Enrichment of Blockchain Transaction Management with Semantic Triples

Kosala Yapa Bandara
The Insight Centre for Data Analytics
National University of Ireland Galway
Galway, Ireland
kosala.yapa@nuigalway.ie

Subhasis Thakur
The Insight Centre for Data Analytics
National University of Ireland Galway
Galway, Ireland
Subhasis.thakur@nuigalway.ie

John Breslin
The Insight Centre for Data Analytics
National University of Ireland Galway
Galway, Ireland
john.breslin@nuigalway.ie

***Abstract*—Enterprise business transactions have both public and private information; hence blockchain adaptation to an enterprise business application needs current blockchain platforms to support both public and private information. Public blockchains (permissionless) are optimized for transparency; hence the sharing of private and sensitive information is challenging. On the other hand, private blockchains (permissioned) separate information about a transaction by generating a public transaction and a set of private transactions and treat them separately. This separation weakens the cohesiveness of transaction information and develops an extra burden when it is necessary to connect both public and private information which is not duly addressed in the literature. For example, auditing, regulatory activities, certifications, and traceability need both the public and private information about transactions. This paper uses semantic triples and introduces the Triples for Transactions(T4T) model to define blockchain transactions, improve cohesiveness and resolve the extra burden of connecting both private and public transactions. This paper presents a user-driven transaction analysis, transaction modelling using the T4T model, semantic querying, and REST endpoints to enrich transaction management. Sets of semantic triples can define both public and private information about a transaction while preserving cohesiveness of the information. This approach supports point-to-point sharing of sensitive information while preserving implicit relationships between both private and public information. We have implemented an auditing scenario in the proposed approach adopting Hyperledger Fabric and compared for performance with Hyperledger Fabric. The results showed that the proposed approach reduces the number of transaction cycles by 66% compared to Hyperledger Fabric and the performance of information retrieval is in O(N). This result is a significant improvement compared to Hyperledger Fabric.**

***Index Terms*—Blockchain, Traceability, Semantic Triples, Hyperledger Fabric**

I. Introduction

Public blockchain platforms create public, immutable and transparent transactions while protecting the user's anonymity, for example, Ethereum [1], Bitcoin [2], and Litecoin [3]. They are permissionless decentralized blockchains where anyone can join, read and write transactions but no one has control over the network. On the other hand, private blockchain platforms place restrictions on who can participate and in what transactions, for example, Hyperledger Fabric [4], and Quorum [5]. They are permissioned blockchains which separate transactions as public and private transactions.

In enterprise business applications, a transaction has both private and public information [6]. For example, information shared between wholesale buyers and sellers can have sensitive pricing details and none-sensitive types of goods and quantity details. However, decentralised ledgers in a blockchain maintain the same state by storing a transaction in all the ledgers. In Hyperledger Fabric, hashes of both private and public transactions are recorded on the public ledger while keeping private records on the participants' private data stores. Hyperledger Fabric introduces private channels and private data collections to manage private transactions.

Permissioned blockchain platforms are mainly used for enterprise business applications [7]–[9]. However, current permissioned blockchain platforms separate transactions into private transactions and public transactions, and treat them separately, for example, Hyperledger Fabric [4] and Quorum [5]. There is no guarantee that public and private transactions are placed in blocks preserving cohesiveness of the information. The privacy of private transactions is preserved since hashes of them are kept on the public ledger. However, this separation destroys the cohesiveness of transaction information and adds an extra overhead when searching and querying of both public and private information of a transaction. This limitation is not duly addressed in the literature so far, and that is the main focus of this paper. For example, analytics for predictions, regulatory activities, auditing, certifications, and traceability need both private and public information about transactions hence destroying cohesiveness is adding an extra overhead for searching and querying transactions in a very large blockchain ledger.

On the other hand, semantic triples enable building both implicit and explicit relationships between subjects and objects [10]. The semantic triples also support rich semantic queries and semantic reasoning for information retrieval and validation of relationships [11]. This is a promising technology to model transactions as semantic triples while preserving cohesiveness of private and public information. Moreover, information retrieval and transactions validations can be further supported by semantic queries and semantic reasoning, respectively.

We introduced the T4T model to define a transaction as a

978-0-7381-0495-9/20/$31.00 ©2020 IEEE
DOI 10.1109/Blockchain50366.2020.00030

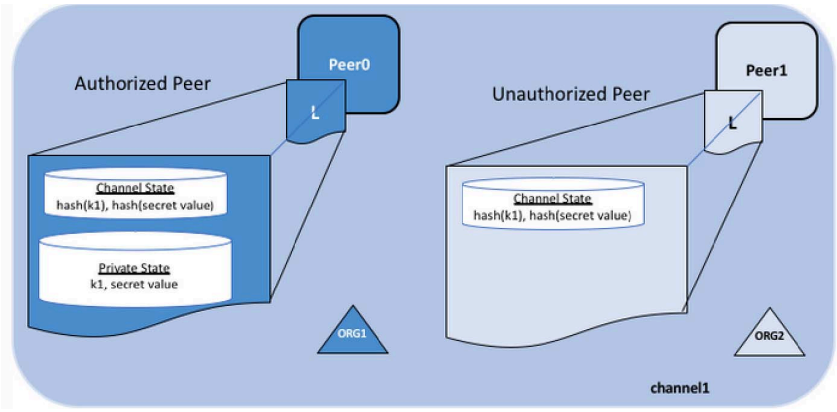

Fig. 1. Private data store in Hyperledger Fabric

collection of semantic triples while preserving cohesiveness of both public and private information about a transaction. This approach supports point-to-point sharing of sensitive information while preserving implicit relationships between both private and public information. In the proposed approach, both private and public information about a transaction is analysed, modelled, encrypted and transferred to public ledger using blockchain network and private data collections using gossip protocol as in Hyperledger Fabric [4]. The transport-level security is governed by public-key cryptography and security certificates [12]. However, the proposed approach does not need one organisation to maintain more than one private data stores as private data collections in Hyperledger [13].

In the remainder of this paper, Section II discusses state of the art regarding private and sensitive data management in blockchains while identifying challenges. Section III presents our proposed model for transaction modelling while detailing the realization in subsection A. Section IV illustrates the transaction flow of the proposed approach, which is near to Hyperledger Fabric. Section V describes the implementation and evaluation of the proposed approach for an auditing scenario. Section VI concludes while discussing contributions and directions for future research.

II. Literature Review

The public (permissionless) blockchain platforms are optimised for transparency, and transactions are public and transparent. For example, Bitcoin [2], Ethereum [1], and Litecoin [3]. However, the permissioned blockchain platforms separate transactions into public transactions and private transactions, for example, Hyperledger Fabric [4] and Quorum [5]. The private transactions share private and sensitive data between participants in a network [14]. The permissioned blockchain platforms are mainly used for enterprise business applications [7]–[9].

Ethereum is a secure decentralised ledger which is optimised for transparency; hence it is challenging to share secrets on the platform [1]. The notion of private transactions and public transactions are introduced in Quorum [5]. Quorum extends the transaction model of Ethereum [1] to include an optional privateFor parameter and a new IsPrivate method to deal with such transactions. On the other hand, Hyperledger Fabric introduces private data collections, which allow a defined subset of organisations on a channel the ability to endorse, commit, or query the private data [13]. The private data is sent peer-to-peer via gossip protocol to only the organisations authorised to see it. The ordering service is not involved here, and orderer does not see the private data. The hash of the private data is endorsed, ordered and written to the ledgers of every peer on the channel as in figure 1 [13].

The hashes of private data go through the orderer to public ledgers and preserve privacy. The hash can be used for state validation and audit purposes. In this approach, if a transaction has both private and public information, it will be decomposed into a public transaction and a set of private transactions, which create several records in the public ledger and private data stores. Moreover, different hashes are created for each transaction. This leads to complicate transaction-specific querying from both public ledger and private data collections because an extra effort is needed to search and join query related information.

Authors in [15] propose a secure-MPC (multi-party computation) protocol to support private data on Hyperledger Fabric. The participants in the network store their private data on the ledger that are encrypted with their own secret key. When private data is needed in a smart contract, the party who has the key decrypts it and uses the decrypted value. However, private data is stored in the distributed ledger enabling access if the secret key is stolen.

A set of successful use cases of blockchain implementations is summarized in [16], for example, Danish shipping company Maersk – a blockchain application for international logistics, Provenance – a pilot project in Indonesia to enable traceability in the fishing industry, Alibaba – a blockchain to fight for food fraud, Walmart – tracking produce from Latin America to the USA, and Intel's solution to track seafood supply chain. The traceability in a supply chain is a challenging

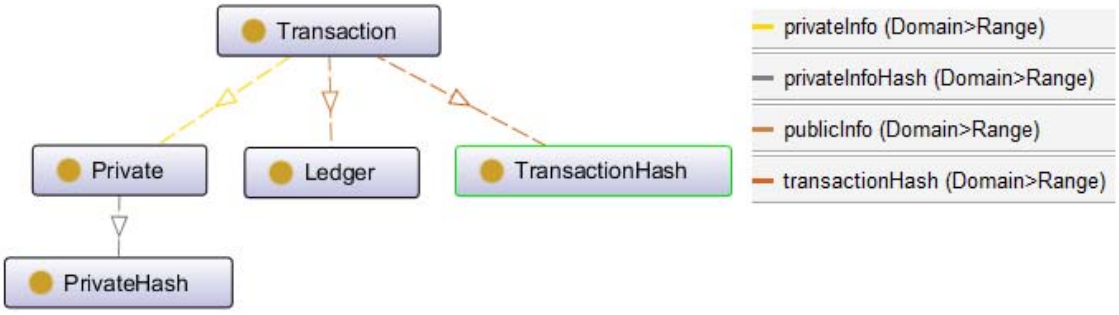

Fig. 2. T4T Model (Triples for Transactions Model)

area to explore [17]. The use of semantics in a blockchain to improve the scalability of IoT is discussed in [18]. In current blockchain architecture, distributed ledgers provide transaction information accessible to all the participants in a blockchain network providing a greater transparency [19]. However, organizations are reluctant to expose sensitive information in a public ledger. The privacy, scalability and lack of governance are still significant concerns for large scale industrial adaptation of blockchain paradigms [20].

The separation of transactions as private and public transactions weakens cohesiveness of the transaction information and adds an extra overhead on searching and querying when both the public and private information about a transaction is needed. For example, analytics for predictions, regulatory activities, traceability, and auditing need both the private and public information about transactions. This separation of concerns and weakening the cohesiveness are not duly addressed in the literature, and further research is needed to resolve these limitations for large enterprise blockchain applications.

III. Triples for Transactions (T4T Model)

In an enterprise application, a single transaction should be capable of holding both public and private information. We combined the concepts separation of concerns [21] in software engineering, principles of public-key cryptography [12] in cybersecurity and RDF triples in semantic web [11], and introduced a T4T model to define transactions in enterprise blockchain applications.

definition 1: The T4T Model is a collection of triples which defines public and private information of a transaction. Triples are made from classes: Transaction, TransactionHash, Private, PrivateHash and Ledger, and object properties: publicInfo, privateInfo, privateInfoHash and transactionHash.

As in definition 1, the T4T model has five main classes connected through four object properties. The classes, domain and range of object properties are illustrated in figure 2. We used OWL functional syntax [10] to present the formalization of the T4T model. The core components of the T4T model are formalized and presented in OWL syntax as follows:

```
:Transaction rdf:type owl:Class
:TransactionHash rdf:type owl:Class
:Private rdf:type owl:Class
:PrivateHash rdf:type owl:Class
:Ledger rdf:type owl:Class
```

The Transaction class instantiates transactions. The TransactionHash class is to define a hash for a transaction. The hash for a transaction is created using both public information and hashes of private information. The hash of the transaction, the hashes of private information and public information are always stored in the public ledger. The Private class is to define private information between participants. The Ledger class is to define public information of a transaction which is visible to all the participants in a network. The PrivateHash class contains hashes of private information.

The object properties of the T4T model are formalized and presented in OWL syntax as follows:

```
:transactionHash
        rdf:type owl:ObjectProperty,
                 owl:FunctionalProperty;
        rdfs:domain :Transaction ;
        rdfs:range :TransactionHash .

:publicInfo
        rdf:type owl:ObjectProperty ,
                 owl:FunctionalProperty;
        rdfs:domain :Transaction ;
        rdfs:range :Ledger .

:privateInfo
        rdf:type owl:ObjectProperty,
                 owl:FunctionalProperty;
        rdfs:domain :Transaction;
        rdfs:range :Private .

:privateInfoHash
        rdf:type owl:ObjectProperty,
                 owl:FunctionalProperty;
        rdfs:domain :Private;
        rdfs:range :PrivateHash .
```

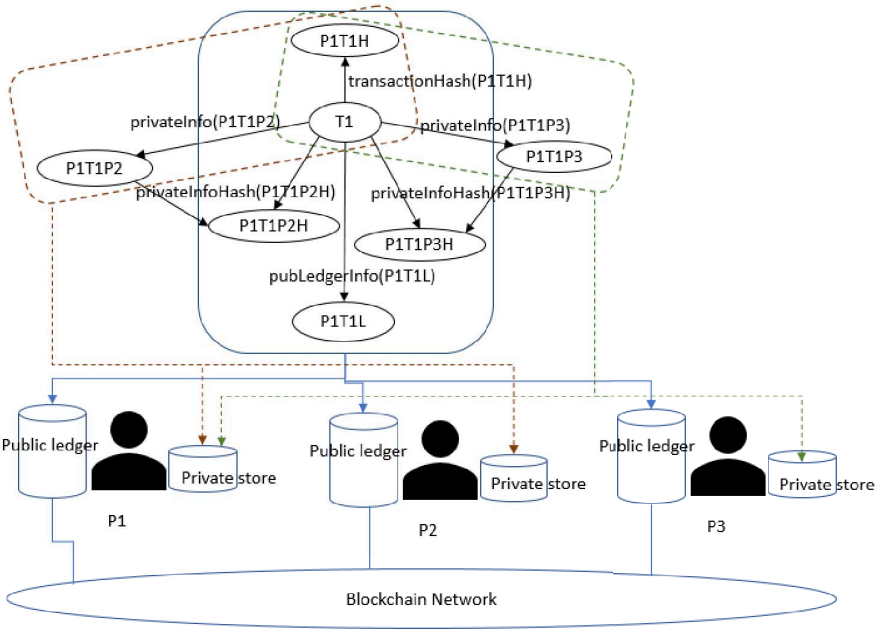

Fig. 3. T4T Realization for Transaction T1

The object property, transactionHash connects a transaction with the hash of the transaction. The object property, publicInfo connects a transaction with the public information of the transaction. The object property, privateInfo connects a transaction with the private information of the transaction. This private information is confidential information of the transaction between participants in the network. The object property, privateInfoHash connects instances of Private and PrivateHash.

User-driven separation of concerns is applied to separate public and private information about a transaction. A collection of triples which define public and private information is auto-generated based on the transaction specification. Public key certificates and private keys will be used to manage transport-level security of sensitive information between participants.

A. Realization of T4T Model for Transactions

For the illustration purpose, we realize the T4T model for transaction T1 in a blockchain network of three participants. The transaction T1 has both public and private information. The P1, P2 and P3 are participants in the blockchain network. In transaction T1, P1 needs to send the commodity C1 to P3 through P2. The P1T1P3 is private information between P1 and P3. The P1T1L is public information shared among all the participants. The P1T1P2 is private information between P1 and P2. The P1T1H is the hash value of the transaction T1 having both public and private information. The P1T1P2H is the hash value of private information between P1 and P2 of the transaction T1. The P1T1P3H is the hash value of private information between P1 and P3 of the transaction T1.

The realization of T4T model for transaction T1 is illustrated in figure 3. When the transaction T1 is committed, Triples [T1, P1T1H, P1T1L, P1T1P2H, P1T1P3H] are stored on the public ledgers of all the participants. The triples [T1, P1T1H, P1T1P2] are stored on the private data store of P2. The triples [T1, P1T1H, P1T1P3] are stored on the private data store of P3. The triples [T1, P1T1H, P1T1P2, P1T1P3] are stored on the private data store of P1. This approach supports point to point sharing of sensitive information while preserving implicit relationships between the private and public information.

The transport-level security is managed by public-key cryptography on the transaction flow. The triples [T1, P1T1H, P1T1P2] are encrypted using P2's security certificate, and the triples [T1, P1T1H, P1T1P3] are encrypted using P3's security certificate. The triples of private information can be decrypted and stored on private data stores of P2 and P3. We introduced triple stores to keep private data at participants.

The type Participant defines a member of a blockchain network. The realisation of T4T model for the transaction T1 is formalised and presented in OWL functional syntax as follows:

```
:P1 rdf:type owl:NamedIndividual,
             :Participant.
:P2 rdf:type owl:NamedIndividual,
             :Participant.
:P3 rdf:type owl:NamedIndividual,
             :Participant.

:P1T1H rdf:type owl:NamedIndividual,
                :TransactionHash .
```

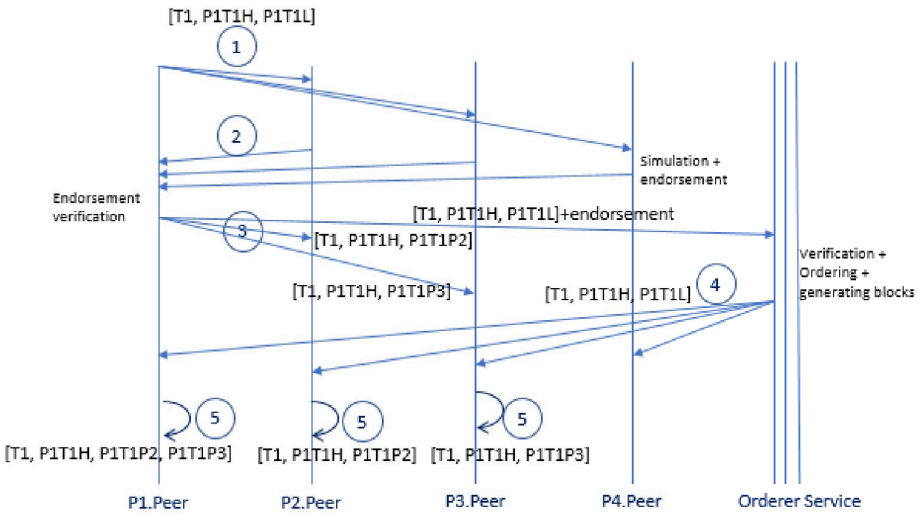

Fig. 4. Transactions Flow

```
:P1T1L rdf:type owl:NamedIndividual,
                :Ledger.
:P1T1P2 rdf:type owl:NamedIndividual.
                :Private.
:P1T1P3 rdf:type owl:NamedIndividual,
                :Private.

:T1 rdf:type owl:NamedIndividual,
            :Transaction;
    :privateInfo :P1T1P2,
                 :P1T1P3;
    :publicInfo :P1T1L;
    :transactionHash :P1T1H.
```

This proposed approach provides scalability in terms of sharing private information between participants and supports semantically rich, efficient querying using collections of triples. The efficiency was assessed based on the performance. This approach improves the cohesiveness of both public and private information of a transaction since they are connected implicitly. Moreover, when the private data needs verification, the relevant hash can be queried from the public ledger. Semantic queries can be applied to private data stores maintained at the participants to query the transaction hash and private data. The necessary validations can be done on the hash generated from the private data, and the hash queried from the public ledger.

IV. Transaction Flow

We extended the transaction flow of Hyperledger Fabric[1] to support our T4T model. In our approach, a transaction is a collection of triples which defines both private and public information. A common hash is generated using public information and hashes of private information about the transaction. The common hash is verified before committing both public and private information about the transaction. The triple stores were introduced for private data stores in Hyperledger Fabric, and the transaction flow was extended to support triple stores. Hyperledger Fabric is using key-value data stores to record private and confidential data. Also, the hash of the private information is stored in the public ledger as a separate transaction weakening cohesiveness of public and private information about a single transaction. The main activities of the extended transaction flow are illustrated in figure 4.

The flow of transactions is nearly similar to Hyperledger Fabric. However, Hyperledger fabric separates public and private information about a transaction as a public transaction and private transactions and stores them on the public ledger. The hash of the private transaction is stored on the leader while private data is stored on private data stores.

We used the transaction defined in figure 3 to illustrate the extended transaction flow in figure 4. The main activities of the extended transaction flow are as follows:

1). The participant P1 sends public triples of the transaction to all the peers. Peers simulate and endorse the transaction as in Hyperledger Fabric.

2). The endorsed transactions are sent to the client. The client verifies the endorsement with the agreed endorsement policy as in Hyperledger Fabric.

3). The endorsed transaction (public triples of the transaction) is broadcasted to ordering service. The private data (private triples of the transaction) is shared with authorized peers upon endorsement and stored in a transient store of each peer. The private triples will not go through the ordering service as in Hyperledger Fabric. The private triples have the transactions ID, transaction hash and private data so that it can connect with ledger information when it is needed for

[1]https://hyperledger-fabric.readthedocs.io/en/release-1.4/arch-deep-dive.html

verification and querying. The transaction hash maintains the integrity of both public and private information.

4). The ordering service verifies endorsement, orders transactions into a block and delivers the block to all the committing peers. The ordering service, which is made up of a cluster of orderers, does not process transactions or maintain the shared ledger. It only accepts the endorsed transactions and specifies the order in which those transactions will be committed to the ledger. The committing peer validates the transaction by checking the current world state. That is endorsers' simulated state is identical to the current world state. After the committing peer validates the transaction, the transaction is written to the ledger, and the world state is updated.

5). While committing a transaction, private records will be committed to private data stores of authorized members from the transient stores. The transaction hash, private hashes and private triples can be verified before committing private information to triple stores. There is no need for a separate transaction on the public ledger to record the hash of private data as in Hyperledger Fabric. This is one of the crucial contributions of this proposed work.

The private transactions in the Hyperledger Fabric execute through the transaction flow to record hashes of the private information. However, our approach does not need extra transactions for private information and reduces unnecessary cycles in the transaction flow, for example, the transaction T1 defined in subsection III-A needs two private transactions in Hyperledger Fabric costing two extra cycles in the transaction flow to record hashes of private information. That is, the proposed approach reduces 66% of transaction cycles for the T1 type transactions. T1 is a simple and the most common type of transactions in enterprise applications; hence this is a significant improvement.

V. Implementation and Evaluation

A. Transaction Flow

We implemented the transaction flow for T1 type transactions, as explained in section 4 and simulated our proposed approach for advanced transactions while increasing the number of participants. For advanced transactions, we used the binomial coefficient to find the distinct combinations of a group of two members for private transactions. The binomial coefficient

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

where n is number of participants and k is 2. We simulated our proposed approach and Hyperledger Fabric for advanced transactions and compared the results in figure 5. Hyperledger Fabric separates public information and private information of a transaction and creates separate transactions. The result shows that for one business transaction, the proposed approach has only one blockchain transaction processing through the transaction flow. However, Hyperledger Fabric needs more than one blockchain transactions (one transaction

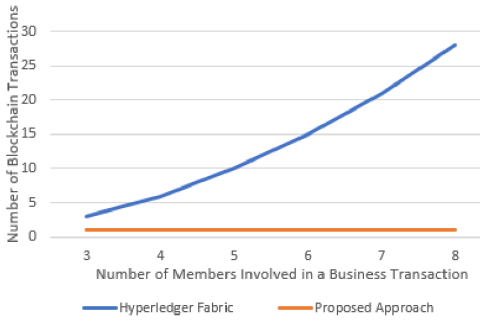

Fig. 5. Compare the number of transactions in the Transaction Flow

for public information and a set of transactions for private information between participants connected through channels and collections) processing through the transaction flow, and that is proportional to the binomial coefficient where n is the number of participants, and k is 2. The result shows that the proposed approach reduces at least 66% of transaction cycles, and that is a significant improvement.

B. Querying Transactions for Traceability

We implemented traceability using an auditing scenario that needs both public and private information of transactions. We implemented the scenario using our proposed approach and Hyperledger Fabric and compared them for performance.

Scenario: An auditor needs both public and private information of the transaction T1. Transaction T1 is initiated by P1 to send goods to P3 through P2. T1 has public information, private information between P1 and P2, and private information between P1 and P3.

In our application, a user-driven analysis of the transaction T1 generates a specification and a collection of triples are auto-generated. The underlying implementation uses Ontology Web Language (OWL)[2].

The transaction model has 6 main classes (:Transaction, :TransactionHash, :Ledger, :Private, :PrivateHash, :Participant of rdf:type owl:Class) and 4 object properties (:transactionHash, :privateInfoHash, :privateInfo, :publicInfo of rdf:type owl:ObjectProperty).

The necessary instances for the transaction T1 are created based on the user-driven analysis of transaction T1. For example, (:T1 rdf:type owl:Transaction, :P1,P2,P3 rdf:type owl:Participant, :P1T1P2, P1T1P3 rdf:type owl:Private :P1T1L rdf:type owl:Ledger,:P1T1H rdf:type owl:TransactionHash, :P1T1P2H, P1T1P3H rdf:type owl:PrivateHash). Then instances are connected using object properties. For example, (:T1 rdf:type owl:Transaction;:privateInfo :P1T1P2, P1T1P3;:publicInfo :P1T1L;:transactionHash :P1T1H). Similarly, private information has private hashes. P1T1P2 has private hash P1T1P2H (:P1T1P2 rdf:type owl:Private; :privateInfoHash:P1T1P2H) and P1T1P3 has

[2]https://www.w3.org/OWL/

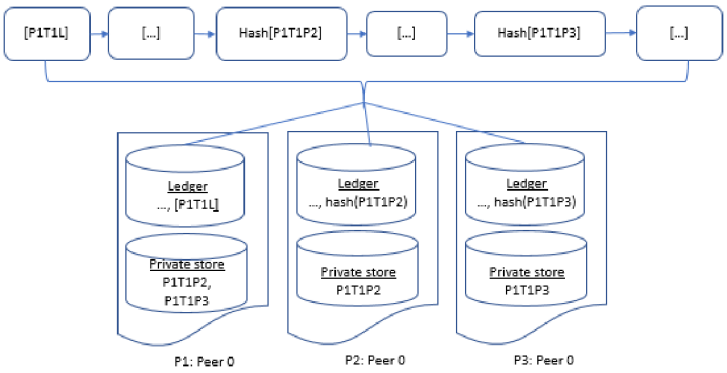

Fig. 6. Ledger and private data stores in Hyperledger Fabric

private hash P1T1P3H (:P1T1P3 rdf:type owl:Private; :privateInfoHash:P1T1P3H).

The private data store of p2 has the record [T1, P1T1H, P1T1P2] and p3 has the record [T1, P1T1H, P1T1P3]. Auditors can use semantic queries on private data stores to retrieve necessary information. We implemented SPARQL[3] queries to retrieve T1 related private information from P2 as follows:

```
PREFIX ab: <http://www.semanticweb.org/
c5282513/ontologies/2019/4/CoTOntology#>
SELECT ?privateInfomation ?TransHash
WHERE {
ab:T1 ab:privateInfo ?privateInfomation.
ab:T1 ab:transactionHash ?TransHash
}
Results:
PrivateInformation  | TransHash
P1T1P2                P1T1H
```

The same query was applied to P3 to retrieve T1 related private information as follows:

```
Results:
PrivateInformation  | TransHash
P1T1P3                P1T1H
```

Now auditors have private information about the transaction T1 as P1T1P2 and P1T1P3. Since there is a common hash for public information and private information, it is a single direct call to query transaction-specific public information from the public ledger. For example, the REST end-point to P1T1H is a single invocation on the public ledger to retrieve P1T1L, P1T1P2H, and P1T1P3H. The performance is in O(1). Now the auditor can verify private information using private hashes - for example, P1T1P2H to verify P1T1P2 and P1T1P3H to verify P1T1P3. This ensures the immutability of private information. Moreover, private information is not exposed to other participants. Auditing Transaction T1 needs the following tasks in the proposed approach:

1) Query private information from a triple store using a semantic query. For P2, private information is [P1T1P2, P1T1H].
2) Retrieve transaction-specific information from the ledger using the REST end-point of P1T1H. Information is [T1: P1T1H, P1T1L, P1T1P2H, P1T1P3H] and performance is in O(1).
3) Verify private information using hashes [P1T1P2-P1T1P2H, P1T1P3-P1T1P3H].
4) All the transaction-specific verified information is ready for auditing [P1T1L, P1T1P2, P1T1P3].

We compared our approach with Hyperledger Fabric [4]. Starting in v 1.2, Hyperledger Fabric offers private data collections. A collection offers two elements, the actual private data, and the hash of that data. Hyperledger Fabric creates three transactions for T1 (P1T1L, P1T1P2, P1T1P3) as illustrated in figure 6 while weakening cohesiveness of the transaction information. There is no guarantee that transactions are in a predefined order in the public ledger.

[P1T1L] needs a public transaction. Similarly, [P1T1P2] and [P1T1P3] need two separate private transactions to record hashes of them. The private store of P2 keeps [P1T1P2] and the hash of [P1T1P2] will be in the public ledger. Similarly, the private store of P3 keeps [P1T1P3] and the hash of [P1T1P3] will be in the public ledger. Auditing Transaction T1 needs the following tasks in Hyperledger Fabric:

1) Querying private information from private data stores [T1: P1T1P2, P1T1P3].
2) Retrieve transaction-specific information from the ledger using the REST end-point of T1 [T1: P1T1L].
3) Generate hashes for private information [P1T1P2, P1T1P3].
4) Retrieve hashes of private information from the public ledger [P1T1P2H, P1T1P3H] and do the necessary verification. A separate invocation is needed for each private data record. The performance is O(N).
5) All the transaction-specific verified information is ready for auditing.

The proposed approach has cohesive information and a common hash. A semantic query can extract the common hash and private information from triples stores. Since the common has is extracted, it is a single REST end-point call

[3]https://www.w3.org/TR/rdf-sparql-query/

to extract all public information from the ledger. However, in Hyperledger Fabric, each private transaction maintains a different hash hence a separate call on the public ledger is needed for each private information as in step 4. Auditing involves thousands of transactions hence making several calls on the public ledger for one transaction is a costly process that can lead to a performance hit of $O(N)^2$.

Similarly, Quorum separates transactions into two categories, called public and private transactions. In enterprise business applications, most of the transactions have both public and private information. Separating transactions makes the information retrieval and auditing process expensive. Several searching, querying and joining are hitting the performance.

VI. Discussion and Future Work

The private blockchains separate information about a transaction by generating a public transaction and a set of private transactions and treat them separately. This separation of concerns weakens the cohesiveness of transaction information and adds an extra overhead for enterprise applications when necessary to consider both public and private transactions. On the other hand, semantic triples enable building both implicit and explicit relationships between subjects and objects and also support rich semantic queries and semantic reasoning for validations.

We used the semantic triples and introduced the T4T model to define blockchain transactions. The T4T model and triple stores for private data persistence enrich cohesiveness of the private and public information about a transaction stored in private data stores and the public ledger and further supports information retrieval and traceability for enterprise blockchains. This approach supports point-to-point sharing of sensitive information while preserving implicit relationships between the private and public information. The privacy and transparency of transactions are preserved while providing scalability for private transactions.

The proposed approach reduces unnecessary cycles in the transaction flow compared to Hyperledger Fabric. The proposed approach reduces at least 66% of transaction cycles which is a significant improvement. Moreover, the proposed approach shows better performance [O(N)] on information retrieval compared to Hyperledger Fabric [$O(N)^2$] which is a significant improvement for much-needed traceability applications.

The proposed approach can be adapted to other blockchain frameworks since the transaction modelling, and private data stores stay outside the blockchain framework. We are working in extending the proposed approach for traceability and item recall in supply chain systems and modelling tokens in token-based blockchains to yield improved performance, scalability and semantically enriched features.

VII. Acknowledgement

This publication has emanated from research supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289_P2 (Insight).

References

[1] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," [Online; Accessed: 2019-05-23]. [Online]. Available: https://gavwood.com/paper.pdf

[2] bitcoin.org, "Open source p2p money," [Online; Accessed: 2019-05-23]. [Online]. Available: https://bitcoin.org/en/

[3] litecoin.org, "Open source p2p digital currency," [Online; Accessed: 2019-05-23]. [Online]. Available: https://litecoin.org/

[4] Hyperledger.org, "Hyperledger fabric," 2019, [Online; Accessed: 2019-05-15]. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/latest/index.html

[5] J. P. Morgan, "Quorum - transaction processing," 2018, [Online; Accessed: 2019-04-23]. [Online]. Available: https://github.com/jpmorganchase/quorum/wiki/Transaction-Processing

[6] J. Holbrook, *Architecting Enterprise Blockchain Solutions*. WILEY, 2020.

[7] F. Yiannas, "A new era of food transparency powered by blockchain," *Innovations: Technology, Governance, Globalization*, vol. 12, no. 1-2, pp. 46–56, 2018.

[8] B. Tan, J. Yan, S. Chen, and X. Liu, "The impact of blockchain on food supply chain: The case of walmart," in *Smart Blockchain*, M. Qiu, Ed. Springer International Publishing, 2018, pp. 167–177.

[9] N. Emmadi, R. Vigneswaran, S. Kanchanapalli, L. Maddali, and H. Narumanchi, "Practical deployability of permissioned blockchains," in *Business Information Systems Workshops*, W. Abramowicz and A. Paschke, Eds. Springer International Publishing, 2019, pp. 229–243.

[10] B. Motik, P. Patel-Schneider, and B. Parsia", "Owl 2 web ontology language: Structural specification and functional-style syntax (second edition)," 2012, [Online; Accessed: 2019-05-14]. [Online]. Available: https://www.w3.org/TR/owl2-syntax/#Functional-Style_Syntax

[11] M. Sintek and S. Decker, "Triple - a query, inference, and transformation language for the semantic web," in *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ser. ISWC '02. Berlin, Heidelberg: Springer-Verlag, 2002, pp. 364–378.

[12] S. Halevi and H. Krawczyk, "Public-key cryptography and password protocols," *ACM Trans. Inf. Syst. Secur.*, vol. 2, no. 3, pp. 230–268, Aug. 1999.

[13] Hyperledger.org, "Private data – hyperledger fabric," 2019, [Online; Accessed: 2019-04-23]. [Online]. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.4/private-data/private-data.html

[14] E. Androulaki, S. Cocco, and C. Ferris, "Private and confidential transactions with hyperledger fabric," [Online; Accessed: 2019-05-22]. [Online]. Available: https://developer.ibm.com/tutorials/cl-blockchain-private-confidential-transactions-hyperledger-fabric-zero-knowledge-proof/

[15] F. Benhamouda, S. Halevi, and T. Halevi, "Supporting private data on hyperledger fabric with secure multiparty computation," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, April 2018, pp. 357–363.

[16] N. Kshetri, "Blockchain's roles in meeting key supply chain management objectives," *International Journal of Information Management*, vol. 39, pp. 80–89, apr 2018.

[17] W. Martin, V. Friedhelm, and K. Axel, "Blockchain-based supply chain traceability: Token recipes model manufacturing processes," in *Proceedings of 2018 IEEE International Conference on Blockchain*. IEEE, 2018.

[18] M. Ruta, F. Scioscia, S. Ieva, G. Capurso, and E. D. Sciascio, "Semantic blockchain to improve scalability in the internet of things," *Open Journal of Internet Of Things (OJIOT)*, vol. 3, pp. 46–61, 2017.

[19] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, pp. 557–564.

[20] W. Li, A. Sforzin, S. Fedorov, and G. O. Karame, "Towards scalable and private industrial blockchains," in *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*. ACM, 2017, pp. 9–14.

[21] H. S. Hamza, "Separation of concerns for evolving systems: A stability-driven approach," *SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1–5, May 2005.