Reliable Collective Cosigning to Scale Blockchain with Strong Consistency

Conference Paper · January 2018

DOI: 10.14722/diss.2018.23005

5 authors, including:

Bithin Alangot
Amrita Vishwa Vidyapeetham
10 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Rahul Krishnan Pathinarupothi
Amrita Vishwa Vidyapeetham
34 PUBLICATIONS **66** CITATIONS
SEE PROFILE

Krishnashree Achuthan
Amrita Vishwa Vidyapeetham
96 PUBLICATIONS **436** CITATIONS
SEE PROFILE

Reliable Collective Cosigning to Scale Blockchain with Strong Consistency

Bithin Alangot*, Maneesha Suresh*, Arvind S Raj*, Rahul Krishnan Pathinarupothi†, Krishnashree Achuthan*
*Amrita Center for Cybersecurity Systems and Networks
†Amrita Center for Wireless Networks & Applications(AmritaWNA)
Amrita School of Engineering, Amritapuri
Amrita Vishwa Vidyapeetham, India
bithina@am.amrita.edu, maneeshas@am.students.amrita.edu, rahulkrishnan@am.amrita.edu,
arvindsraj@am.amrita.edu, krishna@amrita.edu

***Abstract*—The HPlane IoT framework abstracts security and privacy concerns of critical IoT infrastructure in a Remote Healthcare Monitoring (RHM) environment. Despite its usefulness, the framework lacks a scalable access control mechanism leading to performance and scalability challenges. Some of these challenges can be overcome by using Byzcoin blockchain that provides strong consistency guarantee. However, we found limitations in Byzcoin with respect to reliability, performance and high failure probability due to the use of unreliable Collective Cosigning (CoSi) protocol. Our practical analysis shows that on an average 10-30% CoSi protocols fail when it uses a spanning tree topology to scale Schnorr multisignature. Thus use of Byzcoin poses a significant risk to critical IoT infrastructure. In this paper, we present a robust spanning tree topology along with an implementation of BLS multisignature. Our enhanced topology successfully tackles reliability limitation while BLS multisignature improves performance and lowers failure probability. This work also summarizes how blockchains can serve as a controller application to provide an effective scalable access control in HPlane IoT framework.**

I. Introduction

Security and Privacy are major concerns in IoT (Internet of Things) especially for critical IoT infrastructure environments such as with remote healthcare monitoring (RHM). While IoT framework such as HPlane[29] based on Global Data Plane (GDP)[25] has considerably abstracted these concerns for IoT applications, they still lack a scalable access control mechanism to protect data from unauthorized users. The challenges involved in maintaining access permission across multiple untrusted entities such as patients, doctors and hospitals within the RHM makes it difficult to implement this in the HPlane framework. The existing mechanism [30] in HPlane and other systems that provide access control is a centralized one that is susceptible to cyber attacks and limit scalability of IoT infrastructure.

In complex environments such as that of RHM where multiple untrusted parties are involved in maintaining access control policies and/or permissions, we propose a blockchain based solution that provides security and privacy to the medical data. We implement blockchain as a control plane application in HPlane as shown in the Figure 1 with multiple GDP routers (managed by different entities) as validators. The HPlane provides a log based abstraction in which entities are represented as single-write append-only *log*, hence, communication with the entities happens as normal file read and write operations to the *log*. The read/write access request to these *logs* are validated by blockchain validator and once confirmed *log* access details are appended into the blockchain as transactions. This blockchain serves as an immutable access provenance record which could be later referenced for any irregularities.

In designing such a system our first challenge was to select a possible blockchain that provides strong consistency guarantee. This ensures transaction finality without limiting performance which is essential for critical IoT applications. Towards this, variants of BFT (Byzantine Fault Tolerance) are commonly used for underlying blockchain consensus[16], [19], [20], [22], [7], [24]. However, the use of BFT variant consensus algorithm in private blockchains limits its scalability to its high communication complexity (O(n^2))(Here, scalability is in terms of number of validators and not with respect to high transaction process capabilities). This led to our use of Byzcoin[19], that uses combination of CoSi and PBFT (Practical Byzantine Fault Tolerence)[10] algorithm to overcome scalability limitations. Though Byzcoin is a public blockchain, we are adapting it for use in private settings. Despite its scalability, we found several critical limitations:

- **Reliability**: The scalability of PBFT could be accomplished using scalable Collective Co-signing (CoSi)[32] as proposed in Byzcoin. However, this approach has inherent reliability issues in Byzantine setting due to the spanning tree communication topology that CoSi uses to scale.
- **Performance**: Omniledger[20] which uses the same technique as in Byzcoin proposes group communication topology to tackle reliability challenge. In this approach performance degrades beyond 100 nodes and has high bootstrap latency.

Workshop on Decentralized IoT Security and Standards (DISS) 2018
18 February 2018, San Diego, CA, USA
ISBN 1-891562-51-7
https://dx.doi.org/10.14722/diss.2018.23005
www.ndss-symposium.org

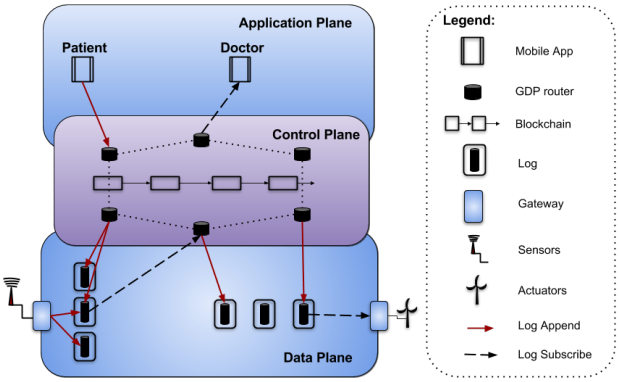

Fig. 1. HPlane architecture with blockchain in the control plane

- **Failure Probability**: CoSi protocol currently use Schnorr multisignature[28] which takes two round trips across the communication topology to sign a statement. The two round trip communication across the existing topology increases the failure probability of protocol.

A. Our Contribution

In this paper, we present an enhancement to the CoSi protocol which enhances both its reliability and performance. The enhanced protocol helps to scale PBFT which is used as underlying consensus protocol for Byzcoin blockchain. The following are the specific contributions of this paper:

- We perform a practical analysis on the reliability of the existing spanning tree topology adopted by CoSi protocol.
- We present a Robust Spanning Tree Topology which enhances the reliability of CoSi protocol that in turn helps build a robust blockchain. The protocol is designed with the assumption that a subset of nodes are trusted. The trust level of nodes are measured based on the QoS (Quality of Service) provided by the cloud providers and its uptime. This assumption is reasonable in private blockchain settings.
- We optimize the CoSi protocol by replacing the Schnorr multi-signature with the BLS multisignature[18], which dwindles the number of messages exchanged thus reducing failure.
- Finally, we discuss how blockchain serve as a control plane application to provide scalable access control mechanism in HPlane framework.

In section II, we give a background on HPlane IoT framework and Byzcoin blockchain. Section III discusses reliability issues of existing CoSi protocol. In section IV, we present our novel robust spanning tree topology that helps increase the reliability of CoSi protocol. Implemention of BLS multisignature and its advantages over Schnorr multisignature is presented in section V. In section VI, we describe the integration of blockchain with HPlane, followed by conclusion in section VII.

II. Background

We give a overview of Byzcoin, a blockchain with strong consistency guarantee and later explain about HPlane, an IoT framework.

A. Byzcoin

Byzcoin blockchain is a Bitcoin-like cryptocurrency enhanced with strong consistency based on the principles of the well-studied PBFT. It builds PBFT atop CoSi protocol, a collective signing protocol that efficiently aggregates hundreds and thousands of signatures. The CoSi protocol uses Schnorr multisignature and scales by arranging nodes in a spanning tree topology. The schnorr multisignature takes four phases i.e. announcement, commitment, challenge and response to successfully sign a statement. These four phases constitute two communication round trips across the spanning tree topology which makes the CoSi protocol highly unreliable. The PBFT view leader uses primitives from CoSi protocol to attest its prepare and commit certificate such that it can assure that at least $2f + 1(f$ faculty nodes) nodes have signed (seen) certificates out of $3f+1$ nodes. The PBFT was not designed to scale to large number of nodes[19], however, CoSi enhanced PBFT can scale to thousands of nodes as CoSi helps to reduces PBFT's communication complexity from O(n^2) to O(n).

B. HPlane: Healthcare Plane

The HPlane IoT framework provides a log based storage abstraction to address various challenges in security, privacy, latency, scalability and bandwidth of RHM IoT infrastructure that follows a three tier architecture. The architecture includes sensors or actuators which generates data at tier 1, gateways to connect to the internet at tier 2 and cloud for storage and interconnecting IoT devices at tier 3. As shown in the Figure 1, the framework divides the IoT infrastructure into three logical planes: application plane, control plane and data plane. The application plane contain the IoT applications such as Amrita Spandanam [21] which runs on the patient's smartphone. The control plane application, also called controller, spans across both the data and application plane. The controller applications help decide the flow of data across the infrastructure and determine access control. The data plane contains the log storage server which is either located at the cloud or edge servers. In this work, we describe how to enhance the blockchain build over CoSi that helps build a scalable controller application for HPlane IoT framework.

III. Analysis of CoSi reliability

In this section, we show why the current implementation of CoSi used in Byzcoin blockchain is unreliable under Byzantine settings. The results from our analysis in this section serves as the motivation for us to build a robust spanning tree topology discussed in the next section.

The practical analysis was done by running CoSi protocol with 1000 nodes arranged in a spanning tree topology with varying branching factors. The idea was to fail 50 and 100 nodes respectively for 100 protocol runs and observe the

TABLE I
NOTATIONS AND DEFINITION FOR TREE CONSTRUCTION

N	Total no. of nodes in the spanning tree
T	No. of trusted nodes in N
U	No. of untrusted nodes in N $(N-T)$
h	Height of a tree
$h(N)$	Height of tree with N nodes
$N_{complete}$	No. of nodes in a complete binary tree of height h
N_{fill}	if $N_{complete} > T : h(N_{complete}) == h(T)$ then $N_{complete} == T + N_{fill} : N_{fill} \subset U$
U_r	$U - N_{fill}$
B_{min}	Min. no of untrusted nodes which a trusted node can have as its children

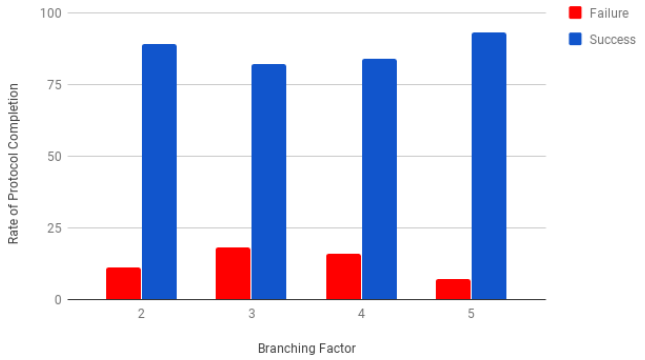

Fig. 2. Trial 1: Fail 50 each in 100 runs of CoSi protocol with 1000 nodes and varying branching factors for the spanning tree.

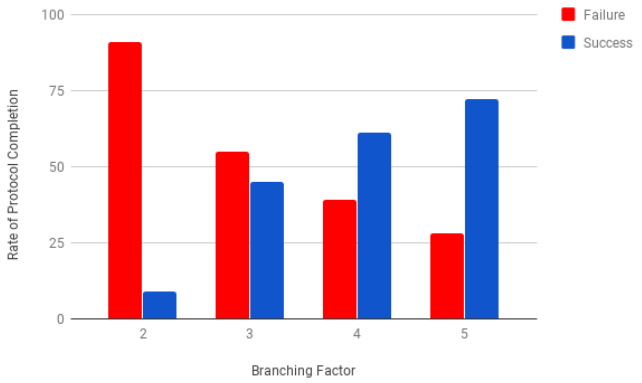

Fig. 3. Trial 2: Fail 100 nodes each in 100 runs of CoSi protocol with 1000 nodes build as a spanning tree topology with varying branching factors.

protocol failure rate. The protocol fails if the effective number of nodes failed goes beyond f. Here, f is the failure factor derived from $f = (n-1)/3$ where n is the total number of nodes participating in the protocol. Although the actual number of nodes that we failed (50 and 100) was always designed to be below the failure factor, the tree topology induced deviations that caused the effective number of failure nodes to go beyond f leading to protocol failure. The results from two trials as shown in Figure 2 and Figure 3 shows that, on average 10% CoSi protocol instance fail when actual number of nodes failed is 50 nodes. As we increase the actual number of nodes failed to 100, the average protocol failure increased to 25-30%. This level of unreliability is not acceptable in case of critical services such as with RHM. The next two sections present our novel approaches to increase the reliability of CoSi.

IV. ROBUST SPANNING TREE TOPOLOGY

Our objective was to develop a robust spanning tree topology construction that provides tree fault tolerance in the event of byzantine behaviour by the nodes. Our tree construction technique assumes that a subset of nodes are trusted and these nodes are placed within the tree such that untrusted nodes cannot compromise liveness by DoS attack. The tree construction only takes $O(n)$ time complexity, hence, the protocol does not lead to significant overhead. The three steps in the tree construction is shown with an example in the Figure 4.

A. Notations and definition

Table I introduces the notations that is followed in this section. Our assumption is that out of N nodes, a subset of T nodes are trusted. Our intelligent tree building strategy caters to this need and makes sure that the traffic load from untrusted nodes are equally distributed across all the trusted nodes in the tree. In future, we also plan to consider the locality of the nodes to increase the performance.

B. Tree construction

The spanning tree is constructed in three logical steps as shown in the Fig. 4. We assume that identities of trusted nodes known. Trusted nodes are those with higher security and uptime (basically with higher QoS in a cloud environment). The details on how trust is defined is kept as a future work.

Step 1: Using Equation 1 calculate the height of the tree which is formed with T nodes. If $N_{complete} < T$, add N_{fill} nodes to T and create a complete binary tree. Now, we have a complete binary tree with $N_{fill} + T$ nodes.

Step 2: Calculate B_{min} using Equation 2 and assign B_{min} nodes to each T nodes from top to bottom until U_r nodes are completely assigned.

Step 3: This step is necessary if U_r is not empty, so we calculate how to distribute the remaining U_r nodes amongst the trusted nodes. Using Equation 3 we calculate B_{extra} which defines how many more trusted nodes (from top to bottom) needs to be assigned additional one more untrusted node.

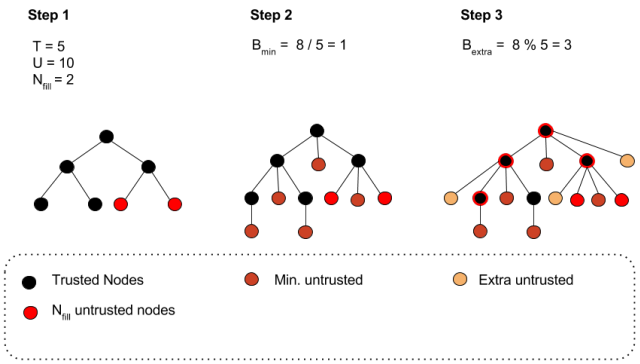

Fig. 4. An example spanning tree construction

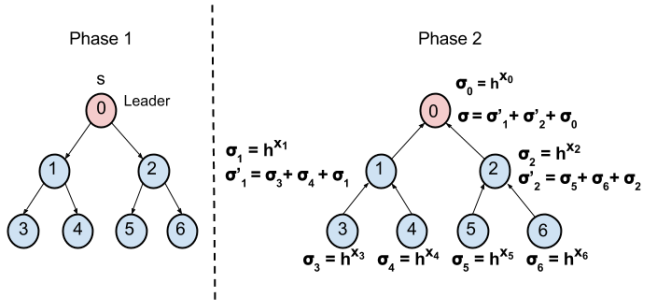

Fig. 5. Signtaure aggregation using spanning tree topology. In Phase 1, leader multicasts(Top down) the statement(S) to be signed through the spanning tree. In Phase 2, the nodes compute a signature and the partial aggregate signature and passes it up to its parent until it reaches the leader.

$$h = log_2(T+1) \tag{1}$$

$$B_{min} = U_r/T \tag{2}$$

$$B_{extra} = U_r\%T \tag{3}$$

Our approach of equally dividing the load from untrusted nodes amongst the trusted nodes helps to increase the performance of CoSi. In depth analysis of distribution of trusted and untrusted nodes for optimal performance will be presented in the final version of this paper. We explained the tree contruction in steps for more clarity, although the actual tree building is done in one single step.

Our tree construction method will replace CoSi's present spanning tree topology that doesn't take into account the reliability or locality of the nodes. In the next section we explain the second contribution which helps to reduce the failure probability of CoSi protocol.

V. BLS Multisignature

In this section, we briefly explain how to implement BLS multisignature which accomplishes the functionalities of Schnorr multisignature in one round trip across the spanning tree topology. Unlike other digital signature schemes, BLS signatures [8] are far shorter in length. For e.g. Digital Signature Standard (DSS) is 320 bit long while BLS signature is only 160 bit long.

BLS signature scheme is based on Gap Diffie Hellman(GDH) class[3] where Computational Diffie Hellman(CDH) problem is hard but Decisional Deffie Hellman(DDH) problem is easy. The scheme uses bilinear pairing over elliptic curves for signature verification. Bilinear pairing is one of the functions on specified elliptic curves. Now in case of BLS multisignature[18], let us consider a group $U = \{u_1, u_2, ..., u_n\}$ of n signers taking part in digitally signing a statement. Here, i denote the index of the signer and e denote the bilinear pairing function.

The key steps involved in the BLS multisignature scheme are:

- **Key Generation**: The secret key of the signer $u_i \in U$ is a random element $x_i \in Z_p^*$ while his public key is $v_i = g^{x_i}$ where g is a generator of a group G.
- **Signing**: H is a one-way hash function, which outputs a random element in the whole group G and m is the statement to be signed. The signer $u_i \in L$ computes $h = H(m)$ and returns $\sigma_i = h^{x_i}$.
- **Aggregation**: The issuer of a multisignature finally collects all σ_i generated by u_i and computes the aggregate signature $\sigma = \sum_{i=0}^{n} \sigma_i$ and returns (m, L, σ).
- **Verification**: When the verifier is given g,m, L and σ, it collects all v_i by L, and computes $v = \sum_{i=0}^{n} v_i$, $h = H(m)$, and verifies $e(g, \sigma) = e(v, h)$.

Using these four steps one can sign a statement and verify the signature. Suppose a signer generates a signature σ_i. If the signature is correct, then the verifier can verify the signature with the result of *bilinear pairing* using the Equation 4 and Equation 5.

$$e(g,\sigma) = e(g, \sum_{i=0}^{n} \sigma_i) = e(g, \sum_{i=0}^{n} h^{x_i}) = e(g,h)^{\sum_{i=0}^{n} x_i} \tag{4}$$

$$e(v,h) = e(\sum_{i=0}^{n} v_i, h) = e(\sum_{i=0}^{n} g^{x_i}, h) = e(g,h)^{\sum_{i=0}^{n} x_i} \tag{5}$$

If $e(g,\sigma) = e(v,h)$, then all the individual signatures σ_i have been generated correctly.

As shown in Figure 5, signature aggregation takes only one round trip across the spanning tree topology. We have implemented BLS multisignature using a library known as Pairing-based Cryptography(PBC)[6]. PBC Library is a free portable C library built on the GMP library that performs the mathematical operations underlying pairing based cryptosystems. It provides routines such as elliptic curve generation, elliptic curve arithmetic and pairing computations. A preliminary implementation of our BLS multisignature can be found in our GitHub[1]. The CoSi with our robust spanning tree

topology and BLS multisignature will be much more reliable than the current implementation. A blockchain using PBFT with enhanced CoSi as the consensus protocol can effectively serve as a controller application for HPlane as described in the next section.

VI. Blockchain as Control Plane Service

Here, we give a brief overview of our idea on how we use the enhanced blockchain in HPlane framework. As explained before blockchain serves as a control plane application in HPlane.

As shown in the Fig. 1, there are multiple entities such as hospitals, doctors and patients in our infrastructure. These entities interact with each other through APIs to provide timely care to the patients located in the remote locations. Here, our primary aim is to validate and record communication between entities with the help of blockchain at the control plane layer. We consider each communication as a transaction that is validated by a quorum of blockchain validators (quorum is decided by the framework based on the privacy settings). In HPlane, the entities are abstracted with a *log*, so communication with entities happen with either a read or write operation to the *log* for which APIs are provided.

The HPlane framework provides two APIs, APPEND and SUBSCRIBE to applications to manage data stored in *logs*. A transaction (it can be either a APPEND or SUBSCRIBE request) first goes to a GDP router that forwards it to appropriate *logs*. Before forwarding the transaction, it is validated as part of blockchain consensus to confirm the authenticity in accordance with access control policies defined using our policy machine [15]. Once validated a transaction along with other transactions validated during a particular time window is batched into a block and added to the blockchain which stays as an immutable access provenance record. The access control policy details are not maintained in the blockchain, but this would be presented as part of the design in the full version of this paper. Here, inorder to validate a transaction, the validators need not traverse through the blockchain like in other cryptocurrenies such as Bitcoin[27]. As our Byzcoin blockchain with enhanced PBFT consensus provides instant transaction finality with high reliability, it helps to build a scalable access control mechanism for HPlane with minimal overhead.

VII. Related Work

A preliminary idea was proposed by Hossein et al. to integrate blockchain with GDP IoT framework[31]. They use blockchain as a auditable access control layer, however, did not explore the scalability challenges of blockchain when used in a IoT infrastructure such as RHM. Inspite of blockchain scalability being actively researched in both public and private settings, it continues to be an open problem. Bitcoin-NG[14] described scaling Bitcoin[27] blockchain by decoupling transaction verification and block mining. However, Bitcoin-NG is susceptible to double spending attack which Lio et al. addresses using Elastico[22]. Though the above blockchains scale well, they have severe performance and computational overhead due to the use of PoW (Proof-of-Work) consensus algorithm. Kyle et al.[11] proposed use of BFT as an efficient consensus protocol for blockchain but using BFT in an open setting (permissionless blockchain) can make them susceptible to Sybil attacks[12]. Eleftherios et al.[19] used a combination of PoW and BFT in ByzCoin blockchain which scales without incurring significant overhead. In OmniLedger[20], Eleftherios et al. manages to get away with PoW as underlying consensus protocol and relied completely on BFT with VRF (verifiable random function)[23]. The VRF provided protection against Sybil attack when deployed in an open setting without incurring computation overhead. Omniledger also enhances CoSi's spanning tree topology with group communication topology to make it more reliable. However, there was a definitive performance degradation when scaled beyond 100 nodes. The popular private blockchain frameworks such as Tendermint[7], Hyperledger Fabric[9], Quorum[26], Kadena.io[5] and Chain[2] also use variants of BFT. It is important to note that these private blockchain frameworks scale only from the perspective of number of transactions and do not scale when the number of validators increase. Hashgraph[4] which uses aBFT (Asynchronous BFT) also has inherent scalability limitation although it can process large number of transactions each second.

The approach of scaling BFT using CoSi helps to improve the above blockchain frameworks. However, the unreliability of CoSi's spanning tree topology comes as a barrier for its use in blockchain which powers critical IoT infrastructure. The reliability of spanning tree topology has been well researched in wireless sensor networks[17] and multicast protocols. Darwin et al.[13] presents a robust spanning tree but nevertheless looked into the idea of constructing spanning tree while subset of nodes are highly reliable.

VIII. Conclusion

The HPlane IoT framework deals with security and privacy concerns of critical RHM IoT infrastructure. However, it lacks a scalable access control mechanism that causes performance challenges. We feel that a blockchain based controller application for HPlane could provide a scalable access control mechanism without affecting the performance. We chose the Byzcoin blockchain to ensure strong consistency guarantee essential for critical IoT infrastructure. But, Byzcoin has reliability concerns due to use of CoSi protocol which limits its application to critical infrastructure. Our approach that uses a combination of robust spanning tree topology and BLS multisignature will increase the reliability of CoSi making Byzcoin blockchain suitable for critical IoT infrastructure.

Acknowledgment

We deeply appreciate the help of our colleagues and friends who reviewed the paper. We are grateful to the Chancellor of Amrita Vishwa Vidyapeetham, Sri Mata Amritanandamayi Devi, for constant support and encouragement in conducting research that has direct societal impact.

REFERENCES

[1] "BLS Multisignature," https://github.com/manishas053/BLS-Multisignature, accessed 2017-12-09.

[2] "Chain Protocol," https://chain.com/docs/1.2/protocol/papers/whitepaper, accessed 2017-11-25.

[3] "GDH class," https://en.wikipedia.org/wiki/XDH_assumption, accessed 2017-11-25.

[4] "Hashgraph," https://hashgraph.com/, accessed 2017-11-25.

[5] "Kadena.io," http://kadena.io/, accessed 2017-11-25.

[6] "PBC library," https://crypto.stanford.edu/pbc/, accessed 2017-11-25.

[7] "Tendermint," https://tendermint.com/, accessed 2017-11-25.

[8] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Advances in CryptologyASIACRYPT 2001*, pp. 514–532, 2001.

[9] C. Cachin, "Architecture of the hyperledger blockchain fabric," in *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[10] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, 1999, pp. 173–186.

[11] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 106–125.

[12] J. R. Douceur, "The sybil attack," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 251–260.

[13] D. England, B. Veeravalli, and J. B. Weissman, "A robust spanning tree topology for data collection and dissemination in distributed environments," *IEEE transactions On parallel and distributed systems*, vol. 18, no. 5, 2007.

[14] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, "Bitcoin-ng: A scalable blockchain protocol." in *NSDI*, 2016, pp. 45–59.

[15] D. Ferraiolo, V. Atluri, and S. Gavrila, "The policy machine: A novel architecture and framework for access control policy specification and enforcement," *Journal of Systems Architecture*, vol. 57, no. 4, pp. 412–424, 2011.

[16] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles*. ACM, 2017, pp. 51–68.

[17] O. Gnawali, R. Fonseca, K. Jamieson, M. Kazandjieva, D. Moss, and P. Levis, "Ctp: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 1, p. 16, 2013.

[18] M. Inamura, K. Iwamura, R. Watanabe, M. Nishikawa, and T. Tanaka, "A new tree-structure-specified multisignature scheme for a document circulation system," in *Security and Cryptography (SECRYPT), 2011 Proceedings of the International Conference on*. IEEE, 2011, pp. 362–369.

[19] E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, "Enhancing bitcoin security and performance with strong consistency via collective signing," in *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016, pp. 279–296.

[20] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger." *IACR Cryptology ePrint Archive*, vol. 2017, p. 406, 2017.

[21] R. Krishnan, M. Ramesh *et al.*, "A low cost remote cardiac monitoring framework for rural regions," in *Proceedings of the 5th EAI International Conference on Wireless Mobile Communication and Healthcare*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015, pp. 231–236.

[22] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 17–30.

[23] S. Micali, M. Rabin, and S. Vadhan, "Verifiable random functions," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*. IEEE, 1999, pp. 120–130.

[24] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 31–42.

[25] N. Mor, B. Zhang, J. Kolb, D. S. Chan, N. Goyal, N. Sun, K. Lutz, E. Allman, J. Wawrzynek, E. A. Lee *et al.*, "Toward a global data infrastructure," *IEEE Internet Computing*, vol. 20, no. 3, pp. 54–62, 2016.

[26] J. Morgan, "Quorum," https://www.jpmorgan.com/global/Quorum, accessed 2017-11-26.

[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[28] K. Ohta and T. Okamoto, "Multi-signature schemes secure against active insider attacks," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 82, no. 1, pp. 21–31, 1999.

[29] R. K. Pathinarupothi, B. Alangot, M. V. Ramesh, K. Achuthan, and P. V. Rangan, "H-plane: Intelligent data management for mobile healthcare applications," in *International Conference on Mobile Web and Information Systems*. Springer, 2016, pp. 283–294.

[30] I. Ray, B. Alangot, S. Nair, and K. Achuthan, "Using attribute-based access control for remote healthcare monitoring," in *Software Defined Systems (SDS), 2017 Fourth International Conference on*. IEEE, 2017, pp. 137–142.

[31] H. Shafagh, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," *arXiv preprint arXiv:1705.08230*, 2017.

[32] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford, "Keeping authorities" honest or bust" with decentralized witness cosigning," in *Security and Privacy (SP), 2016 IEEE Symposium on*. Ieee, 2016, pp. 526–545.