Brief Announcement: Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP

Shir Cohen  
Technion  
shirco@campus.technion.ac.il

Idit Keidar  
Technion  
idish@ee.technion.ac.il

Alexander Spiegelman∗  
Novi  
sasha.spiegelman@gmail.com

1 INTRODUCTION

Byzantine Agreement (BA) [14] has been studied for four decades by now, but until recently, has been considered at a fairly small scale. In recent years, however, we begin to see practical use-cases of BA in large-scale systems, which motivates a push for reduced communication complexity. In deterministic algorithms, Dolev and Reischuk’s renown lower bound stipulates that $\Omega(n^2)$ communication is needed [9], and until fairly recently, almost all randomized solutions have had (expected) quadratic word complexity. Recent work has broken this barrier [10, 11, 16], but not in asynchronous settings. In our work, we present the first sub-quadratic asynchronous Byzantine Agreement algorithm. Our algorithm is randomized and solves binary BA with high probability (whp), i.e., with probability that tends to 1 as $n$ goes to infinity.

We consider a system with a static set of $n$ processes, in the so-called “permissioned” setting, where the ids of all processes are well-known. Our algorithm tolerates $f$ failures for $n = 4.5f$ (asymptotically). The adversary in our settings may adaptively corrupt up to $f = (\frac{1}{3} - \epsilon)n$ processes in the course of a run, where $\max\left\{\frac{3}{8n\ln(n)}, 0.109\right\} + \frac{1}{8n\ln(n)} < \epsilon < \frac{1}{3}$. In addition, we assume a trusted public key infrastructure (PKI) that allows us to use verifiable random functions (VRFs) [15].

We assume a strong adversary that can adaptively take over processes, whereupon it has full access to their private data. It further sees all communication in the system. But we do limit the adversary in two ways. First, we assume that it is computationally bounded so that we may use the PKI. Second, as proven in [1] for the synchronous model, achieving sub-quadratic complexity is impossible when the adversary can perform after-the-fact removal, meaning that it can delete messages that were sent by correct processes before corrupting these processes. Here, we adapt the no-after-the-fact removal assumption to the asynchronous model, and define a delayed-adaptive adversary based on causality [13].

We formalize the concept of VRF-based committee sampling as used in Algorand [7, 10], and adapt it to the asynchronous model. In a nutshell, the idea is to use a VRF seeded with each process’s private key in order to sample uniformly at random $O(\log(n))$ processes for a committee, and to have different committees execute different parts of the BA protocol. Each committee is used for sending exactly one protocol message and messages are sent only by committee members, thus reducing the communication cost. Whereas in Algorand’s synchronous model a process can be sure it receives messages from all correct committee members by a timeout, in the asynchronous model this is not the case. Rather, processes make progress by waiting for some threshold number of messages. Without committees, this threshold is normally $n - f$ (waiting for more than $n - f$ processes might violate termination).

But since committees are randomly sampled, we do not know the

*The work was done when the author was working at VMware Research.
committee’s exact size or the number of Byzantine processes in it. Thus, adapting committees to this model is somewhat subtle and requires ensuring certain conditions regarding the intersection of subsets of committees. In this work we identify sufficient conditions on sampling, which ensure safety and liveness with high probability.

Randomized BA algorithms can be seen as if processes toss a random coin at some point during the protocol. While some protocols toss a local coin [3, 4] and require exponential expected time to reach agreement, others use the abstraction of a shared coin [5, 6, 10, 12, 17], which involves communication among processes and results in the same coin toss with some well defined success rate. In this work we present an asynchronous shared coin algorithm that uses a VRF and provides a constant success rate with an equal probability for tossing 0 and 1. Unlike previous shared coin implementations, our solution does not require a priori knowledge of the set of participants, which makes it useful in committee-based constructions. We then adapt our coin to work with committees and use it to devise a sub-quadratic BA algorithm.

In this brief overview we overview the properties of our committee sampling and the shared coin protocol. Their details, as well as the consensus algorithm, appear in our full paper [8].

2 COMMITTEE SAMPLING

In order to reduce the number of messages and achieve sub-quadratic word complexity, it is essential to avoid all-to-all communication phases. Instead, a subset of processes is sampled to a committee and only processes elected to the committee send messages [10, 11]. To prevent the adversary from corrupting committee members, each protocol message is sent by a different committee, and committee members cannot predict the next committee sample. Therefore, they send their messages to all other processes. If the committee is sufficiently small, this technique results in sub-quadratic word complexity.

Validated committee sampling is a primitive that allows processes to elect committees without communication and later prove their election. It provides every process $p_i$ with a private function $\text{sample}_i(s, \lambda)$, which gets a string $s$ and a threshold $1 \leq \lambda \leq n$ and returns a tuple $(\sigma_i, \sigma_f)$, where $\sigma_r \in \{\text{true, false}\}$ and $\sigma_f$ is a proof that $\sigma_i = \text{sample}_i(s, \lambda)$. If $\sigma_i = \text{true}$ we say that $p_i$ is sampled to the committee for $s$ and $\lambda$. The primitive ensures that $p_i$ is sampled with probability $\frac{\lambda}{n}$. In addition, there is a public (known to all) function, $\text{committee-val}(s, \lambda, i, \sigma_f)$, which gets a string $s$, a threshold $\lambda$, a process id $i$, and a proof $\sigma_f$, and returns $\text{true}$ or $\text{false}$.

Consider a string $s$. For every $i$, $1 \leq i \leq n$, let $\langle \sigma_i, \sigma_f \rangle$ be the return value of $\text{sample}_i(s, \lambda)$. The following is satisfied for every $p_i$:

- $\text{committee-val}(s, \lambda, i, \sigma_f) = v_i$.
- If $p_i$ is correct, then it is infeasible for the adversary to compute $\text{sample}_i(s, \lambda)$.
- It is infeasible for the adversary to find $(\sigma, \sigma')$ s.t. $\sigma \neq \sigma_i$ and $\text{committee-val}(s, \lambda, i, \sigma) = \text{true}$.

We refer to the set of processes sampled to the committee for $s$ and $\lambda$ as $C(s, \lambda)$. Here we set $\lambda$ to $8\ln(n)$. Let $d$ be a parameter of the system such that $\max(\frac{\lambda}{3}, 0.0362) < d < \frac{5}{2} - \frac{1}{\lambda^2}$. Committee-based protocols can not wait for $n - f$ processes. Instead, they wait for $W \triangleq \left\lceil \frac{n}{3} + 3d\right\rceil \lambda$ processes. We show that whp at least $W$ processes will be correct in each committee sample and hence waiting for this number does not compromise liveness. In addition, instead of assuming $f$ Byzantine processes, our committee-based protocols assume that whp the number of Byzantine processes in each committee is at most $B \triangleq \left\lceil \frac{n}{3} - d\right\rceil \lambda$. The following claim is proven in the full version of this paper [8] using Chernoff bounds.

For a string $s$ and $\lambda = \text{const} \cdot \ln(n)$ the following hold with high probability:

- $(S1)$ $|C(s, \lambda)| \leq (1 + d)\lambda$.
- $(S2)$ $|C(s, \lambda)| \geq (1 - d)\lambda$.
- $(S3)$ At least $W$ processes in $C(s, \lambda)$ are correct.
- $(S4)$ At most $B$ processes in $C(s, \lambda)$ are Byzantine.

If a protocol uses a constant number of committees, then with high probability, $S1$-$S4$ hold for all of them. If, however, a protocol uses a polynomial number of committees then these are not ensured whp. The following corollaries are derived from S1-S4 and are used to ensure the safety and liveness of our protocols that use committees (a full proof is in the full paper). Intuitively, S3 allows the protocol to wait for $W$ messages without forgoing liveness. Property S5 below shows that if two processes wait for sets $P_1$ and $P_2$ of this size, then they hear from at least $B + 1$ common processes of which, by S4, at least one is correct.

Corollary 2.1 (S5). Consider $C(s, \lambda)$ for some string $s$ and some $\lambda = \text{const} \cdot \ln(n)$ and two sets $P_1, P_2 \subseteq C(s, \lambda)$ s.t. $|P_1| = |P_2| = W$. Then, $|P_1 \cap P_2| \geq B + 1$.

The following property is used to show that if $B + 1$ correct processes hold some value, and some correct process waits for messages from $W$ processes, then it hears from at least one correct process that holds this value.

Corollary 2.2 (S6). Consider $C(s, \lambda)$ for some string $s$ and some $\lambda = \text{const} \cdot \ln(n)$ and two sets $P_1, P_2 \subseteq C(s, \lambda)$ s.t. $|P_1| = B + 1$ and $|P_2| = W$. Then, $|P_1 \cap P_2| \geq 1$.

3 WHP COIN

We employ committee sampling and describe here an asynchronous protocol for a shared coin with a constant success rate against the delayed-adaptive adversary. Our coin works whp and has a word complexity of $\tilde{O}(n)$ in expectation. We now define the WHP coin abstraction:

Definition 3.1 (WHP Coin). A WHP coin with success rate $\rho$ is a shared object exposing $\text{whp}_\text{coin}(r), r \in \mathbb{N}$ at each process. If all correct processes invoke $\text{whp}_\text{coin}(r)$ then, whp $(1)$ all correct processes return, and $(2)$ all of them output the same value $b$ with probability at least $\rho$, for any value $b \in \{0, 1\}$.

The $\text{whp}_\text{coin}$ protocol is presented in Algorithm 1. It samples two committees, one for each communication step. In each step, only the processes that are sampled to the committee send messages. However, since the committee samples are unpredictable, messages are sent to all processes. Processes wait for $W$ messages in each step. Since a constant number of committees is sampled in the protocol, S1-S6 hold for all of them. In particular, by S3, all processes receive $W$ messages, ensuring liveness.
In the first step, each committee member first samples the VRF with its private key and the protocol’s argument in order to generate a random initial value. For brevity, we denote by \( VRF_i \) the VRF with \( p_i \)'s private key. Using a VRF to generate a random initial value effectively weakens the adversary as Byzantine processes can neither choose their initial values nor equivocate. If a Byzantine process would try to act maliciously, the VRF proof would easily expose it and its message would be ignored.

In each phase of the protocol, processes that are sampled to the relevant committee send one value to every other process. The receiver validates the received values using the VRF proofs, which are sent along with the values. We omit the proof validation from the code for clarity. After two phases of communication, each process chooses the minimum value it received in the second phase and outputs its least significant bit.

We follow the concept of a common core, as presented by Attiya and Welch for the crash failure model [2], adapt it to committees' settings and argue that if a core of \( B + 1 \) correct processes hold the global minimum value at the end of phase 1, then by the end of the following phase all processes receive this value. We exploit the \( d \) parameter in our committee sampling definition to bound the number of values held by \( B + 1 \) correct processes. We show that this number is linear in \( n \) and hence with a constant positive probability, by the end of the second phase, all correct processes receive the global minimum among the VRF outputs and therefore produce the same output.

Algorithm 1 Protocol whp_coin(\( r \)): code for process \( p_i \)

1. Initially first-set, second-set = ∅, \( v_i = \infty \)
2. if \( \text{sample}_\ell(\text{FIRST}, \lambda) = \text{true} \)
3. \( v_i \leftarrow VRF_\ell(\lambda) \)
4. send \( \langle \text{FIRST}, v_i \rangle \) to all processes
5. upon receiving \( \langle \text{FIRST}, v_j \rangle \) with valid \( v_j \)
6. from validly sampled \( p_j \) do
7. if \( \text{sample}_\ell(\text{SECOND}, \lambda) \)
8. if \( v_j < v_i \) then \( v_i \leftarrow v_j \)
9. first-set \( \leftarrow \) first-set \( \cup \{ j \} \)
10. when \( \text{first-set} = W \) for the first time
11. send \( \langle \text{SECOND}, v_i \rangle \) to all processes
12. upon receiving \( \langle \text{SECOND}, v_j \rangle \) with valid \( v_j \)
13. from validly sampled \( p_j \) do
14. if \( v_j < v_i \) then \( v_i \leftarrow v_j \)
15. second-set \( \leftarrow \) second-set \( \cup \{ j \} \)
16. when \( \text{second-set} = W \) for the first time
17. return LSB(\( v_i \))

In the full paper we prove the following theorem:

**Theorem 3.2.** Algorithm 1 implements a WHP coin with a constant success rate.

**Complexity.** In each whp_coin instance using committees all correct processes that are sampled to the two committees (lines 2,6) send messages to all other processes. Each of these messages contains a VRF output (including a value and a proof), a VRF proof of the sender’s election to the committee and a constant number of bits that identify the type of message that is sent. Therefore, each message’s size is a constant number of words and the total word complexity of a WHP coin instance is \( O(nC) \) where \( C \) is the number of processes that are sampled to the committees. Since each process is sampled to a committee with probability \( \frac{1}{2} \), we get a word complexity of \( O(n\lambda) = O(n\log(n)) = \tilde{O}(n) \) in expectation.

**4 CONCLUSIONS**

In summary, our paper [8] presents the first formalization of randomly sampled committees using cryptography in asynchronous settings. Based on this technique, it presents the first sub-quadratic asynchronous shared coin and BA wph algorithms. Our algorithms have expected \( \tilde{O}(n) \) word complexity and \( O(1) \) expected time.

**REFERENCES**

[1] Ittai Abraham, TH Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. 2019. Communication complexity of byzantine agreement, revisited. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. 317–326.

[2] Hagit Attiya and Jennifer Welch. 2004. Distributed computing: fundamentals, simulations, and advanced topics. Vol. 19. John Wiley & Sons.

[3] Michael Ben-Or. 1983. Another advantage of free choice (extended abstract): Completely asynchronous agreement protocols. In Proceedings of the second annual ACM symposium on Principles of distributed computing. ACM, 27–30.

[4] Gabriel Bracha and Sam Toueg. 1983. Resilient consensus protocols. In Proceedings of the second annual ACM symposium on Principles of distributed computing. ACM, 12–26.

[5] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random oracles in Constant-time: Practical asynchronous Byzantine agreement using cryptography. Journal of Cryptology 18, 3 (2005), 219–246.

[6] Ran Canetti and Tal Rabin. 1993. Fast asynchronous Byzantine agreement with optimal resilience. In STOC. Vol. 93. Citeseer, 42–51.

[7] Jing Chen, Sergey Gorbanov, Silvio Micali, and Georgios Vlachos. 2018. ALGOREAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. IACR Cryptology ePrint Archive 2018 (2018), 377.

[8] Shiri Cohen, Idit Keidar, and Alexander Spiegelman. 2020. Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP. arXiv preprint arXiv:2002.06545 (2020).

[9] Danny Dolev and Rüdiger Reischuk. 1985. Bounds on Information Exchange for Byzantine Agreement. J. ACM 32, 1 (Jan. 1985), 191–204.

[10] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling byzantine agreements for cryptocurrencies. In Proceedings of the 26th Symposium on Operating Systems Principles. 51–68.

[11] Valerie King and Jared Saia. 2011. Breaking the O(\( n^2 \)) bit barrier: scalable byzantine agreement with an adaptive adversary. Journal of the ACM (JACM) 58, 4 (2011), 1–24.

[12] Valerie King and Jared Saia. 2013. Byzantine agreement in polynomial expected time. In Proceedings of the forty-fifth annual ACM symposium on Theory of computing. ACM, 401–410.

[13] Leslie Lamport. 1978. Time, Clocks, and the Ordering of Events in a Distributed System. Commun. ACM 21, 7 (July 1978), 558–565.

[14] Leslie Lamport, Robert Shostak, and Marshall Pease. 1982. The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst. 4, 3 (July 1982), 382–401.

[15] Silvio Micali, Michael Rabin, and Sahil Vadhan. 1999. Verifiable random functions. In Foundations of Computer Science, 1999. 40th Annual Symposium on. IEEE, 120–130.

[16] Satoshi Nakamoto. 2019. Bitcoin: A peer-to-peer electronic cash system. Technical Report. Man大户.

[17] Michael O Rabin. 1983. Randomized byzantine generals. In 24th Annual Symposium on Foundations of Computer Science (sfcs 1983). IEEE, 403–409.