal{A}$ and have agreed on the specification ϕ, payments p, and the deposit D.

2.4 Assumptions

The verifier in the system is able to detect any faults by Alice and is able to prove that Alice was at fault. This means, that the specification ϕ of the protocol π has some "proof". For example, this could be the hash input of a boolean circuit as in FairSwap, a transaction inclusion proof as required by XCLAIM, or fraud-proofs [2].

We further assume that the protocol utilizing Promise implements payments and deposits through a ledger that provides the functionalities as describes in, e.g. [11]. Also, there is a one to one mapping between the collateral and a user, i.e. the collateral of an intermediary is not split between multiple users. Last, agents in the system can be identified with their public/private key pair.

2.5 Utilities

In our model, we assume that agents are economically rational and self-interested. An agent will therefore decide on a course of action depending on the utility associated with those actions. We use a simplified model here, were the intermediary Alice can choose between two actions and Bob has no choice once he committed to the agreement $\mathcal{A}$.

Alice can either behave honestly or malicious, with the following pay-offs one period ahead. V_A denotes the numerical value that Alice attaches to being malicious, where $V_A \geq 0$. We chose to only include a valuation to the malicious side to Alice as Alice could e.g. be bribed to be malicious. This is a worst case assumption: Alice can only be influenced increasing her incentive to misbehave. While we could also include a positive valuation for honest behaviour, this wouldn't strengthen our security assumptions.

V_B denotes the numerical value that Bob attaches to receiving the service. Note that we assume *private information*, i.e. Bob does not know Alice's private valuation V_A and Alice does not know Bob's valuation of the service V_B.

Last, c denotes the cost of an individual transaction. $\mathrm{E}[r]D$ reflects the expected opportunity cost of locking the capital for one period.

$$u_A = \begin{cases} p - \mathrm{E}[r]D, & \text{if Alice is honest} \\ V_A - \mathrm{E}[r]D - D, & \text{if Alice is malicious} \end{cases} \tag{2}$$

$$u_B = \begin{cases} V_B - p - c, & \text{if Alice is honest} \\ D - V_B - c, & \text{if Alice is malicious} \end{cases} \tag{3}$$

Each round the game resets. Therefore, Alice will be honest iff $p > V_A - D$. Assuming that $V_B - p > 0$, otherwise Bob would not seek the service from Alice in the first place, he stands to gain utility if Alice behaves honestly.

2.6 Security

Following the rational agents assumptions and the utility definitions in Eq. (2) and (3), we define a secure cryptoeconomic protocol as follows.

Definition 1 (Security). *Assuming rational intermediary A with a private valuation V_A, a cryptoeconomic protocol π is secure if A's utility u_A for behaving honest is higher than her utility for being malicious, i.e. $p > V_A - D$.*

In the following, we introduce Promise in detail. We use Def. 1 to show that integrating Promise into a generic protocol π does not affect its security. The core proof is to show that π and π_{P} are equivalent with respect to their security.

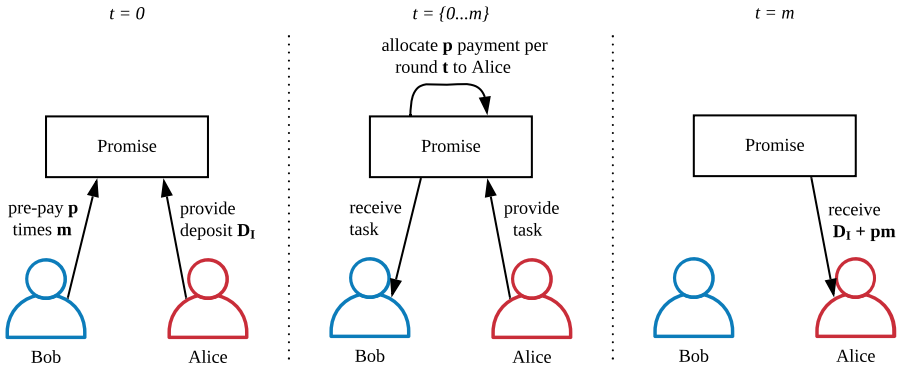

Fig. 1: Promise allows intermediaries (Alice) to lock less initial deposit D_I and use payments p_i provided by users (Bob) as additional deposit. The initial deposit and payments are locked for a period of time m determined by Bob. Only when Alice behaves honestly for the entire period m can Alice withdraw her initial deposit D_I and the payments pm.

3 Promise

In Promise we allow Bob to provide multiple payments in advance and delay the receipt of the payments to Alice. In turn, Alice is able to reduce the initially provided collateral from D to D_I such that $D_I < D$. At $t = 0$ Bob is able to lock m future payments $\{p_0, ..., p_m\}$ in escrow and determine a period τ after which Alice can receive the payments. When $t < \tau$, Alice continues to accumulate collateral as time passes by keeping the cumulative total of her payments p_i in escrow. We provide an intuition in Fig. 1. Promise has the following advantages for Alice and Bob.

Alice: the barrier to entry as an intermediary is lowered, as in the first period Alice only needs to provide a lower initial deposit D_I as opposed to D.

Bob: the aggregation of multiple payments allows Bob to reduce transaction costs and guarantees Bob that he only pays Alice if she fulfils all tasks for the given period m.

3.1 Protocol

The Promise protocol consists of three steps. We denote the intermediary as A, the user as B, and the smart contract implementing Promise as P. We assume that A and B have agreed the total payment and the period over which the payment is to-be-paid in advance.

1. At $t = 0$: B locks m times p payments in P. A locks the initial deposit D_I in P.

2. At $t = \{0...m\}$: A provides m times the agreed task to B. P allocates one payment p to A, if (i) A provides a proof to P that fulfils the specification ϕ, or (ii) B does not provide a fraud proof that A did not provide the task within a determined time [2].
3. At $t = m$: A withdraws pm and D_I from P.

To argue about the security of Promise, we introduce two concepts: (i) temporality and (ii) a likelihood of users exiting the system upon the counterparty cheating.

3.2 Introducing Time

We introduce temporality as follows. Denoting time by t, we define the period for which Bob locks Alice's payments as τ, such that if Bob makes a payment every period t then $\tau = m$. Let m denote the number of payments that Bob makes into escrow P.

Discounting is also introduced, where $0 < \delta < 1$ denotes the discount factor of an agent's valuation of future utility, i.e. the notion that the future is worth less than the present. We argue that an agent can spend received payments somewhere else or potentially invest the payment for a profit. The potential profit is encoded with $\mathrm{E}[r]$, as the expected rate of return. Hence, the agent faces an opportunity cost for delayed payments. The pay-offs to Alice, if she follows the same course of action over every round, are as follows.

$$u_A(t) = \begin{cases} \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (p - t\mathrm{E}[r]p - \mathrm{E}[r]D_I), & \text{if Alice is honest} \\ \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (V_A - \mathrm{E}[r]D_I - D_I), & \text{if Alice is malicious} \end{cases} \tag{4}$$

Bob receives the following pay-off, depending on Alice's behaviour.

$$u_B(t) = \begin{cases} \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (\frac{1}{m}V_B - p - (m-t)\mathrm{E}[r]p) - c, & \text{if Alice is honest} \\ \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (D_I - V_B - c), & \text{if Alice is malicious} \end{cases} \tag{5}$$

3.3 Termination Probability

Lowering Alice's initial collateral to D_I increases the risk of Alice cheating Bob. Specifically, in the first round, Alice's collateral is the lowest since she has not provided the service yet and has not added any payment into her collateral pool.

Eq. (2) and (4) implicitly assume that Alice and Bob are playing a one-shot game ad infinitum. We argue that Bob exits a protocol after being cheated too often, encoded in the function $\beta \rightarrow [0, 1)$ describing the likelihood that Bob remains in the protocol. Each time Bob gets cheated, n_c, the lower the probability that Bob continues to participate. Each user can have its own β function where users might choose to never participate again in a protocol after

being cheated once and others might tolerate a higher number of incidents. This changes Alice's pay-off for the protocol.

$$u_A(t) = \begin{cases} \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (p - t\mathrm{E}[r]p - \mathrm{E}[r]D_I), & \text{if Alice is honest} \\ \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t \beta(V_A - \mathrm{E}[r]D_I - D_I), & \text{if Alice is malicious} \end{cases} \quad (6)$$

As β decreases, the payoff to Alice converges to 0 as $n \to \infty$. For Alice, we increase the motivation to behave honestly by (i) providing a sum of payments mp that Bob locks in the protocol, and (ii) the fear of Bob leaving the protocol altogether if she acts malicious. As Bob chooses m, he has a direct influence on Alice's expected pay-off. By setting large m and being able to quit the protocol upon Alice's misbehaviour, he can motivate rational Alice to behave honestly. Overall, Alice can join protocols with less initial collateral while providing Bob with the option to use ongoing payments as an additional security measurement.

4 Security Analysis

Promise ensures that funds are secure. Payments p (future and current) provided by a user as well as deposits D by intermediaries cannot be stolen. This property is fulfilled by implementing the escrow as a smart contract on a ledger with a functionality and appropriate security parameters as described in [11,12]. Apart from this, Promise aims to fulfil the following properties:

1. **Security equivalence**: Given Def. 1, introducing Promise to a protocol π, does not increase the incentive for an intermediary A to behave malicious.
2. **Sybil resistance**: It is not individually rational for Alice to increase her pay-off u_A by creating multiple Sybil identities A' to increase.

4.1 Security equivalence

We argue that π_{P} and π are equivalent in terms of security considering an economically rational intermediary A.

Theorem 1 (Security equivalance). *Given an economically rational intermediary A with a private valuation V_A and an unknown termination probability β, a cryptoeconomic protocol with Promise, π_{P}, is secure if A's utility u_A for behaving honest is higher than her utility for being malicious.*

To prove Theorem 1, we consider both action choices of Alice. Given Eq. (6), Alice has different pay-offs depending on her being honest or malicious. We can calculate the decision bound for Alice's individually rational choice to cheat Bob by comparing the two options.

$$\sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (p - t\mathrm{E}[r]p - \mathrm{E}[r]D_I) = \sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t \beta(V_A - \mathrm{E}[r]D_I - D_I) \quad (7)$$

Using Eq. (7) above, we can determine a decision bound based on the β function of Alice to behave honestly. The β function, in turn, determines how large the initial deposit D_I needs to be to prevent Alice from cheating.

Proof. Assume A is individually rational, A will choose an action based on the highest pay-off. We re-arrange Eq. (7) for β:

$$\beta = \frac{\sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (p - t\mathrm{E}[r]p - \mathrm{E}[r]D_I)}{\sum_{t=0}^{m} \left(\frac{\delta}{1+r}\right)^t (V_A - \mathrm{E}[r]D_I - D_I)} \tag{8}$$

From Eq. (8), we can notice that the smaller β becomes, the smaller the pay-off for being malicious for A becomes. Hence, if $\beta \to 0$, A has no incentive to be malicious and should behave honest. However, if $\beta \to 1$, Alice's incentive to be malicious is largely determined by the $V_A - \mathrm{E}[r]D_I - D_I$ term. Further, we can clearly see that at $\beta = 1$, Eq. (8) and Eq. (4) are equivalent. From this follows that if $\beta \to 1$ then the initial deposit locked must also approach the deposit locked as without Promise, i.e. $D_I \to D$. As long as β and D_I are set such that the utility for A to behave honestly is greater, a protocol implying Promise is considered secure.

Of course, A has an interest to lower the initial deposit D_I, so it is hard for the user to determine if D_I is set adequately. In a practical setting, for example in payment channels or commit-chains, D_I can be set to exactly the value V_B that the user associates with the task. The additional payments can then be used to increase the amount of transactions that can be made through the payment channel or the amount of assets that can be finalized via the commit-chain.

4.2 Sybil Resistance

We argue that if β is lowered aggressively, Alice can do no better than being honest. However, if there is a non-negligible probability that Bob remains in the protocol after being cheated once, Alice can try to create a Sybil identity A' to (i) cheat Bob in round $t = 0$ and (ii) behave honestly in round 1 to $m+1$. Hence, Bob can only be sure to prevent Alice from executing a Sybil attack, if $\beta = 0$.

4.3 Beta Factor Over Time

The β factor set by Bob is the crucial security parameter for Bob given we are in a permissionless system. We argue that setting the β factor depends on Bob's reliance on participating in a protocol with a specific intermediary. If Bob *requires* the service Alice provides and there is *no alternative* within the same protocol in which Alice operates, Bob is faced with a *monopoly*. In a monopoly Bob β likely never approaches 0 as Bob is forced to tolerate Alice's misbehaviour. To protect Bob, Promise requires that D_I of Alice is close or equal to D in a monopoly.

The opposite market situation is *perfect competition*. Bob can choose from a wide selection of service providers to interact with[2]. Assuming perfect competition exists, Bob has the choice to pick another intermediary any time. As an example, we could imagine Bob choosing between several operators in NOCUST or vaults in XCLAIM. If Bob would be cheated on by an intermediary in NOCUST, he could switch to using another ooperator. In turn, Bob can set β to 0. In these cases, we can substantially lower the initial collateral D_I using Eq. (8).

5 Economic Benefits

1. **Cost reduction for users**: Bob is able to reduce transaction costs c by paying for multiple rounds m of services for payment p in advance.
2. **Collateral reduction for intermediaries**: Alice is able to provide a lower deposit D_I than the deposit required in a single-shot game setting D while Bob enjoys the same level of security against Alice.

5.1 Cost Reduction for Users

Assume that Alice behaves honestly. If a user pays every round t for the service provided by Alice, then his pay-off per round is $V_B - p - c$ as described in Eq. (3). However, locking multiple payments incurs opportunity cost. This cost is lowered at every time step as the payments are assigned to the intermediary, as expressed in Eq. (5).

Bob starts with an opportunity cost of $\mathrm{E}[r]pm$ at $t = 0$. The opportunity cost is reduced to $\mathrm{E}[r]p(m-1)$ at $t = 1$ as the payment is allocated to Alice. Generalizing this for t rounds, leaves us with $\mathrm{E}[r]p(m-t)$ at every time step t from today's perspective.

The user locks future payments when the sum of the transaction costs c for m payments is greater than the opportunity cost for locking additional payments plus the single transaction cost for making the prepayments. Hence, the boundary for a user to choose Promise as individually rational choice maximizing his pay-off is given by:

$$\sum_{t=1}^{m} \big(\frac{\delta}{1+r}\big)^t c = \sum_{t=0}^{m} \big(\frac{\delta}{1+r}\big)^t \mathrm{E}[r]p(m-t) \tag{9}$$

$$c = \mathrm{E}[r]p(m-t) \tag{10}$$

Provided the right hand of Eq. (10) is smaller, Bob should use Promise to lock multiple payments pm as it is his individually rational choice that maximizes his pay-off u_B.

[2] However, as we lack strong identities, these service providers could all be Sybil identities of Alice. Recognizing that perfect competition exists is a challenging task and we leave this as future work.

6 Applications

In this section we apply Promise to the NOCUST and XCLAIM protocols. We show analytically how Promise is able to reduce the initial amount of locked deposits.

6.1 NOCUST

NOCUST is a second-layer payment protocol whereby an untrusted intermediary operates a commit-chain to facilitate payments between its users [14]. We consider a scenario where Alice is the intermediary commit-chain operator, and Bob is a payment recipient. In this setting we propose to employ Promise as follows: Any fee to be paid by Bob to Alice in exchange for the delivery of an incoming payment would be locked as collateral that Bob could claim if the NOCUST protocol fails. Over time, the fees locked in Promise would grant Bob instant finality over larger payments, increasing the utility of the service.

6.2 XCLAIM

XCLAIM is a protocol that allows users to transfer assets between heterogeneous decentralized ledgers using a collateralized intermediary [20]. Instead of relying on a trusted third party like a centralized exchange, the intermediary must provide a collateral to ensure that she does not steal the coins she holds in custody. She has to verify correctness of her actions by submitting transaction inclusion proofs to the smart contracts that augments the protocol. Promise can be applied such that Alice locks some initial collateral D_I and issues backed-tokens using this collateral. Bob, using her services, is able to lock the future payments of Alice to allow him to transfer more assets between the ledgers.

6.3 Implementation

We implement Promise in Solidity in around 100 LoC. We use the implementation to experimentally assess the cost of executing the contract functions. Our cost calculations are summarised in Table 1 based on an Ether exchange rate of USD 172.61 and 1.5 Gwei gas price. The implementation is available as an open source project[3].

7 Related Work

There are two strands of related literature. The first one comes from the financial world covering (advance) payments for financial contracts. The second strand comes from the more recent work in decentralized ledgers. In the economics literature, a wide range of work focuses on secured debt, such as [16,17].

[3] `https://github.com/nud3l/Promise/tree/master/src`

Table 1: Overview of Promise functions and their cost.

Function	Description	Gas cost	Cost
create	Setup function.	112196	USD 0.02895
deposit	Called by intermediary to provide deposit.	43291	USD 0.01116
payment	Called by user to provide pre-payment.	43770	USD 0.0113
deliver	Called as part of task provision.	50703	USD 0.01309
withdraw	Called by intermediary after the service period is up to receive payment and deposit.	31788	USD 0.0082

However, these concepts rely on trust on third parties to maintain security in the debt and payment positions. Promise replaces this third-party trust by holding advance payments in a smart contract escrow.

On the second strand, Balance is a protocol that allows intermediaries to lower their collateral over time [13]. It operates at the other end of Promise: instead of lowering the initial collateral, the more an agent behaves honestly, the higher the reduction of collateral. Balance requires the highest collateral to be provided at the start of the interaction between agents and makes the assumption that payments are close to 0 (i.e. there is perfect competition). Promise and Balance can be combined together to first reduce initial collateral when bootstrapping a new protocol and then lower collateral requirements for established agents over time. Teutsch et al. discuss bootstrapping a token for verifiable computations [18]. This work discusses how to enable users, like Bob, to obtain the required funds to participate in TrueBit. Their proposal includes a governance game that allows to exchange special governance tokens into collateral tokens (for intermediaries) and utility tokens (for users). Last, the idea of bundling payments together is also introduced in [6] to create subscriptions for services of agents. Promise extends this idea to allow collateral reduction for intermediaries.

8 Conclusion

We present Promise, a system that allows intermediaries to lower the initially locked collateral. The core assumption for the security of Promise is that a user Bob is able to lock a number of payments up front and exit the protocol when Alice cheats. This allows Alice to reduce her initial collateral requirement, allowing protocols that adopt Promise to lower the burden on intermediaries. Bob is able to reduce his transaction cost as he transfers a sum of payments. Further, Bob is able to receive the service in full — for the entire period he pre-paid — or he is refunded the entire sum of payments. We have introduced a semi-formal model for Promise. We discuss the security and the effect of the β parameter, but leave formal proofs of the security properties as future work.

References

1. Aiyer, A.S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.P., Porth, C.: Bar fault tolerance for cooperative services. In: ACM SIGOPS operating systems review. vol. 39, pp. 45–58. ACM (2005)
2. Al-Bassam, M., Sonnino, A., Buterin, V.: Fraud Proofs: Maximising Light Client Security and Scaling Blockchains with Dishonest Majorities (2018), http://arxiv.org/abs/1809.09044
3. Avarikioti, G., Kogias, E.K., Wattenhofer, R.: Brick: Asynchronous state channels. arXiv preprint arXiv:1905.11360 (2019)
4. Avarikioti, G., Laufenberg, F., Sliwinski, J., Wang, Y., Wattenhofer, R.: Towards secure and efficient payment channels. arXiv preprint arXiv:1811.12740 (2018)
5. Bartoletti, M., Zunino, R.: BitML : A Calculus for Bitcoin Smart Contracts. CCS '18 Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security pp. 83–100 (2018). https://doi.org/10.1145/3243734.3243795
6. Berg, P.R.: ERC-1620: Money Streaming (2018), https://github.com/ethereum/EIPs/issues/1620
7. Böhme, R.: A primer on economics for cryptocurrencies (2019)
8. Dziembowski, S., Eckey, L., Faust, S.: FairSwap: How To Fairly Exchange Digital Goods. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18. pp. 967–984. ACM Press, New York, New York, USA (2018). https://doi.org/10.1145/3243734.3243857, http://dl.acm.org/citation.cfm?doid=3243734.3243857
9. Dziembowski, S., Faust, S., Hostáková, K.: General state channel networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 949–966. ACM (2018)
10. Fanti, G., Kogan, L., Oh, S., Ruan, K., Viswanath, P., Wang, G.: Compounding of Wealth in Proof-of-Stake Cryptocurrencies. In: Financial Cryptography and Data Security 2019 (2019)
11. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol with chains of variable difficulty. http://eprint.iacr.org/2016/1048.pdf (2016), accessed: 2017-02-06
12. Gervais, A., Karame, G.O., Wüst, K., Glykantzis, V., Ritzdorf, H., Capkun, S.: On the security and performance of proof of work blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 3–16. ACM (2016)
13. Harz, D., Gudgeon, L., Gervais, A., Knottenbelt, W.J.: Balance: Dynamic Adjustment of Cryptocurrency Deposits. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19). ACM, New York, NY, USA (2019), https://eprint.iacr.org/2019/675.pdf
14. Khalil, R., Gervais, A., Felley, G.: NOCUST - A Securely Scalable Commit-Chain (2019), https://eprint.iacr.org/2018/642
15. McCorry, P., Bakshi, S., Bentov, I., Miller, A., Meiklejohn, S.: Pisa: Arbitration outsourcing for state channels. IACR Cryptology ePrint Archive **2018**, 582 (2018)
16. Scott Jr, J.H.: Bankruptcy, secured debt, and optimal capital structure. The Journal of Finance **32**(1), 1–19 (1977)
17. Stulz, R., Johnson, H.: An analysis of secured debt. Journal of Financial Economics **14**(4), 501–521 (1985)
18. Teutsch, J., Mäkelä, S., Bakshi, S.: Bootstrapping a stable computation token (2019), http://arxiv.org/abs/1908.02946

19. Teutsch, J., Reitwießner, C.: A scalable verification solution for blockchains. https://truebit.io/ (March 2017), accessed:2017-10-06
20. Zamyatin, A., Harz, D., Lind, J., Panayiotou, P., Gervais, A., Knottenbelt, W.J.: XCLAIM: Trustless, Interoperable, Cryptocurrency-Backed Assets. In: Proceedings of the IEEE Symposium on Security & Privacy, May 2019. pp. 1254–1271 (2019), https://eprint.iacr.org/2018/643.pdf