**Fall 2023 ISEC 885: Idea Concept**

**Introduction:**

The exploratory research for the Fall 2023 ISEC 885 course aims to develop a problem direction for a doctoral research idea concept paper, leading to an idea paper, with the ultimate objective of improving certain aspects within the field of asynchronous consensus such as throughput, latency, security, or decentralization, yet to be determined.Asynchronous consensus refers to the ability of a distributed system to reach an agreement without any timing assumptions even in the events of faults, message delays, and unpredictable delivery times. Asynchronous consensus ensures that the system can reach an agreement even when there is no bound on the time it takes for messages to be delivered or for nodes to respond. Not knowing the bound on the time in an asynchronous setting directly contrast synchronous consensus, which does know its bound on time, along with partially-synchronous consensus, which knows there is a bound on time, but doesn't know what that bound is (Miller et al., 2016 "Many Forms of Timing Assumptions," p. 3). A problem with synchronous or partially synchronous protocols are their failure or suffering of performance degradation when network conditions are unpredictable, such as on the open internet. This is directly supported in the works of the Honey Badger Byzantine Fault Tolerant (HBFT) protocol whose research highlights the ability to bring down a well researched partially-synchronous protocol called Practical Byzantine Fault Tolerant (PBFT). This is done by the means of an adversarial network scheduler abusing the leader selection and halting progress by making the intermittent synchronicity connections too small for the network to catch up (Miller et al., 2016 "When Weak Synchrony Fails," p. 4). In contrast, the advantage of asynchronous consensus and by nature the HBFT, is that during these smaller intermittent windows of synchronicity, the HBFT still makes progress. Highlighting the benefit that the HBFT can synchronize relatively instantaneously when the network connects again, in contrast to synchronous protocols which might not have a long enough synchronization window to catch up. It is important to note that the HBFT is significant to the realm of asynchronous consensus as it touts itself as “the first practical asynchronous Byzantine Fault Tolerant (BFT) protocol, which guarantees liveness without making any timing assumptions” (Miller et al., 2016 "Abstract," p. 1). This is relevant because of how HBFT’s asynchronous consensus relates to the Fischer, Lynch, Paterson (FLP) theorem. The FLP theorem states “an asynchronous distributed system where even a single process can crash, it is impossible to achieve consensus materialistically if there is a possibility of message delays and process failures.”(Fischer et al., 1985) It is crucial to note that HBFT argues to practically get around the FLP theorem asynchronously by using cryptographic techniques and assumptions, such as threshold signatures, rather than strictly refuting the FLP result. This is supported in future asynchronous consensus research such as in another asynchronous consensus protocol called BEAT, who states “However, state machine replication cannot be achieved in asynchronous environments (Fischer et al., 1985), unless it uses randomization to circumvent this impossibility result. HoneyBadgerBFT and BEAT fall into this category.”(Duan et al., 2018 “Related Work” p. 3 ) From this statement we can see research that supports the FLP theorem can be circumvented through randomization and that other research has supported HBFT’s claims of being one of the first practical asynchronous BFT protocols to do so. HBFT will be chosen as a bench mark for this research as it has been done before in the field of asynchronous consensus through newer protocols such as BEAT, DUMBO, and Asynchronous Byzantine Fault Tolerance (ABFT) that will later be discussed (Knudsen et al., 2021,Duan et al., 2018, and Guo, Lu, Tang, Xu, & Zhang, 2020). The ABFT protocol was chosen for the upper bounds of this research because it is an amalgamation of the works of previous asynchronous consensus protocols like HBFT, BEAT, and DUMBO (Knudsen, Li, Notland, Haro, & Ræder, 2021 "ABFT Design and Implementation on EC2" p. 9). Noting that ABFT claims empirical gains in the fields of throughput and latency when compared to HBFT, DUMBO, and inherently BEAT (Knudsen et al., 2021 “Results of RQ1” p. 6).

**Problem:**

The objective of this section is to highlight the problems outlined in the ABFT literature, providing a foundation for prospective research directions. These challenges serve as potential avenues for future research. By exploring and addressing these issues, the goal is to contribute to the advancement of the field of asynchronous consensus through a doctoral research concept paper based off the research of ABFT and the previous asynchronous consensus protocols that contributed to the work. The potential problem directions defined by ABFT are presented below:

*Threshold Elliptic Curve Digital Signature Algorithm Scheme Problem*:

In (Knudsen et al., 2021, "Limitation of ABFT," p. 7) it is stated that ABFT does not fully implement the threshold ECDSA scheme and relies on a trusted dealer for precomputing signing material. A problem to address is the need to develop a method of implementing the full threshold Elliptic Curve Digital Signature Algorithm (ECDSA) scheme within ABFT with out a trusted dealer. Further research is needed to quantify the performance benefits of threshold ECDSA signatures versus threshold Boneh-Lynn-Shacham (BLS) signatures. When it comes to ECDSA threshold signatures, the precomputed partial signature mechanism can add to the setup cost while both increasing run-time savings and also limiting scalability. In contrast, BLS signatures often have lower setup costs due to the simplified key management and consistent signature sizes. BLS signatures individually are generally not as fast or efficient as ECDSA signatures when signing a single transaction, but can be advantageous in scenarios where multiple signatures are needed to be combined, as in the case of consensus algorithms (Knudsen et al., 2021, "Threshold ECDSA Signatures" p. 2). The ABFT research directly states “concern is related to the amount of additional computational resources needed to maintain enough precomputed signing material for each round of ABFT and how the cost of this compares to the performance benefit of threshold ECDSA signatures over the previously used threshold BLS signatures” (Knudsen et al., 2021, "Limitation of ABFT," p. 7) . Signifying the need for research comparing a new fully implemented dealer-less threshold ECDSA schema, the original trusted dealer threshold ECDSA schema, and the previously used BLS signatures.

*Experimental Setup Problem*:

In (Knudsen et al., 2021, "Threats to Validity," p. 7) it is stated that the experimental setup of ABFT is based on previous related work, which might have limitations in terms of external validity. Research is needed to explore different experimental setups and network environments to validate ABFT and previous protocol's performances under various conditions and deployments. The ABFT research along with the DUMBO and BEAT research all support this need by redeploying HBFT via the original Python code or just comparing the results of the original HBFT research. A potential research direction could be to replicate HBFT in a different language such as Rust or deployment such as Azure. Given the HBFT testing fault tolerance is defined as F=N/4 (Miller et al., 2016 "Experiments on EC2" p. 9), an experimental network setup could also include a more fault tolerant network of F=N/3 instead of F=N/4. Measurements of latency, throughput, and message complexity could be measured and compared to the previous results of HBFT .

*Scalability Problem*:

In (Knudsen et al., 2021, "Results of RQ1," p. 6) it is highlighted that as the number of nodes in the network increases, the communication overhead grows, which can lead to network congestion with less throughput and increased latency. The scalability problem with ABFT makes it less practical for very large networks or networks with a high churn rate. There is a need to develop asynchronous consensus in such a way that ABFT can scale practically. More research can be done on the ECDSA threshold signatures vs the BLS signatures dilemma. By eliminating the need for ECDSA’s precomputed signatures and using BLS aggregated signatures, there could be a potential to increase scalability with performance trade-offs. The previous research of ABFT that uses ECDSA signatures deploys 100 nodes while also deploying the same amount of HBFT using BLS signatures (Knudsen et al., 2021, "Results of Q1" p. 6). Because the research only scaled to 100 nodes, more research can be given in the context of larger scale networks. Because ABFT uses ECDSA with the precomputed signing material, more pre-signing materials are needed to scale compared to using BLS signatures that can be used to aggregate multiple signatures (Knudsen et al., 2021, "Threshold ECDSA Signatures" p. 2). The research of scalability could be measured in throughput vs latency where the goal would be to see if BLS signatures outperform ECDSA in throughput as the network scales, while realistically giving up latency. Then after providing evidence off BLS signatures outperforming ECDSA on throughput scalability past (N) amount of nodes, further research could be proposed on how to decrease latency while still maintaining gains in throughput as the network scales.

*Handling Network Degradation Problem*:

In (Knudsen et al., 2021, "Results of RQ2," p. 7) it states that ABFT performs well when the number of affected nodes is less than the fault tolerance, but has a problem of degrading as the number of affected nodes becomes greater than the fault tolerance, there is need for improving the handling of network degradation growing larger than the fault tolerance. The ABFT research found that performance didn't degrade even when some network nodes were affected, as long as the number of affected nodes (M) is less than the fault tolerance (F). In the research scenario of 8 nodes and a fault tolerance of 2, having 2 nodes affected did not significantly impact performance. However, ABFT mentions that if M exceeds the fault tolerance, performance degrades and latency increases. ABFT still ensures termination and security without needing protocol changes, but expects performance degradation when certain message thresholds are not met. There is a need to explore adaptive strategies to mitigate the harsh performance degradation observed under such conditions. These exploratory strategies can include load balancing, selective switching, selective recovery, or adjusting the fault tolerance threshold based on network conditions. Previous research supports these similar techniques such as BEAT2 opportunistically moving the threshold encryption to the client side (Duan et al., 2018 “BEAT2” p. 7) and the how the optimistic mode of BEAT4 requests fingerprinted cross-checksum from a single or multiple servers depending on network congestion (Duan et al., 2018 “BEAT4” p. 8).

**Goal:**

The goal of the exploratory research, conducted as part of the Fall 2023 ISEC 885 course, is to establish a problem direction for a doctoral research idea concept paper, which will lead to a research idea paper. The goal is centered around the improvement of the field of Asynchronous Byzantine Fault Tolerance through the ABFT protocol. The final outcome will be a concrete problem, need, and goal. The technical goals of the research is to implement and deploy the HBFT protocol to create a baseline. This baseline will serve as a reference point for comparing both the original work of HBFT and any future research endeavors. After creating a baseline the next technical goal will be replicating and contributing to ABFT such that previous results can be compared to and new research proposed.

Review of the Literature

*HoneyBadgerBFT:*

As aforementioned in the introduction, HBFT claims to be the first practical asynchronous BFT protocol and is important to the field of asynchronous consensus because HBFT argues to solve the Fischer Lynch Patterson (FLP) theorem that states “No completely asynchronous consensus protocol can tolerate even a single unannounced process death”(Fischer et al., 1985).The high level design of the HBFT protocol can be summed up in the abstract of DUMBO stating “The core of the HBFT protocol is to achieve batching consensus using Asynchronous Common Subset (ACS) protocol of (Ben-Or, Kelmer, & Rabin, 1994) constituted with (n) Reliable Broad Cast protocol (RBC) to have each node propose its input, followed by (N) Asynchronous Binary Agreement protocol (ABA) to make a decision for each proposed value (N is the total number of nodes)” (Guo, Lu, Tang, Xu, & Zhang, 2020)**.** While the HBFT is regarded as an advancement in asynchronous consensus and practical example of solving the FLP theorem by handling byzantine faults asynchronously, it is not with out its own issues. As its successor BEAT alludes too, “the performance (latency, throughput) issues, compared to partially synchronous BFT protocols such as the Practical Byzantine Fault Tolerant protocol (PBFT), HoneyBadgerBFT has significantly higher latency and lower throughput, in part due to its use of expensive threshold cryptography (specifically, threshold encryption and threshold signatures)” (Duan et al., 2018 “Challenges and Opportunities in Adopting Asynchronous Permissioned Blockchains p.1). Summarizing BEAT and supporting that HBFT is a step in the right direction for asynchronous consensus, but the problem being HBFT is not yet competitive to the industry standard synchronous protocols like the PBFT protocol.

*BEAT:*

BEAT takes the works of HBFT and creates “five asynchronous BFT protocols that are designed to meet different goals (e.g., different performance metrics, different application scenarios)” (Duan et al., 2018 “Abstract” p. 2). The BEAT instantiations act as follows when compared to HBFT: BEAT0 uses a different threshold encryption, BEAT1 uses a different erasure-coded broadcast, BEAT2 changes the HBFT logic by opportunistically moving the encryption part of the threshold encryption to the client, BEAT3 changes the HBFT primitive and becomes a BFT storage system by replacing the RBC with Bandwidth-efficient Asynchronous Verifiable Information Dispersal (AVID-FP), and BEAT4 reduces read bandwidth making it more suitable for clients who read only a fraction of stored transactions (Duan et al., 2018 “The BEAT protocols” p. 2). BEAT was chosen for this research because it builds upon the logic of HBFT and creates different instances with different design goals.

*DUMBO:*

The DUMBO protocol was chosen for the research because its extends the research of BEAT and HBFT by creating two protocols DUMBO1 and DUMBO2 based off the HBFT codebase and the findings from BEAT. DUMBO1 runs a small K (independent of N) instances of ABA instances, while DUMBO2 reduces it further down to a constant. The premise behind DUMBO being that “(1) reducing the number of ABA instances significantly improves efficiency; and (2) using multi-valued validated Byzantine agreement (MVBA) which was considered sub-optimal for an Atomic Common Subset (ACS) in HBFT, in a more careful way could actually lead to a much more efficient ACS” (Duan et al., 2018 “Abstract” p. 1). DUMBO was chosen for this research because it extends the research of BEAT and HBFT and directly refutes that MVBA is sub-optimal for ACS as proposed in HBFT.

*ABFT:*

The ABFT protocol was chosen for the research because it is an amalgamation of the works of HBFT, BEAT, and DUMBO (Knudsen, Li, Notland, Haro, & Ræder, 2021 "ABFT Design and Implementation on EC2" p. 9). The basis of the ABFT logic when compared to HBFT is that ABFT “integrates threshold Elliptic Curve Digital Signature Algorithm (ECDSA) signatures and optimization of erasure coding parameters, as well as additional, implementation-level optimizations” (Knudsen et al., 2021, "Abstract," p. 1). ABFT deploys two scenarios in its research. Scenario one assess ABFT's performance in a stable network. Showing gains in throughput and latency when comparing it with HBFT,BEAT, and DUMBO (Knudsen, et al., 2021"Results of RQ1" p. 6). Scenario two explored the impact of asymmetric, high packet delay, and packet loss on ABFT's performance. ABFT demonstrated improved transaction latency and throughput in both LAN and WAN configurations when compared to the existing HBFT, BEAT, and DUMBO protocols (Knudsen, et al., 2021"Results of RQ2" p. 7). Highlighting ABFT's ability to handle network degradation within certain thresholds. Because ABFT extends the works of HBFT, BEAT, and DUMBO, it was chosen for the upper bounds of this research.

Approach

*Redeploy HBFT*

The initial goal of the research will be to redeploy an instance of HBFT and establish a baseline which compares previous and future work to. The deployment will mimic that of the HBFT deployment of 32, 40, 48, 56, 64, and 104 Amazon EC2 t2.medium instances uniformly distributed throughout its 8 regions spanning 5 continents. The batch sizes of these transactions will mimic that of the HBFT such that each node will propose 256, 512, 1024, 2048, 4096,8192, 16384, 32768, 65536, or 131072 transactions (Miller et al., 2016 "Experiments on Amazon EC2," p. 9). The size of each transaction will be a constant of 250 Bytes each (Miller et al., 2016 "Bandwidth Breakdown and Evaluation." p. 9). The results of the HBFT findings should mimic the original findings that state the upper bound limits of “throughput exceeding 20,000 transactions per second for medium size networks of up to 40 nodes. For a large 104 node network, we attain more than 1,500 transactions per second.” (Miller et al., 2016 "Experiments on EC2" p. 10) Note that the HBFT fault tolerance parameter is set to Faulty Node (F) = Nodes (N) / 4, such that 32N/8F 40N/10F 48N/12F 56N/14F 64N/16F 104N/26F was used. Also note that the formula F=N/4 was chosen instead of the greater fault tolerance of F=N/3 for ease of division. (Miller et al., 2016 "Experiments on EC2" p. 9) . It is important to establish a baseline deployment of HBFT that replicates the original findings before creating newer research to test against.

## **References**

Abraham, I., Malkhi, D., & Spiegelman, A. (2019). Asymptotically optimal validated asynchronous byzantine agreement. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (pp. 337–346). ACM. <https://doi.org/10.1145/3293611.3331612>

Ben-Or, M., Kelmer, B., & Rabin, T. (1994). "Asynchronous secure computations with optimal resilience." In Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing (pp. 183–192). ACM.

Duan, S., Reiter, M., & Zhang, H. (2018). BEAT: Asynchronous BFT Made Practical. *In ACM SIGSAC Conference on Computer and Communications Security,* pp. 2028–2041. <https://doi.org/10.1145/3243734.3243812>

Guo, B., Lu, Z., Tang, Q., Xu, J., & Zhang, Z. (2020). Dumbo: Faster Asynchronous BFT Protocols. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20), 803–818. <https://doi.org/10.1145/3372297.3417262>

Knudsen, H., Li, J., Notland, J., Haro, P., & Ræder, T. (2021). High-Performance Asynchronous Byzantine Fault Tolerance Consensus Protocol. *IEEE International Conference on Blockchain*, pp. 476-483. <https://doi.org/10.1109/Blockchain53845.2021.00073>

Miller, A., Xia, Y., Croman, K., Shi, E., & Song, D. (2016). The Honey Badger of BFT Protocols. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 31–42. <https://doi.org/10.1145/2976749.2978399>