Evaluation of using Pairing-based cryptography in ByzCoin

Swarali Karkhanis

School of Computer and Communication Sciences

Decentralized and Distributed Systems lab

Semester Project

January 2019

Responsible
Prof. Bryan Ford
EPFL / DEDIS

Supervisor
Lefteris Kokoris Kogias
EPFL / DEDIS

Contents

1 Introduction

Bitcoin's[10] meteoric rise has paved way for thousands of crypto-currencies in the past decade. Distrust in banks and government in the wake of the 2008 financial crises fueled the need to remove third-party single guarantors through decentralizing authority. Due to the *weakest-link* security of the centralized systems, various centralized internet services like certificate authorities (CA), domain name systems (DNS), software repos have routinely fallen prey to hackers on the internet. Although decentralized authority improves the security of the system, it is difficult to efficiently reach consensus among these authorities in an asynchronous environment like the internet.

PBFT protocol [6] achieves strongly consistent consensus in a byzantine environment but is not scalable due to $O(n^2)$ communication between the participating nodes. Bitcoin on the other hand, does offer lesser transaction latency in a large network but it only provides probabilistic guarantees. ByzCoin[1] improves on Bitcoin's consistency using the principles of PBFT with proof-of-membership mechanism and scalable collective signing(CoSi) [11].

As a naive solution, collective signing could be simply done by appending signatures of the participating authorities to a message. However as the number of signatures increase, the payload size and verification cost increases linearly, making it infeasible at scale. Therefore CoSi uses Schnorr multisignatures[5] to generate compact signatures from multiple cosignatures, that can be verified in constant time. Using tree-pattern communication and aggregation at every level allows CoSi to also efficiently combine cosignatures.

A CoSi protocol consists of four phases- announcement, commitment, challenge and response. The two round-trip protocol requires the network structures to be stable after commitment in the first round. The $\sum$-protocol nature of CoSi can make it unusable in unstable/high churn networks. As a single round-trip protocol, the Boneh-Lynn-Shacham signature scheme [7] on pairing-based elliptic curves can eliminate this problem. It also allows for incremental aggregation in an asynchronous setting. Further, with Schnorr signatures m-of-n multisignature verification can be done with merkel trees[2] but it grows exponentially in size. The public key aggregation with pairing-based keys allows for efficient verification for m-of-n multisignatures.

In this report we present and evaluate BLS-CoSi, a collective signing protocol using BLS signature scheme with pairing-based elliptic curves. Section 2 presents the background for BLS-CoSi. Section 3 then presents the design and implementation of BLS-CoSi. Section 4 experimentally evaluates the protocol and Section 5 concludes.

2 Background

This section presents the technical and mathematical background required to understand the report.

2.1 Practical Byzantine Fault Tolerance (PBFT)

Distributed systems are characterized by presence of independently failing components. Therefore fault tolerance is an important property for a distributed system that should remain available even in the presence of these failures. Derived from Byzanine General's problem[8], a system is Byzantine Fault Tolerant if it functions as desired even in the presence of malicious entities that can fail or pass incorrect information to other peers. It can be shown that in the presence of f traitors, a byzantine fault tolerant system can exist only if the total number of generals is $> 3f$.

PBFT algorithm [6] provides a practical implementation for such a system through a 3-phase protocol: pre-prepare, prepare and commit. One active node of the system acts a primary and the rest act as backups.The primary receives the requests from the client and initiates the pre-prepare phase by broadcasting the message in the network. In the prepare every node broadcast authenticated messages in the network to signals its participation. After receiving $2f + 1$ threshold prepare message, the nodes against broadcasts its commitment to the network. After receiving $2f + 1$ commitments, the node accepts the message. We easily see that this requires $O(n^2)$ communication, therefore does not scale well. Further the consensus group is closed, making it unusable for open permission-less systems like crypto-currencies.

2.2 BitCoin

Bitcoin is digital currency introduced in 2009 on a decentralized peer-to-peer network. Without centralized authority, bitcoin operates on a public distributed ledger called blockchain. Bitcoin uses proof-of-work mechanism to achieve consensus among thousands of nodes. Miners are the participating nodes that mine for new blocks to append new transactions to the blockchain. New block must contain proof-of-work that solves a computationally difficult condition (eg: finding a nonce such that when hashed with the block content, it is smaller than some target) but is easily verifiable. Due to its decentralized nature, a single malicious miner cannot attack the system. For a confirmed success, a 51% attack on bitcoin is required which can cost over $6 billion US dollars [3]. However due to the probabilistic nature of Bitcoin's consensus, the consistency of a transaction is never guaranteed. Forks can occur in the blockchain which are resolved by the longest fork. Thus a committed transaction on a dissolved fork will be reverted.

2.3 Collective signing (CoSi)

CoSi is protocol that performs efficient signing by multiple entities. Using Schnorr signatures, CoSi outputs a compact collective signature that is easily verifiable by the client. CoSi protocol follows a tree-based communication to scale computation and communication. One round of CoSi protocol consists of four phases i.e. two communications round-trips:
1) Announcement: The root initiates the protocol by multicasting the message to be signed.
2) Commitment: Each node commits to a random secret and generates a public commit which it sends up the tree. On receiving all commitments, non-leaf nodes aggregate the commit before sending it up the tree.
3) Challenge: The leader multicasts a collective challenge down the tree.
4) Response: The nodes compute their response using their committed secret keys. They further aggregate the responses received from their children and send them further up the tree.

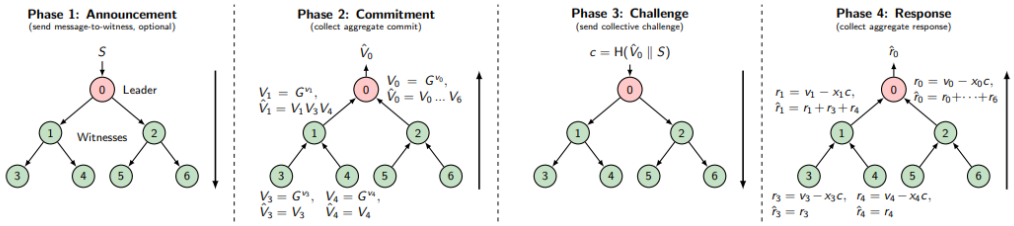

Figure 1: CoSi protocol with Schnorr multisignatures from [11]

Aggregation at every level allows nodes to check for dishonest descendants. The final collective signature can be verified as a Schnorr signature.

2.4 ByzCoin

Byzcoin provides strong consistency on Bitcoin while preserving its open membership, scalability and transaction rate. Bycoin employs proof-of-membership through a sliding window share over proof of work. For consensus among the members, Byzcoin uses PBFT with collective signing. ByzCoin replaces the direct MAC-authenticated communication among nodes with digital signatures that allows for indirect communication. This allows Byzcoin to scale with sparser tree-based communication patterns. CoSi protocol creates an efficient compact multi-signature which can be verified with aggregate public key as efficiently as an individual key.

2.5 Boneh-Lynn-Shacham (BLS) signature scheme

The Boneh-Lynn-Shacham (BLS) signature scheme [7] generates signatures that are elements of elliptic curve and uses bilinear pairings for verification.

Let G_1, G_2 and G_T be multiplicative groups where $P \in G_1, Q \in G_2$ are generators of G_1, G_2 respectively.
A bilinear pairing is a map, $e : G_1 \times G_2 \rightarrow G_T$ for which the following holds:
1) Bilinearity: $\forall a, b \in Z : e(P^a, Q^b) = e(P, Q)^a b$
2) Non-degeneracy: $e(P, Q) \neq 1$
3) e is efficiently computable.

Groups with bilinear pairing are useful in a signature scheme because they give us groups in which the Computational Diffie Hellman(CDH) problem is difficult but the Decisional Diffie-Hellman(DDH) problem is easy. BLS signature scheme consists of 3 phases:
1) Key generation: The signer selects a random secret key $x \in [0, r-1]$ as its private key. It computes and publishes the public key, g^x.
2) Signing: To sign a message, m with private key x, we compute signature on the hash of the message, $h = H(m)$ and output signature, $\sigma = h^x$.
3) Verification: To verify a signature σ and public key g^x, we verify that $e(\sigma, g) = e(H(m), g^x)$.

For aggregating multiple signature for CoSi, given the n cosigners generate public keys: $pk_1, pk_2..pk_n$ and signatures $\sigma_1, \sigma_2..\sigma_n$. BLS scheme allows us generate an aggregated multisignature, $\sigma = \sigma_1\sigma_2..\sigma_n$. The multisignature can be simply verified by checking $e(\sigma, g) = e(H(m), pk_1..pk_n)$. Using this signature scheme in a tree-structured communication model we can produce a collective signature in a single round-trip protocol

3 Design

BLS-CoSi is designed for sparse communication pattern over the network. The nodes are arranged in a 3-level tree structure. The node at the root acts as the leader of the tree and the remaining non-leaf nodes act as the sub-leader of their corresponding subtree.

The BLS-CoSi protocol runs as follows:
1) The root initiates the protocol by multicasting the message to be signed to its children, who further multicast it down the tree.
2) Each leaf node, n_i responds by returning the signature σ_i signed by their own private key. The subleader aggregates the received signatures along with their own signature and sends it to the root. The root, similarly aggregates the received aggregated signatures and the own signature to generate a single collective-signature, that is sent to the client.

The single-round trip protocol eliminates the risk faced in the two-round CoSi where the nodes not participating in the first round might disappear in the second round. Therefore BLSCoSi is useful in highly unstable networks. BLS signatures can be aggregated incrementally thus allowing participants to generate fast responses on an asynchronous network.

Non-responsive nodes do not affect the availability of the protocol. If the subleader fails to respond within a specified time period, the subtree is reorganized with a new subleader is initiated. The unresponsive leaf nodes are simply not included during aggregation.

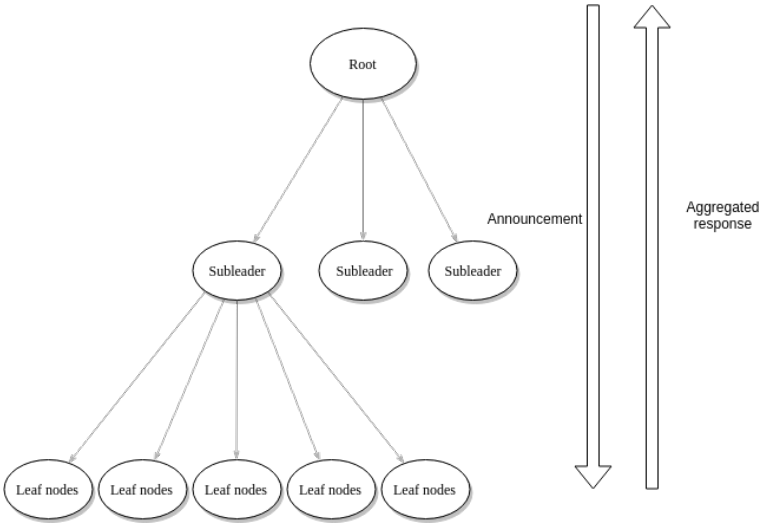

Figure 2: Design of BLSCoSi

4 Evaluation

We have implemented the BLSCoSi protocol described in Section 3 in Golang and made it available on Github [1]. For BLS signatures we use the bilinear bn256 [9] group of elliptic curves. We evaluated it on Deterlab[4] using upto 20 physical machines with 16 GB RAM to simulate upto 1,000 CoSi witnesses. The simulated network was set with a roundtrip delay of 5 milliseconds on a 1GBps link between the physical servers. We used a 3-level tree-structure with $O(\sqrt{n})$ branching factor. Acceptance threshold was set to $2f + 1$ on the network of $3f$ nodes to simulate a Byzantine-fault tolerant system that can tolerate upto f malicious nodes. In Figure 3, we evaluate scalability of BLSCoSi under different block sizes and failure rates.

4.1 Scalability

For BLSCoSi rounds with 0% failures in Figure 3, we observe a linear increase in latency with increasing number of nodes. The latency also doubles with doubling the block-size, confirms linear increase for the same. Signing 8 MB block on BLS-Cosi can account can incur latency ∼3 sec with 1k cosigners.

Compared with Flat-BLS which is a 2-level tree-structure where only the root performs all the aggregations, we see that increasing the block-size can exponentially increase the latencies as the aggregations cost at the root is affect by both the size of the signature and the number of signatures.

4.2 Fault tolerance

We simulated node failures in BLSCoSi by randomly choosing x% nodes to become unresponsive during the protocol run. For the experiment we ran 10 rounds of the protocol and averaged their latencies. The BLSCoSi protocol with 10% and 20% failures sees an increase of 2.5x and 4x in the latencies. While the failures at the leaf nodes does not affect the protocol, the failures at the subleader makes the signatures from the entire subtree unavailable. Subleader-change is triggered only through timeout and the current timeout (5 sec for the protocol, 2.5 sec for the sub-protocol) on the BLSCoSi allows for only a single subleader change. In the cases where the subsequent subleaders of a subtree are unresponsive, in the entire subtree fails to participate in the cosigning round.

4.3 Comparison with CoSi

We evaluated CoSi on a similar tree structure as BLSFTCoSi with 0% failures. The smaller block-sizes (¡ 1 MB) showed smaller latencies for BLSCoSi

[1] At https://github.com/dedis/student_18_blsftcosi

but for a larger block-size of 4 MB, showed better latency for CoSi. This can be attributed to the fact the BLS signatures are computationally much more intensive than ECDSA signatures. With larger block sizes and larger number of nodes in BLSCoSi, the computational latencies likely overtake the commmunication latencies. This can also be seen from the fact that the latency graph for BLSCoSi gets much more steeper with block-size increase than the same for CoSi.

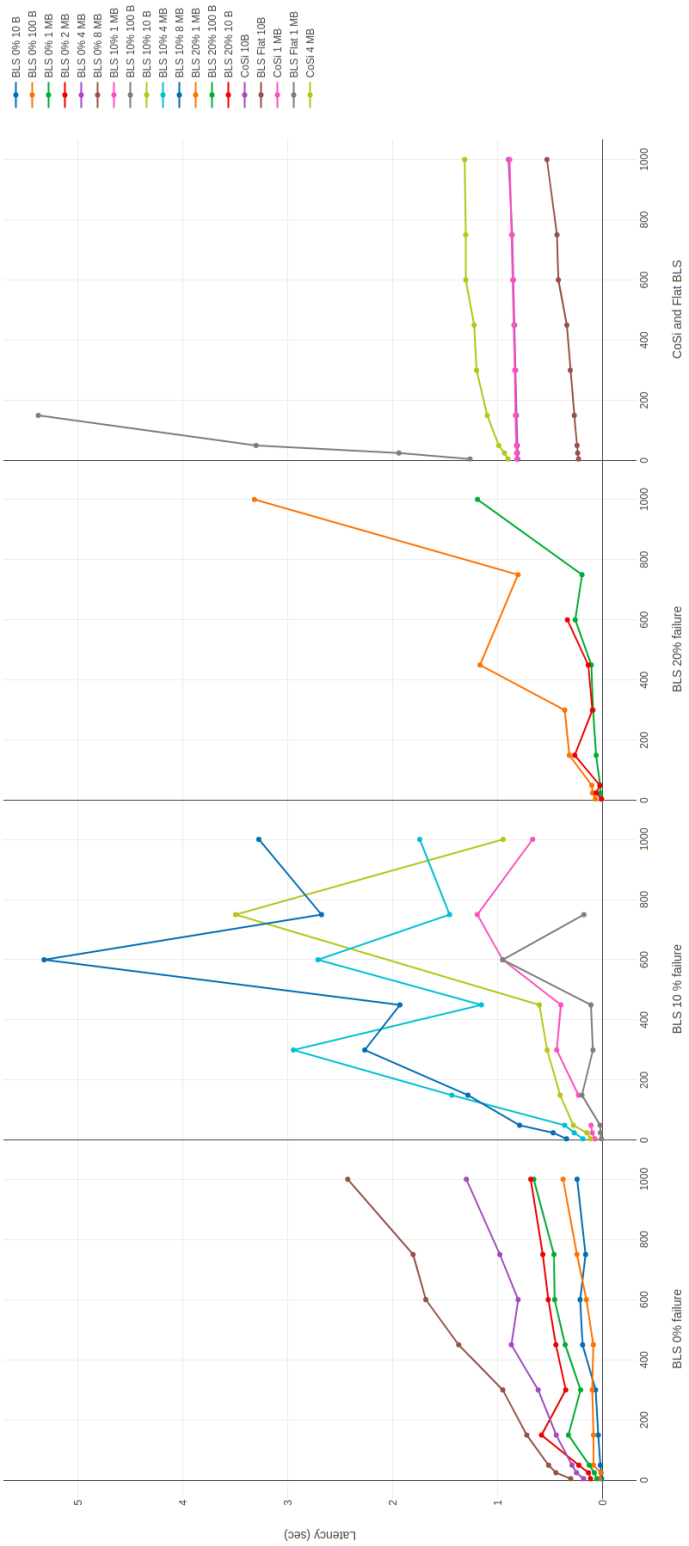

Figure 3: Latency of BLSCoSi round under different scenarios

5 Conclusion

In this report we presented and evaluated BLS-CoSi, a collective signing protocol with BLS signature scheme. The CoSi is prone to Denial-of-Service attacks and loss of availability in highly unstable networks due to its two-round-trip protocol with Schnorr signatures. As a single round-trip protocol, BLS-CoSi prevents these issue. The bilinearity property of groups used in BLS-CoSi also allows for incremental aggregation, thus allowing fast responses on satisfying some threshold.

We evaluated BLS-CoSi with different block-sizes, failure rates and tree structures. For the standard 1 MB blocks, BLS-CoSi showed latencies slightly better than the existing CoSi. The latencies were almost directly proportional to block-size and the number of nodes. But the signature generation complexity factor with elliptic curves can overshadow the communication complexity on large setups with large block-size.

References

[1] Poster: Bitcoin meets collective signing. http://www.ieee-security.org/TC/SP2016/poster-abstracts/16-poster_abstract.pdf.

[2] Merkle tree, Dec 2018. https://en.wikipedia.org/wiki/Merkle_tree.

[3] Cost of a 51% attack, Jan 2019. https://gobitcoin.io/tools/cost-51-attack/.

[4] Deterlab network security testbed, Jan 2019. http://isi.deterlab.net/.

[5] Schnorr signature, Jan 2019. https://en.wikipedia.org/wiki/Schnorr_signature.

[6] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation. OSDI 99. New Orleans, Louisiana, USA: USENIX Association*, pages 173–186, 1999.

[7] Ben Lynn Dan Boneh and Hovav Shacham. Short signatures from the weil pairing. pages 514–532. Springer-Verlag, 2001.

[8] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, July 1982.

[9] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. New software speed records for cryptographic pairings. *Lecture Notes in Computer Science Progress in Cryptology LATINCRYPT 2010*, page 109123, 2010.

[10] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009.

[11] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. *2016 IEEE Symposium on Security and Privacy (SP)*, 2016.