ETH Relay: A Cost-efficient Relay for Ethereum-based Blockchains

Philipp Frauenthaler*, Marten Sigwart*, Christof Spanring†, Michael Sober*, Stefan Schulte*
* *Christian Doppler Laboratory for Blockchain Technologies for the Internet of Things*
Distributed Systems Group, TU Wien, Vienna, Austria
{p.frauenthaler, m.sigwart, m.sober, s.schulte}@dsg.tuwien.ac.at
† *Pantos GmbH, Vienna, Austria*
contact@pantos.io

***Abstract*—Current blockchain relay schemes require the immediate validation of each relayed block header by the destination blockchain. This leads to high operating cost when deploying these relays between Ethereum-based blockchains where validating block headers on-chain is computationally expensive.**

To overcome these limitations, we introduce a novel relay scheme that employs a validation-on-demand pattern combined with economic incentives to reduce the cost of operating a relay between Ethereum-based blockchains by up to 92%. With this relay scheme, decentralized interoperability between blockchains like Ethereum and Ethereum Classic becomes feasible.

***Index Terms*—blockchain interoperability, cross-blockchain communication, blockchain relay, simplified payment verification**

I. Introduction

Because of its ability to store data and execute code in a decentralized and immutable way, blockchain technology is seen as potentially disruptive in areas such as finance [1], business process management [2], data provenance [3, 4], supply chain management [5], or healthcare [6]. The growing popularity among industry and research communities has led to a plethora of new projects creating their own blockchains, either by developing an entirely new one or by forking an existing code base. However, interoperability between these blockchains is often not foreseen. As a result, a heterogeneous landscape of independent and unconnected platforms has emerged [7]. Today, even blockchains that share the same code base operate isolated from each other. Prominent examples of such blockchains are Ethereum-based blockchains, e.g., Ethereum and Ethereum Classic.

One promising way to break this isolation are blockchain relays [8]. Relays operate by having off-chain clients forward block headers of some source blockchain to a destination blockchain virtually replicating one blockchain within the other. This way, it becomes possible for the destination blockchain to independently verify that certain pieces of data (e.g., transactions) exist on the source blockchain [8].

To ensure that only valid block headers are replicated within the destination blockchain, current relay solutions [9, 10] require the destination blockchain to validate each relayed block header according to the protocol rules of the source blockchain (e.g., the consensus algorithm). With each block header being validated on-chain, the need for explicit trust in off-chain clients is eliminated.

However, Ethereum-based blockchains such as Ethereum and Ethereum Classic use a consensus algorithm called Ethash that is computationally expensive to validate on-chain. Even with gas-optimized implementations [11], the validation of Ethash for a single block header still costs around 3 million gas. Thus, operating a relay that requires each block header to be fully validated on submission between Ethereum-based blockchains leads to exorbitant operating cost.

To overcome this issue, we introduce ETH Relay—a novel relay scheme that uses a validation-on-demand pattern together with a sophisticated incentive structure to achieve a cost reduction of up to 92% over traditional relay solutions when deployed for Ethereum-based blockchains. We show that ETH Relay is fully decentralized and secure as long as at least one off-chain client acts honestly.

To this end, the paper is organized as follows. We first provide background information on blockchain relays (Section II), and take a look at existing relay solutions (Section III). In Sections IV and V, we then describe the concepts and implementation of ETH Relay. In Sections VI and VII, we evaluate the proposed relay scheme with regards to security and operating cost, respectively. Finally, Section VIII concludes the paper.

II. Background

In blockchain relays, off-chain clients relay the block headers of a source blockchain to some destination blockchain [8]. Having access to the source blockchain's block headers, the destination blockchain can use a technique called Simplified Payment Verification (SPV) to verify the existence of certain pieces of the source blockchain's state (e.g., the existence of certain transactions). In this section, we first explain the concept of SPV and then describe how SPVs facilitate blockchain relays.

A. Simplified Payment Verification

SPV is a technique that can be used to cryptographically verify that a particular transaction is part of a blockchain while only having knowledge of a blockchain's block headers and not

978-0-7381-0495-9/20/$31.00 ©2020 IEEE
DOI 10.1109/Blockchain50366.2020.00032

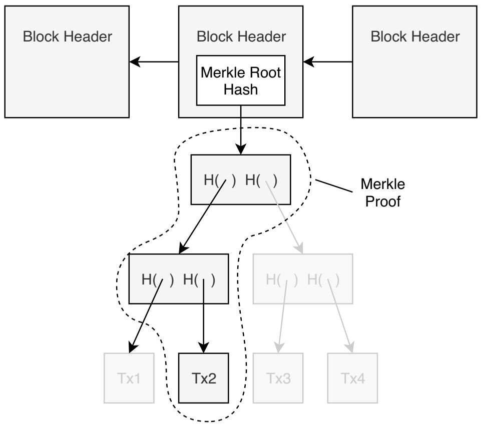

Fig. 1. A block consists of a header and a Merkle tree containing the block's transactions. Merkle trees enable concise proofs of membership, as illustrated for Tx2.

the individual blockchain transactions [1]. SPV is commonly used in light nodes such as wallet software [12].

In blockchains such as Bitcoin and Ethereum, transactions in a block are organized in a so-called Merkle tree [13], a data structure in which each node uses hash pointers to reference its child nodes. As seen in Fig. 1, the leaves of the Merkle tree consist of the individual transactions with the hash of the Merkle tree's root (Merkle root hash) stored as part of the block header. Therefore, a light node has access to the Merkle root hash. With this, light nodes can verify the inclusion of a particular transaction in the blockchain by leveraging a so-called Merkle proof of membership. Such a proof consists of all tree nodes that make up the path from the transaction (leaf) up to the root node (see Fig. 1), and can be retrieved from full nodes. When retrieving such a proof, the light node recalculates the hashes of all nodes along the path from the leaf (i.e., the transaction) up to the root node. If the final hash matches the Merkle root hash of the block header stored by the light node, the membership of the transaction within the corresponding block is successfully verified. Light nodes thus are able to verify the existence of transactions (e.g., payments) while only consuming a fraction of the space as they do not need to store the transaction history.

B. Relay Schemes

Light nodes use SPVs to verify whether or not a particular transaction exists on some blockchain. Blockchain relays leverage this capability by having a smart contract on some destination blockchain act as a light node for some source blockchain [8]. A prominent example for this is BTC Relay [9]. BTC Relay is essentially a Bitcoin light node running on the Ethereum blockchain in the form of a smart contract, the so-called *relay contract*. The relay contract is able to verify the inclusion of Bitcoin transactions on the Ethereum blockchain by means of SPVs.

For successful SPVs, the relay contract needs to know about the block headers of the source blockchain. However, contrary to off-chain light nodes, the relay contract cannot proactively query headers from full nodes. Instead, headers of the source blockchain need to be constantly submitted to the contract by off-chain clients. As any off-chain client can submit block headers, potentially illegal block headers may arrive at the relay contract. Hence, blockchain relays need to make sure that only valid block headers are used for SPVs [8].

Furthermore, in Proof of Work (PoW) blockchains like Ethereum, multiple valid blocks with the same block height can exist in parallel, forming multiple blockchain "branches". While there can be several branches at the same time, only one of these branches represents the current main chain of the blockchain, e.g., in PoW blockchains, the main chain is identified by searching for the branch with the highest total difficulty [8]. As more block headers are appended to branches, the main chain of a blockchain may change over time. This represents a challenge to relays since SPVs should only be successful if the requested block header is part of the main chain. When an SPV is requested on a certain block, the relay contract thus needs to determine whether the block is part of the main chain of the source blockchain. The likelihood of a block remaining part of the main chain increases with each succeeding block that is appended to it. These block confirmations should be taken into account by the relay contract as well when performing SPVs.

III. Related Work

As blockchain relays are seen as one way to achieve blockchain interoperability [8, 14], a couple of relays have been implemented and conceptualized [9, 10, 15, 16, 17]. In the following section, we explain how these relays verify the validity of submitted headers and discuss current limitations.

BTC Relay [9] was the first and—to the best of our knowledge—so far only relay solution to be operational. It allows relaying block headers from the Bitcoin blockchain to the Ethereum blockchain. Similarly, Waterloo [15, 16] attempts to provide a relay between the Ethereum and EOS blockchains. Waterloo is bi-directional, i.e., it consists of two separate relays, from Ethereum to EOS and from EOS to Ethereum. To ensure that only valid block headers are used for SPVs, both relays—BTC Relay and Waterloo—fully validate newly submitted block headers according to the header validation procedure of the source blockchain, i.e., the blockchain from which the headers originate (e.g., Bitcoin in case of BTC Relay). Among other things, this usually involves validating the consensus algorithm, e.g., for PoW blockchains it needs to be verified that enough work was performed for constructing a block [8].

For both relays, fully validating every submitted block header is economically viable due to the specifics of the involved blockchains. In case of BTC Relay, validating each Bitcoin block header on Ethereum is feasible as Bitcoin headers only have a size of 80 bytes and validating the Bitcoin hashing algorithm SHA-256 on Ethereum is computationally inexpensive.

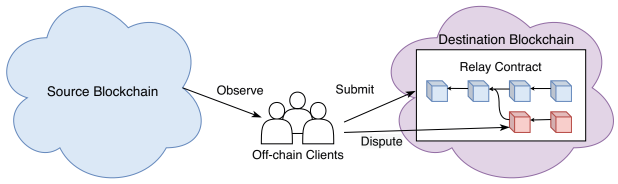

Fig. 2. The ETH Relay Scheme

In Waterloo, relaying headers from Ethereum to EOS is economically viable because EOS uses delegated Proof of Stake as consensus mechanism. By design, this allows cheaper on-chain computations and thus makes the on-chain validation of Ethash, the consensus algorithm of Ethereum, feasible. For the other direction (EOS to Ethereum), delegated Proof of Stake relies on a changing set of block producers, which on average happens every 8 hours [15]. Consequently, when relaying block headers from EOS to Ethereum, it suffices to validate only those headers where the block producers change.

BTC Relay and Waterloo both work in their specific settings. However, when deploying one of these schemes between Ethereum-based blockchains such as Ethereum and Ethereum Classic, validating each block header would lead to high operating cost: validating Ethash, the PoW algorithm used by Ethereum-based blockchains, on another Ethereum-based blockchain costs about 3 million gas even with gas-optimized solutions (see Section VII). Hence, the underlying strategy employed by BTC Relay and Waterloo is not viable for relays between Ethereum-based blockchains.

A relay for Ethereum-based blockchains that mitigates high validation cost is PeaceRelay [10]. In its current implementation, PeaceRelay relies on trusted, authorized clients to submit valid block headers. While this makes PeaceRelay relatively cheap to operate, it also leads to a high degree of centralization. PeaceRelay is therefore rather a notary scheme than a relay.

Another approach for reducing operating cost is to leverage zero-knowledge proofs as done in zkRelay [17]. In zkRelay, clients validate a batch of Bitcoin headers off-chain and submit a proof certifying the successful validation to the relay contract on the Ethereum blockchain. While the block header validation of Bitcoin can be implemented as zero-knowledge proof enabling such succinct batch submissions, it is uncertain whether this approach can be leveraged for block headers of Ethereum-based blockchains.

To conclude, a number of blockchain relays have been conceptualized so far. However, their applicability largely depends on the underlying blockchains. Deploying these relays between Ethereum-based blockchains either leads to high operating cost, requires explicit trust in third parties, or may be technologically infeasible. The following sections show how operating cost of relays can be kept to a minimum without placing explicit trust in third parties.

Algorithm 1 Procedure performed by the relay contract when receiving a new header of the source blockchain

```
1:  function SUBMITBLOCKHEADER($header, submitter$)
2:      if $headers$.contains(HASH($header$)) == $true$ then
3:          return $false$
4:      $parentHash = header.parentHash$
5:      if $headers$.contains(parentHash) == $false$ then
6:          return $false$
7:      $headers$.put(HASH($header$), $header$)
8:      $header.m.lockedUntil = now + LOCK\_PERIOD$
9:      $parent = headers$.get($parentHash$)
10:     $parent.m.chldn$.append(HASH($header$))
11:     $branchHeads$.add(HASH($header$))
12:     if $branchHeads$.contains($parentHash$) then
13:         $branchHeads$.remove($parentHash$)
14:         $header.m.branchId = parent.m.branchId$
15:         $header.m.junction = parent.m.junction$
16:     else
17:         $lastBranchId = lastBranchId + 1$
18:         $header.m.branchId = lastBranchId$
19:         $header.m.junction = parentHash$
20:         if $parent.m.chldn.length == 2$ then
21:             SETJUNCTION($parent.m.chldn[0], parentHash$)
22:     $mainChainHead$ = GETMAINCHAINHEAD( )
```

IV. ETH Relay

This section introduces ETH Relay, a novel relay scheme that keeps operating cost low while remaining fully decentralized and secure. The relay scheme consists of a relay contract (i.e., a smart contract) running on the destination blockchain and off-chain clients (see Fig. 2).

A. Validation-on-demand

In ETH Relay—just like in any relay scheme—off-chain clients are responsible for continuously relaying block headers of the source blockchain to the relay contract on the destination blockchain. Algorithm 1 shows the pseudo code that is executed for each newly submitted block header.

Right after the arrival of a new header, it is checked that the retrieved header has not been submitted to the relay contract before (Line 2) and that the parent block referenced by the header is known to ensure that only a continuous chain of block headers is replicated within the relay contract (Line 5). If both checks are successful, the header is stored in a hashmap with the header's hash as key and the header itself as value (Line 7).

Notably, a full header validation as done by traditional relay solutions such as BTC Relay is *not* carried out. Instead, the key to reducing operating cost in ETH Relay lies in a validation-on-demand pattern. That means, newly submitted block headers do not immediately undergo the expensive full header validation procedure of the source blockchain. Rather, they are optimistically accepted by the relay contract. Of course, this may lead to invalid block headers entering the relay contract. Therefore, each newly received header is "locked" for a certain amount of time (Line 8), during which a block header cannot be used for SPVs.

To filter out the invalid block headers that may have entered the relay contract, locked headers can be disputed by the off-chain clients as shown in Fig. 2. That is, the clients monitor the relay contract and the source blockchain. Whenever they detect a block header that is submitted to the relay contract and that does not constitute a valid block header of the source blockchain, they send a dispute request to the relay contract. In case of a dispute, the full header validation according to the rules of the source blockchain is carried out by the relay contract. If the validation fails, the invalid block header is removed from the contract. If in the meantime, block headers have been submitted to the relay contract that derive from the invalid block header, these descendants are removed as well (see Fig. 2).

Once the lock period has passed, block headers are considered valid. From then on, SPVs on these headers can be carried out. While the validation-on-demand pattern already reduces cost of submitting block headers, ETH Relay applies a further optimization for reducing operating cost by using a modified version of the so-called content-addressable storage pattern [18].

B. Content-Addressable Storage Pattern

The idea of the content-addressable storage pattern is that not all data needs to be stored directly in the relay contract. Instead, the remaining data is stored externally. As Ethereum transactions also include the parameters of smart contract invocations, submitted block headers are implicitly recorded in the blockchain's transaction history. We can take advantage of this fact by storing only the hash of the block header, the block number and certain meta data in the relay contract itself. Whenever clients initiate a dispute or an on-chain SPV, they read the required full header data from the corresponding submit-transactions recorded in the transaction history and provide it to the relay contract. The contract can then verify the provided headers' integrity by recalculating their hashes and comparing them to the hashes stored in the relay contract. This way, no trust in the client invoking an SPV or a block header dispute is required. With fields such as the parent hash or the Merkle root hash no longer being kept in the relay contract directly, the amount of stored data per block header is reduced—subsequently further reducing submission cost. Notably, while the content-addressable storage pattern leads to reduced submission cost it may increase cost of executing SPVs and header disputes since more data needs to be passed with each request (see Section VII for details).

Obviously, the correct functioning of ETH Relay is only ensured if off-chain clients continuously submit block headers of the source blockchain to the relay contract on the destination blockchain and dispute any invalid block headers entering the relay contract. However, clients that submit and dispute block headers incur cost. Thus, an incentive structure for encouraging (honest) participation is needed. The details of this incentive structure are explained in the next section.

C. Incentive Structure

Without an incentive structure that compensates off-chain clients for submitting and disputing block headers, clients may have no interest in participating in the proposed relay scheme. The incentive structure we propose rewards off-chain clients for submitting and disputing block headers and also discourages submission of invalid block headers.

To hold clients that submit invalid block headers accountable, clients are required to deposit a stake for every submitted header. The stake is locked for the duration of the lock period of newly submitted block headers. While the stake is locked, it cannot be withdrawn and cannot be used for submitting further block headers. After a submitted header has passed the lock period without a dispute, the client that submitted the header gets back control of the corresponding stake. However, in case the block header is disputed successfully within the lock period, i.e., the validation of the block header fails, the client that triggered the dispute earns the locked stake of the submitter as well as any stake that was locked for any descendant of the illegal block header. Not only does this incentivize disputes, it also discourages submission of invalid block headers as clients risk losing the deposited stake. Of course, clients are only incentivized to dispute headers if the potential reward is higher than the cost of executing the dispute.

To encourage the submission of block headers, clients receive a fee every time their submitted headers are used for SPVs. This verification fee is paid by the client requesting the verification. To fully compensate submitting clients, the total verification fees earned on each header need to be greater than the initial submission cost for that header (Eq. (1)).

$$\textit{fee} \times \textit{no. of verifications} > \textit{submission cost} \tag{1}$$

The minimum verification fee can thus be calculated as the average submission cost divided by the expected number of verifications per block header (Eq. (2)).

$$\textit{fee} > \frac{\textit{submission cost}}{\textit{no. of verifications}} \tag{2}$$

With headers replicated within the relay contract and an incentive structure in place to encourage participation and honest behavior, the next section looks at how SPVs are executed within ETH Relay.

V. Simplified Payment Verification in ETH Relay

Once headers are replicated within the relay contract, clients (e.g., other smart contracts) can send a request to the relay contract in the form of "Is transaction tx of block b part of the source blockchain and confirmed by at least n blocks?". To answer the request, the relay contract executes an on-chain SPV. Since SPVs should only be successful if the requested block is part of the main chain of the source blockchain,

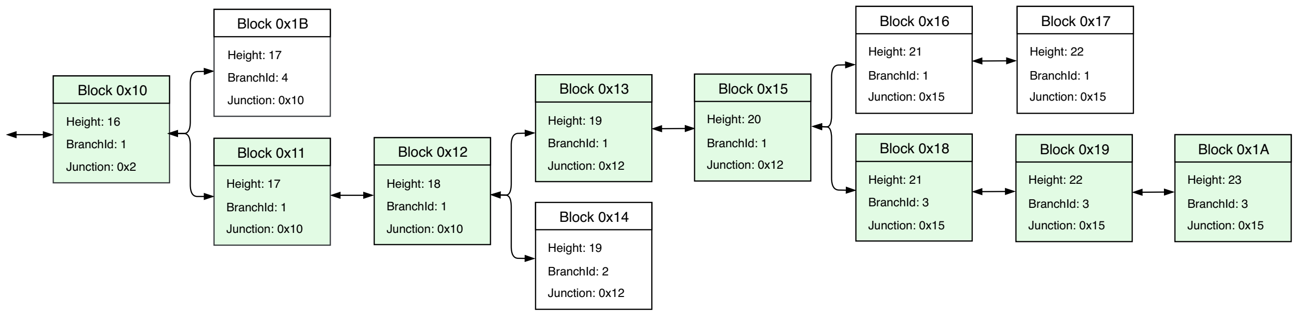

Fig. 3. An example illustrating the replication of a source blockchain within the relay contract. Headers are double-linked (denoted by arrows pointing in both directions) as ETH Relay stores in each block header the parent hash as well as the hashes of its children. Green headers represent the current main chain of the source blockchain. For the sake of simplicity, block hashes are in ascending order to make it clearly evident which block headers have been submitted before others to the contract, e.g., block header 0x11 was submitted after block header 0x10, 0x1B after 0x1A, and so on. Block headers 0x1B, 0x14, 0x17, and 0x1A are heads of the corresponding branch. Block headers 0x10, 0x12, and 0x15 represent branch junctions.

the relay contract performs the following four steps. First, it determines the current main chain of the source blockchain (see Section V-A). Second, the contract checks whether h is unlocked (i.e., the lock period has passed) and part of the main chain (see Section V-B). Third, the contract verifies that h is confirmed by at least n succeeding block headers (see Section V-C). Finally, it is checked whether *tx* is actually included in block b by means of a Merkle proof of membership (see Section V-D).

A. Determining the Main Chain

As mentioned in Section II, the main chain of a blockchain may change over time as new blocks are appended to different branches. In ETH Relay, the relay contract tracks the head of each branch, i.e., the latest block header that was appended to the branch. We consider Algorithm 1 again. Whenever a new block header is received, it is added to the set of branch heads (Line 11) as it either starts a completely new branch or it becomes the new head of an existing branch in which case it replaces its parent (Lines 12ff). The contract then re-evaluates the branch head with the greatest difficulty storing it in a global variable *mainChainHead* (Line 22). This branch head represents the current head of the main chain of the source blockchain.

B. Verifying Main Chain Membership

With the main chain determined, the relay contract now needs to verify that header h is unlocked and actually part of the main chain. As blockchains with multiple branches represent a Directed Acyclic Graph (DAG) [19], determining main chain membership boils down to a classical problem in graph theory, the so-called reachability problem. In graph theory, reachability refers to the ability of some node v to reach some other node s within a graph [20]. Verifying the membership of some block b on the main chain (which is a specific branch of the DAG) is done by checking whether the main chain's head (i.e., the latest block of the main chain) can reach block b.

As each block header contains a hash pointer to its parent, we could simply trace from the main chain's head all the way back until we reach the requested block header or the genesis block. However, depending on how far back the requested block lies, this traversal can be expensive.

To make this traversal more cost-efficient, ETH Relay follows a design inspired by the chain decomposition approach [20]. Within the relay contract, the source blockchain (i.e., the DAG) is partitioned into its individual branches with each branch being assigned a unique identifier. To track branch memberships, each block header gets assigned the id of the branch it belongs to. Furthermore, for each received header, the relay contract stores a reference to the preceding branch junction, i.e., a reference to the header where the branch of the newly submitted block header branches off. This meta data is stored in the fields *m.branchId* and *m.junction*, respectively.

In case the submitted header continues an existing branch, *m.branchId* and *m.junction* are set to the corresponding field values of its parent, as shown in Lines 14 and 15 of Algorithm 1. If a new branch occurs, *m.branchId* gets assigned the maximum branch id used so far incremented by one and *m.junction* of the submitted header is set to the parent's hash (Lines 17 to 19).

The introduced helper fields *m.branchId* and *m.junction* enable a more efficient search when verifying the membership of some block header on the main chain, as the backwards traversal can be executed in jumps from branch junction to branch junction rather than from block header to block header.

Consider the example illustrated in Fig. 3 which shows a source blockchain replicated within the relay contract. All block headers between two consecutive branch junctions or between a branch junction and a branch head have the same branch id *BranchId*. Since a submitted header's branch id is either set to its parent's branch id or to the maximum branch id incremented by one, we know that all descendants of some block header (i.e., headers that build upon this header) have a branch id equal to or greater than the branch id of that header. Analogous, all predecessors of some header have a branch id equal to or lower than the branch id of that header.

This constraint can be used to efficiently verify whether block header h is part of the main chain of the source blockchain, i.e., whether the block header h is a predecessor of the main chain's head.

If block header h has a branch id that is greater than the branch id of the main chain's head (i.e., *mainChainHead*), we know—without having to traverse any header—that header h cannot be a predecessor of *mainChainHead* and thus not be part of the main chain.

If block header h has the same branch id as *mainChainHead*, we know that h is part of the main chain since header h and *mainChainHead* belong to the same branch. Again, no block header needs to be traversed.

In case the branch id of header h is smaller than the branch id of the main chain's head, we cannot know with certainty whether h is part of the main chain. Consider block header 0x16 in Fig. 3. Despite having a branch id lower than header 0x1A (i.e., the head of the main chain), header 0x16 is not part of the main chain. Hence, the relay contract starts at the main chain's head and traverses each reachable branch junction until a branch junction j is reached that has a branch id lower than or equal to the branch id of header h. Note that the next branch junction can be retrieved from the field *m.junction* of the currently traversed branch junction.

If the branch id of j is lower than the branch id of h, we know that h is not part of the main chain as h cannot be a predecessor of junction j due to the higher branch id. In case both headers have the same branch id, we know that they are both part of at least one common branch. However, it can still be the case that h is not part of the main chain. Consider block headers 0x15 and 0x16 in Fig. 3. If 0x15 is the branch junction j and 0x16 the header h, 0x15 is part of the main chain while 0x16 is not. As such, the relay contract compares the block heights of j and h. If *h.blockHeight* $>$ *j.blockHeight*, h is not part of the main chain. If *h.blockHeight* $\leq$ *j.blockHeight*, j and h are either the same or j is a branch junction between h and the main chain's head. Hence, in both cases, h is part of the main chain.

We note that it is completely sufficient to only traverse branch junctions since all headers between two consecutive branch junctions or between a branch junction and a branch's head always have the same branch id. As all headers in-between are skipped, the relay contract only checks a few block headers instead of traversing all headers between the header h and the main chain's head as done in a naïve search.

C. Counting Block Confirmations

With main chain membership determined, it needs to be verified that header h is already confirmed by a certain number of block headers to increase the likelihood of the block remaining part of the main chain [1]. For that purpose, the relay contract maintains a reference to each header's immediate children. Whenever a new block header is received, the relay contract adds its hash to its parent's child list (Line 10 of Algorithm 1). Typically, a header only has one child. If the header is a branch junction, the list contains at least two children (i.e., the hashes of the block headers branching off from the header).

Of course, for each confirming block header the lock period must be over. To verify the number of confirmations, the relay contract temporarily stores a reference to the last unlocked branch junction that was encountered while determining main chain membership of header h. Starting at that branch junction, all its descendants are traversed in the direction of the main chain until enough unlocked (confirming) block headers are found (as all headers between h and the starting point are unlocked, these header count already as confirming headers). If a block header is encountered that is locked or there is no descendant to check (i.e., the main chain's head is reached), we know that h has not enough confirmations. In this case, the SPV fails.

Once more, consider the example in Fig. 3. Assuming membership of block header 0x11 should be verified and block header 0x15 is the most recent unlocked branch junction. Therefore, the starting point for the confirmation verification is block header 0x18.

D. Verifying the Merkle Proof of Membership

After verifying that header h is unlocked, part of the source blockchain's current main chain and that h is confirmed by at least n succeeding block headers, the relay contract checks the Merkle proof of membership.

If the verification of the Merkle proof fails, transaction tx is not part of the corresponding block b of header h. If it is successful, tx is included within block b. Since the relay contract has already verified h's membership in the main chain of the source blockchain and n headers succeeding h, it can be concluded that transaction tx is in fact included in the source blockchain.

By specifying a sufficiently large number of confirmations, clients requesting a verification increase the probability that transaction tx remains in the main chain of the source blockchain. Further, as the verification procedure relies on the source blockchain's headers being exactly replicated within the relay contract on the destination blockchain, it must be difficult to tamper with the relay. The next section analyzes the security of the relay.

VI. Security Analysis

This section provides a security analysis of ETH Relay by looking at possible attack scenarios and investigating consequences on the relay in case changes to the involved blockchains occur.

For the following discussion, we suppose the set of off-chain clients to remain static during an attack. Furthermore, our analysis is based on the following assumptions: (a) no off-chain client is guaranteed to follow the proposed relay scheme, (b) the actions of many clients are driven by self-interest, and (c) some clients may categorically deviate from the scheme. Accordingly, we categorize off-chain clients into three groups according to the BAR (Byzantine, Altruistic, Rational) model [21]. This model has found application in security analysis for blockchain protocols and extensions before, e.g., [22, 23]. Under this model, byzantine clients may depart arbitrarily from the relay scheme for any reason, e.g., they may be faulty or may just follow strategies optimizing an unknown utility function. Altruistic clients always follow the proposed scheme, regardless of whether deviations would lead to a higher profit. They exhibit no adversarial behavior. Finally, rational clients are self-interested, aiming at maximizing their profit according to a known utility function. These clients will depart from the scheme if they expect doing so to yield a higher profit than being honest.

A. *Relay Poisoning*

To reliably execute on-chain SPVs, the relay contract depends on off-chain clients constantly submitting block headers. Thus, an attacker may try to poison the relay contract with invalid block headers. For that, the attacker must attempt a chain re-organization within the relay contract according to the consensus rules of the source blockchain, i.e., the attacker must submit enough block headers such that these headers form the new main chain within the relay contract. A chain re-organization allows the attacker to request SPVs on invalid block headers. For instance, an application relying on the relay contract can be tricked into performing actions on the basis of transactions that never occurred on the source blockchain.

Essentially, an attacker can choose between two approaches. Either the attacker sends invalid block headers to the relay contract or the attacker submits headers which are themselves valid but belong to blocks containing invalid transactions (i.e., a header validation in case of a dispute would not detect any anomaly). We discuss these two attack models in the next subsections.

1) Incentive Attacks on Disputes:
Option one to achieve relay poisoning is for the attacker to submit illegal block headers while preventing other clients from disputing these headers. The advantage of this approach is that the attacker does not have to follow the source blockchain's consensus rules for creating block headers, e.g., the attacker does not have to solve the PoW for each header. This enables the attacker to create block headers at a much faster rate. However, disputes of these illegal headers would be successful since the block header validation would inevitably fail. Hence, to launch a successful attack, the attacker needs to convince all participating clients to not dispute any illegal headers for the duration of the headers' lock periods, e.g., by launching incentive attacks [23].

Imagine all off-chain clients to act rationally. Even with incentives perfectly aligned, rational clients seeking to maximize profit will deviate from the relay scheme if they expect doing so to yield a higher profit. For instance, an attacker may offer these clients an alternative reward that is more profitable than a successful dispute of invalid block headers. Naturally, the more clients participate in the relay, the more expensive the attack becomes since each client needs to be convinced to follow the attack. However, if all clients act rationally and the attacker has sufficient funds, an attack may be successful. In general, correct behavior is never guaranteed in systems that rely on all clients acting rationally since clients can always find ways to yield larger profits in a greater ecosystem [24].

Fortunately, the relay contract can determine the validity of headers by itself. It just needs a single client to trigger this validation in form of a dispute. Thus, if just one client acts altruistically, incentive attacks always fail ensuring the correct functioning of the relay scheme. As building open and permissionless systems that withstand all participants potentially deviating from the intended rules appears to be fundamentally impossible [24], an acceptable trade-off is a system that only requires a single altruistic participant out of all participants.

2) Incentive Attacks on Submissions:
Option two for achieving relay poising is for the attacker to submit block headers that are valid according to the source blockchain's header validation procedure but belong to blocks containing illegal transactions. Disputing these headers would not be successful since the header validation performed by the relay contract does not include the validation of transactions. Hence, the attacker could request SPVs on illegal transactions. If other clients continue to submit the correct headers from the source blockchain, the only way this attack can be successful is if the attacker is able to create and submit valid block headers at a faster rate than the network of the source blockchain, e.g., by launching a 51% attack [13].

Alternatively, the attacker may try to convince the other clients to refrain from submitting block headers for the duration of the attack. This way, the attacker's block headers are the only ones arriving at the relay contract on the destination blockchain. If all clients act rationally, the attacker may convince them to join the attack, e.g., by offering a bribe. However—analogue to what has been discussed above—since all clients need to be convinced, the cost of this attack grows proportionally with the number of participants. Again, if there is just one altruistic client that continuously submits the block headers created by the network of the source blockchain, the success of this attack is about as likely as a successful 51% attack on the source blockchain.

B. *Changes to the Source Blockchain*

Besides deliberate attacks as discussed in the prior sections, changes to the source blockchain may affect the reliability of the relay contract as well.

When changing the block header validation procedure of the source blockchain, it becomes either less or more restrictive. In the first case, headers adhering to the new validation rules would be rejected by the relay contract when being received, or clients would be able to successfully dispute block headers that are actually valid under the new rules. If the header validation becomes more restrictive, newly introduced validation rules are not enforced by the relay contract, possibly leading to the acceptance of headers invalid under the new rules. Thus, any change to the header validation procedure of the source blockchain requires an update of the header validation procedure performed within the relay contract. On the other hand, if changes to the source blockchain do not affect the header validation procedure, the relay contract does not need to be updated.

In case the source blockchain's network does not reach consensus on an upcoming update, the source blockchain may be split up, resulting in multiple instances of the same blockchain. Technically, such instances are just branches originating from the same blockchain. While the relay contract is able to keep track of branches, only one branch is used for on-chain SPVs. Hence, there may be a competition of multiple instances of the source blockchain to form the main chain within the relay contract. Which branch eventually overtakes the others may be unclear at the time the blockchain is split up. If multiple blockchain instances were to be supported, additional deployments of the relay contract would be required.

VII. Quantitative Analysis

To evaluate ETH Relay, we implemented a total of three prototypes (*Baseline*, *ETH Relay 1*, and *ETH Relay 2*, respectively) for Ethereum Virtual Machine (EVM)-based blockchains such as Ethereum and Ethereum Classic.

As the name implies, *Baseline* acts as baseline for our experiments. This prototype does not implement the validation-on-demand pattern. Instead, it fully validates each block header at submission as done by existing relays such as BTC Relay. Furthermore, *Baseline* implements a naïve search algorithm for verifying main chain membership starting from the main chain's head, traversing each header until the header supposedly containing the transaction is found or the genesis block is reached.

In contrast, the prototypes *ETH Relay 1* and *ETH Relay 2* both implement the validation-on-demand pattern and implement the more efficient search for verifying main chain membership as explained in Section V-B. *ETH Relay 2* additionally applies the content-addressable storage pattern as explained in Section IV-B.

The functionality of each prototype is summed up in Table 1. A fully functional reference implementation of all concepts and algorithms of the relay contract, an off-chain client written in Go, and the evaluation are available as open-source projects on GitHub[1][2][3]. For repeatability, the evaluation project not only contains the three prototypes used for the evaluation but also the evaluation scripts, the necessary block header data as SQL dump, and the results.

TABLE 1
PROTOTYPE FUNCTIONALITY

Functionality	*Baseline*	*ETH Relay 1*	*ETH Relay 2*
Validation-on-submission	✓		
Validation-on-demand		✓	✓
Content-adressable storage			✓
Naïve search	✓		
Optimized search		✓	✓

A. Experiments

To evaluate the operating cost of the relay contract, we used a Geth light client (version 1.9.10) to collect 154,445 block headers containing 2,542 branches from the Ethereum main network from 17.12.2019 to 14.01.2020. Note that we also count uncle blocks as branches, since—when submitted to the relay contract—they would introduce a new branch. We then feed these block headers into the three prototypes that are deployed as smart contracts on a private development blockchain running on a Parity Ethereum node (version 2.6.8-beta, –config dev). All three prototypes are initialized with block #9121452 as genesis block.

In the first experiment, we analyze the operating cost (i.e., the cost of submitting block headers), the cost of on-chain SPVs, and whether the source blockchain is correctly replicated within the relay contract. For that, we continuously submit all block headers of our dataset to each prototype. The headers are submitted in ascending order according to their block numbers and timestamp fields. After each submission, an SPV on the genesis block (block #9121452) is triggered. Since the replicated header chain within the relay contract grows after each submission, the algorithm checking whether block #9121452 is part of the main chain has to deal with a growing number of headers. This allows us to observe the cost of executing SPVs with an increasing search depth.

We measure the cost of each operation by the gas consumption of its corresponding Ethereum transaction. Furthermore, after each submission, we log the head of the main chain and the currently submitted header's branch id and branch junction within the relay contract. This enables us to verify that the submitted headers of the source blockchain (i.e., the Ethereum main network) are correctly replicated within the prototypes running on the destination blockchain (i.e., private development blockchain).

To measure the cost of header disputes, we repeat the first experiment, however, instead of performing an SPV after each submission, we trigger a dispute on the genesis block (block #9121452). For simplicity, as we are primarily interested in the cost caused by the removal of branches, we remove the branch originating from the disputed header regardless of the actual result of the header validation. After each dispute, all removed headers are resubmitted to restore the state. This allows us to observe the dispute cost with a growing number of headers that have to be pruned. The cost of each dispute (in gas) is logged only for prototypes *ETH Relay 1* and *ETH Relay 2*, since prototype *Baseline* already performs the full header validation at time of submission.

[1]https://github.com/pantos-io/ethrelay
[2]https://github.com/pantos-io/go-ethrelay
[3]https://github.com/pantos-io/ethrelay-evaluation

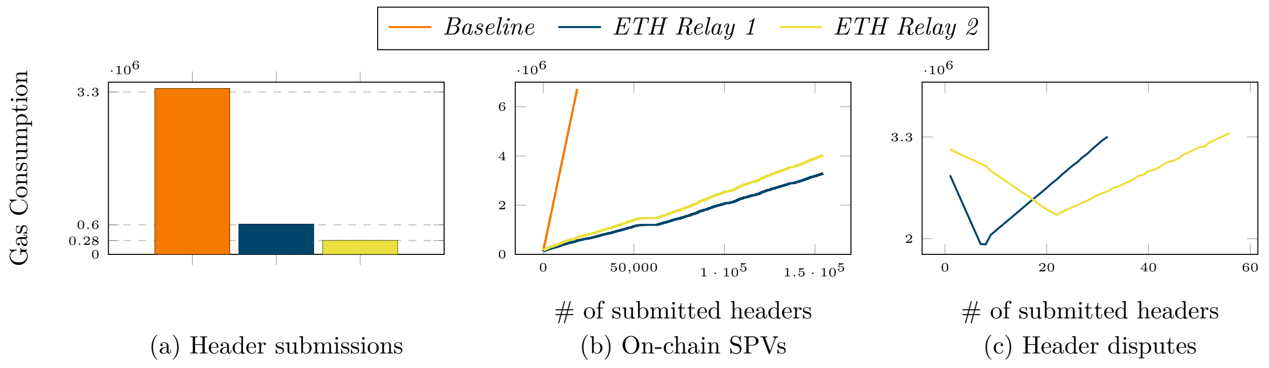

Fig. 4. Gas consumption of the relay contract

B. Results

Figure 4 depicts the results of our experiments. Figure 4(a) shows the average gas consumption per header submission for each prototype. With 612,348 gas (standard deviation of 6,592 gas), *ETH Relay 1* achieves a significant cost reduction of 82% over *Baseline* (average gas consumption of 3,369,653 gas, standard deviation of 5,101 gas). By applying the content-addressable storage pattern, *ETH Relay 2* reduces the average gas consumption of *ETH Relay 1* by 54%, resulting in gas cost of 284,041 for every submitted header (standard deviation of 3,679 gas). Compared to *Baseline*, *ETH Relay 2* reduces submission cost by 92%.

Figure 4(b) depicts the cost for executing on-chain SPVs on the genesis block (block #9121452) for each prototype. The x-axis denotes the number of succeeding block headers that have already been submitted to the relay. Since SPVs are always performed on the genesis block, the search algorithms verifying the membership on the main chain have to cope with an increasing search depth. Prototype *Baseline* using the naïve search algorithm reaches the private blockchain's block gas limit of 6.7 million gas already after 18,766 submitted headers. *ETH Relay 1* and *ETH Relay 2* can cope with the growing search depth at much lower cost.

Notably, *ETH Relay 2* is slightly more expensive than *ETH Relay 1* due to the implementation of the content-addressable storage pattern requiring the full block header to be provided at every execution of an SPV. Hence, applying this pattern is a trade-off between low submission cost and slightly higher cost for SPVs. Notably, gas consumption is measured in a worst-case scenario where each block header is submitted to the relay even if it may not be part of the actual main chain of the source blockchain. In practice, the verification cost measured for *ETH Relay 1* and *ETH Relay 2* may be much lower since clients may be reluctant to submit headers which are not part of the main chain since these headers will not yield a profit.

Figure 4(c) shows the dispute cost measured for *ETH Relay 1* and *ETH Relay 2* (*Baseline* has no dispute cost at all, since it does not implement the validation-on-demand pattern). Despite the fact that a growing number of headers has to be removed with each dispute, the dispute cost of both prototypes temporarily declines. This is because freeing up contract storage in Ethereum-based blockchains yields a so-called gas refund which is given at the end of a successful transaction execution [25]. From a certain point on (after nine headers for *ETH Relay 1* and 23 headers for *ETH Relay 2*), the dispute cost starts to rise which is caused by the design rationale that the gas refund is capped up to a maximum of the half of the total gas consumed by a transaction [25]. Furthermore, as shown in the figure, *ETH Relay 1* reaches the block gas limit much earlier than *ETH Relay 2*. Notably, the gas refund is given only after the successful execution of a transaction, i.e., reaching the block gas limit makes the transaction fail without yielding any gas refund. This is why both graphs stop at around 3.3 million gas (last successful disputes) before reaching the block gas limit. If a branch is too long to be disputed within a single invocation, the dispute function can be called multiple times with each invocation pruning one part of the illegal branch.

In the first experiment, the branch id and branch junction of each submitted header as recorded by each of the three prototypes were logged. This data allows us to verify whether all branches of the dataset have been correctly replicated within the relays. In particular, we extracted all unique junctions from the results as well as from our dataset. A comparison of both lists shows that all 2,542 branches were correctly recognized by the three prototypes.

VIII. Conclusion

For Ethereum-based blockchains, existing relay schemes are either very costly, rely on authorized parties, or may not be technologically feasible. As a solution to this problem, we introduced ETH Relay, a novel relay scheme especially suited for Ethereum-based blockchains. ETH Relay uses a validation-on-demand pattern combined with a sophisticated incentive structure to motivate honest participation. Our evaluation shows that ETH Relay is able to reliably verify the inclusion of transactions across blockchains while reducing the operating cost in comparison to traditional blockchain relays by up to 92%. ETH Relay does not require trust in a centralized party. As such, it provides a further step in enabling decentralized interoperability between blockchains. While the cost benefits of ETH Relay are evident for EVM-based blockchains, the proposed relay scheme can be leveraged whenever the block header validation of the source blockchain is so costly that validating every single block header on the destination blockchain is too expensive. Hence, the basic approach presented in this paper could also be applied with regard to other blockchain technologies.

In the current approach, every block header of the source blockchain needs to be submitted to the destination blockchain. In future work, in an effort to reduce submission cost even further, we will investigate the feasibility of batch submissions of Ethereum block headers using zero-knowledge proofs similar to the approach taken by zkRelay for Bitcoin block headers. Further, we will look into optimization opportunities for verifying main chain membership. While there exist graph algorithms capable of verifying main chain membership in constant time, these algorithms often require a precomputation phase with additional storage demands possibly increasing submission cost [20]. In upcoming research, we will investigate the trade-off between submission cost and optimizations of the on-chain execution of SPVs.

ACKNOWLEDGEMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

REFERENCES

[1] S. Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. White Paper, Accessed 2019-09-19. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[2] C. Prybila, S. Schulte, C. Hochreiner, and I. Weber. "Runtime verification for business processes utilizing the Bitcoin blockchain". In: *Future Generation Computer Systems* 107 (2020), pp. 816–831.

[3] P. Ruan, G. Chen, T. T. A. Dinh, Q. Lin, B. C. Ooi, and M. Zhang. "Fine-Grained, Secure and Efficient Data Provenance on Blockchain Systems". In: *PVLDB* 12.9 (2019), pp. 975–988.

[4] M. Sigwart, M. Borkowski, M. Peise, S. Schulte, and S. Tai. "Blockchain-based Data Provenance for the Internet of Things". In: *9th International Conference on the Internet of Things*. ACM, 2019, 15:1–15:8.

[5] F. Tian. "An agri-food supply chain traceability system for China based on RFID & blockchain technology". In: *2016 13th International Conference on Service Systems and Service Management*. IEEE, 2016, pp. 1–6.

[6] M. Mettler. "Blockchain technology in healthcare: The revolution starts here". In: *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services*. IEEE, 2016, pp. 1–3.

[7] S. Schulte, M. Sigwart, P. Frauenthaler, and M. Borkowski. "Towards Blockchain Interoperability". In: *Business Process Management: Blockchain and Central and Eastern Europe Forum*. Vol. 361. LNBIP. Springer, 2019, pp. 1–8.

[8] V. Buterin. *Chain Interoperability*. Accessed 2019-09-19. URL: https://www.bubifans.com/ueditor/php/upload/file/20181015/1539602892605747.pdf.

[9] *BTC Relay*. Accessed 2019-09-25. 2016. URL: http://btcrelay.org/.

[10] L. Luu. *PeaceRelay: Connecting the many Ethereum Blockchains*. Accessed 2019-09-25. 2017. URL: https://medium.com/@loiluu/peacerelay-connecting-the-many-ethereum-blockchains-22605c300ad3.

[11] L. Luu, Y. Velner, J. Teutsch, and P. Saxena. "SmartPool: Practical Decentralized Pooled Mining". In: *26th USENIX Security Symposium*. USENIX, 2017, pp. 1409–1426.

[12] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber. "On the privacy provisions of bloom filters in lightweight bitcoin clients". In: *Proceedings of the 30th Annual Computer Security Applications Conference*. 2014, pp. 326–335.

[13] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. *Bitcoin and cryptocurrency technologies: A comprehensive introduction*. Princeton University Press, 2016.

[14] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. "A Survey on Blockchain Interoperability: Past, Present, and Future Trends". In: *arXiv:2005.14282* (2020).

[15] T. Baneth. *Waterloo—a Decentralized Practical Bridge between EOS and Ethereum (Part 1)*. Accessed 2019-11-19. 2019. URL: https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-1c230ac65524.

[16] T. Baneth. *Waterloo—a Decentralized Practical Bridge between EOS and Ethereum (Part 2)*. Accessed 2019-11-19. 2019. URL: https://blog.kyber.network/waterloo-a-decentralized-practical-bridge-between-eos-and-ethereum-c25b1698f010.

[17] M. Westerkamp and J. Eberhardt. "zkRelay: Facilitating Sidechains using zkSNARK-based Chain-Relays". In: *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2020.

[18] J. Eberhardt and S. Tai. "On or off the blockchain? Insights on off-chaining computation and data". In: *6th European Conference on Service-Oriented and Cloud Computing*. Vol. 10465. LNCS. Springer, 2017, pp. 3–15.

[19] F. Tschorsch and B. Scheuermann. "Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies". In: *IEEE Communications Surveys & Tutorials* 18 (2016), pp. 2084–2123.

[20] R. Jin, N. Ruan, Y. Xiang, and H. Wang. "Path-Tree: An Efficient Reachability Indexing Scheme for Large Directed Graphs". In: *ACM Transactions on Database Systems* 36.1 (2011).

[21] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. "BAR Fault Tolerance for Cooperative Services". In: *20th ACM Symposium on Operating Systems Principles*. ACM, 2005, pp. 45–58.

[22] M. Herlihy, L. Shrira, and B. Liskov. "Cross-chain Deals and Adversarial Commerce". In: *PVLDB* 13.2 (2019), pp. 100–113.

[23] A. Judmayer, N. Stifter, A. Zamyatin, I. Tsabary, I. Eyal, P. Gazi, S. Meiklejohn, and E. R. Weippl. "Pay-To-Win: Incentive Attacks on Proof-of-Work Cryptocurrencies". In: *Cryptology ePrint Archive* (2019). https://eprint.iacr.org/2019/775.

[24] B. Ford and R. Böhme. "Rationality is Self-Defeating in Permissionless Systems". In: *arXiv:1908.03999* (2019).

[25] G. Wood. *Ethereum yellow paper*. Accessed 2019-09-25. 2014. URL: https://ethereum.github.io/yellowpaper/paper.pdf.