

Trusted Data Notifications from Private Blockchains[#]

Dushyant Behl, Palanivel Kodeswaran, Venkatraman Ramakrishna, Sayandeep Sen, Dhinakaran Vinayagamurthy
IBM Research India

Abstract—Private blockchain networks are used by enterprises to manage decentralized processes without trusted mediators and without exposing their assets publicly on an open network like Ethereum. Yet external parties that cannot join such networks may have a compelling need to be informed about certain data items on their shared ledgers along with certifications of data authenticity; e.g., a mortgage bank may need to know about the sale of a mortgaged property from a network managing property deeds. These parties are willing to compensate the networks in exchange for privately sharing information with proof of authenticity and authorization for external use. We have devised a novel and cryptographically secure protocol to effect a fair exchange between rational network members and information recipients using a public blockchain and atomic swap techniques. Using our protocol, any member of a private blockchain can atomically reveal private blockchain data with proofs in exchange for a monetary reward to an external party if and only if the external party is a valid recipient. The protocol preserves confidentiality of data for the recipient, and in addition, allows it to mount a challenge if the data turns out to be inauthentic. We also formally analyze the security and privacy of this protocol, which can be used in a wide array of practical scenarios.

I. INTRODUCTION

The recent past has seen the emergence of private blockchain platforms as viable decentralized transaction-processing systems for businesses and consortia [1]–[4]. Similar to public blockchain platforms like Ethereum [5], private blockchain platforms support smart contracts running on networks of peers, updating and recording data on a shared replicated ledger through a distributed consensus protocol. However, network membership and access to ledger information is governed by designated network authorities, enabling such networks to guarantee higher privacy and accountability that are critical for enterprise scenarios. Furthermore, a private network can use a consensus protocol suited to the applications running on the network (e.g., PBFT [6], Raft [7]) rather than expensive ones like PoW [8], delivering higher performance. This enables consortia to balance performance and scalability for their specific applications by picking and choosing the right configuration and policy parameters. It is this customizability that makes private blockchains amenable to large scale industry adoption.

Motivated by the aforementioned benefits, a number of businesses and consortia have invested significantly to create and manage private blockchains, running a wide

range of applications including trade and supply chains [9], finance [10], [11], regulatory compliance [12], provenance [13], and real estate [14], [15]. Network participants, typically members of a consortium, collectively control network membership and access to ledger information. As a result, private blockchain networks have turned into “walled gardens”, and the data and assets managed by them lie within silos. But this presents a challenge to real world business processes, which typically involve far more parties that have an interest in a private blockchain network’s data than that network can accommodate either due to scalability or privacy constraints. Clearly there is a need for **trusted data notifications** of private blockchain data to an extended set of interested parties while still respecting the privacy concerns of the private blockchain members.

To elucidate, we refer to Figure 1 depicting a real-estate blockchain network tracking the lifecycle and ownership of property titles. The network can be constituted with membership restricted to property owners and government regulators. We note that properties registered in the network can be involved in transactions outside the scope of this network. Land can be mortgaged or used as collateral in a business transaction, where the bank or counterparty involved is not a member of the real estate network. A bank, for example, has no visibility into the network’s ledger and does not know whether the property has changed ownership or undergone transformation of any kind such as division amongst inheritors. Given the bank’s lack of visibility into the network, a dishonest property owner has the opportunity to defraud the bank by re-mortgaging the land to another bank. In fact, in a blockchain ecosystem, the bank cannot rely on information from any single member of the network because ledger data is collectively agreed upon, and a blockchain’s trust emanates from this collective endorsement of data. Property owners who have no connection with the bank may not want to expose their assets and transactions to it for privacy and business reasons. The right approach to solve this problem is to enable the network to collectively share just the necessary data with the bank, on a need to know basis, in a secure manner, without leaking ledger data to unintended recipients.

It is important to note that the value of the ledger data stems from it being *collectively agreed upon by the members of the network*. We term this as *extrinsic* value as opposed to *intrinsic* value, where the data content is valuable to external participants regardless of who supplied it. Blockchain data

[#] Authors are listed in alphabetical order.

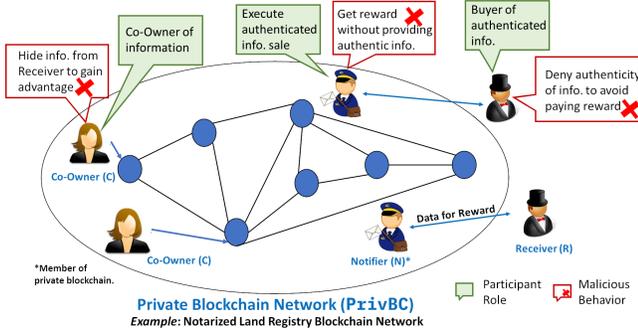


Figure 1. Private Blockchain network for real-estate ownership

sans evidence of collective agreement by network members has no extrinsic value to an external recipient, say as evidence in a court of law. Further, in business processes where data with extrinsic value is required, it is crucial that the data is known to have been procured through legitimate means; i.e., the data carries information clearly indicating that the network members collectively authorize a particular non-member to use it. This is necessary to protect the privacy and business interests of the network, and is evident in our property mortgage use case: registry network members sharing records with unauthorized banks may face audits and penalties. This condition becomes important in our solution design as described in later sections.

Note that usage of trusted data notifications is well-entrenched and legally accepted in modern economic and commercial processes. Examples include getting electronic signatures endorsed by respective agencies [16], credit-worthiness statements endorsed by ratings agencies [17], warrants of physical presence of goods endorsed by warehouses [18], and documents of ownership of real estate (or other assets) endorsed by respective agencies [19].

Unfortunately, no methods currently exist for private networks to supply trusted notifications of extrinsically valuable data to outsiders. *Members of private blockchains* as a group can themselves act as trusted sources of data and provide notary services to external members in specific scenarios [20], [21]. As an added incentive, supplying data in exchange for money may also enable a consortium’s members to monetize and recoup their investments in maintaining a blockchain network.

Therefore, we set out to create *trusted* mechanisms to enable a private blockchain network to share private ledger data authorized for a non-member by the network securely and confidentially in exchange for a monetary reward. In our model, protecting the privacy and business interests of the network is crucial: the consortia members should collectively endorse not only the veracity of data but also the set of external members with whom that data can be shared in an authorized manner. The solution we will propose relies on existing mechanisms to generate collective endorsements,

typically using digital signatures [1].

Our problem is essentially a variant of *Fair Exchange* [22] between two untrusting parties: private blockchain network members (as a collective) and an external data recipient. Each party faces the risk of being cheated by its counterparty unless precautions are taken. For example, a naive solution in our property scenario would involve the interested bank periodically querying peers (owned by members) of the private network (at least a majority) to get updates on properties of interest. But this solution is neither scalable (because of polling overhead) nor lucrative (lack of incentives for members to share data). An alternate solution, where the bank is notified by any member of the private network in return for a reward, is no better as the member may supply fake data or the bank may renege on making a payment.

Two-party fair exchange is known to be infeasible in the absence of a trusted third party [23]. The solution that we will present uses a public blockchain with programmable money constructs as a trusted arbitrator to guarantee fairness for both the external recipient (like a bank) and the private network member group. In our solution model, a transaction (data in exchange for money) is executed by a *Notifier*, a role that can be assumed by any member of the private network to supply data and earn a pre-determined reward (for itself and members of the private blockchain) if the data is authentic (see Figure 1). At the heart of our solution lies a pair of interlocking contracts to ensure that the receiver gets authentic information (containing private blockchain network endorsements) if and only if all private network members get their reward. Our protocol, while inspired by Hashed Time Lock Contracts (HTLC) [24], ensures authenticity with confidentiality by leveraging zero knowledge and encryption mechanisms.

Key contributions: In this paper, we make the following contributions 1) We motivate and present the design of our solution for trusted data notification, *Evidence Time Locked Contracts (ETLC)* which uses zero knowledge proofs and standard cryptographic techniques to guarantee fairness in the exchange of private blockchain data with external authorized recipients. 2) We formally define the claims made by our solution and present a security analysis of our protocol. In particular, our solution guarantees fair exchange between the private network members’ group and the receiver, and confidentiality of blockchain data 3) Our solution further guarantees that the network’s privacy policies are always enforced and even a rogue notifier cannot exfiltrate valid notifications to unauthorized recipients.

We organize the rest of the paper as follows. First, we discuss our problem setting and list the building blocks for our solution in Section II. We describe our protocol ETLC in detail in Section III. We formally analyze the correctness, security, and privacy features of the protocol in Section IV, survey related work in Section V, and conclude with our ideas for future research in Section VI.

II. PROBLEM SETTING AND PRELIMINARIES

In this section, we model the problem and describe the building blocks of our proposed solution.

A. Problem Setting

A private blockchain network, denoted by *PrivBC*, is collectively governed by a group of entities (sometimes called *consortium*) that need to cooperate for business reasons but do not completely trust each other. Though the technologies they are built on vary widely in structure and function [1]–[4], we can model them commonly as $\langle \text{Peers}, \text{SC}, \text{C}, \text{L}, \text{M} \rangle$, where:

- **Peers** denotes the peers belonging to different business entities (sometimes called *organizations*)
- **SC** denotes the smart contracts running business logic
- **C** denotes the consensus protocol to achieve finality and ordering of transactions
- **L** denotes the ledgers maintaining chains of blocks and state snapshots
- **M** denotes membership policy, governing entry and access to *PrivBC*

A transaction is typically submitted by an application client, pre-authorized using **M**, to the network. Signatures (endorsements) are collected from a subset of **Peers** (the list is determined by consensus policy) running appropriate **SC** functions. Subsequently, the transaction is validated using pre-configured consensus policies before it is committed independently by each peer to its replica of a ledger. Such a commitment consists of a given ledger data **LD** item with an updated value, and this is stored in the ledger along with peers' endorsements.

The consensus protocol **C** ensures that data committed to the blockchain represents the consortium's collective will, the proof of which lies in the endorsements associated with ledger data (**LD**). To an external entity, **LD** coupled with endorsements may possess extrinsic value (see Section I), however, the membership policy **M** ensures that *PrivBC* ledger data is accessible only to the consortium's members.

One approach to make **LD** available outside the network is for *PrivBC* to expose a *Verifiable Read Service (VRS)* [25] API that allows external members to query for a given **LD**. In response, *PrivBC* will return **LD** along with endorsements and other metadata if the requestor passes an access control check enforced collectively by **Peers**. However, the use of a *VRS* raises two challenges: (i) the peers lack incentive to offer such a service freely to external members, and (ii) the pull model requires the external member to continuously poll for updates from *PrivBC*. In our solution *ETLC*, we address (i) using an incentive mechanism for network members, and (ii) using a notification based push model. In this push based model, any member of *PrivBC* may assume the role of notifier **N**, sharing updates to **LD** with a designated external receiver **R**. In return, **R** pays a reward to the members of *PrivBC* after verifying the authenticity of the

data. This now reduces to a fair exchange problem [22] between the **N** and **R** requiring the use of a trusted third party arbitrator to guarantee fairness [23]. In *ETLC*, we leverage a public blockchain network (accessible to **R** and members of *PrivBC*) as a trusted arbitrator, exploiting its programmable money constructs to guarantee fairness of the ledger data-reward exchange.

The core exchange mechanism we will use in *ETLC* is inspired by *Hashed Time Lock Contracts (HTLC)* [24], a mechanism that allows two parties on two different public blockchains to swap assets with each other. In an *HTLC* instance, one of the parties deploys a contract on one network promising to transfer an asset to the other party when a secret preimage x , known only to it and corresponding to a hash $H(x)$ encoded in the contract, is supplied ($H(\cdot)$ is a standard hash function like SHA-256). The other party deploys a similar contract encoded with the identical hash on the other network, promising to transfer another asset to the first party. Upon production of a secret s , the hash of which matches x , transfer of assets happen in both directions without either party getting an opportunity to renege on its pledge (hence *hash lock*). These contracts are also *time locked* i.e. if either party fails to submit a transaction with the right preimage within a given time t , the escrowed asset is returned to its owner.

B. Trusted Data Notification Model

We now formally describe the *PrivBC* trusted data notification problem and assumptions of our threat model.

We consider the problem of enabling an external receiver **R** (that is not a member of *PrivBC*) to be notified of a blockchain event e of interest in return for paying a reward. In our current model e represents the creation, updation, or deletion of a specific ledger data **LD** item that **R** is interested in. We define *Notifiers* $\{N\}$ (consisting of *PrivBC* members) to be any set of entities that notify e to **R** in exchange for a reward a . *PubBC* is a public blockchain such that **R**, **N** and all members of *PrivBC* (including **Peers** and clients) own accounts and can transfer money (native assets, like cryptocurrency) to others on the network. Importantly, we assume that e holds *extrinsic* value for **R** and hence carries utility only if it can be proven to (i) emanate from *PrivBC* and (ii) authorized by *PrivBC* for **R**.

1) *Threat Model*: We make the following assumptions:

- **R is rational**: Receiver **R** has a genuine interest (economic or otherwise) in receiving event e and has no incentive to broadcast private blockchain data to others.
- **N is rational**: Notifier **N** is incentivized to notify **R** about e if reward is guaranteed in return.
- **PrivBC is trustworthy**: The private blockchain is viewed as a collection of peers trusted to finalize transactions and record data on the ledger using an advertised consensus protocol, even if some peers may act dishonestly. We assume that the members of *PrivBC* will not collectively collude to

cheat R by supplying fake data or endorsements. We also assume the membership service of the network is trustworthy and issues valid certificates to the PrivBC members.

- **PubBC is trustworthy:** The public blockchain’s consensus protocol is trusted to accurately execute smart contracts and record transactions. We also assume that PrivBC members cannot collude and corrupt the consistency of PubBC.

2) *Goals:* Our protocol must ensure the following:

- **Fair Exchange :** Each member of PrivBC (including N) must obtain a reward in PubBC for every valid e that is successfully delivered to R. This protects N from a malicious R that receives an endorsed LD but later denies payment to N by claiming non-delivery. Similarly, R is protected against a malicious N that delivers an e that either (i) does not represent ledger data or (ii) has no associated endorsements from peers proving validity or authorization.

- **Confidentiality:** A notification can be delivered only to an R that is authorized by PrivBC to receive notifications. This prevents unintended recipients from receiving an LD carrying endorsements proving both its validity and the network’s authorization for R. Since we are using a public blockchain as an arbitrator, this is a salient concern.

III. SOLUTION: TRUSTED DATA NOTIFICATION PROTOCOL

In this section, we will describe our protocol ETLC for the notification of valid ledger data from a private blockchain (PrivBC) to an intended external recipient (R) in exchange for a reward.

A. Smart Contracts

ETLC utilizes a set of smart contracts deployed on PrivBC and the PubBC. These contracts embed logic to record and verify various artifacts of hashes, ciphertext, keys, signatures etc. to enable the guarantees provided by our solution. For instance, some of these contracts utilize *hash and time locks* as discussed in Section II-A, supported by public blockchains like Ethereum [5] to guarantee liveness of the protocol. The list of contracts and their functions are listed in Table I. We will now describe how the contracts are used in our protocol construction.

B. ETLC Protocol

In ETLC, every unique external recipient of a private ledger data element goes through four sequential stages, as illustrated in Figures 2–5.

Note that in each stage diagram, we use P to denote all members of PrivBC, excluding N; C (Co-Owner) is any member in P that creates or updates ledger data (LD) of interest to R. Any member of PrivBC can assume the role of N in any given instance without requiring changes to the protocol. For rewarding purposes, we will assume that every member in P has an identity and crypto-currency account in PubBC for conducting transactions. We now describe each process in more detail.

Table I
SYSTEM CONTRACTS

Name	Network	Description	Operations
SC-ACL	PrivBC	Maintains access control list: <i>subjects</i> are external recipients (identified by their public keys) and <i>objects</i> are ledger data elements. Encrypts updated ledger data for external notifications.	<i>PermitAccess, RevokeAccess, RecordEncDataHashProof</i>
SC-Reward	PubBC	Guarantees rewards to notifiers after due verification of the recipient having received authentic data, utilizing HTLC techniques.	<i>RecordPubKey, RecordCipherTextHashProof, RecordSignature, VerifySignature, Validate</i>
SC-R-Sign	PubBC	Guarantees amount in exchange for authentic signature submitted before timeout.	<i>RecordSignature</i>
SC-N-Key	PubBC	Guarantees amount (identical to SC-R-Sign) in exchange for valid decryption key (symmetric) submitted before timeout.	<i>RecordKey</i>

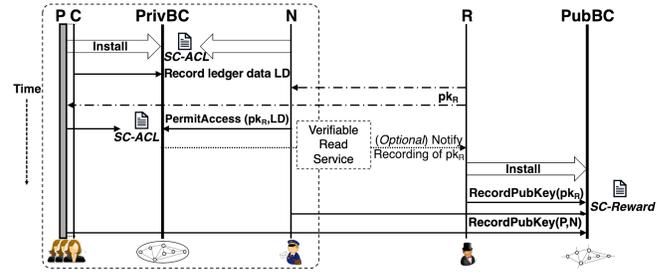


Figure 2. End-to-End Protocol: Bootstrap Stage

1) *Bootstrap Stage:* The *Bootstrap stage* need only be done once for a given $\langle \text{recipient}, \text{LD} \rangle$ pair. In Figure 2 first, the *SC-ACL* contract must be installed in PrivBC through the network’s consensus process. This needs to be done just once in the lifecycle of the network. This contract supports the registration of external recipients’ public keys (or certificates) for identification and cryptographic purposes. We assume that recipient R can communicate its public key (pk_R) separately to every member of the network as indicated in Figure 2, and that the key is committed through consensus to ledger using *SC-ACL*’s *PermitAccess* operation. Optionally, R may submit pk_R within a certificate signed by authorities that can be validated in this operation. As the figure also indicates, we assume the data item that R has been given access to (LD) is already recorded on the ledger.

R can confirm through a *VRS* (see Section II-A) that (i) pk_R has been associated with LD on PrivBC’s ledger, and (ii) that PrivBC has authorized R to receive any future updates made to LD.

2) *Generation Stage:* The goal of the generation step is for N to commit to the contents of the notification while also proving (using a zero knowledge proof) that the notification contains valid updates from the ledger. When LD is updated on PrivBC’s ledger by any C, a call to *SC-ACL* is triggered using PrivBC’s transaction submission mechanism. At this point, LD already has signatures endorsing it on the chain (ledger), so this call can package LD with the signatures

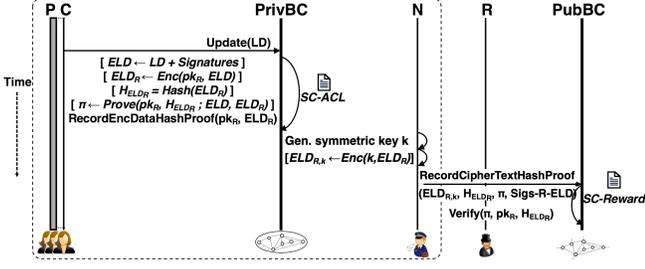


Figure 3. End-to-End Protocol: Generation Stage

into ELD (or endorsed LD). The following are produced and recorded on the ledger using the *RecordEncDataHashProof* operation, as illustrated in Figure 3:

- ELD_R : a deterministic public-key encryption of ELD using pk_R ensuring only R can decrypt the notification
- H_{ELD_R} : a hash of ELD_R as a commitment¹
- π_{ELD_R} : a zero-knowledge proof that can be used to verify, without knowing ELD_R , that ELD_R was produced by encrypting some plaintext under pk_R .

We will use the Encrypt-with-Hash construction in Bellare et al. [26] as our deterministic encryption scheme with the ElGamal encryption scheme [27] as the underlying randomized public-key encryption scheme. This deterministic encryption provides PRIV-CCA security [26] and it's good enough to provide confidentiality to ELD since the underlying unforgeable signatures are 'unpredictable'. For this encryption, the zero-knowledge proof π_{ELD_R} is generated using a pair of discrete-log exponent checks followed by a hash check.

N now privately generates a symmetric key k (we use AES-CTR²) and further encrypts ELD_R into $ELD_{R,k}$. This second layer of encryption is to prevent R from learning ELD_R (and hence ELD) before the next step. On PubBC, N records the following through the *RecordCipherTextHashProof* operation on the *SC-Reward* contract:

- Doubly encrypted data ($ELD_{R,k}$)
- Hash H_{ELD_R} of the singly encrypted ELD_R , which is recorded on PrivBC's ledger after consensus among its members
- Zero-knowledge proof π_{ELD_R}
- Signatures validating PrivBC's collective endorsement of R's authorization to access LD: *Sigs-R-ELD*; generated on the private chain upon successful conclusion of the *RecordEncDataHashProof* operation

The proof π_{ELD_R} is now verified using pk_R , which was recorded on the ledger in the bootstrap stage, and H_{ELD_R} . If the verification succeeds, i.e., the proof and the hash are

¹a randomized commitment scheme is used, with the randomness/opening to the commitment recorded on PrivBC and later recorded on PubBC during the KeyTransfer stage.

²IND-CPA security provided by AES-CTR is sufficient as we show in section IV

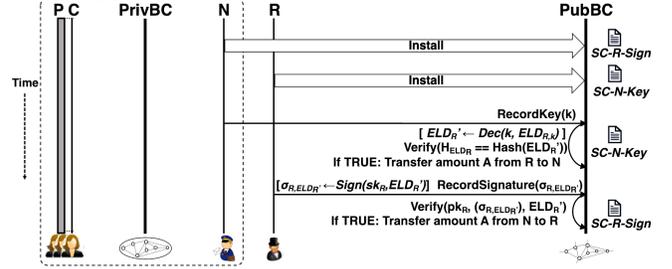


Figure 4. End-to-End Protocol: Key Transfer Stage

authentic, it implies that the hash was produced from data which itself was generated by encrypting a different data element using pk_R . (We will see why this is useful when the preimage of the hash is revealed in the next stage.) If the verification fails, the protocol terminates at this stage.

The remaining steps of the protocol occur on PubBC and do not involve PrivBC.

3) *Key Transfer Stage*: The goal of the key transfer stage is to enable N to share the symmetric key k with R in exchange for collecting a signature from R that allows N to collect its reward in later stages of the protocol. The main innovative feature of our protocol, illustrated in Figure 4, is a pair of interlocking contracts, inspired by the HTLC mechanism, that allow N to share its symmetric key (k) in exchange for R's signature on the resulting decrypted data. First, N must install contract *SC-R-Sign*, which obligates N to transfer an amount A to R if the latter submits a verifiable signature over data generated by decrypting $ELD_{R,k}$ using k . Once R can see this contract installed, it must install *SC-N-Key*, which promises to transfer the same amount A to N if the latter provides a valid key k .

Let us assume N supplies k to *SC-N-Key*. Decryption of $ELD_{R,k}$ produces ELD'_R , which ought to be identical to ELD_R produced in the generation stage. Verification of this involves checking that the hash H_{ELD_R} is indeed a hash of ELD'_R . If this check succeeds, it also proves that the preimage ELD_R is ciphertext that was generated by encrypting some plaintext using pk_R ; this follows from the successful verification of the zero-knowledge proof π_{ELD_R} at the end of the Generation Stage. A is transferred to N as promised if verification succeeds; otherwise, no amount is transferred and the protocol terminates.³

Now R must play its part by supplying a receipt in the form of a signature over the decrypted data ELD'_R to *SC-R-Sign*: σ_{R,ELD'_R} . If it provides a valid signature that can be verified using pk_R , amount A is transferred back from R to

³Stopping the protocol if N produces an ELD_R encrypted under a different public key is not mandatory. As an alternative, we can delay the validity check on ELD_R till the final verification stage. This will make R proceed till the final step even if ELD_R is invalid. But, the collision resistance property of the hash function will be enough to ensure honest behaviour of R and N without the need for zero-knowledge proofs.

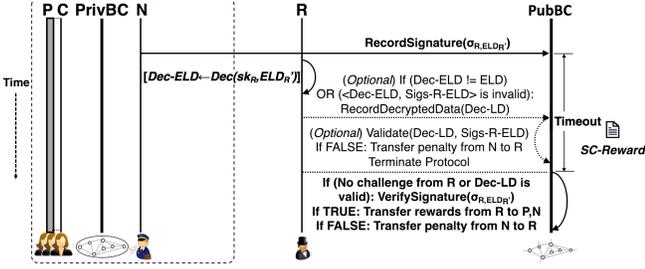


Figure 5. End-to-End Protocol: Verification and Reward Stage

N, and the net monetary exchange is zero. If R does not supply a signature within a given time period or supplies an invalid signature, it ends up forfeiting A.

Last but not least, both *SC-R-Sign* and *SC-N-Key* contain embedded timeouts (similar but not identical to time locks). *SC-N-Key* has a timeout t just to ensure that R does not wait indefinitely for N to provide k . If t expires, the protocol terminates without anyone losing money. But if N does supply a valid k within t , R is obliged to provide its receipt signature within a timeout t' encoded in *SC-R-Sign*. t' must be chosen to be significantly greater than t so that R gets ample opportunity to supply a signature after N supplies k .

4) *Verification and Reward Stage*: In the final stage, our protocol verifies the validity of the notification received by R and rewards PrivBC members for a successful notification. To claim the reward (a) for itself and all members of PrivBC, N can now submit R's signature σ_{R, ELD'_R} , obtained in the previous stage as evidence to *SC-Reward* (see Figure 5). A timer starts upon this submission, and R is given the opportunity to challenge the veracity of the received data within a given time period (call it Timeout). R now has ELD'_R , which can be decrypted using sk_R , the private key corresponding to the public pk_R .

The produced Dec-LD ought to be well-formed, contain LD and enough signatures from PrivBC members to prove consensus. The set of enclosed signatures should have been created by members whose public keys were recorded to PubBC in the Bootstrap Stage, and the set of signatories should match those of Sigs-R-ELD, supplied in the Generation Stage to prove that R was authorized to get access to LD. Signatures in Sigs-R-ELD should be validated as well, given that ELD (LD plus signatures) has been decrypted. The value of LD should be fresh and not an older version (a form of replay attack mounted by N).

If any of these conditions are unsatisfied, R can choose to record Dec-LD on the ledger and *SC-Reward* can verify whether R's claims about receiving invalid data are correct. If that is the case, no reward is transferred to members of PrivBC, an amount is transferred from N to R as penalty for spurious information, and the protocol terminates. Otherwise, when Timeout occurs, R's receipt signature σ_{R, ELD'_R} is verified. Upon success, reward amount a is transferred from

R to PrivBC's members, with N getting an extra amount to cover its transaction costs in PubBC. Upon failure, R pays no reward and N pays an amount in penalty to R. The protocol now terminates.

C. Discussion

We now discuss some specific aspects of ETLC in more detail.

- **Handling Multiple Concurrent Notifiers**: It is possible that multiple N concurrently try to run our protocol for the same LD. To protect against this, we suggest that an additional version attribute be attached to each LD (to reject older updates to LD) and the smart contracts follow a first-come-first-serve policy to select and designate an N.
- **Collective Authorizations**: We encode access control rules in a smart contract *SC-ACL* to ensure that a decision to allow a given R access to a given LD passes through network consensus, hence expressing collective consent of the network members rather than a decision made by a single authority.
- **Incentive Structure**: A central enabler for our protocol is a cryptocurrency-based public blockchain with contracts deployed on it that incentivize participants to be honest. In ETLC, the intent is to ensure that PrivBC members get their rewards *only* when they fulfill their commitments. We lock the reward in the form of crypto-currency in the *SC-Reward* contract during the bootstrap phase, which can only be claimed by members of PrivBC possessing accounts in PubBC when they provide evidence of fulfilling their tasks. This is done by (i) submitting valid data in the Generation Stage, and (ii) submitting a valid signature from R in the Verification and Reward Stage. Needless to say, the reward amount should be commensurate with the perceived value of information being acquired (including any additional costs e.g. transaction fees on PubBC) and hence left as a use case-specific configurable parameter.

IV. ANALYSIS

We will now analyze ETLC and show that it satisfies the goals listed in Section II-B while relying on the assumptions listed in that section. Cryptographic primitives are expected to provide standard security properties, which we will specify as needed. Additionally, we assume that the public-key encryption scheme is robust [28].

A. Protecting against a malicious member of PrivBC

We now show the guarantees ETLC provides in the presence of a malicious co-owner C and/or notifier N.

1) *Notification authenticity*: ETLC ensures that the notification received by R accurately represents LD after it has been updated in PrivBC.

Claim IV.1. *A rational R accepts a notification as valid only if it corresponds to a valid LD in PrivBC.*

Proof sketch. The members of PrivBC obtain a reward from *SC-Reward* at the end of a *successful* run of the protocol. This happens when R does not submit an invalid Dec-ELD to SC-Reward. At a high level, the proof of this claim will follow the following outline: the unforgeability of signatures and the soundness of zero-knowledge proofs will ensure that a rational R will not submit Dec-ELD only if it corresponds to a valid LD in PrivBC.

To notify R, N submits two transactions to PubBC:

- N first submits $ELD_{R,k}$, H_{ELD_R} , π , Sigs-R-ELD to SC-Reward.
- Then, N records a key k .

These actions lead to the following *automatic* actions by the smart contracts which cannot be altered by N or the members of PrivBC after they are installed:

- *SC-Reward* verifies the zero-knowledge proof π – ensures that N knows of an entry (i) which is an encryption of a message under the public key pk_R and (ii) whose hash is H_{ELD_R}
- *SC-N-Key* decrypts $ELD_{R,k}$ with k , verifies its hash with ELD_R and finally transfers the amount A from R to N if hash is valid – given the above zk proof, collision resistance of H ensures that ELD_R is an encryption of some message under pk_R ; note that the secret key encryption scheme need not be robust [28], even if N has the opportunity to provide a different key k' to decrypt to a different ELD'_R post the submission of $ELD_{R,k}$, since the hash collision-resistance and proof's soundness ensure that the resulting ciphertext is an encryption under pk_R .

The above construct ensures that ELD_R encrypts a message that can be decrypted using R's secret key. Here, R can decrypt and check the validity of the message, confirming that that it is a valid LD with signatures from PrivBC. Even after R acknowledges receipt of k from N by submitting its signature on ELD_R , as a rational actor, it can subsequently submit a decrypted ELD if it turns out to be invalid. This will ensure that N does not get the reward, and additionally is penalized.

2) *Information assurance:* This property of ETLC ensures that R gets notified of any update to LD.

Claim IV.2. *ETLC ensures the delivery of a notification to R for each of its subscribed updates in PrivBC.*

Proof sketch. This is ensured by the rationality of the members of PrivBC, who are incentivized to claim a reward from R through the smart contract *SC-Reward* upon sending an endorsed LD.

B. Protecting against a malicious receiver R

We now discuss the security properties of ETLC to protect against a malicious R.

1) *Reward fairness:* Members of PrivBC and R decide on a reward to be paid by R when R receives a valid notification on LD from N. ETLC ensures that they obtain the agreed upon reward when the notification is received by R.

Claim IV.3. *Members of PrivBC obtain the agreed upon reward if a rational R accepts a notification as valid.*

Proof sketch. R receives the ciphertext ELD_R containing the notification only after N records the key k on PubBC. Even though N commits $ELD_{R,k}$, H_{ELD_R} , π and Sigs-R-ELD to PubBC, R does not learn any information about ELD_R , and hence about ELD, before N records k due to the semantic security of the secret key encryption scheme, hiding of the commitment scheme and the zero-knowledge property of the proof system. After N records k , *SC-N-Key* transfers the amount A from R to N. Since this amount A is higher than the reward that it would send to members of PrivBC, a rational R would send its signature on ELD'_R to confirm its receipt. Unless R submits decryption of an invalid Dec-ELD before the pre-decided challenge timeout, the reward is automatically transferred to the PrivBC members just on R's confirmation of receipt of ELD'_R . If the ELD'_R sent by N is valid, to prevent the reward from being transferred, R should submit a Dec-ELD which upon encryption should yield ELD'_R , since we use a deterministic encryption scheme. But it is not possible for R to submit a different Dec-ELD which results in the same ELD'_R due to the robustness of the public-key encryption scheme.

2) *Notification-receipt undeniability:* This property prevents a malicious R from denying knowledge of ELD after it has obtained ELD through ETLC.

Claim IV.4. *A malicious R will not be able to obtain knowledge of ELD while simultaneously denying receipt of the notification leading to that knowledge.*

Proof sketch. As mentioned in Claim IV.3, R does not learn any information about ELD, and hence about LD, before N records k due to the semantic security of the secret key encryption scheme and the zero-knowledge property of the proof system. Hence, R has to wait for N to record k on PubBC. But a rational R has to send a signature confirming its receipt of ELD_R since we set $A \gg a$. This signature along with the hash and zero-knowledge proof (that H_{ELD_R} is an encryption under pk_R) serve as a proof recorded on PubBC that R has received a message from N that only it can decrypt.

C. Fair exchange

Now, we are ready to prove that ETLC ensures a fair exchange between R and the members of PrivBC.

Claim IV.5. *R receives a valid notification if and only if the members of PrivBC obtain their reward for the notification.*

Proof sketch. Claim IV.3 covers the *only if* part of this

claim. We will argue the *if* part of the claim now. If the members of PrivBC receive their reward, it means that R did not provide an invalid Dec-ELD to PubBC in the Verification and Reward Stage. This implies either that R accepted the notification as valid or that *SC-Reward* rejected a challenge from R. The unforgeability of signatures and Claim IV.1 ensure that a rational R received a valid notification.

D. Validity of all exchanges in ETLC

We now prove an interesting and powerful claim: ETLC ensures that only authorized recipients of ELD end up possessing it.

Claim IV.6. *Only authorized Rs can receive notifications through ETLC.*

Proof sketch. If $\text{PermitAccess}(pk_R, LD)$ (in SC-ACL) had not been completed, there will not be signatures on PrivBC to prove the consensus among PrivBC members that pk_R is permitted to access LD. The unforgeability of the signature scheme ensures that N cannot generate valid signatures on its own. Now, if N does proceed with our protocol to obtain a fair exchange of reward for the data ELD, a rational R after it decrypts ELD'_R to obtain ELD can still proceed to show the absence or the invalidity of the signatures to PubBC to deny N a reward and additionally claim a penalty from it. This prevents a rational N from even initiating the transaction through our protocol if R is not permitted access to LD. Additionally, the PRIV-CCA security of the deterministic encryption scheme ensures that ELD remains confidential from any observer of ETLC messages other than members of PrivBC and R.

V. RELATED WORK

Fair Exchange: As we stated in Section I, trusted data notification, or the exchange of a reward for authentic private network data can be modeled as a form of *Fair Exchange* [22] between the notifier and receiver, where PubBC acts as the trusted third party mediating the exchange. The exchange supported by our protocol has a few salient differences from the classic fair exchange problem that are not covered in literature: (i) one of the counterparties is a decentralized group collectively agreeing to do an exchange and allowing a single spokesperson (N) to execute that exchange in a way that gives both receiver and all group members their due, (ii) the item being exchanged for a reward must be kept confidential from the trusted third party, which in our protocol is a public blockchain.

Atomic Transactions across Networks: HTLC [24] and Atomic Swaps [29] have been proposed as techniques to swap assets between multiple parties across blockchain networks. The InterLedger Protocol (ILP) [30] was proposed to transfer assets between entities across multiple network hops, using HTLC to transfer an asset across each hop.

These mechanisms rely on counterparties having visibility into the participating network ledgers, and therefore cannot be directly applied to the trusted data notification scenario where one network is private and the counterparty (R) has no visibility into the that network’s ledger.

Blockchain Interoperability Frameworks: Relay networks like Cosmos [31] and PolkaDot [32] enable inter-blockchain network transactions of a similar flavor to our scenario, using central blockchain as mediators. Abebe et al. [25] support similar interoperation, at least for data transfer, without relying on a mediating network (serving as a candidate for our *VRS* building block). However, while these protocols enable interoperation, none of these frameworks provide out-of-the-box support for the kind of novel trusted data notification between a private blockchain and an external party which our protocol enables.

Cross Blockchain Data Transfer With Proofs: Numerous mechanism have been proposed to prove the authenticity of data shared from one blockchain to another. In public blockchains, various off-chain proof mechanisms are used to verify the validity of one public blockchain’s transactions in another [33]–[36]. In private blockchains, *proof-of-authority*, or a quorum of signatures, are typically used to attest to the authenticity of network data [2], [25], [31], [32]. However, these techniques were not designed to handle trusted data notifications, and can at best act as supporting features for our novel protocol.

VI. CONCLUSION AND FUTURE WORK

We have presented the design of a trusted data notification protocol for a private blockchain network to shared valid notifications of updates to its ledger data with authorized external entities. Our solution introduces incentives for private blockchain members to participate in, and comply with the rules of, such a protocol. Using a public blockchain as trusted arbitrator, standard cryptographic mechanisms for data confidentiality and integrity, and blockchain patterns like HTLC, our protocol guarantees fairness for both network and recipient. We formally defined the ideal properties of this protocol and proved that our solution satisfies those properties. We believe our protocol is practically usable and serves as a mechanism for private blockchain network processes to interoperate with external processes, the lack of which has inhibited wider adoption of private blockchains in industry and government. In the future, we plan to build a prototype to further validate our design and evaluate its performance for practical needs. We will also extend the protocol to allow other business networks (built on blockchain or other technology) to be receivers, thereby enhancing the abilities of networks built on diverse technology stacks to interoperate with each other.

REFERENCES

- [1] E. Androulaki et al., “Hyperledger fabric: a distributed operating system for permissioned blockchains,” in *EuroSys 2018*.

- [2] M. Hearn and R. G. Brown, "Corda: A distributed ledger," <https://www.r3.com/reports/corda-technical-whitepaper/>, November 2016.
- [3] "Quorum," <https://github.com/jpmorganchase/quorum>, 2019.
- [4] Hyperledger, "Hyperledger sawtooth," <https://www.hyperledger.org/projects/sawtooth>, 2019.
- [5] "Ethereum," <https://www.ethereum.org/>, 2019.
- [6] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *OSDI*. USENIX Association, 1999.
- [7] D. Ongaro, "Consensus: Bridging theory and practice," Ph.D. dissertation, Stanford University, 2014.
- [8] "Bitcoin wiki: Proof of Work," https://en.bitcoin.it/wiki/Proof_of_work, 2019.
- [9] "Tradelens," <https://www.tradelens.com>.
- [10] "We.trade," <https://we-trade.com>, 2019.
- [11] "Marco polo network," <https://www.marcopolo.finance/>, 2019.
- [12] "Ibm verify credentials," <https://www.ibm.com/blockchain/solutions/identity>, 2019.
- [13] "Ibm food trust," <https://www.ibm.com/in-en/blockchain/solutions/food-trust>, 2019.
- [14] Lantmäteriet et al., "The land registry in the blockchain," http://ica-it.org/pdf/Blockchain_Landregistry_Report.pdf, 2016.
- [15] Qburst, "Blockchain based land registration system," <https://www.qburst.com/resources/case-studies/blockchain-land-registry/>.
- [16] European Commission. eSignature - create and verify electronic signatures in line with european standards. <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eSignature>.
- [17] Fitch Ratings, "Fitch ratings: Credit ratings & analysis for financial markets," <https://www.fitchratings.com/>.
- [18] International Trade Center, "What are warehouse receipt systems?" <http://www.intracen.org/What-are-warehouse-receipt-systems/>.
- [19] European Land Registry Association, "European land registry network," <https://www.elra.eu/european-land-registry-network/>.
- [20] "Blocknotary journal: Modern official electronic journal for us notaries." <https://www.blocknotary.com/journal>.
- [21] Bloom, "Take control of your credit and identity," <https://bloom.co/>.
- [22] N. Asokan et al., "Optimistic fair exchange of digital signatures (extended abstract)," in *EUROCRYPT '98*. Springer, 1998.
- [23] H. Pagnia and F. C. Gärtner, "On the impossibility of fair exchange without a trusted third party," Tech. Rep., 1999.
- [24] "Bitcoin wiki: Hashed time-locked contracts," https://en.bitcoin.it/wiki/Hash_Time_Locked_Contracts, 2018.
- [25] E. Abebe et al., "Enabling enterprise blockchain interoperability with trusted data transfer (industry track)," in *Middleware Conference Industrial Track, 2019*.
- [26] M. Bellare et al., "Deterministic and efficiently searchable encryption," in *CRYPTO, 2007*.
- [27] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, 1985.
- [28] M. Abdalla et al., "Robust encryption," in *TCC, 2010*.
- [29] M. Herlihy, "Atomic cross-chain swaps," in *Symposium on Principles of Distributed Computing*. ACM, 2018.
- [30] (2019) Interledger protocol v4. <https://interledger.org/rfcs/0027-interledger-protocol-4/>.
- [31] "Cosmos inter-blockchain communication (ibc) protocol," <https://cosmos.network/docs/spec/ibc/>, 2018.
- [32] "Polkadot," <https://polkadot.network/>, 2019.
- [33] BitcoinWiki, "Simplified payment verification," https://en.bitcoinwiki.org/wiki/Simplified_Payment_Verification.
- [34] A. Zamyatin et al., "XCLAIM: trustless, interoperable, cryptocurrency-backed assets," in *2019 IEEE Symposium on Security and Privacy*.
- [35] A. Kiayias et al., "Proofs of proofs of work with sublinear complexity," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016.
- [36] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work." *IACR Cryptology ePrint Archive*, 2017. [Online]. Available: <https://eprint.iacr.org/2017/963.pdf>