A cross-chain rating system: bridging EVM-based blockchains with Chainbridge

Andrea Lisi*†, Nunzio Lopardo*, Domenico Tortola*, Paolo Mori†, Laura Ricci*, and Fabio Severino‡
*University of Pisa, Pisa, Italy,
†Consiglio Nazionale delle Ricerche - IIT, Pisa, Italy,
‡Traent, Pisa, Italy
Emails: andrea.lisi@iit.cnr.it, n.lopardo@studenti.unipi.it, domenico.tortola@phd.unipi.it,
paolo.mori@iit.cnr.it, laura.ricci@unipi.it, fabio.severino@traent.com

2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS) | 979-8-3503-4647-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/COINS57856.2023.10189274

***Abstract*—Rating systems are services collecting users' opinions about any kind of item, such as movies, locations, or hobbies, typically characterized by centralized governance making them susceptible to censorship or rating manipulation. This paper leverages blockchain technology to build a decentralized system to secure the ratings and operations performed by the users.**

We assume a scenario where many rating systems coexist and are hosted by different blockchains, and these blockchains are connected with one that could handle monetary-like operations common to the rating systems. The paper describes an experiment with Chainbridge, a tool to perform transfers across two Ethereum-like blockchains, applied to a cross-chain rating system prototype involving two blockchains: one hosting the ratings and one handling the payments. The results include the costs in units of gas and USD, i.e. the blockchain fees, and the latency time to perform a transfer across two test networks, Goerli and Mumbai.

***Keywords*—Blockchain, Smart contract, Cross-chain, Chainbridge**

I. Introduction

Still today, blockchain technology attracts a lot of interest from both industry and research given its property to securely store transactions in a peer-to-peer fashion while guaranteeing determinism, i.e. the peers store the same transactions in the same order. Its first implementation, Bitcoin [1], built a decentralized currency secured by cryptography, thus called cryptocurrency, and it evolved with newer projects such as Ethereum [2] to support arbitrary computation with programs called smart contracts. Indeed, Ethereum is currently a very popular hosting platform for Decentralized Applications (DApps), applications based on a decentralized architecture running with a, possibly, decentralized governance thanks to the integration of tokens, with ERC-20 (fungible) and ERC-721 (non-fungible) being the most supported standards. Given the widespread of DApps across different blockchains and the willingness to exchange tokens or cryptocurrency, which builds Decentralized Finance ecosystems, designing and deploying interoperability mechanisms between blockchains is a new challenge that is tackled by many [3], [4].

We study the application of a cross-chain technology called Chainbridge, developed by Chainsafe, to a rating system DApp. A rating system stores reviews, or ratings, about several kinds of *items* [5], such as goods, locations, experiences, and much more, given by the users who consumed them. A rating system is also queried by its users to find the most suitable items for them. The design of the system is based on two distinct exchanges: *i)* a user pays for a service and receives in exchange the permission to review the service, *ii)* a user reviews the service in exchange for a reward. The permission to review bound to the payment ensures the customer cannot give a review without having had any interaction with the item, and the reward bound to the review should push customers to share their opinion. Tripadvisor, Yelp, IMDb, online shops, and social media posts, to name a few, are popular examples. To overcome the limitations due to the centralized nature of such systems, blockchain technology has been studied as an alternative hosting platform [6].

The use of a decentralized and transparent platform such as a blockchain should help make the reviews more trustable for the users since they cannot be modified or censored by the service provider [7]. Given the widespread use of blockchain platforms and the scattering of DApps, a rating system can be modularized so that each blockchain exposes a component, e.g. smart contracts and cryptocurrency, acting as a service. This approach suits better real world use cases than a monolithic deployment on a singular blockchain. In this paper we tackle the research challenge of designing such a cross-chain system, ensuring a trusted transfer of assets among the involved blockchains. The contributions are: *i)* the design of the cross-chain rating system whose transfers are coordinated following the Chainbridge protocol; *ii)* description of the key functions of the smart contracts triggering the cross-chain transfer process prototype; *iii)* a set of measurements conducted exploiting two test networks and an analysis.

The paper is structured as follows: Section II describes the basis of rating systems, blockchain, and cross-chain technologies; Section III lists the works related to this paper; Section IV describes the high-level design of the system; Section V provides the implementation details and their evaluation in terms of costs and latency; Section VI discusses the solution; Section VII concludes the paper.

II. Background

A. Rating systems

Rating systems collect ratings that reflect individuals' positive or negative opinions towards a platform's content, such

as a tourist location, a good, a place to sleep or to eat, or a service to a customer. As examples of rating systems, IMDb focuses on movies and TV series, Tripadvisor and Yelp on tourist locations and experiences, and RateMDs on medical professionals. The combination of all the ratings dictates the reputation of the content. We use the nomenclature of the model of Ricci et al. [5] concerning recommender systems, i.e. *users* interact with the system with *transactions* targeting different *items*. Transactions are the operations a user performs on the system, such as submitting a rating. A rating may be a numerical score (e.g. from 1 to 5), a binary expression such as *like* or *dislike*, or multiple scores each associated to a specific aspect of the item (e.g. quality, user-friendliness, clarity, etc). To be general, we use the term *review*. Users rate the items driven by a sense of community or by an incentive, e.g. rewards. A reward can be in form of reputation, to improve the status of an individual, reciprocity, to increase mutual trust, and monetization, as an economic reward [8].

B. Blockchain technologies

A blockchain is an append-only list of blocks maintained and upgraded by the nodes of a peer-to-peer network. A block contains data submitted with transactions and each block is hash-linked to the previous one. Each peer stores the entire blockchain and all the peers periodically select one peer to produce the new block following a consensus protocol. Joining the consensus can either be open, i.e. permissionless and public, or have restrictions, i.e. permissioned and (optionally) private [9]. A smart contract is a software program whose execution, performed by the nodes of a blockchain, is triggered by a transaction. Ethereum popularized smart contracts since they are, instead, general-purpose programs written in a "quasi"-Turing [2] complete language compiled in bytecode, and executed by the Ethereum Virtual Machine (EVM). A smart contract function is executed by the nodes of the blockchain, making contracts non-repudiable, immutable, and transparent software. Their behavior is then visible to anyone and unchangeable, making them good candidates as "notaries" between untrusted users although their execution typically requires the sender of a transaction to pay a fee.

C. Cross-chain and Chainbridge

A blockchain, thanks to cryptography and consensus, builds a trusted execution environment on top of a set of untrusted nodes [10]. However, since a blockchain alone may not satisfy all the requirements of Decentralized Applications (DApps), techniques and ideas of interoperability between blockchains emerged [3], [4], [11] aiming at connecting multiple different blockchains in order to benefit from the advantages brought by each of them and minimize the disadvantages. Blockchain interoperability can be implemented by cross-chain technologies involving either heterogeneous blockchains, such as Bitcoin and Ethereum, or homogeneous blockchains, i.e. working with the same protocol such as in Cosmos [12] or Polygon [13], with the latter being a project that creates EVM-based blockchains secured by Ethereum.

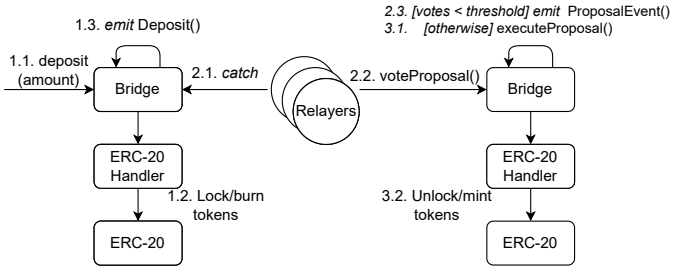

Fig. 1. An overview of Chainbridge.

In this paper, we focus on Chainbridge [14], a tool developed to transfer ERC-20 tokens, ERC-721 tokens, and arbitrary data across two EVM-based blockchains: from a source chain to a destination chain. Chainbridge involves one or more *Relayers*, off-chain nodes that carry out the cross-chain transaction interacting with `Bridge` smart contracts deployed on each blockchain. A Relayer has the following modules: the listener module observes events in the source chain; the router module processes the event input, e.g. what to transfer and where (which destination chain); the write module signs the transaction on the destination chain. To reduce the centralization factor, Chainbridge supports more Relayers carrying out the same cross-chain transaction. The `Bridge` emits the events about new cross-chain transfers and exposes the functions to finalize them.

As illustrated in Figure 1, a cross-chain transfer is triggered by the invocation of the `deposit()` function exposed by the `Bridge` (step 1.1): the function emits a `Deposit()` event (note capital "D") to notify the Relayers, which listen for these events, about a new cross-chain transfer from the source blockchain to a destination blockchain. Chainbridge mainly provides functions to transfer ERC-20 and ERC-721 tokens connecting the `Bridge` to `Handler` smart contracts, adapters for ERC-20 and ERC-721 smart contracts. The default approach locks, or burns, tokens in the source blockchain and mints new ones in the destination blockchain. The first Relayer that catches a `Deposit()` event (step 2.1) triggers the transfer with a `voteProposal()` transaction on the destination `Bridge` (step 2.2), which emits `ProposalEvent()` for logging purposes. The Relayers keep submitting their vote until a threshold number of Relayers is reached: the `voteProposal()` execution reaching the threshold finalizes the transfer by invoking the `executeProposal()` function on the `Bridge` (step 2.3).

III. Related work

The design of decentralized rating systems is an active research topic to securely store ratings, reviews, or any kind of data then used to build recommendations (see Almasoud et al. for an in-depth review [6]). Shaker et al. [15] study a system storing ratings on a smart contract to grant them transparency, immutability, public verifiability (since transactions are signed), and privacy since a user's identity is not known (although they are pseudonymous and not anonymous).

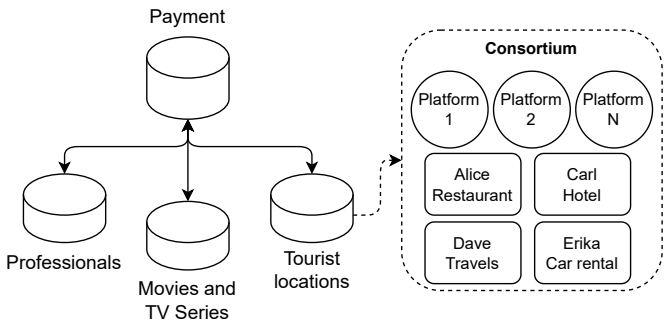

Fig. 2. High-level model of a cross-chain rating system. Each cylinder represents a blockchain with the "Tourist location" highlighted to show an example of participants in that consortium.

Salah et al. [16], instead, propose to store the ratings in the InterPlanetary File System (IPFS) and use a smart contract to reward reviewers with tokens. Other works study the feasibility of supporting recommendation algorithms, however since such algorithms are complex the authors focus on permissioned blockchains [17]–[19]: scalability, either in terms of computational capacity and transaction throughput, is generally higher and the costs lower in permissioned blockchains than permissionless blockchains due to their more restricted environment.

Given the new research directions towards cross-chain technologies aiming at finding a trade-off of the advantages by connecting existing blockchains, this paper aims at designing a decentralized rating system involving two blockchains. The atomic cross-chain swap [20] is a popular technique based on Hash Time Locked Contracts (HTLCs) to exchange value between two different blockchains, e.g. bitcoins for ether (ETH). This technique is purely decentralized because no other party besides the participants in the swap is involved, however, the blockchains must support the same hash functions and it requires the participants to be online. Similarly, cross-chain deals [21] extends cross-chain swaps supporting the exchange of assets not yet owned by one of the parties, e.g. by a middleman buying a ticket from a seller and selling it to a customer: the middleman does not own the ticket when establishing the deals.

This paper builds on the idea of cross-chain rating systems expressed in [22], which identifies two pairs of operations in a decentralized rating system that could be connected with atomic swaps: the exchange of a payment for a permission to rate; the exchange of a rating for a reward. This paper explores an alternative approach based on bridging in order to automatize the cross-chain operations across the components of the rating system. As a consequence, a decentralized rating system can be modularized across various blockchains to find a trade-off between the advantages and disadvantages of each one, e.g. the scalability potential of permissioned blockchains and the security of permissionless ones.

IV. Cross-chain bridging rating system

Rating systems are characterized by the type of items they handle, they are typically managed by different service providers, and they could be deployed on various blockchains, depending on the preference of each provider. Let us assume that service providers, such as online platforms, run rating systems about similar items, such as tourist locations, and together with item owners, such as restaurant and hotel owners, form a *consortium* (see Figure 2). The members of a consortium participate to a blockchain to deploy their services, the rating system platforms and items, and they define the structures of the reviews and the algorithms to process the reviews to produce the final rating to display to the users navigating the system. Each participant in the consortium, e.g. the service providers but also the item owners, hosts one or more nodes to maintain a blockchain similar to a permissioned setting. Since permissionless and public blockchains that natively support cryptocurrency already exist and are highly secure, such as Bitcoin or Ethereum, each consortium could utilize them to carry on the payment-related operations instead of defining their own. This arises the need for a cross-chain protocol to form a topology as shown in Figure 2, where various consortia could share the same payment blockchain. Let us define the operation flow between the consortium and payment blockchains, inherited from [22], as follows:

1) A payment performed on the payment blockchain for a given item gives the permission to rate such item in the related consortium blockchain (called *Payment-Permission* transfer);
2) The rating of the item in a consortium blockchain unlocks a reward in the payment blockchain (called *Review-Reward* transfer).

Since both the previously described transfers involve two distinct blockchains, their execution requires a cross-chain technology. Inspired by this scenario, we designed a system connecting the blockchains with Chainbridge, therefore involving the `Bridge` and `Handler` contracts and the Relayers as illustrated in Figure 3. As described above, one of these blockchains is dedicated to processing payments (*Sales blockchain*), while the other one handles user's ratings (*Review blockchain*). In the following, we showcase our approach with a use case involving restaurants, but the approach is general purpose and directly applicable to any rating and payment systems that work similarly to how described. Although Chainbridge supports only the EVM, the approach and the functionalities also work with blockchains with capabilities equivalent to the EVM. We consider as example a restaurant as an item owned by Alice, one of its customers as a user named Bob, and a numerical rating in the range [1, 10] as a review. When Bob pays a bill in such a restaurant he receives a token enabling him to release a review about the restaurant. Users are incentivized to release reviews through a reward system: once a user releases a review, they will get a discount token that can be used to pay less the next bill. The Sales blockchain handles bill payments and discounts, while the Review blockchain manages permission tokens to review and it stores the reviews.

The high-level workflow of the Payment-Permission transfer

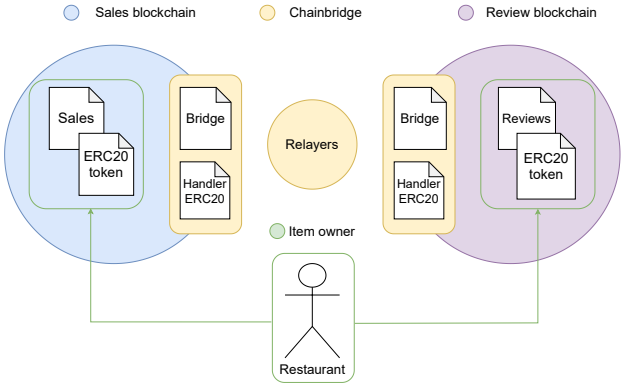

Fig. 3. Cross-chain rating system architecture.

with Chainbridge is the following:

1) Bob consumes a meal at Alice's restaurant. Alice sets Bob's bill in the Sales blockchain;
2) Bob pays Alice's bill with cryptocurrency, or tokens. This action triggers the `Bridge` contract that emits a `Deposit()` event to unlock a permission for Bob to rate Alice's restaurant in the Review blockchain;
3) The Relayers perform the cross-chain transfer invoking the `voteProposal()` function on the Review blockchain as described in Section II-C;
4) The `executeProposal()` in the Review blockchain unlocks Bob's permission to rate Alice's restaurant.

If Bob reviews Alice's restaurant, an equivalent sequence of operations, the Review-Reward transfer, will be carried out with the Review blockchain being the source blockchain and the Sales blockchain being the destination one. In this case, the Sales blockchain will reward Bob with a currency, for example tokens, that he can spend in the rating system (for example at Alice's restaurant next time).

V. Implementation

To validate our proposal, we developed a prototype to be deployed on Ethereum and Polygon networks bridged by Chainbridge. We used the *Polygon Edge v. 0.6.0* framework, which instantiates independent EVM-based blockchains, to set up two local blockchains for testing. We used *HardHat v. 1.3.3* to compile, deploy and test Ethereum software, alongside *Ether.js v. 5.7.2* to interact with the blockchains using JavaScript. We worked with Chainbridge Solidity v. 2.14 smart contracts[1] and Chainbridge-core[2] v. 3.0. The former repository also contains the OpenZeppelin implementation of ERC-20 contracts. The latter repository provides the executable files written in Go to run a Relayer node. Each Relayer is associated with an account to participate in all the connected networks. In the Relayer's local storage, it is necessary to create a keystore file *<public_address>.key* storing the private keys of the Relayer's accounts on the blockchains (obviously password protected in order to avoid impersonation of the Relayer). In addition, it is necessary to create a configuration file in JSON format by entering a list of objects representing blockchains, such as the Remote Procedure Call (RPC) endpoint or the address of the `Bridge` contract. To execute a Relayer node it is necessary to run the Go files via command line by passing the paths to the configuration and the keystore files as parameters. For the use case, we developed two different Solidity smart contracts, `Sales` and `Review`. As the naming suggests, the former contract handles the operations executed in the Sales blockchain, while the latter is dedicated to the review management in the Review blockchain. The source code is available on GitHub[3].

As shown in Figure 3, Chainbridge's contracts, `Bridge` and `Handler`, are deployed on both the blockchains only once since they are shared among different items, while each new restaurant requires its owner to deploy a `Sales`, a `Review`, and a `ERC-20` contract on each chain. To comply with the strategy used by Chainbridge to identify resources, a restaurant has unique 32 bytes long identifier named `resourceID`.

A. Use case contracts

Sales contract: the `Sales` contract stores the restaurant balance, the users' bills, and the respective payments. The `Sales` contract keeps two map data structures, `userBills` and `userDiscounts`, to keep track of users' bills and discounts: the prototype represents the reward as a discount granted by Alice's restaurant to Bob in a future purchase. The main functions exposed by the `Sales` contract, omitting the getters, are:

- `setUserBill(userAddress, bill)`: a function called by the restaurant owner, e.g. Alice, to set up the customer's bill. The function adds a new entry in the `userBills` map;
- `payBill()`: a customer, e.g. Bob, pays the bill using this function. The amount to pay is sent in cryptocurrency and retrieved by the contract with `msg.value` Solidity field;
- `setUserDiscount(userAddress, discount)`: this function is invoked by the `Bridge` contract when executing the `executeProposal()` (see Section II-C for the functions related to Chainbridge) at the end of the *Review-Reward* transfer, and adds a new entry to the `userDiscounts` map to create a new discount token for the customer `userAddress`.

Figure 4 illustrates the flow of function calls during the *Payment-Permission* transfer. The `setUserBill()` function is the first function to be called, where Alice sets Bob's bill. Assuming Bob has no discounts, Bob invokes the `payBill()` to pay for the bill (step 3.1): this is the entry point of the cross-chain transfer that invokes the `deposit()` function of the `Bridge`. The function checks if the customer

[1]https://github.com/ChainSafe/chainbridge-solidity/ [Accessed on 10 January 2023].

[2]https://github.com/ChainSafe/chainbridge-core/ [Accessed on 10 January 2023].

[3]https://github.com/gasparax/review-reward-chainbridge [Accessed on 10 January 2023]

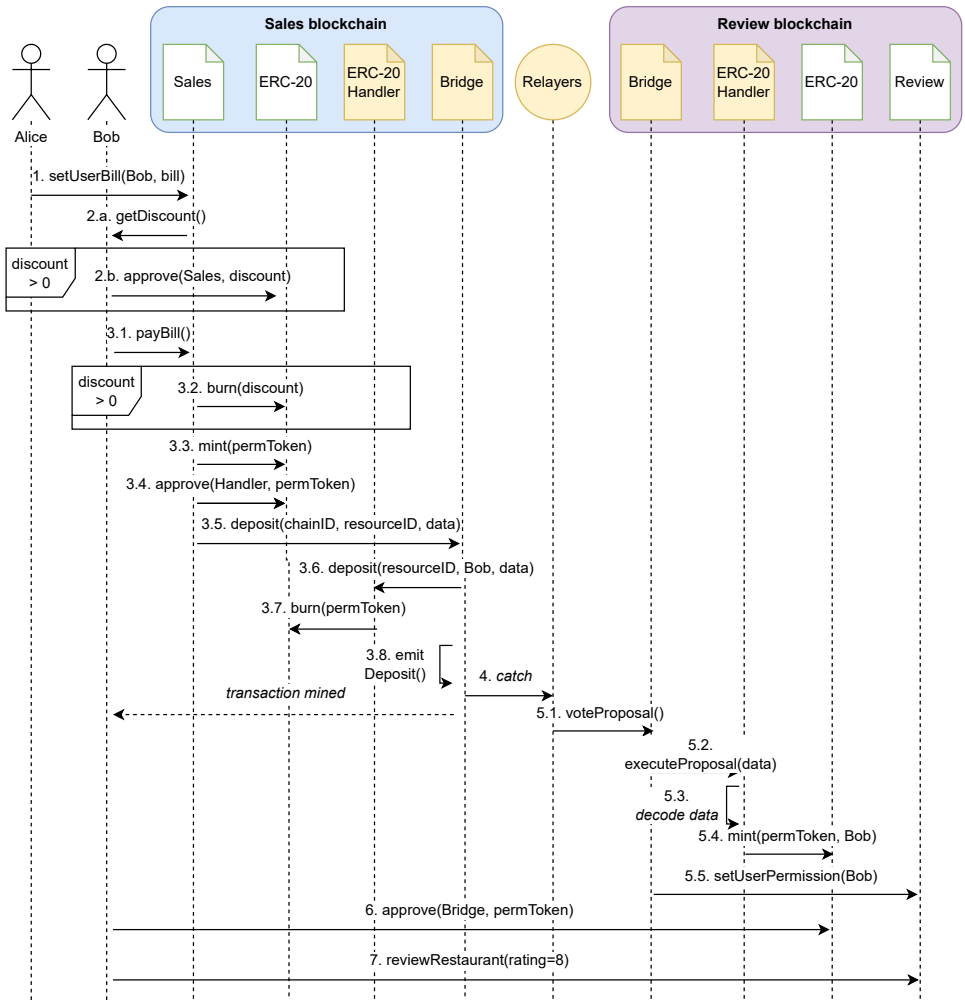

Fig. 4. Detailed diagram showing the *Payment-Permission* transfer and the beginning of the *Review-Reward* transfer (last two steps). The colors follow those of Figure 3.

meets some requirements, such as the amount sent by Bob (`msg.value`) matching the bill. If the requirements are met, the function mints a new permission token (step 3.3 in Figure 4), represented by 1 ERC-20 token, and gives to the *Handler* contract the permission to burn it (step 3.4). This complies with Chainbridge's default behavior of burning tokens in the source blockchain and minting them in the destination blockchain (see Figure 1). Then, a string containing a hexadecimal value, the address length, and the receiver's address on the Review blockchain is built. Finally, the `deposit()` function of the `Bridge` contract is called (step 3.5) passing the just built string, the Review blockchain identifier, and the resource identifier.

Assuming Bob accumulated a discount from a previous purchase, he can apply it by approving the `Sales` contract (step 2.b) to handle the ERC-20 tokens representing the discount on Bob's behalf. This approval is needed because the ERC-20 contract sees Bob as the owner of the tokens. Without approval to the `Sales` contract to grant it the permission to use the tokens, the ERC-20 contract would reject any usage request incoming from `Sales`, even if Bob issued the transaction to `Sales`. Indeed, if Bob approves `Sales`, the contract will request the tokens to be burned (step 3.2), and the bill will be reduced accordingly.

Review contract: the `Review` contract exposes all the functions necessary to manage the restaurant's rating, store customer reviews, and set the permissions to review. Similarly to the `Sales` contract, the `Review` contract keeps the *userPermission* and *userReviews* maps: the former keeps track of the users holding a permission to review while the latter stores the reviews (numerical value in the range [1, 10]). To keep the solution consistent, a permission token is represented as a single ERC-20 token. The main functions, omitting the getters, are:

- `setReward(amount)`: the restaurant owner, e.g. Alice, invokes this function to set the reward value, i.e. a discount equal to `amount`;
- `reviewRestaurant(rating)`: when called by the customer, e.g. Bob, the function checks the presence of a permission associated with the caller and stores the

`rating`;

- `setUserPermission(userAddress)`: this function is invoked by the `Bridge` contract when executing the `executeProposal()` at the end of the *Payment-Permission* transfer, and adds a new entry in the *userPermission* map to associate a review permission with the customer `userAddrress`.

Let us take again Figure 4 at the point the Relayers are finalizing the cross-chain transfer (step 5.1 onward). When enough Relayers voted for the transfer, the last vote triggers the `executeProposal()` function. As part of this invocation, the permission token, previously burned in the Sales blockchain, is minted and assigned to Bob (step 5.4) and the `setUserPermission()` (step 5.5) stores a flag in the *userPermission* map associated with Bob.

The `reviewRestaurant()` function plays a similar role as the *payBill()* function of the `Sales` contract being the entry point of the *Review-Reward* transfer invoking the `deposit()` function of the `Bridge`.

Similarly to the application of the discount, the `Review` contract cannot consume, burn, the permission token without the approval from the owner, i.e. the customer. Therefore, before reviewing a restaurant the customer needs to `approve()` the `Review` contract to burn the token (step 6). This approval adds an overhead, also in terms of user experience, but it is a required operation unless a change to the ERC-20 standard behavior is made. Finally, Bob invokes the `reviewRestaurant()` to rate the restaurant (say with a score of 8 as shown in Step 7), which follows a flow similar to Figure 4 with the exception that this transfer mints the tokens in the Sales blockchain representing the discount Bob could use in his next purchase.

B. Chainbridge contracts

As described in Section II-C, the `Bridge` contract handles the cross-chain messages and the communication with the `Handler` contract to mint or burn tokens. The `Bridge` has been extended to map restaurants, to their `Sales` and `Review` contract addresses. The `Handler` is linked both to the `Bridge` contract and to the `ERC-20` contract: the former link is needed in order to take part in the `Deposit()` event, while the latter link is necessary to handle the token changes, i.e. burning, locking, or minting tokens following the protocol's execution stage. In our use case, this feature of the `Handler` contract was used to ensure that each restaurant can only manage its own token (in terms of both review permissions and discounts).

C. Experiments

We executed some experiments on the prototype we developed according to the previous description. The experiments have been conducted on a virtual machine based on *Ubuntu 22.04*, with an *Intel™ core i7 8750H* CPU and 16 GB of RAM. We evaluated the system costs in terms of gas to understand the effective usability in a real-world scenario. The testing phase was divided into two parts: local tests and remote tests. In the local tests, we measured the costs in units of gas of the main functionalities, including the deployment of the contracts, in a local environment. In the remote tests, we deployed the contracts on two working testnet: *Goerli* (Ethereum testnet) and *Mumbai* (Polygon testnet).

1) Local tests (costs): An Ethereum transaction consumes an amount of *gas* units proportional to its complexity. Also, a caller provides a *gas price* as a fee to the block miners. This gas price is typically measured in gwei, with 1 gwei equal to 10^{-9} ETH in Ethereum or 10^{-9} MATIC in Polygon. The transaction costs are important to understand the real usability of the prototype. We evaluated the gas units needed to deploy and interact with the proposed system and estimated the related cost in USD according to the following parameters taken on 10 January 2023:

- Ethereum: gas price of 25 gwei and ETH:USD exchange rate of 1:1,335.42 (Source: Etherscan);
- Polygon: gas price of 98 gwei and MATIC:USD exchange rate of 1:0.86 (Source: Polygonscan).

TABLE I
SMART CONTRACTS DEPLOYMENT COSTS.

Contract	Gas	ETH	MATIC	USD (ETH)	USD (MATIC)
Bridge	5,396,151	0.135	0.529	180.153	0.455
Handler	1,370,145	0.034	0.134	45.743	0.115
Sales	1,345,570	0.034	0.132	44.932	0.113
Reviews	1,283,940	0.032	0.126	42.865	0.108
ERC-20	3,089,345	0.077	0.303	103.139	0.26

Table I shows the deployment cost for each smart contract in the architecture. The deployment of the `Bridge` and the `Handler` contracts requires the most units of gas, about 5M units for the `Bridge` contract, but recall they are shared among different items, and therefore their cost is paid only once per consortium. Instead, a new item requires the item owner to deploy the `Sales`, `Review`, and ERC-20 contracts. Assuming the exchange rates reported in the previous paragraph, the item owner would pay about 188.94 USD (about 0.14 ETH) in Ethereum and about 0.48 USD (about 0.56 MATIC) in Polygon. Recall that the aforementioned contracts need to be deployed in both the blockchains.

Table II shows the costs of the main functionalities and the caller who is going to pay for them among the restaurant owner, the customer, or the Relayers. The Relayers invoke the most expensive operation, the `voteProposal()`, which would cost about 4.90 USD in the Ethereum blockchain or 0.04 USD in the Polygon blockchain. The `reviewRestaurant()` is the most expensive operation for the user, which requires a fee of about 4.75 USD in the Ethereum or 0.01 USD Polygon blockchains respectively. The only fee required by the restaurant owner after the deployment is for setting the bill, and it equals 1.56 USD if executed on Ethereum or 0.004 USD in Polygon.

The two tables highlight significant differences between the costs on the Ethereum blockchain and on the Polygon blockchain. This difference comes from the fact that ratings

TABLE II
OPERATIONAL COSTS FOR USERS AND RELAYERS. (RO): RESTAURANT OWNER, (C): CUSTOMER, (R): RELAYER.

Operation	Gas	ETH	MATIC	USD (ETH)	USD (MATIC)
setBill (RO)	46,702	0.001	0.005	1.559	0.004
approve (C)	46,902	0.001	0.005	1.566	0.004
payBill (C)	138,344	0.003	0.014	4.619	0.012
review Restaurant (C)	142,393	0.004	0.014	4.754	0.012
approve (C)	46,902	0.001	0.014	4.898	0.012
vote Proposal (R)	146,697	0.013	0.051	17.395	0.044

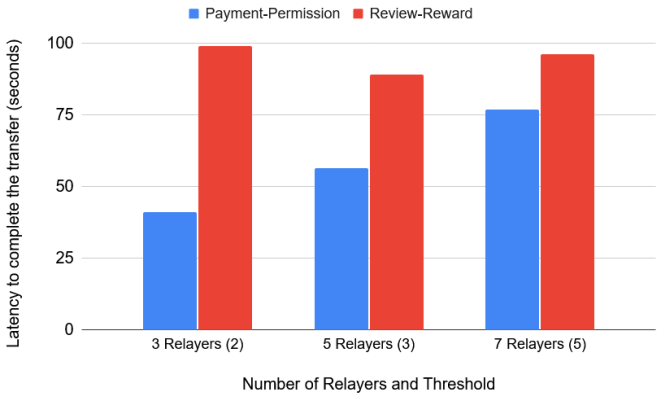

Fig. 5. Latency to complete a cross-chain transfer with 3, 5 and 7 Relayers.

and payments are secured by the respective blockchains and their security depends on their characteristics. At the time of writing, Ethereum is secured by more than 9000 nodes[4] while Polygon by 100 validators[5]. Higher security comes at a cost since gas costs in USD for Ethereum are significantly higher due to the higher value of the ETH with respect to MATIC.

2) Remote tests (latency): We measured the latency it takes the Relayers, hosted in our local device, to finalize a cross-chain transfer deploying the contracts to the Goerli and Mumbai test networks. We collected the measurements with 3, 5, and 7 Relayers. The Goerli test network simulates the Review blockchain whose `Bridge` address is 0x8a27e2475741aa9aba0afca2193e20f1406be530, while the Mumbai test network the Sales blockchain whose `Bridge` address is 0x83f2518414C639B2edEC42C906f473679318165d.

Figure 5 shows our test results. We evaluated separately the time needed to execute the *Payment-Permission* transfer (marked in blue) and to execute the *Review-Reward* transfer (marked in red). We notice the time to perform the *Payment-Permission* transfer, i.e. invoking the `voteProposal()` function on the `Bridge` contract deployed on the Goerli test network, increases with respect to the number of Relayers from 40 to 77 seconds. Instead, the time to perform the *Review-Reward* transfer, i.e. invoking the `voteProposal()` function on the `Bridge` contract deployed on the Mumbai test network, remains around 90 seconds regardless of the number of Relayers. The total amount of time needed to complete the entire protocol goes from 140 seconds with 3 Relayers to 170 seconds with 7 Relayers.

[4]https://etherscan.io/nodetracker [Accessed on 10 January 2023].
[5]https://staking.polygon.technology/ [Accessed on 10 January 2023].

VI. DISCUSSION

As anticipated in Section III, this paper follows the idea of a cross-chain rating system supporting the *Payment-Permission* and *Review-Reward* exchanges with a bridging-based approach instead of an atomic swap-based one. Following the atomic cross-chain swap approach [20] the two transfers are synchronized with hashlocks and timelocks. This protocol does not involve any intermediary besides the two participants, Alice and Bob, making it fully decentralized. However, the protocol requires active communication and coordination between the two participants, which may be cumbersome in terms of actions, especially for the item owner in case of multiple transfers happening at the same time. With a bridging solution the actions required by the participants can be minimized since the bridge tool, e.g. the Relayers, carry out the transfer automatically. The optimal number of inputs from the participants would be 1 or 2 per transfer: setting the bill and paying, or submitting a review (the reward, differently from the bill, could be a fixed amount set once and changed only a few times). In our prototype, we deemed necessary an additional operation from the user that is the approval to the `Bridge` to use their tokens (either permission or discount). This was driven by the standard token management of Chainbridge since the `Bridge` contract cannot ask the `Handler` to burn Bob's tokens since he is the owner and the `Handler` would reject the transaction. Alternatively, the `Bridge` contract could be customized to remove the management of tokens and use the Relayers only to pass messages across blockchains. Fewer actions from the users of the cross-chain rating system mean fewer times the user has to pay for the blockchain fees (although Chainbridge includes a fee to pay to the Relayers). Additionally, the Relayers introduce a centralization point in the system that, if attacked, may cause a denial of service since they carry out the cross-chain transfers. As a consequence, setting a permission in the Review blockchain or sending a reward in the Sales blockchain would require manual intervention of the item owner.

An argument could be reducing the number of cross-chain transfers to one, i.e. sending the review together with the payment to unlock the reward, but we preferred to keep the payment and rating actions logically separated to not enforce the submission of a rating alongside the payment. Moreover, payment and rating operations depend on the nature of an item: in a restaurant context, a customer pays for a meal they already consumed and they could give an opinion at roughly the same time; in a movie context, a customer pays before watching the movie, and therefore tying the rating to the payment is not intuitive.

Finally, the overhead introduced by Chainbridge is related to the `Bridge` and `Handler` smart contracts, which are a

one-time cost to pay for. Also, the Chainbridge protocol could require a fee to pay to the Relayers to carry on the transfers since they need to submit transactions. The overhead of a potential solution based on HTLCs, instead, depends on the structure of the HTLCs and their number, for example using a single contract to deal all the swaps or deploying a single contract per swap with the former solution costing less in terms of fees while the latter being more modular. A similar trade-off could be done with the ERC-20 token contract, which in the prototype is one per restaurant while it could be shared among multiple restaurants: the approach followed in the paper gives to the item owners more governance on such contract, while the shared approach would relief the item owner from deploying a contract thus reducing the deployment costs.

VII. Conclusions

This paper proposes a solution to support cross-chain rating systems based on the bridging approach. The rating system is based on two cross-chain transfers, a payment on a blockchain unlocks a permission to review in the other blockchain and, equivalently, a review unlocks a reward. We tested an implementation integrating Chainbridge that natively handles the transfer of ERC-20 tokens. Therefore, we represented the permission and the reward, the data to transfer, as ERC-20 tokens. The prototype demonstrates the feasibility of supporting cross-chain rating systems with Chainbridge, with the only caveat being an approval operation before initiating a cross-chain transfer. The experiments show that the costs in gas are the typical spent by Ethereum smart contracts with the deployment costs being, as expected, the largest. Finally, the experiments with two test networks allow us to provide a tangible measurement of the time it takes to carry on a cross-chain transfer, a time that remains acceptable even in the worst test case since it is spent by the Relayers in the background from the point of view of the user and the item owner.

As future improvements, we plan to structure the prototype in a way to minimize the input transactions required by the users, potentially achieving a single operation per cross-chain transfer. It is interesting to test the same use case, or others, with alternative cross-chain technologies, for example with the already mentioned Cosmos, to improve our knowledge and understanding of cross-chain DApps by analyzing their potential and their limitations. With all the gained insights we plan to identify missing features and propose cross-chain methods to support them. Finally, given the presence of regulations that might limit the deployment of our proposal, such as the General Data Protection Regulation (GDPR), we plan to study an alternative approach to bridge not only blockchains, but also off-chain authenticated storage means to support the actual deletion of sensitive data to comply with such regulations.

Acknowledgments

This work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2009.

[2] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[3] V. A. Siris, P. Nikander, S. Voulgaris, N. Fotiou, D. Lagutin, and G. C. Polyzos, "Interledger approaches," *IEEE Access*, vol. 7, pp. 89 948–89 966, 2019.

[4] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on blockchain interoperability: Past, present, and future trends," *ACM Comput. Surv.*, vol. 54, no. 8, pp. 168:1–168:41, 2022.

[5] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: introduction and challenges," in *Recommender systems handbook*. Springer, 2015, pp. 1–34.

[6] A. S. Almasoud, F. K. Hussain, and O. K. Hussain, "Smart contracts for blockchain-based reputation systems: A systematic literature review," *J. Netw. Comput. Appl.*, vol. 170, p. 102814, 2020.

[7] A. Lisi, A. D. Salve, P. Mori, L. Ricci, and S. Fabrizi, "Rewarding reviews with tokens: An ethereum-based approach," *Future Gener. Comput. Syst.*, vol. 120, pp. 36–54, 2021.

[8] A. Bogliolo, P. Polidori, A. Aldini, W. A. Moreira, P. Mendes, M. Yildiz, C. B. Lafuente, and J. Seigneur, "Virtual currency and reputation-based cooperation incentives in user-centric networks," in *8th International Wireless Communications and Mobile Computing Conference, IWCMC 2012, Limassol, Cyprus, August 27-31, 2012*. IEEE, 2012, pp. 895–900.

[9] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE, 2018, pp. 45–54.

[10] D. Mazzei, G. Baldi, G. Fantoni, G. Montelisciani, A. Pitasi, L. Ricci, and L. Rizzello, "A blockchain tokenizer for industrial IOT trustless applications," *Future Gener. Comput. Syst.*, vol. 105, pp. 432–445, 2020.

[11] T. Koens and E. Poll, "Assessing interoperability solutions for distributed ledgers," *Pervasive and Mobile Computing*, vol. 59, p. 101079, 2019.

[12] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," Online, https://cosmos.network/whitepaper [Accessed on 10 January 2023], 2016.

[13] Polygon, "Introduction to Polygon PoS," https://wiki.polygon.technology/docs/develop/getting-started/ [Accessed on 10 January 2023].

[14] Chainsafe, "Chainbridge documentation," Online, https://chainbridge.chainsafe.io/ [Accessed on 10 January 2023].

[15] M. Shaker, F. S. Aliee, and R. Fotohi, "Online rating system development using blockchain-based distributed ledger technology," *Wirel. Networks*, vol. 27, no. 3, pp. 1715–1737, 2021.

[16] K. Salah, A. Alfalasi, and M. Alfalasi, "A blockchain-based system for online consumer reviews," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2019, Paris, France, April 29 - May 2, 2019*. IEEE, 2019, pp. 853–858.

[17] F. Casino and C. Patsakis, "An efficient blockchain-based privacy-preserving collaborative filtering architecture," *IEEE Trans. Engineering Management*, vol. 67, no. 4, pp. 1501–1513, 2020.

[18] Q. Ye, T. Zhao, G. Sun, and X. Feng, "A recommendation scheme with reputation-based incentive mechanism on consortium blockchain," in *2021 International Conference on Networking and Network Applications, NaNA 2021, Lijiang City, China, October 29 - Nov. 1, 2021*. IEEE, 2021, pp. 313–318.

[19] T. Hai, J. Zhou, S. R. Srividhya, S. K. Jain, P. Young, and S. Agrawal, "BVFLEMR: an integrated federated learning and blockchain technology for cloud-based medical records recommendation system," *J. Cloud Comput.*, vol. 11, p. 22, 2022.

[20] M. Herlihy, "Atomic cross-chain swaps," in *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, C. Newport and I. Keidar, Eds. ACM, 2018, pp. 245–254.

[21] M. Herlihy, B. Liskov, and L. Shrira, "Cross-chain deals and adversarial commerce," *VLDB J.*, vol. 31, no. 6, pp. 1291–1309, 2022.

[22] A. Lisi, A. D. Salve, P. Mori, and L. Ricci, "Practical application and evaluation of atomic swaps for blockchain-based recommender systems," in *ICBTA 2020: The 3rd International Conference on Blockchain Technology and Applications, Xi'an, China, December, 2020*. ACM, 2020, pp. 67–74.