Towards Micropayment for Intermediary based Trading

Anupa De Silva, Subhasis Thakur, and John Breslin

Abstract Involving an intermediary between producer-consumer environments is a common practice that reduces the managerial overhead on both parties. However, this mediation has both pros and cons. For example, it can overtake the power of producers in pricing, which could cause unfair revenue distribution. We present a novel micropayment protocol called MARI, which enables verifiable pricing and facilitates payment delegation for the prod-con environment. The proposed protocol is a blockchain-based unidirectional off-chain payment mechanism using a novel Merkel tree-based smart contract. MARI allows producers to communicate the fee structure directly to their consumers. It grants total control over producers' inbound payment flow, which an intermediary mediates and preserves the low overhead on the producers and consumers at the same time. Further, we explore how MARI meets the general requirements as a payment protocol and its scalability.

1 Introduction

Intermediary involvement between producer and consumer is a common practice in the real world. It decouples the consumers of a service from its provider and assists in managerial tasks, including access management, communication, and quality of service [4]. Hence, producers can minimize the cost and administrative overhead,

Anupa De Silva
National University of Ireland, University Rd, Galway H91 TK33, Ireland, e-mail: anupa.shyamlal@insight-centre.org

Subhasis Thakur
National University of Ireland, University Rd, Galway H91 TK33, Ireland e-mail: subhasis.thakur@insight-centre.org

John Breslin
National University of Ireland, University Rd, Galway H91 TK33, Ireland e-mail: john.breslin@nuigalway.ie

allowing them to focus more on their core products. Furthermore, it is beneficial in providing recurring services where producers have to maintain persistent links with consumers. Also, resource-constrained producers cannot afford to manage a large number of consumers. For example, IoT-based systems are resource-constrained and produce data as a recurring service. Generally, they rely on a third party due to the lack of resources. Additionally, intermediaries can handle multiple such connections seamlessly and create highly scalable solutions.

However, having intermediaries is not always a desired feature in producer-consumer environments. The centralized nature of the intermediary role can cloud the views of both producer and consumer in terms of service and pricing [3]. As a result, consumers have to bear a higher cost for the services than the actual fee of the producer. Considering the pros and cons, we can realize that the presence of an intermediary is useful, but its position enables them to manipulate the link between producer and consumer.

This paper proposes the MARI payment protocol for a recurring producer-consumer service environment in which intermediaries perform managerial tasks. We deny the existence of an intermediary neither in service nor in payments. Hence, the intermediary also mediates in the compensation from consumer to producer in the same way. We facilitate payment delegation and define a verifiable mechanism to communicate the fee structure directly to the consumer. It makes sure that the producers get compensated through the delegation of intermediaries but without depending on it. First, we identify micropayment as a viable mechanism for recurring services. Then, we introduce a novel blockchain-based smart contract called Layered Multi-Lock Contract (LMLC), which can be released gradually by a third party who is not a contract participant. It allows making high-resolution micropayments with minimum on-chain cost. Then we present the flow of MARI followed by potential applications. Finally, we analyze and evaluate the performance, scalability and security aspects of the protocol. MARI demonstrates a strong form of authenticity where an entity can control an external payment flow towards itself with minimal additional overhead. Both consumers and producers can benefit from the transparency it creates. Further, it is scalable to facilitate multiple producers, consumers, and intermediaries with minimum additional cost.

2 Background and Related Works

2.1 Smart Contracts

Smart contracts are a digital form of real-life agreements created on top of blockchain technology. These pieces of software programs are subjected to be verified by the blockchain participants. The decentralized and distributed nature of blockchain makes them reliable with minimal trust on third parties [12].

MultiSig contract is one of the simplest yet powerful forms of contracts that can be implemented even using Turing incomplete scripts. They can enforce one or more participants to involve in spending. This nature enables MultiSig transactions to form offline payment channels [6] between two parties without making costly on-chain transactions for each payment. For example, when b needs to make periodic payment of v_δ to a, b initially funds an amount of v and publish a MultiSig transaction where both a and b has to sign to spend it. Afterwards, b keeps updating the balance and present the transaction to a (off the chain) as commitments. Once verified, a has the assurance of the payment although, it is not recorded on the chain as it is publishable at any time (i.e. if a dispute has occurred). There is no extra cost apart from the computation for signature verification and communication.

Apart from the signatures, we can design the MultiSig contract to integrate additional locks that need to be unlocked to move the funds. Time locks can mandate activation and expiration boundaries, whereas hash locks can be released by producing the pre-image of the hash value. The expiration time lock prevents funds to be locked forever because of an unresponsive counterparty in the contract. Activation time locks enable the transactions after a specific time, preventing old states from being published on the chain in offline payment channels. Hash locks are primarily valuable for the atomic swapping of assets between two parties [7]. It is the underlying technology used to create a network of payment channels [6] where payments are passed from one entity to another. The process is offline and atomic.

2.2 Micropayments

Trust is a fundamental requirement for trading unless transactions are atomic. In a transaction, whoever makes the first move (i.e. payment by the consumer or service by the producer) can lose if the counterparty avoids getting back. Either of the involved parties has to trust the other to get compensated or receive the service. If that is micropayment-enabled trading, the victim can minimize the loss by lowering the quantity at stake. Micropayment allows transactions in tiny amounts. Its resolution can vary depending on the application context and the nature of the payment system. Further, it implies the swiftness of making transactions and delivering service or product in exchange for the payment.

The concept of micropayment originated even before the World Wide Web (WWW). In 1960, Theodor Nelson [15] envisaged micropayments as a compensation mechanism for hypertext contents. Párhonyi [10] describes two generations of micropayment systems in the 1990s and 2000s. The first-generation micropayment emerged in the mid-'90s. However, it failed to provide a pragmatic solution for micropayment, although their innovative contribution is not trivial. The second generation started around 2000 with the advancement of internet technologies. Unlike the first generation, most of them survived for at least half a decade and even up until now. We argue that micropayment bounced back in its third generation with the emergence of Blockchain and, specifically, off-chain payment channels. Their decentralized nature eradicates the starting friction caused by the financial institutes and trust in third parties.

The contribution by Rivest et al. in cryptography-based micropayments protocols is notable. Payword [11] chain is one such innovation in commitment based micropayment protocols. A financial institute issues certificate for a PayWord chain, which is a chain of hashes. Then the recipient can sequentially release PayWords as payments. This method requires only one interaction with the broker during the payer-payee engagement, and it is an offline interaction. However, multiple Payword chains are needed for a client to engage with multiple merchants. Kim et al. [9] propose a method to mitigate this drawback where the client has to keep only one concatenated chain. In addition to the original system, the client has to embed a hash value given by the broker when making the payments. Therefore, the broker can keep track of multiple payments. However, the chain grows linearly along with new additions. Jutla et al. [8] propose an alternate solution using Merkle authentication trees, called PayTree.

In recent works, Zhi-Guo Wan et al. [14] developed MicroBTC based on Blockchain technology, inspired by PayWord. They modified the core Bitcoin implementation as the process demands programmable unspent outputs and loops. Galal et al. [5] proposed Merkel tree-based scheme, PayMerkel. Merkel tree root reduces the on-chain verification logarithmically. In essence, commitment-based schemes avoid costly signature verifications. It is particularly advantageous on Ethereum like blockchains that charge per operation. Otherwise, they make the payers restrain themselves to a particular frequency, whereas MultiSig based payment channel granularity is infinite. We exploit this limitation while designing the proposed protocol.

3 MARI Payment Protocol

MARI is a unidirectional payment protocol that supports payment from one entity to another through a third party. For example, a producer a is compensated by its service consumer c through an intermediary b. We call this setting a-b-c, and henceforth, we denote a producer, an intermediary, and a consumer by a, b, and c, respectively and superscript to denote a set of entities. For example, c-B-A denotes a consumer c connects with a set of intermediaries B to connect with a set of producers A. We first introduce a novel smart contract in subsection 3.1, followed by its usage, protocol flow and applications in subsections 3.2 and 3.3.

3.1 Layered Multi Lock Contract (LMLC)

Here, we present LMLC based on a modified Merkel Tree for the MARI payment protocol. The on-contract lock is LMLC's primary component, a Merkel Root encapsulating multiple locks from multiple parties. The complete Merkel Tree is a collection of subtrees and layered so that the below layer produces the leaves to the upper layer. For example, in the c-B-A setting, the on-contract lock is created at the top layer (θ), obtaining the locks from individual intermediaries as the leaves for the on-contract root. Similarly, the middle layer (β) consists of Merkel Trees based on the locks from the producers engaged with each intermediary. At the bottom layer (α), producers' hashed secrets act as the leaves to create producer-wise locks. We also embed different information, including monetary values and expiration times, alongside the locks using a prefix function at each layer. Further, we can independently release each lock at the α layer, making the on-contract lock openable multiple times. We define the LMLC and its components in detail as follows.

Definition 1 Let $\mathcal{L}$ be the recursive function which accepts n sized ordered lists of keys, K_n^λ to build the Merkel locks in each λ layer.

$$\mathcal{L}(K_n^\lambda) = \mathcal{F}^\lambda()\|\mathcal{H}(log(n)\|\mathcal{L}(K_n^\lambda[0:j])\|\mathcal{L}(K_n^\lambda[j:n])) \tag{1}$$

where $\mathcal{L}([k_i^\lambda]) = k_i^\lambda$ and j is the largest power of 2 less than n. $\mathcal{F}^\lambda$ is the prefix function and $\|$ denotes the concatenation. $K_n^\lambda[j_1 : j_2]$ indicates $[k_{j_1}^\lambda, k_{j_{1+1}}^\lambda, \ldots, k_{j_{2-1}}^\lambda]$ which is a sub-list of K_n^λ. $\mathcal{H}$ is a collision-resistant hash function.

Let $\mathcal{F}^\lambda$ be the prefix function that takes an ordered list of keys (i.e.leaves) K^λ at layer $\lambda \in \{\alpha, \beta, \theta\}$. When $\lambda = \alpha$, let $k_i^\alpha (\in K_n^\alpha) = v_i^\alpha \| e_i^\alpha \| \mathcal{H}(x_i^\alpha)$ where $v_i^\alpha \in V_n^\alpha$ is a monetary value, $x_i^\alpha \in X_n^\alpha$ is a random secret, and $e_i^\alpha \in E_n^\alpha$ indicates the expiration time. V_n^α and E_n^α are in ascending order. Then, $\mathcal{F}^\alpha(K_n^\alpha) \leftarrow v_{n-1}^\alpha \| e_{n-1}^\alpha$ (i.e. the last/maximum elements). The output is in the form of $v_n^\alpha \| e_n^\alpha \| l^\alpha$, where l^α is a hash value. We consider these outputs as the keys of the β layer (K^β). Then, we define $F^\beta \leftarrow \sum v_i^\beta$ (i.e. the sum of all the last/maximum values at α). When $\lambda = \theta$, we consider the outputs of the β layer along with the address of the respective intermediary (add_b) as the input leaves and $F^\theta \leftarrow \sum v_i^\theta$.

Definition 2 Let $\mathcal{W}$ be a recursion function that outputs the inclusion proof of i^{th} item inK_n^λ when $\mathcal{W}(0, [k_0]) = []$.

$$\mathcal{W}(i, K_n^\lambda) = \begin{cases} if\ i < j, \\ \mathcal{W}(i, K_n^\lambda[0:j]), [(\mathcal{F}^\lambda(K_n^\lambda), \mathcal{L}(K_n^\lambda[j:n]))] \\ \\ if\ i >= j, \\ \mathcal{W}(i-j, K_n^\lambda[j:n]), [(\mathcal{F}^\lambda(K_n^\lambda), \mathcal{L}(K_n^\lambda[0:j]))] \end{cases} \tag{2}$$

Definition 3 Let i be the position of $k_i^\lambda \in K_n^\lambda$ and W^λ the output of $\mathcal{W}^\lambda$. Then $\mathcal{V}$ is the inclusion verification function, defined in algorithm 1.

Definition 4 LMLC is a tuple, $(\tilde{l}, \tilde{v}, \tilde{e}, K^\lambda, \mathcal{F}^\lambda, \mathcal{L}, \mathcal{W}, \mathcal{V})$ where $\tilde{l}$ is the root lock which carries a total of $\tilde{v}$ and expires at $\tilde{e}$. It is created passing $K^\alpha, K^\beta, K^\theta$ to the function $\mathcal{L}$ in each layer separately.

Algorithm 1 Verification, $\mathcal{V}$

```
Input: $i, k_i^\lambda, W^\lambda$
Output: $l'$
Require: $i < n$
    $Init: l' \leftarrow k_i^\lambda$
    $Init: j \leftarrow 1$
1: for all $w_i \in W^\lambda$ do
2:     if $i$ is even then
3:         $l' \leftarrow w_i[0], \mathcal{H}(j \| l' \| w_i[1])$
4:     else
5:         $l' \leftarrow l'[0], \mathcal{H}(j \| w_i \| l')$
6:     end if
7:     $i = \lfloor i/2 \lceil$
8:     $j++$
9: end for
```

Algorithm 2 Claim (On-chain)

```
Require: $\tilde{l}, claimed$
Input: $W^\alpha, W^\beta, W^\theta, x^\alpha, v^\alpha, e^\alpha, i^\alpha, i^\beta, i^\theta$
Input: $sig, add_b$
Output: payment settlement or error
1: Claimable check $\triangleleft\, i^\beta$
2: Signature check $\triangleleft\, sig$
3: $(v^\beta \| e^\beta, h^\beta) \leftarrow$
4:     $\mathcal{V}(i^\alpha, v^\alpha \| e^\alpha \| \mathcal{H}(x^\alpha), W^\alpha)$
5: Expiration check $\triangleleft\, e^\beta$
6: $(v^\theta, h^\theta) \leftarrow \mathcal{V}(i^\beta, v^\beta \| \mathcal{H}(h^\beta), W^\beta)$
7: $\tilde{l}' \leftarrow \mathcal{V}(i^\theta, v_\theta \| \mathcal{H}(add_b \| h_\theta), W^\theta)$
8: if $\tilde{l} == \tilde{l}'$ then
9:     $claimed \Leftarrow (i^\alpha, v_i^\alpha)$
10:    transfer $v_i^\alpha$ to $add_b$
11: end if
```

3.2 Design

Here we present the usage of LMLC in an A-B-C environment in different stages. They include initialization, verification, service, and closing.

Initially, producers connect with intermediaries, ideally with a MultiSig based payment channel. Producers generate an ordered list of random secrets (later take the hash value) for each segment of their services, along with the expiration time and monetary value. Altogether, they act as the producer's fee structure and the keys (or leaves) for the LMLC at layer α. The consumer picks them chronologically and builds a lock using $\mathcal{L}$, and it is verifiable (using $\mathcal{V}$) individually without revealing the producer secrets. There can be multiple such locks from different producers, managed by different intermediaries. Combining them layer by layer according to LMLC creates the root lock $\tilde{l}$, the overall expiry time $\tilde{e}$ and the total monetary value, $\tilde{v}$. Consumer place on the blockchain contract with funding $\tilde{v}$.

Both producers and intermediaries individually verify, 1) the existence of the contract, 2) the lock within (given the witnesses $\mathcal{W}^\beta$ and $\mathcal{W}^\theta$ by the consumer) using $\mathcal{V}$ which can verify inclusion of k_i in the lock l when $l^\lambda == \mathcal{V}(i, k_i^\lambda, W^\lambda)$, 3) adequate $\tilde{v}$, and 4) $\tilde{e}$ is after the individual expiration. Successful verification indicates the consumer's willingness to pay intermediaries according to the respective producers' fee structure. However, they are not claimable by the intermediaries by themselves. The producers can also determine the number of consumers and their identities. It allows a to predetermine the charge which is to be claimed from its intermediary.

The service can begin after the verification. To compensate, c presents an off-chain commitment to the respective b in the form of a signed key $k_i \in K^\alpha$, that holds a value of v_i. b can claim it only if b is aware of x_i, the secret of k_i which belongs to a. Hence, b requests x_i from a in the form of an offline payment with the same lock where a has to reveal x_i to claim the payment, making the exchange atomic. Here, the intermediary can also include its brokerage fee. Similarly, b can handle multiple consumers and compensate a in bulk. Note that there is no direct interaction with the blockchain here. The signature verification (SV_o) assures the payment receipt.

Algorithm 2 gives the on-chain operation for an intermediary to claim a value. b should present the key with the secret along with its inclusion witness. The algorithm checks the inclusion level by level, and if successful, the respective amount is transferred to the given address (denoted by PT_c). The b's address (add_b) is also part of the verification. The contract allows several unlocks on different occasions by different parties. To avoid double-spending, we blacklist the already claimed positions at α layer along with their value. Contract owner can claim unspent amount after $\tilde{e}$ time.

Lemma 1 *Verification fails, provided* $\mathcal{V}(i', v_i^\alpha \| e_i^\alpha \| \mathcal{H}(x_i^\alpha), W^\alpha)$ *where* $i \neq i'$*.*

Proof First, we prove that the key position, i in the Algorithm 1 creates a unique sequence of odd and even that follows the binary sequence of value i. According to $\mathcal{V}$ and its notation, $i_j = 2 * i_{j+1} + r_j$ where r_j determines both the parity and j^{th} position of binary representation of the initial i (say i_1) such that $i_1 = 2^{n-1}r_{n-1} + 2^{n-2}r_{n-2}, + \cdots + 2r_1 + r_0$ (by basis representation theorem [1] and note that $|W| = log(n)$ and $i_1 < n$). The series, $\forall j\ r_j$, is the odd-even pattern the determines the control flow of the algorithm $\mathcal{V}$. From the same basis representation theorem, base two representation is unique. Therefore, we can derive that there exists a unique flow in $\mathcal{V}$, given i. i' gives a different flow and fails the verification because $\mathcal{H}$ receives a different set of inputs. □

3.3 Application

Our focus is on the compensation scenarios where an intermediary mediates between two parties in a continuous fashion. Such contexts are prevalent in the real world, although they were not distinguishably identified as such. For example, when a web platform mediates publishing videos for viewers, the producer becomes the publisher, consumers are viewers, and the platform becomes the intermediary. Here, the publisher's gain and platform revenue are generally decoupled and undisclosed. Another example is the intermediary involvement in data service domains due to the limited resources of data producers. However, it is not limited to digital services. For example, even a seamstress is a partial producer of a branded costume where the brand acts as an intermediary. MARI is applicable in these environments to improve the transparency of monetisation. The only limitation is that it is not pragmatic to apply it to services with no or low cost of reproduction without the producer's consent (e.g., videos in the first example).

MARI needs a blockchain with Turing complete scripting capability to implement the contract, defined in the algorithm 2 as $\mathcal{V}$ consists of a loop. Hence, we used Ethereum to build the proof of concept of the protocol, which can be found here[1].

[1] https://github.com/anupasm/MARI

Table 1 Producer's perspective

	Setting I: a-C		Setting II: a-b-C		
	a	C	a	b	C
PT_c	$\|C\|$	$2\|C\|$	1	$\|C\|+1$	$2\|C\|$
DS_c	-	$3\|C\|$	-	-	$3\|C\|$
SV_c	-	-	1	$\|C\|$	-
HG_c	$log(\eta)\|C\|$	-	1	$log(\eta)\|C\|$	-
PV_o	$\|C\|$	-	$1+\|C\|$	$\|C\|$	-
SV_o	-	-	η	$\eta\|C\|$	-
SG_o	-	-	-	η	$\eta\|C\|$
DS_o	$\eta\|C\|$	-	η	-	-
HG_o	$2\eta\|C\|$	$\eta\|C\|$	2η	η	$\eta\|C\|$

However, the MultiSig contract between producer and intermediary is not necessarily on the same blockchain. Therefore, it is an added advantage to integrate entities on different payment services. The rest of the functions are offline.

4 Analysis

Here, we analyse the amount of work required by each entity involved in the protocol from the positions of producers and consumers. The goal is to evaluate the overhead on a single entity. We examined it in different settings as well. In the first setting, there is a peer to peer engagement with a set of entities. It is analogous to the setting described in [5] when a payer pays multiple payees and vice versa. In the second setting, there is an intermediary in-between, as we described in the MARI protocol. Selected actions are high level and require notable effort both on-chain (c) and off-chain (o). They are, payment verification[2] (PV), signature verification[3] (SV) , hash generation (HG), signature generation (SG), payment transfer [4] (PT), and data storage[5] (DS).

We assume payment resolution per cycle(denoted by η) is the same for all producers and consumers for simplicity. Further, we consider the worst-case where all sessions are consumed per cycle. Summary of the analysis is tabled in Table 1 and 2.

[2] Contract verification process mentioned in subsection 3.2.

[3] Off-chain signature verification of commitment transactions and on contract signature verification in line 2 of algorithm 2.

[4] On-chain monetary value transfer (i.e. initial funds transfer, claim funds by payee, or claim balance by payer).

[5] Storage for secret values.

Table 2 Consumer's perspective

	Setting I: c-A		Setting II: c-B-A		
	c	A	c	B	A
PT_c	$2\|A\|$	$\|A\|$	2	$\|A\| + \|B\|$	$\|A\|$
DS_c	$3\|A\|$	-	3	-	-
SV_c	-	-	-	$\|A\|$	$\|A\|$
HG_c	-	$log(\eta)\|A\|$	-	$log(\eta)\|A\|$	$\|A\|$
PV_o	-	$\|A\|$	-	$\|B\|$	$\|A\| + \|B\|$
SV_o	-	-	-	$\eta\|A\|$	$\eta\|A\|$
SG_o	-	-	$\eta\|A\|$	$\eta\|A\|$	-
DS_o	-	$\eta\|A\|$	-	-	$\eta\|A\|$
HG_o	$\eta\|A\|$	$2\eta\|A\|$	$\eta\|A\|$	$\eta\|A\|$	$2\eta\|A\|$

5 Evaluation

MARI provides payment delegation, low overhead, verifiable pricing, and scalability. It inherits several security features by leveraging blockchain technology as an underline layer. However, they consist of both desirable and undesirable points. Here, we evaluate how the proposed protocol achieves those features and satisfies the requirements of a state of the art payment protocol [2].

In general, blockchain-based payment systems provide pseudo-anonymity for the participants. However, transaction details, including monetary values and on-chain data, are publicly available for anyone to verify the details. From the perspective of providers and intermediaries, this can be beneficial to build up their reputation. However, consumers may concern about disclosing their consumed services even though pseudo-anonymity prevents revealing their true identity. In terms of the proposed LMLC, the map between consumer and service is not placed on-chain, and it is not exposed until the intermediary claims the respective values.

$\mathcal{H}$ is assumed to be a collision-resistant hash function. Hence, it can be proved that the Merkel tree of LMLC is also collision-resistant. Furthermore, we embed a level number inside hashing which prevents a second preimage attack. Therefore, given the key, an adversary cannot find its secret on its own. MARI payments are unidirectional, and both payment recipient has no incentive to publish old states whereas payment senders are bounded by expiration time. However, there is an opportunity for a double-spending in extended LMLC that allow multiple withdrawals. For example, suppose an adversary can provide a substitute number for the key position. In that case, the contract appends an invalid position to the claimed list, and the adversary can withdraw the amount multiple times. However, we prove that this is not possible in the LMLC in lemma 1. Considering both facts, we can conclude that MARI preserves the integrity of the payments.

Distributed nature of blockchain-based systems provides reliability and availability for a system based on this protocol. One drawback of the system is that unresponsive clients of b can incur a loss. Although it is unavoidable without additional precautions, we argue that the loss is minimum in micropayment systems.

Fee structure is clearly stated for the agreed period and verifiable in MARI. It avoids the micropayment's mental cost as raised in [13]. In the event of a change of mind, incomplete or inaccurate decision to choose the service, a consumer can withdraw from the contract, which basically stops paying. It does not affect the compensation of the service provided by the producer thus far.

Producer's perspective in Table 1 demonstrates that the overhead on a has primarily moved to the intermediary with compared to the peer to peer engagement in on-chain operations. However, the signature verification incurs an additional cost. In the first setting, payments are solely based on hashing [5]. However, MARI occupies signatures and provides additional security for the payments compared to the hashes. Further, off-chain operations of a in the second setting are independent of the number of consumers, implying the ability to handle a higher number of consumers with static operations. Although it increases the overhead on the intermediary, off-chain operations do not cause a direct fee. Besides, intermediaries are supposed to be resource-rich entities. According to the consumer's perspective in Table 2, the on-chain operations of consumers is static. It means that consumers can engage any counterparty without additional on-chain costs. In general, out of all on-chain operations, the resolution value (η) is involved logarithmically, and it is only in hash verification. Therefore, Merkel tree-based micropayments can grow exponentially. It does not inflate the local storage as space can be recycled with new items.

Here, we can conclude that MARI largely preserves the low overhead on producers and the total on-chain cost compared to the first setting. As a result, off-chain operations are higher but primarily on intermediaries. Further, it can scale at a fixed cost for a consumer.

6 Conclusion

This paper presents MARI payment protocol, a novel approach to implement micropayment for intermediary-based trading. We create a novel blockchain-based smart contract where multiple parties can open the same lock multiple times and claim different monetary values, yet protected from double-spending. MARI facilitates producers to directly communicate the pricing structure to their consumers and assure the compensation even in the presence of an intermediary. The analysis of required operations shows that MARI is scalable and reduces on-chain cost and overhead on both producer and consumer. Additionally, we evaluate the protocol in terms of the security requirements of general payment protocols. We intend to integrate this protocol with existing communication protocols and develop an eco-system with the ability to update the connections dynamically.

Acknowledgements This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) and the Department of Agriculture, Food and the

Marine on behalf of the Government of Ireland under Grant Number SFI 16/RC/3835 (VistaMilk), and also by a research grant from SFI under Grant Number SFI 12/RC/2289_P2 (Insight), with both grants co-funded by the European Regional Development Fund.

References

1. Andrews, G.E.: Number theory. Courier Corporation (1994)
2. Asokan, N., Janson, P., Steiner, M., Waidner, M.: State of the art in electronic payment systems. Advances in Computers **53**, 425–449 (2000)
3. Bessy, C., Chauvin, P.M.: The power of market intermediaries: From information to valuation processes. Valuation studies **1**(1), 83–117 (2013)
4. Bhargava, H.K., Choudhary, V.: Economics of an information intermediary with aggregation benefits. Information systems research **15**(1), 22–36 (2004)
5. Galal, H.S., ElSheikh, M., Youssef, A.M.: An efficient micropayment channel on ethereum. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology, pp. 211–218. Springer (2019)
6. Gudgeon, L., Moreno-Sanchez, P., Roos, S., McCorry, P., Gervais, A.: Sok: Layer-two blockchain protocols. In: International Conference on Financial Cryptography and Data Security, pp. 201–226. Springer (2020)
7. Herlihy, M.: Atomic cross-chain swaps. In: Proceedings of the 2018 ACM symposium on principles of distributed computing, pp. 245–254 (2018)
8. Jutla, C.S., Yung, M.: Paytree:" amortized signature" for flexible micro-payments. IACR Cryptol. ePrint Arch. **2012**, 10 (2012)
9. Kim, S., Lee, W.: A pay word-based micropayment protocol supporting multiple payments. In: Proceedings. 12th International Conference on Computer Communications and Networks (IEEE Cat. No. 03EX712), pp. 609–612. IEEE (2003)
10. Párhonyi, R.: Handbook of financial cryptography and security, chap. Micropayment Systems. CRC Press (2010)
11. Rivest, R.L., Shamir, A.: Payword and micromint: Two simple micropayment schemes. In: International workshop on security protocols, pp. 69–87. Springer (1996)
12. Rouhani, S., Deters, R.: Security, performance, and applications of smart contracts: A systematic survey. IEEE Access **7**, 50759–50779 (2019)
13. Szabo, N.: Micropayments and mental transaction costs. In: 2nd Berlin Internet Economics Workshop, vol. 44, p. 44 (1999)
14. Wan, Z.G., Deng, R.H., Lee, D., Li, Y.: Microbtc: efficient, flexible and fair micropayment for bitcoin using hash chains. Journal of Computer Science and Technology **34**(2), 403–415 (2019)
15. Wikipedia contributors: Micropayment — Wikipedia, the free encyclopedia (2021). URL https://en.wikipedia.org/wiki/Micropayment. [Online; accessed 29-June-2021]