Brief Announcement: Malicious Security Comes for Free in Consensus with Leaders

Mark Abspoel
CWI Amsterdam
the Netherlands

Thomas Attema
CWI Amsterdam
Leiden University, Mathematical Institute
TNO, Cyber Security and Robustness
the Netherlands

Matthieu Rambaud
Telecom Paris
Institut polytechnique de Paris
France

ABSTRACT

We consider consensus protocols in the model that is most commonly considered for use in state machine replication, as initiated by Dwork-Lynch-Stockmeyer, then by Castro-Liskov in 1999 with "PBFT". Such protocols guarantee, assuming n players out of which $t < n/3$ are maliciously corrupted, that the honest players output the same valid value within a finite number of messages, after the (unknown) point in time where both: the network becomes synchronous, and a designated player (the leader) is honest. The state of the art (Hotstuff, PODC'19), achieves linear communication complexity, but at the cost of additional latency, due to one more round-trip with the leader. Furthermore, it relies on constant-size threshold signatures schemes (TSS), for which all prior-known constructions require a costly interactive (or trusted) setup.

We remove all of these limitations. The communication bottleneck of PBFT lies in the subprotocol, denoted as "view change", in which the leader forwards $2t + 1$ signed messages to each player. Then, each player checks that these $2t + 1$ messages satisfy some predicate, which we denote "non-supermajority". We replace this with a responsive subprotocol, with linear communication complexity, that enables players to check this predicate. Its construction is elementary, since it requires only black box use of any TSS.

In the full version of our paper [5] we achieve three things. Firstly, we further optimize this subprotocol from succinct arguments of many signed messages, which we instantiate from Attema-Cramer-Rambaud [8, 2021-3-9 version]. As an introduction to these methods, we discuss here the simplest case, which is the construction in [8] of the first logarithmic-sized TSS with transparent setup. Second, we also address another complexity challenge pointed in Hotstuff, namely, that protocols with fast termination in favorable runs, have so far quadratic complexity, due to an even more complex view change. Third, we enable halting in finite time with (amortized) linear complexity, which was an unsolved question so far when external validity is required.

CCS CONCEPTS

• Theory of computation → Cryptographic protocols; **• Computing methodologies → Distributed algorithms**.

KEYWORDS

BFT; consensus; state machine replication; succinct proofs

ACM Reference Format:
Mark Abspoel, Thomas Attema, and Matthieu Rambaud. 2021. Brief Announcement: Malicious Security Comes for Free in Consensus with Leaders. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC '21), July 26–30, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3465084.3467953

We account for part of the first version of [31] (2020/11/29) and of [8]. This work was supported to EU H2020 projects No 780701 (PROMETHEUS) and No 74079 (ALGSTRONGCRYPTO).

1 THE SO-FAR TRADE-OFFS IN COMPLEXITY

The problem of consensus is the oldest and most studied task in distributed systems [10, 11, 21–23, 28]. We consider deterministic algorithms for consensus, where $n = 3t + 1$ players want to agree on some value in the presence of a malicious adversary that corrupts up to t of them. Each player inputs a value, and the goal is for the honest (i.e., non-corrupted) players to output the same value (consistency), while also ensuring that the output value satisfies some publicly checkable validity predicate. This constraint, known as "external validity" ([18, Def 4.1]), naturally shows up in the related problem of fault-tolerant state machine replication, which can be seen as an ordered sequence of consensus instances [20, 27, 33]. Typical predicates are a proof of work, or the signature of a request by some authorized client.

Players are connected through an asynchronous network controlled by the adversary, that may inspect all messages, and arbitrarily alter, reroute, drop, delay, or replay them. We assume a plain bulletin board where players can publish public keys (PKI). This enables digital signatures, and thus in turn guarantees the integrity and authenticity of messages. In this model, asynchronous consensus is not solvable ([22]). Various ways around this exist [1, 13, 19] that assume a negligible probability of failure, but these protocols require a shared key or a common coin, whose implementation requires a cubic communication complexity ([4, 25]). This is why, instead, we consider the classical *conditional* termination condition known as partial synchrony ([21]), that is used most of the deterministic state machine replication algorithms [9, 20, 24, 33]. These protocols proceed by consecutive timeframes which we denote as "phases". Each phase corresponds to a publicly known player known as the "leader", that may or not stay the same throughout the protocol. Termination in these protocols guarantees that, if from some point in time the leader is honest and the network fast enough, then all honest players will output in this phase. This is why both the bit communication complexity and latency to output, are traditionally measured from this particular moment in time. Indeed, nothing is guaranteed as long as both of these conditions are not satisfied.

The apparent latency cost of responsiveness with linear communication complexity in every phase. In this model, [33, PODC'19] observe that the communication complexity was so far at least *quadratic* in the number n of players since [20]. They write: "This scaling challenge plagues not only PBFT, but many other protocols developed since then, e.g., Prime, Zyzzyva [26], Upright, BFT-SMaRt, 700BFT [9], and SBFT [24]." They observe that this complexity comes from the subprotocol denoted "view change" in PBFT, which "requires the new leader to relay [to all n players] information from $(n - t)$ [players]". The main contribution of [33] is to circumvent this requirement, to achieve communication complexity linear in n. But they pay the price to sacrifice the latency. Namely, as they write: "HotStuff achieves these properties by adding another [round trip] to each [phase], a small price to latency in return for considerably simplifying the leader replacement protocol." Their argument tends to show that this price is *unavoidable*, namely, they write about previous protocols with better latency: "Unfortunately, [phase]-change based on the [two round trips] paradigm is far from simple, is bug-prone [3], and incurs a signicant communication penalty for even moderate system sizes." To further show that this price may be unavoidable, they insist that ways around preserving linear complexity and two-round trips latency do exist: Tendermint [16], Casper [17] (and also the first version Hotstuffv1 [2]), but that all of them sacrifice, on the other hand the important property denoted "responsiveness", which guarantees output at the actual network's speed. Indeed, in [2], the leader is instructed to *wait* at the beginning of a phase for some fixed delay Δ, which is the upper bound on the network delay after some (unknown) in unknown some point in time, because their termination requires that the leader collects messages from *all* the honest players. This inefficiency is pointed in [33] as follows: "However, these systems are built around a synchronous core, where in proposals are made in pre-determined intervals that must accommodate the worst-case time it takes to propagate messages". Then, later page 9 of [33], they emphasize that termination in the previous systems would not be guaranteed without waiting for Δ (under the paragraph "Livelessness with two-phases").

2 A SIMPLE SOLUTION, FROM ANY TSS

2.1 Further Model, Definitions and Notations

Baseline protocol. We provide, in 2.2, a rephrasing of the protocol denoted as "linear PBFT" in [24]. Borrowing the classical terminology of [12], the protocol proceeds in intervals denoted "phases", which are controlled by a global clock. We assume given by the model, in each phase $\phi \geq 1$, the publicly known identity of a player, denoted L_ϕ: the "leader". The first phase $\phi = 1$ differs from the higher ones $\phi \geq 2$, in that it does not contain the first instruction **0 Report**. Players in a phase ϕ perform these instructions *as soon as they can*, and the numbered steps *do not* denote any waiting instruction. The three message types that need to be made explicit in the description of the protocol below, are denoted as: propose, lock vote, decision vote.

A k-threshold signature scheme (TSS). is defined in [8, §5] as follows. Considering a baseline digital signature scheme, a k-threshold signature provides, for any integer k, an algorithm $\textsc{aggregate}_k$ which, on input a set $\mathcal{S}$ of k messages of *identical content m*, signed by any *distinct* k out of n players, outputs a *proof of knowledge* of such a set $\mathcal{S}$ (including the predicate that the k signers are distinct). This new definition of a TSS implies the classical one, e.g., of [32].

The TSS of [8, §5] proves knowledge of k out of n signatures of the type BLS [15]. Their sizes are in $O(\log(n))$, and it has the additional properties that k can actually be dynamically chosen by the prover, and that the identities of the signers are hidden. They require no interactive (nor trusted) setup.

By contrast, let us recall the following approach used by the decentralized transaction system Diem [29] and by [30]. Every player generates its own public-private key-pair. A threshold signature is computed as the sum of k individual BLS signatures, and it can be verified by running the BLS verification algorithm using the sum of the public keys of the k signers. Hence, the threshold signature should contain a list of the k signers, i.e., it is of size $O(n)$ or $O(k\log(n))$ depending on the exact encoding of this list.

2.2 Baseline protocol: instructions in a phase ϕ

For clarity, we denote as lock certificate the data type output by $\textsc{aggregate}_{2t+1}$ on messages of type lock vote, and decision certificate the data type output by $\textsc{aggregate}_{2t+1}$ on messages of type decision vote.

0 Report Every player P_i sets $\phi_i \leq \phi$ the highest phase up to ϕ for which he received a lock certificate. By convention $\phi_i = 0$ if he did not so far. He sends to the leader L_ϕ: a report(ϕ_i, ϕ), appended with a lock certificate(v_i, ϕ_i) if $\phi_i \geq 1$.

1 Propose The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t+1$ distinct players, then, letting $\phi_{\max}$ be the highest ϕ_j in $\mathcal{R}$:

(i) If $0 < \phi_{\max}$, he received at least one $\mathsf{lc}_{\max}$:= lock certificate$(v_{\max}, \phi_{\max})$. Then he multicasts propose$(v_{\max}, \phi)$, appended with the justification $\{\mathcal{R},\ \mathsf{lc}_{\max}\}$.

(ii) Otherwise, he sets v_L equal to his own input and multicasts propose(v_L, ϕ) appended with the justification $\mathcal{R}$.

2 Lock vote Players, upon receiving propose(v, ϕ) from L_ϕ for the first time (whatever v is), appended with a set $\mathcal{R}$ of report messages relative to ϕ, check if:

(i) *Either* the justification comes with some lock certificate(v, ϕ') for v, such that ϕ' is the highest ϕ_j reported in $\mathcal{R}$;

(ii) *Or* the justification contains no lock certificate, and *all* messages in $\mathcal{R}$ report $\phi_j = 0$;

then if the check passes, they reply with a signed lock vote(v, ϕ).

3 Lock certificate Leader L_ϕ, upon receiving lock votes for the same (v, ϕ) from $2t + 1$ distinct players, $\textsc{aggregate}_{2t+1}$ them into a "lock certificate(v, ϕ)", which he multicasts.

4 Decision vote Players, upon receiving from L_ϕ a lock certificate(v, ϕ) for the first time (for whatever value of v), reply with a signed decision vote(v, ϕ).

5 Decision certificate Leader L_ϕ, upon receiving decision vote(v, ϕ) from $2t+1$ distinct players for the same v, AGGREGATES$_{2t+1}$ them into a decision certificate(v), which he multicasts.

6 Output Upon receiving a decision certificate for v, output v.

2.3 Linear Complexity, from a PnS subprotocol

The forwarding of the $2t+1$ messages of $\mathcal{R}$ to all players, done in **1**, has quadratic bit complexity. The key to remove this, is the observation that the steps **0,1,2** contain a subprotocol whose purpose is to *prove* players that the value proposed in **1** is such that: it comes with a lock certificate formed in some phase ϕ', and that at most t honest players could possibly have received lock certificates formed in phases strictly higher than ϕ'. We specify this as follows:

*Definition 1 (*PnS*).* For every honest player i in phase ϕ , denote $\phi_i \le \phi$ the highest phase number, up to ϕ, for which i saw a valid lock certificate lc_i. Then a *Proof of non Supermajority*: PnS(ϕ_{max}, ϕ) for some valid ($\phi_{max}, \mathsf{lc}_{max}$) is the data of: $\phi_{max} \le \phi$, along with some data proving that *no* $t+1$ honest players have their $\phi_i > \phi_{max}$.

Relatively to such a specific set of inputs (ϕ_i, lc_i): a PnS protocol is one in which honest players in phase ϕ send one message to a designated prover L_ϕ among them, such that, upon receiving $2t+1$ well-formed messages, a (honest) prover is able to output: a phase number $\phi_{max} (\le \phi)$, a lock certificate lc_{max} relative to ϕ_{max}, and a PnS(ϕ_{max}, ϕ).

The following is easily seen to be a PnS protocol in one round trip, as proven in [31, §6], with total bit complexity $O(n\phi|TSS|)$, where $|TSS|$ denotes the size of a $|TSS|$.

1 Every player P_i sends to L_ϕ a report (ϕ_i, ϕ), appended with a lock certificate in ϕ_i if $1 \le \phi_i$; along with, for *each* $\phi' \in [\phi_i, \ldots, \phi]$, one signed testimony consisting of the string: "my locked phase number up to ϕ is lower than or equal to ϕ' ".

2 Prover L_ϕ, *upon receiving* from $2t+1$ players such well formed messages, i.e., containing a lock certificate for the claimed ϕ_i, a list of testimonies which is *consistent* with the claimed ϕ_i, and such that all the messages specify "up to ϕ" with ϕ the current phase number. Then, define ϕ_{max} the *lowest* value for which there exists $2t+1$ identical testimonies: "my locked phase number up to ϕ is lower or equal to ϕ_{max} ". Leader L_ϕ then:
- extracts this ϕ_{max} *and a* lock certificate lc_{max} relative to ϕ_{max} from one of the $2t+1$ messages received. *Claim 1*: he is always able to do so;
- AGGREGATES$_{2t+1}$ the testimonies. *Claim 2*: this constitutes a PnS on (ϕ_{max}, ϕ).

In conclusion, plugging the above PnS protocol in the steps **0,1** of the baseline consensus of §2.2 makes useless the forwarding of $\mathcal{R}$, and thus yields:

Theorem 1. *There exists a consensus with communication $O(n\phi|TSS|)$ per phase, such that players output within 6 actual message delays, as soon as both the leader is honest and the network synchronous.*

3 A LOGARITHMIC SETUP-FREE TSS

Let us give the intuition of the setup-free and logarithmic-sized TSS of the other work [8, §5]. This TSS is a short argument of knowledge of k out of n signed messages, in the simplest case (all messages are the same), and thus provides a warmup for our long version [5].

The TSS is instantiated with the BLS signature scheme [15] defined over a bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$. Let us now briefly recall the BLS signature scheme, instantiated in our n-player setting. All players i, $1 \le i \le n$, generate their own private key $u_i \in \mathbb{Z}_q$, and publish the associated public key $P_i = u_i H \in \mathbb{G}_2$. To sign a message $m \in \{0,1\}^*$, player i computes signature $\sigma_i = u_i \mathcal{H}(m) \in \mathbb{G}_1$, where $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}_1$ is some public hash function. The public verification algorithm accepts a signature σ_i if

$$(1) \qquad e(\sigma_i, H) = e(\mathcal{H}(m), P_i).$$

By the bilinearity of e, all honestly generated signatures are accepted. The unforgeability follows from the co-CDH assumption [14]. The AGGREGATE$_k$ algorithm takes as input a message m, along with the signatures from a subset $\mathcal{S}$ of k out of n signers on m, and simply outputs a proof of knowledge of this input, namely, of the following relation:

$$\left\{(P_1, \ldots, P_n, m; \mathcal{S}, (\sigma_i)_{i \in \mathcal{S}}) : |\mathcal{S}| = k,\ e(\sigma_i, H) = e(\mathcal{H}(m), P_i)\, \forall i \in \mathcal{S}\right\}.$$

The main challenge is that the prover only knows k-out-of-n signatures. This k-out-of-n case is reduced to the n-out-of-n, using a recent result on *k-out-of-n proofs of partial knowledge* [7] as follows. Let $p(X) = 1 + \sum_{j=1}^{n-k} a_j X^j \in \mathbb{Z}_q[X]$ be the unique polynomial of degree at most $n-k$ with $p(i) = 0$ for all $i \in \{1, \ldots, n\} \backslash \mathcal{S}$. Note that this polynomial defines an $(n-k, n)$ secret sharing of 1, with shares $s_i = 0$ for all $i \notin \mathcal{S}$. The k-aggregator defines $\widetilde{\sigma}_i = p(i)\sigma_i$, where $\widetilde{\sigma}_i$ is understood to be equal to 0 for $i \notin \mathcal{S}$, i.e., the secret sharing defined by $p(X)$ *eliminates* the signatures $(\sigma_i)_{i \notin \mathcal{S}}$ that the k-aggregator does not know. Subsequently, the k-aggregator commits to

$$\mathbf{x} = (a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_n) \in \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n.$$

Now note that the committed vector $\mathbf{x}$ satisfies

$$(2) \quad f_i(\mathbf{x}) = f_i(a_1, \ldots, a_{n-k}, \widetilde{\sigma}_1, \ldots \widetilde{\sigma}_n) = e(\mathcal{H}(m), P_i)\ \forall i \in [n]\ ,$$

$$\text{where} f_i : \mathbb{Z}_q^{n-k} \times \mathbb{G}_1^n \to \mathbb{G}_T, \quad \mathbf{x} \to e(\widetilde{\sigma}_i, H) - \sum_{j=1}^{n-k} a_j i^j e(\mathcal{H}(m), P_i).$$

Hence, by proving that the committed vector satisfies these relations, it follows that the k-aggregator knows a non-zero polynomial $p(X)$ of degree at most $n-k$ and group elements $\widetilde{\sigma}_1, \ldots \widetilde{\sigma}_n \in \mathbb{G}_1$ such that $e(\widetilde{\sigma}_i, H) = p(i) e(\mathcal{H}(m), P_i)$ for all $1 \le i \le n$. Therefore, the k-aggregator must know valid signatures for all indices i with $p(i) \neq 0$, and since $p(X)$ is non-zero and of degree at most $n-k$ at least, k of its evaluations are non-zero.

It remains to prove the n relations of (2) with low communication. Since they are n linear relations on the *same* committed vector $\mathbf{x}$, a random linear combination technique ([6]) enables to prove them for the price of only *one* relation. The relation is then proven using the compressed Σ-protocol Π_c of [8, §4].

REFERENCES

[1] Ittai Abraham, T.-H. 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, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.* ACM, 317–326.

[2] Ittai Abraham, Guy Gueta, and Dahlia Malkhi. 2018. Hot-Stuff the Linear, Optimal-Resilience, One-Message BFT Devil. *CoRR* abs/1803.05069 (2018). https://eprint.iacr.org/2020/406.

[3] Ittai Abraham, Guy Gueta, Dahlia Malkhi, Lorenzo Alvisi, Ramakrishna Kotla, and Jean-Philippe Martin. 2017. Revisiting Fast Practical Byzantine Fault Tolerance. *CoRR* abs/1712.01367 (2017).

[4] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. 2021. Reaching Consensus for Asynchronous Distributed Key Generation. In *Podc'21*.

[5] Mark Abspoel, Thomas Attema, and Matthieu Rambaud. 2020. Malicious Security Comes for Free in Consensus with Leaders. Cryptology ePrint Archive, Report 2020/1480. version 2021-04-26 https://eprint.iacr.org/2020/1480.

[6] Thomas Attema and Ronald Cramer. 2020. Compressed Sigma-Protocol Theory and Practical Application to Plug & Play Secure Algorithmics. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 12172)*. Springer, 513–543.

[7] Thomas Attema, Ronald Cramer, and Serge Fehr. 2021. Compressing Proofs of k-Out-Of-n Partial Knowledge. *Crypto'21* (2021).

[8] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. 2020. Compressed Sigma-Protocols for Bilinear Circuits and Applications. Cryptology ePrint Archive, Report 2020/1447. version 2021/3/10 https://eprint.iacr.org/eprint-bin/getfile.pl?entry=2020/1447&version=20210310:160359&file=1447.pdf.

[9] Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knezevic, Vivien Quéma, and Marko Vukolic. 2015. The Next 700 BFT Protocols. *ACM Trans. Comput. Syst.* 32, 4 (2015), 12:1–12:45.

[10] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. 2017. Bitcoin as a Transaction Ledger: A Composable Treatment. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I.* Springer, 324–356.

[11] Michael Ben-Or. 1983. Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols (Extended Abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 17-19, 1983.* ACM, 27–30.

[12] Piotr Berman and Juan A. Garay. 1989. Asymptotically optimal distributed consensus. In *Automata, Languages and Programming*.

[13] Erica Blum, Jonathan Katz, Chen-Da Liu Zhang, and Julian Loss. 2020. Asynchronous Byzantine Agreement with Subquadratic Communication. *IACR Cryptol. ePrint Arch.* 2020 (2020), 851.

[14] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *ASIACRYPT (Lecture Notes in Computer Science, Vol. 2248)*. Springer, 514–532.

[15] Dan Boneh, Ben Lynn, and Hovav Shacham. 2004. Short Signatures from the Weil Pairing. *J. Cryptology* 17 (2004), 297–319.

[16] Ethan Buchman. 2016. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains.* Ph.D. Dissertation. University of Guelph.

[17] Vitalik Buterin and Virgil Griffith. 2017. Casper the Friendly Finality Gadget. *CoRR* abs/1710.09437 (2017).

[18] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and Efficient Asynchronous Broadcast Protocols. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2139)*. Springer, 524–541.

[19] Ran Canetti and Tal Rabin. 1993. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*.

[20] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation* (New Orleans, Louisiana, USA) *(OSDI '99)*. USENIX Association, Berkeley, CA, USA.

[21] Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. 1988. Consensus in the presence of partial synchrony. *J. ACM* (1988).

[22] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. 1985. Impossibility of Distributed Consensus with One Faulty Process. *J. ACM* 32, 2 (1985), 374–382.

[23] Peter Gazi, Aggelos Kiayias, and Alexander Russell. 2020. Tight Consistency Bounds for Bitcoin. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*.

[24] Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. 2018. SBFT: a Scalable Decentralized Trust Infrastructure for Blockchains. *CoRR* abs/1804.01626 (2018). arXiv:1804.01626 http://arxiv.org/abs/1804.01626

[25] Eleftherios Kokoris-Kogias, Alexander Spiegelman, and Dahlia Malkhi. 2020. Asynchronous Distributed Key Generation for Computationally-Secure Randomness, Consensus, and Threshold Signatures. In *ACM CCS 2020*.

[26] Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. 2009. Zyzzyva: Speculative Byzantine fault tolerance. *ACM Trans. Comput. Syst.* 27, 4 (2009), 7:1–7:39.

[27] Leslie Lamport. 1998. The Part-time Parliament. *ACM Trans. Comput. Syst.* 16, 2 (May 1998), 133–169.

[28] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. 1982. The Byzantine Generals Problem. *ACM Trans. Program. Lang. Syst.* (1982).

[29] Libra Team. 2019. *State Machine Replication in the LibraBlockchain.* Version 2019-10-24.

[30] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. 2020. Improved Extension Protocols for Byzantine Broadcast and Agreement. In *DISC (LIPIcs, Vol. 179)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:17.

[31] Matthieu Rambaud. 2020. Malicious Security Comes for Free in Consensus with Leaders. Cryptology ePrint Archive, Report 2020/1480. version 2020-11-29 https://eprint.iacr.org/eprint-bin/getfile.pl?entry=2020/1480&version=20201129:224740&file=1480.pdf.

[32] Victor Shoup. 2000. Practical Threshold Signatures. In *EUROCRYPT (Lecture Notes in Computer Science, Vol. 1807)*. Springer, 207–220.

[33] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. 2019. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.* ACM, 347–356.