(Re)-envisioning Approximate Agreement for Distributed Cryptography and Oracles

Akhil Bandarupalli
Purdue University

Saurabh Bagchi
Purdue University

Aniket Kate
Purdue University and Supra Research

***Abstract*—Asynchronous agreement protocols are crucial in various emerging applications, spanning from blockchains to fault-tolerant cyber-physical systems. Present asynchronous agreement protocols employ either randomization, incurring substantial computation overhead, or approximate agreement techniques, leading to high $\tilde{\mathcal{O}}(n^3)$ communication for an n-node system. We address this inefficiency of approximate agreement protocols in the context of two different applications: a) Convex Agreement for Distributed Oracles, and b) Asynchronous random beacons, by proposing two protocols titled DELPHI and HASHRAND, respectively.**

1. Introduction

Consensus protocols are the bedrock of secure and dependable distributed systems, which enable parties or nodes to reach agreement on their states. Protocols that achieve consensus in the presence of malicious faults (termed Byzantine faults) are called Byzantine Agreement (BA) protocols. Much research has been done on building efficient BA protocols under varying network and fault assumptions. We primarily consider the asynchronous network model, where messages can be delayed by an arbitrary unknown amount of time. This model represents networks ranging from the internet connecting geo-distributed blockchains to ad-hoc networks connecting Cyber-Physical Systems (CPS).

Asynchronous BA is bound by the famed FLP impossibility result [16], which states that it is impossible for any deterministic protocol to achieve asynchronous BA in the presence of a single faulty node. Protocols circumvent this impossibility using tools of randomization like common coins, which enable nodes to terminate with a probability $p > 0$ in each round.

However, common coins have three major drawbacks. First, they require computationally expensive threshold cryptographic operations. The most efficient and popular implementation based on threshold BLS signatures [4] requires $\mathcal{O}(n)$ bilinear pairing computations per coin [3], [6]. Each pairing requires $1000\times$ more time and energy than symmetric key primitives used in TLS. Second, they often require a threshold cryptographic setup amongst the participant nodes. Deploying this setup involves running an expensive Asynchronous Distributed Key Generation (ADKG) procedure amongst nodes [12], [18]. Third, most current implementations rely on the discrete log or factoring hardness assumptions, which are not true in the advent of quantum computers [24].

The Asynchronous Approximate Agreement (AAA) primitive offers an alternate way to circumvent the FLP impossibility without using common coins. This primitive enables honest nodes to agree on values within ϵ distance of each other. AAA protocols do not require any expensive public-key or threshold cryptography, which make them computationally efficient and also Post-Quantum secure by default. However, prior AAA protocols have at least $\mathcal{O}(n^3)$ communication complexity [1], which makes them communication inefficient and causes problems with their scalability. We address this inefficiency of AAA protocols and demonstrate their effectiveness in two applications.

2. Convex Agreement for Distributed Oracles

Many applications in fault-tolerant distributed CPS [15], [20], [23], and oracle networks [5], [7] involve a set of n nodes or oracles, each with an input coming from its sensor device, agreeing on a common output. In particular, oracle networks on blockchains empower applications in the DeFi space, like provisioning loans and trading futures, by providing reliable market data about stocks and cryptocurrencies [19]. In this context, the output of an ideal system is close to the range of honest inputs and must be representative of the ground truth value. The convex agreement problem [8] captures this requirement.

Definition 2.1. ***Convex Agreement****: We denote the set of honest nodes' inputs by V_{h}. A protocol Π_{AA} for n nodes where node i inputs v_i and outputs o_i solves the convex agreement problem with ρ-relaxed validity if it satisfies the following properties.*

1) ***Termination****: Each honest node must eventually produce an output o_i.*
2) ***ρ-relaxed Min-Max Validity****: Each honest node's output must be within the ρ-relaxed interval formed by honest inputs. For every honest node $i \in \mathcal{P}$: $m - \rho \leq o_i \leq M + \rho$, where $m = \min(V_{\mathsf{h}})$ and $M = \max(V_{\mathsf{h}})$.*
3) ***ϵ-agreement****: The outputs of any pair of honest nodes i and j are within ϵ of each other, i.e., $|o_i - o_j| < \epsilon$.*

2.0.1. Claims Overview. We propose DELPHI, an efficient convex agreement protocol for distributed Oracles. DELPHI has a communication complexity of $\mathcal{O}(\ell n^2 \log(\lambda \log(n)))$ bits, where ℓ is the size of the input and λ is a statistical security parameter. In exchange for this efficiency, DELPHI sacrifices Validity by allowing nodes to output values ρ

2833-292X/24/$31.00 ©2024 IEEE
DOI 10.1109/DSN-S60304.2024.00026

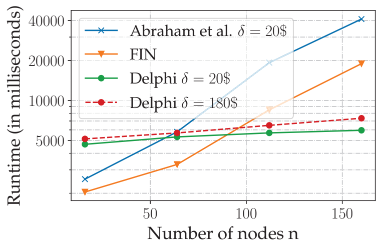

Figure 1: DELPHI **evaluation: runtime vs** n **on AWS**: Nodes are agreeing on the price of Bitcoin. $\delta = M - m$ in US Dollars. DELPHI's runtime increases much slowly with n than FIN [14], the SOTA convex BA protocol and Abraham et al. [1], the best AAA protocol, and does not vary much with range δ.

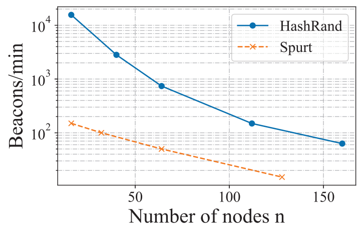

Figure 2: HASHRAND **evaluation: Throughput vs** n **on AWS**: HASHRAND's computational efficiency allows it to produce to produce 1 beacon per second at $n = 136$ without a threshold setup, which is 5x higher than Spurt [11] at $n = 128$.

distance outside the convex-hull of honest inputs. DELPHI is also the first work to explore a tradeoff between efficiency and Validity of distributed consensus protocols.

2.0.2. Techniques. We propose BINAA, the first building block of DELPHI. BINAA achieves approximate agreement with binary inputs, i.e. either 0 or 1, where parties output values within ϵ distance with $\mathcal{O}(n^2 \log \frac{1}{\epsilon})$ communication. Next, we divide the real number line into checkpoints separated by distance ρ and run a BINAA instance for each. A node i inputs 1 to a checkpoint μ_i only if $|v_i - \mu_i| \leq \rho$. If all inputs v_i are within range ρ, then at least one checkpoint μ_i will have weight $w_i = 1$. Finally, we run BINAA for checkpoints with increasing separation $\rho_j \in [\rho_0, \Delta]$. We set Δ based on the distribution of inputs such that honest input range is $\leq \Delta$ with probability $p = 1 - \frac{1}{2^\lambda}$. After ensuring at least one checkpoint's weight is 1, we use a novel weighted average aggregation to ensure ϵ-agreement within ρ-relaxed convex hull of inputs. We conducted experiments in a geo-distributed testbed on AWS for the oracle network application. We demonstrate the results in Fig. 1.

3. Asynchronous Random Beacons

Random beacon schemes [17], [26] emit random numbers at regular intervals. They are often used as a source of secure randomness for common coins in asynchronous BA [2], [22], SMR [10], [21], MPC [9], [25], and also in layer-2 apps in the DeFi and GameFi space [19].

Definition 3.1. *A beacon protocol B consists of two subprotocols $B.Prep(.,.)$ and $B.Open(.)$, with outputs $\langle x, b_x\rangle$: $x \in \mathbb{N}, b_x \in \mathcal{D}$. Nodes invoke $B.Prep(x,y)$: $x, y \in \mathbb{N}, x \leq y$ to prepare beacons from x to y. Upon terminating $B.Prep(x,y)$, and outputting $\langle x, b_x\rangle$ $\forall x \in [1,y)$, nodes invoke $B.Open(y)$ to generate $\langle y, b_y\rangle$.*

1) ***Agreement**: If honest nodes j and k output $\langle x, b_{xj}\rangle$ and $\langle x, b_{xk}\rangle$ respectively, then $b_{xj} = b_{xk} \forall x \in \mathbb{N}$.*
2) ***Liveness**: Every honest node j must eventually output a beacon $\langle x, b_x\rangle$ for all $x \geq 1$.*
3) ***Bias-resistance and Unpredictability**: Given that no honest node invoked $B.Open(x)$ for $x \geq 1$, the adversary $\mathcal{A}$ should not have any advantage in predicting b_x. $Pr[\mathcal{A}(x) = b_x] = \frac{1}{|\mathcal{D}|}$*

Current asynchronous random beacon protocols either rely on a threshold setup or are highly computationally expensive with $\Omega(n^2)$ discrete log exponentiations per beacon [11]. This computational cost is a scalability bottleneck.

3.0.1. Claims Overview. We present HASHRAND, a beacon that uses hash functions, a secure channel setup, and an efficient AAA protocol. HASHRAND has an amortized $\mathcal{O}(\lambda n \log(n))$ communication complexity and requires amortized $\mathcal{O}(n \log(n))$ Hash computations per beacon per node. HASHRAND is computationally efficient because it only uses lightweight cryptographic primitives like hash functions. HASHRAND is also post-quantum secure assuming that hash functions like SHA256 behave like a Quantum Random Oracle.

3.0.2. Techniques. Conventional random beacon protocols consist of two building blocks: Verifiable Secret Sharing (VSS) and consensus [11]. Each node uses VSS to share a uniformly random element and then nodes agree on $t+1$ nodes whose VSS instances have terminated to ensure the contribution of at least one honest node. However, this design does not work in asynchrony because consensus itself requires a beacon. We utilize the design in [13] to break this circularity by using the AAA primitive, which enables nodes to approximately agree on a random number. Then, nodes round off their outputs to the nearest checkpoint, which results in agreement with high probability (termed Monte-Carlo agreement). Further, HASHRAND incorporates a novel Hash-based weak AVSS protocol with the same communication as discrete-log-based AVSS, but with much lesser computational cost. It also uses the efficient BINAA protocol for AAA. We conducted experiments on a geo-distributed testbed on AWS and measured the throughput of HASHRAND in Fig. 2.

4. Conclusion

We demonstrated the effectiveness of AAA protocols in two different applications. Finally, we note that improvements in communication complexity of AAA protocols offer a compounding advantage over asynchronous BA protocols because of AAA protocols' inherent computational efficiency and Post-Quantum security properties.

References

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th International Conference on Principles of Distributed Systems*, OPODIS'04, page 229–239, Berlin, Heidelberg, 2004. Springer-Verlag.

[2] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 337–346, 2019.

[3] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[4] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 416–432. Springer, 2003.

[5] Lorenz Breidenbach, Christian Cachin, Alex Coventry, Ari Juels, and Andrew Miller. Chainlink off-chain reporting protocol. *URl: https://blog. chain. link/off-chain-reporting-live-on-mainnet*, 2021.

[6] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: Practical asynchronous byzantine agreement using cryptography (extended abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, page 123–132, New York, NY, USA, 2000. Association for Computing Machinery.

[7] Prasanth Chakka, Saurabh Joshi, Aniket Kate, Joshua Tobkin, and David Yang. Dora: Distributed oracle agreement with simple majority. In *Proceedings of the 43rd International Conference on Distributed Computing Systems(ICDCS'23)*, pages 1–13. IEEE, 2023.

[8] Andrei Constantinescu, Diana Ghinea, Roger Wattenhofer, and Floris Westermann. Meeting in a convex world: Convex consensus with asynchronous fallback. *IACR Cryptol. ePrint Arch.*, page 1364, 2023.

[9] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *International workshop on public key cryptography*, pages 160–179. Springer, 2009.

[10] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. Narwhal and tusk: A dag-based mempool and efficient bft consensus. In *Proceedings of the Seventeenth European Conference on Computer Systems*, EuroSys '22, page 34–50, New York, NY, USA, 2022. Association for Computing Machinery.

[11] Sourav Das, Vinith Krishnan, Irene Miriam Isaac, and Ling Ren. Spurt: Scalable distributed randomness beacon with transparent setup. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2502–2517, 2022.

[12] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 2518–2534. IEEE, 2022.

[13] Luciano Freitas de Souza, Petr Kuznetsov, and Andrei Tonkikh. Distributed randomness from approximate agreement. In Christian Scheideler, editor, *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*, volume 246 of *LIPIcs*, pages 24:1–24:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[14] Sisi Duan, Xin Wang, and Haibin Zhang. Fin: Practical signature-free asynchronous common subset in constant time. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 815–829. Association for Computing Machinery, 2023.

[15] El Mahdi El-Mhamdi, Sadegh Farhadkhani, Rachid Guerraoui, Arsany Guirguis, Lê-Nguyên Hoang, and Sébastien Rouault. Collaborative learning in the jungle (decentralized, byzantine, heterogeneous, asynchronous and nonconvex learning). *Advances in Neural Information Processing Systems*, 34:25044–25057, 2021.

[16] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[17] John Kelsey, Luís TAN Brandão, Rene Peralta, and Harold Booth. A reference for randomness beacons: Format and protocol version 2. Technical report, National Institute of Standards and Technology, 2019.

[18] Eleftherios Kokoris Kogias, Dahlia Malkhi, and Alexander Spiegelman. *Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures.* Association for Computing Machinery, New York, NY, USA, 2020.

[19] Chainlink Labs. 77+ smart contract use cases enabled by chainlink. https://blog.chain.link/smart-contract-use-cases/#post-title, 2019.

[20] Jiani Li, Waseem Abbas, Mudassir Shabbir, and Xenofon Koutsoukos. Byzantine resilient distributed learning in multirobot systems. *IEEE Transactions on Robotics*, 2022.

[21] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of bft protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42, 2016.

[22] Achour Mostéfaoui, Hamouma Moumen, and Michel Raynal. Signature-free asynchronous binary byzantine consensus with t¡ n/3, o (n2) messages, and o (1) expected time. *Journal of the ACM (JACM)*, 62(4):1–21, 2015.

[23] Hyongju Park and Seth A Hutchinson. Fault-tolerant rendezvous of multirobot systems. *IEEE transactions on robotics*, 33(3):565–582, 2017.

[24] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, oct 1997.

[25] Victor Shoup and Nigel P. Smart. Lightweight asynchronous verifiable secret sharing with optimal resilience. Cryptology ePrint Archive, Paper 2023/536, 2023. https://eprint.iacr.org/2023/536.

[26] Open source contributors. Drand - a distributed randomness beacon daemon - https://github.com/drand/drand, 2019.