The Global Synchronization Log

Digital Asset

November, 2016

Executive Summary

This paper introduces the Global Synchronization Log (GSL), a component that can serve as a common foundation for distinct Distributed Ledger Technology (DLT) implementations. The GSL is a natural evolution of current technologies in the face of new and evolving requirements.

The purpose of this paper is to encourage collaborative work in the Hyperledger community towards an implementation of a GSL using projects currently under the Hyperledger umbrella. It is our belief that the GSL can be a distinct component in any number of DLT solutions. Digital Asset is currently working on an implementation as a component of its Digital Asset Platform and would like to encourage other implementations in the interests of promoting compatibility across different platforms. A more technical and formal discussion will be presented in subsequent communications. We look forward to learning more about whether aspects of the forthcoming Corda[1] release are compatible with this approach and/or would equally welcome development of the GSL as part of another Hyperledger project, Fabric[2].

1.0 Introduction

The 2009 release of Bitcoin[3] sparked wide interest in and recognition of the value of DLT. Since then, the growing interest in applications for use within regulated financial institutions has driven increasingly stringent requirements for privacy, throughput, and operational scale. In response, many DLT implementations have begun experimenting with novel cryptographic techniques, ledger segmentation, and external uniqueness services in search of an enterprise-grade solution.

There have been several proposals and designs for enterprise DLT solutions. To date, most efforts to create standardization of the technology have come in the form of full-stack solutions, complete with applications, languages, execution capabilities, and unique ledger services. Many of these proposals are competing, duplicative, or otherwise incompatible. It is our belief that the advancement of this technology, and hence this industry, is best served by the creation of smaller, more modular components that are reusable across a range of differing implementations. A collection of components and customized implementations would allow standardization to emerge naturally across popular components, yet allow for alternatives for different use cases. Therefore, we strongly support the Hyperledger Foundation's efforts to serve as an umbrella organization for enterprise-grade, open source software components[4].

Working with clients who are active in wholesale regulated financial markets and through our experience in attempting to leverage a number of existing DLT implementations, we have identified a set of specialized components and services that combine to form a full DLT solution. We believe this solution is suitable for systemically important financial institutions. We further believe that these components are reusable across many use cases, platforms, and industries, which would all benefit from wider collaboration.

The first and most fundamental requirement for the application of DLT in regulated financial services is to preserve the privacy of sensitive information stored and coordinated by the DLT. Taking into consideration all the risks and factors, we believe that the physical segmentation of private data is currently the only viable way to achieve this requirement. Some solutions have sought to solve this issue with the use of encryption, but from our experience with regulations governing data sharing and persistence, analysis of revealed data to uncover patterns, and forward secrecy (meaning that if there is a compromise later, it does not compromise the full history), we do not believe this is yet a viable solution for this industry.

We have designed a component[1] called the Global Synchronization Log (GSL) to provide the same integrity, privacy, and transparency guarantees found in shared, replicated ledgers to a distributed network of physically segregated transaction data. Our implementation of the GSL employs a method of using a blockchain as a privacy-preserving uniqueness service with a built-in notification mechanism. We believe other DLT implementations that aim to solve privacy requirements through segmentation such as Quorum[5], would benefit from such a component. Furthermore, we believe some of the current frameworks such as Fabric[2], Corda[1], and MultiChain[6] can implement the same interface as our GSL. Such implementations would allow gathering of comparable metrics and increased emphasis on developing key quality attributes of these frameworks such as scale, throughput, supportability, and version management.

The GSL is a log of commitments and notification sets that guarantees the integrity of the distributed data stores and the auditability of the stakeholders to the contracts in the Distributed Ledger (DL). The GSL establishes a common and complete set of valid transactions that, when combined with the corresponding off-chain transaction data, comprises a DL. The GSL is a communication layer designed to deliver network-wide integrity guarantees of transaction commitments and notifications. This paper addresses features of the GSL but does not address or dictate features of higher layer components.

2.0 Problems

There are numerous criteria crucial to evaluating DLT implementations for production deployments, including confidentiality, throughput, and scale of storage. Typical blockchain implementations require the full, transparent replication of data to ensure its validity, which compromises confidentiality. The alternatives to full transparency are improving and advancing steadily. As of publication of this paper, opportunities to improve privacy generally fall into the following categories:

1. Obfuscation: transactions are visible to all parties, but the parties themselves are pseudonymous, unique public keys are used, or the identities are otherwise obscured.[7]

2. Encryption: each transaction carries ciphertext and decryption keys may be made available to eligible parties.[2][8][16]

3. Zero Knowledge Proofs: the transaction carries a cryptographic proof of its validity. The proof is convincing to all parties of the network but data is only revealed selectively to eligible parties.[9]

[1] While we refer to the GSL as a software component, in this paper we will use GSL interchangeably to refer to the component, the service which it implements, and the distributed data structure.

4. Segregated Ledgers: an interconnected network of segregated low population ledgers (e.g., bilateral, trilateral, etc.) utilizing common protocols or trusted intermediaries to move or interoperate across ledgers.

5. Data and Execution Commitments: a network-wide blockchain carries fingerprints of sensitive data. The actual data is segregated and communicated over private channels only to eligible parties.[5]

The first two options involve all network participants, including parties who are not entitled to certain data, to hold that data in encrypted or obfuscated form. Due to data domicile regulations and forward secrecy concerns the first two options are currently not viable options for regulated financial institutions.

Zero Knowledge Proofs (ZKP) are a promising direction but, as of yet, implementations are immature and unproven in production environments. The most efficient ZKP schemes[10] applicable to DLT leverage cryptographic techniques that have not yet had the benefit of scrutiny over time.[2]

Because of the limitations of the first three options, Digital Asset has further explored the remaining two; Segregated Ledgers and Data and Execution Commitments. These are similar in concept, in both the components responsible for producing evidence of the validity of a transaction are separated from the transaction data itself. This separation guarantees that information is segregated and shared only with entities entitled to access this information.

However, segregated ledgers introduce some unique problems of their own:

- Transfer of assets or contracts across segregated ledgers with disjoint or overlapping sets of participants requires proof of integrity of the ledger originating the asset. Disparate ledgers must be synchronized. Furthermore, for n participants each participant must maintain up to 2^{n-1} ledgers with other participants, introducing complex logic needed to manage these ledgers and keep the scale practical.[3]

- In order to guarantee that all participants have a complete set of data, either all participants of a specific ledger must sign every transaction, or there must be a mechanism for assured delivery for each participant of every ledger. We will elaborate on the importance of assuring a complete set of data below.

[2] Specifically, none of these techniques are certified under common standards such as FIPS 140-2[17]

[3] This is not an insurmountable problem and is a viable design for a GSL. However, at Digital Asset we found the solutions in this design space to be much more complex than the alternatives when designing for market infrastructure scale.

We have designed the GSL to be one possible way to solve these problems without sacrificing privacy or integrity. The solution is described at a high level in the following section.

3.0 The Global Synchronization Log

The GSL serves three primary functions when added to a distributed network with segregated data:

1. To serve as the arbiter of relative order[4] between dependent transactions.
2. To ensure uniqueness of mutually exclusive events and maintain the state of the ledger data. This state is derived from the stream of transactions.
3. To serve as an assured notification mechanism. Any stakeholder affected by a state change of data or contracts must be notified, or more precisely have the assurance that it will be made aware that this state change has occurred.

3.1 Active Contracts

"Contracts" refer to the off-chain business logic, whether this is transactional data or common workflow models of behavior within the system. Contracts are not necessarily the legally binding documents that govern the higher order rules of a market, but rather the code-enforced constraints and agreements by which participants are bound.

In the Digital Asset stack, the state of the DL is defined by the set of current "active contracts" between parties on the ledger. This notion of Contracts is a specific instantiation of the broader concepts such as "State Objects" as defined in the Corda framework or "Chaincode State" as defined in Fabric[11]. Active contracts are those that have been instantiated but not yet referenced or archived by a subsequent contract[5]. Referencing a previous contract replaces, or consumes, the referenced contract such that it is no longer active and cannot be referenced in the future[6]. Unreferenced contracts can also be archived to render them inactive. The storage and interpretation of contracts are the responsibilities of separate services.

A GSL that appends transactions receives the unique contract IDs from a separate service. One purpose of the GSL is to ensure that a contract is

[4] While relative order is sufficient, the Digital Asset implementation uses strict ordering via a blockchain.

[5] The active contract set resembles the UTXO set in Bitcoin.

[6] Purely as an optimization, the Digital Asset stack allows a specific type of non-consuming reference. This permits evidence that a specific contract existed at the time of a transaction, e.g., a Master Service Agreement existed at the time of a trade registration.

referenced or archived only once. The GSL will record a fingerprint of the active contract ID set in every block, creating an immutable version or state of the ledger at the moment of transaction recordation. Those who are entitled to see the contract or to generally audit the active contract set can create a compact proof of the contract's existence (as of any block) using the active contract set's fingerprint.

3.2 Transactions

Committing a transaction to the GSL changes a subset of the DL by instantiating or archiving contracts relevant to a subset of the network participants. The semantics of a transaction are jointly defined by the GSL and a separate service, but the GSL does not have to understand, or make transparent, the semantics defined by that separate service.

Digital Asset's separate contract-validating service does not directly support general purpose procedural computation and accumulation of arbitrary state. However, according to others' use case requirements, these can be implemented on top of the GSL and replicated on a network-wide or bilateral basis. Any state change can be expressed by replacing contracts and any procedure can be expressed within the semantics of the contract language . At Digital Asset, we use our domain-specific language — DAML[12] — to model and execute agreements between financial institutions with certainty and finality, but the GSL aims to support multiple implementations.

Transactions can be validated using the state of the DL as it was at the point in time of their recording. The validation of a transaction may only use information obtained from the state of the DL. These rules ensure that decisions on validity are unconditionally reproducible by the network and the network is therefore able to mutually agree on a consistent DL.

If implemented naively, the set of events referenced in a given transaction commitment, i.e., the creation and archival of contracts, could leak confidential information by exposing patterns of these contract creations and archivals. Furthermore, their connections and patterns on the GSL data structure could be correlated with observable events in the open market and leak meaningful information.[7]

To address this risk, every committed GSL transaction is split into two logical parts:

[7] The Digital Asset implementation requires that the equivalent of a UTXO graph cannot be deduced by parties observing the public data structure. In order to achieve this, a subset of permissioned parties shares an auxiliary data structure similar to Bitcoin's segregated witness[13].

- a blinded network-wide evidence of events (a Merkelized hash) and notification of involved parties,[8]
- privately shared references of contract archival.

3.3 Notifications

In any system, a number of state changes initiated and authorized by a set of parties may affect parties outside of the primary authorization workflow. As a simple example, the owner of an American call option has the right to exercise and take delivery of the underlying asset prior to expiration. In doing so, the seller of the option is not involved in the choice to exercise, but must be notified that the buyer has done so. In many systems today, this notification tends to come in a daily report, via messaging or other lines of communication. In a DL, there are a few ways in which one could guarantee that notification was delivered to the affected stakeholder(s). In a network of physically segmented transaction data, this notification is especially important as the relation between data can be opaque to certain parties in the network.

A notification is a cryptographic shared secret that a notified party or set of parties can recognize[9]. Crucially, this shared secret cannot be understood by any other party.

Notifications ensure that an involved party will be aware of all transactions that affect it. The notified party can further request the full transaction data from the appropriate parties. In the Digital Asset implementation, a transaction that does not correctly notify all affected stakeholders to a transaction will not be considered valid by the contract validating service.

Notifications must be compact to store, inexpensive to create, and easy to recognize so that the GSL service is capable of efficiently processing transactions involving thousands of parties to a single transaction over an active contract set that exceeds billions of instantiated contracts.

3.4 Ordering transactions

The order of transactions in a GSL is irrelevant if they have no direct or transitive dependency across the events. Some networks may have completely disjoint transaction dependencies, in which case those networks could operate completely independently (although they might use a shared network purely for technical reasons). In the following paragraphs, we describe transactions for which order matters.

[8] The notification mechanism in Digital Asset's implementation is based on ECDH shared secrets.
[9] Notifications resemble (confidential) addresses of public blockchains.

If a market and its regulatory framework permits multiple parties to append to the GSL, a consensus protocol must be used to establish mutually agreed upon order and validity of transactions.

The validity of a transaction can be pre-computed under an assumed order (which is published alongside the transaction) to reduce the problem to a consensus on order. If there are transactions with contradicting assumed orderings (a race condition) only one will be accepted[14].

If ledger-appending parties cannot trust each other's integrity, the consensus protocol must handle Byzantine Faults, any fault presenting different states to different participants. Most current Byzantine Fault tolerant protocol implementations are either practically limited to work with tens of appending parties or provide only probabilistic convergence. Although the probability of convergence can with time become arbitrarily close to certainty, probabilistic protocols do not guarantee finality on their own.

Our description of the GSL component does not depend on a BFT consensus protocol, but we expect it to be able to leverage advances in this area of research.[10]

3.5. Ledger Integrity

In order to maintain integrity of the DL, all invariants and predicates of the ledger must be validated. In the Digital Asset ledger, the GSL software validates that the GSL stream is well formed and that the state updates are correct. A separate service ensures that all affected parties are notified. The list of parties that are expected to be notified, as well as a list of referenced contracts is provided as a predicate to the GSL software upon submission or receipt of a transaction.

Some networks may restrict appending transactions, or "writing," to the ledger to a single Operator of a market, or a small group of Operators who are responsible for updating the GSL on behalf of the network at large. However, it is critical to the integrity of the ledger that all participants have the ability to verify the updates of the Operator(s). Thus each participant behaves as a real- time auditor to the validity of commitments to the ledger as they pertain to the subset of the ledger visible to them.

Furthermore, different participants can verify different sets of invariants and predicates of the DL, according to their data entitlements. Thus a regulator may be permissioned to audit only a specific aspect of the ledger, for example that the current state follows directly from the previous state and the current events. A "notification auditor" role allows a participant of the network to audit the

[10] A notable contribution here is the work in progress in the Hyperledger Sawtooth Lake project[15].

invariant that all stakeholders of a transaction have been notified on the GSL, without having visibility into the business logic of the transaction. By carefully constructing the different integrity audit roles in the market with enough overlap, the DL is verified completely while no entity ever sees data to which it is not entitled.

4.0 Further Requirements

The following is a subset of additional client requirements of a GSL:

1. A GSL must be able to process several thousand transactions per second. A transaction will typically create, replace, or archive several contracts.
   - 1.1. Some contracts have thousands of involved parties that need to be notified.
   - 1.2. Some transactions have millions of contracts updated in an atomic event.
2. A GSL must have the capacity to scale in terms of periodic (daily, weekly, etc.) transaction peak volumes (e.g., millions of transactions per day) but also have the ability to scale over time (historical transaction set - hundreds of millions per year).
3. The state of the GSL, the active contract set, is on the order of a few billion active contracts at any given time in moderately sized markets.
4. A GSL must be resilient and able to recover from system or component failures in a timely manner.
5. The GSL must be resilient to data corruption and be able to return to normal operation quickly.
6. The GSL must ensure the transactionality and idempotency of all actions executed against the system.
7. The GSL must be extensible to cope with system, protocol, data and contract changes over time and not require network-wide coordination of changes.
8. Multi-party transactions must not leak confidential information to parties who are not the authorized stakeholders to that information.
9. GSL must support deployment to enterprise environments and integrate with enterprise operational support processes. This includes flexibility with respect to choice of compute, storage and network technologies across deployed participants.
10. A GSL must be deployable to many different network and connectivity architectures (Public, Private, DMZ), including segmented network address space. A GSL should be deployable across both the Internet and private, dedicated network topologies.
11. GSL use of cryptography must align with auditable, enterprise scale security operations procedures (key rotation, revocation, lifespan, etc), including use of PKI and HSM (Hardware Security Modules).

5.0 Future publications

The purpose of this document is to encourage discussion and promote compatibility across different projects within Hyperledger. This document is not intended as a specification and Digital Asset will release a subsequent publication on the API that our GSL implements. We hope to evaluate the ability to implement this API with different Hyperledger projects, including Corda and Fabric. From our initial work with Fabric, we believe this to be possible and from our current understanding of Corda we believe this may also be possible with some additional development.

6.0 Conclusion

We have given a high-level overview of the Global Synchronization Log, the component the Digital Asset stack uses as the underlying network synchronization and notification mechanism.

Digital Asset's implementation of the GSL is currently being developed as part of a suite of related components initially custom tailored for specific client needs and environments, therefore only limitedly applicable to other uses. Our work there however provided valuable insights to modularization of a DL, which we believe would support a modular build-up of the DL technology within Hyperledger, in-line with its vision to of enterprise-grade, open source software components.

6.0 Bibliography

[1] Richard Gendal Brown, James Carlyle, Ian Grigg, Mike Hearn: *Corda: An Introduction*.
https://r3cev.com/s/corda-introductory-whitepaper-final.pdf
[2] Christian Cachin: *Architecture of the Hyperledger Blockchain Fabric.* 2016
https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf
[3] Satoshi Nakamoto: Bitcoin: *A Peer-to-Peer Electronic Cash System.* 2008
https://bitcoin.org/bitcoin.pdf
[4] Behlendorf: *Meet Hyperledger: An "Umbrella" for Open Source Blockchain & Smart Contract Technologies.* 2016
https://www.hyperledger.org/blog/2016/09/13/meet-hyperledger-an-umbrella-for-open-source-blockchain-smart-contract-technologies
[5] J.P. Morgan: *Quorum: A Permissioned Implementation of Ethereum Supporting Data Privacy.* 2016
https://drive.google.com/file/d/0B42vMkapQi1MSEVaa2tuVEtXZXM/view
[6] Gideon Greenspan: *Introducing MultiChain Streams*. 2016
http://www.multichain.com/blog/2016/09/introducing-multichain-streams/
2016 http://r3cev.com/s/corda-introductory-whitepaper-final.pdf
[7] Greg Maxwell: *Coinjoin, Bitcoin privacy for the real world*. 2013
https://bitcointalk.org/index.php?topic=279249.0
[8] Shen Noether: *Ring Confidential Transactions. 2015*
https://eprint.iacr.org/2015/1098.pdf
[9] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, Madars Virza: Zerocash: *Decentralized Anonymous Payments from Bitcoin*. 2014
http://zerocash-project.org/media/pdf/zerocash-oakland2014.pdf
[10] SCIPR-Lab: *libsnark*. 2014
https://github.com/scipr-lab/libsnark
[11] Hyperledger: *Fabric Protocol Specification.* 2016
https://github.com/hyperledger/fabric/blob/master/docs/protocol-spec.md#3324-chaincode-state
[12] Digital Asset: *Introducing the Digital Asset Modelling Language*. 2016
https://digitalasset.com/press/introducing-daml.html
[13] Eric Lombrozo, Johnson Lau, Pieter Wuille: *BIP 141: Segregated Witness*. 2015
https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
[14] Elli Androulaki, Christian Cachin, Konstantinos Christidis, Chet Murthy, Binh Nguyen, and Marko Vukolić: *Next Consensus Architecture Proposal.* 2016
https://github.com/hyperledger/fabric/blob/master/proposals/r1/Next-Consensus-Architecture-Proposal.md
[15] Intel: *Sawtooth Lake*. 2016
https://intelledger.github.io/introduction.html
[16] Adam Back: *Bitcoins with homomorphic value.* 2013
https://bitcointalk.org/index.php?topic=305791.0
[17] NIST: *FIPS pub 140-2.* 2001
http://csrc.nist.gov/groups/STM/cmvp/standards.html