Faster Hash-based Multi-valued Validated Asynchronous Byzantine Agreement

Hanwen Feng*, Zhenliang Lu†§, Tiancheng Mai*, Qiang Tang*
*University of Sydney, Australia
†Nanyang Technological University, Singapore
{hanwen.feng, tiancheng.mai, qiang.tang}@sydney.edu.au, zhenliang.lu@ntu.edu.sg

***Abstract*—Multi-valued Validated Byzantine Agreement (MVBA) is vital for asynchronous distributed protocols like asynchronous BFT consensus and distributed key generation, making performance improvements a long-standing goal. Existing communication-optimal MVBA protocols rely on computationally intensive public-key cryptographic tools, such as non-interactive threshold signatures, which are also vulnerable to quantum attacks. While hash-based MVBA protocols have been proposed to address these challenges, their higher communication overhead has raised concerns about practical performance. We present a novel MVBA protocol with adaptive security, relying exclusively on hash functions to achieve post-quantum security. Our protocol delivers near-optimal communication, constant round complexity, and significantly reduced latency compared to existing schemes, though it has sub-optimal resilience, tolerating up to 20% Byzantine corruptions instead of the typical 33%. For example, with $n = 201$ and input size 1.75 MB, it reduces latency by 81% over previous hash-based approaches.**

I. Introduction

Byzantine fault-tolerant (BFT) consensus is one of the most crucial topics in distributed computing, providing a means for individuals to establish a consistent view in a distributed environment, and facilitating the execution of higher-level functionalities. With the surge of distributed applications over the global Internet in the last few decades, asynchronous BFT protocols are gaining re-surged attention and substantial progress in recent years, due to their resilience to network churns and ease of implementation.

Multi-Valued Validated Byzantine Agreement (MVBA), introduced in the seminal work of Cachin et al. [1], stands out as one of the most critical tools for asynchronous distributed protocols, particularly the recent *practical* ones. In an MVBA protocol, each node provides a multi-bit value as input, collectively deciding on one of the input values to output, which has to satisfy a pre-defined condition. MVBA remained as theoretical study, [1]–[3], until Dumbo [4] re-established its critical importance to construct practical asynchronous BFT protocols. Since then, it has started to play a pivotal role in many *practical* distributed protocols, including asynchronous distributed key generation [5]–[7], dynamic-committee proactive secret sharing [8], [9], optimistic asynchronous consensus protocols [10], [11], and network agnostic distributed protocols [12]. Moreover, MVBA plays a crucial role in achieving an efficient Asynchronous Common Subset (ACS), vital for Asynchronous Multi-party Computation to agree on sets of inputs [13]–[21]. Notably, as observed in [4], [22], [23], MVBA usually constitutes the bottleneck of its applications, and high communication complexity is the main reason when the system scales up to a moderate size, thus reducing the communication of MVBA is greatly desired.

The original MVBA of Cachin et al. [1] is with $\mathcal{O}(\ell n^2 + \lambda n^2 + n^3)$ bits of communication. Here, ℓ is the bit length of input, n is the number of participants, and λ is the security parameter that captures the signature size etc. Its large communication complexity becomes the major bottleneck of its practical use. After 20 years, the communication complexity was reduced by Abraham et al. [2] to be $\mathcal{O}(\ell n^2 + \lambda n^2)$. However, when the input size is moderate, such as $\mathcal{O}(n)$ (e.g., a vector of input bits), the ℓn^2 term becomes n^3 and dominates again. For this reason, Lu et al. [3] gave an extension framework that finally led to an optimal MVBA in Dumbo-MVBA⋆ [3], with $\mathcal{O}(\ell n + \lambda n^2)$ bits of communication, that matches the lower bound [2], [24]. Subsequently, Guo et al. [22] gave further concrete optimizations on rounds, and constructed Speeding MVBA, which eventually led to Dumbo-NG [25], a fast asynchronous BFT with very high throughput. While those recent communication efficient MVBA protocols made use of some "heavy-weight" cryptography, particularly non-interactive threshold signatures like BLS [26]–[28]. Relying on those tools raises *both* security and performance (particularly computation) concerns. Specifically, the underlying algebraic assumptions are vulnerable to quantum attackers, the pairing operations are considerably expensive (e.g., 10^5 times slower than computing a hash), and they may require a private setup for decentralized application scenarios like blockchains. Despite the trusted setup possibly being eliminated by using distributed key generation [29] or recently introduced transparent threshold signatures [30], [31], more communication and computation will be incurred, further hindering the performance.

Considering practical concerns associated with using threshold signatures and motivated by a desire to minimize cryptographic assumptions for enhanced security (e.g., plausible post-quantum security), there is renewed interest in exploring MVBA in the information-theoretical (IT) setting [29], [32], [35], or in solely using collision-resistant hash functions for better performance than their IT-secure analogs. In the

§ Zhenliang Lu is the corresponding author.

TABLE I: Comparison of the Multi-valued Validated BA protocols with ℓ-bit input [1]

Protocol	Resilience	Adaptive? [2]	Communication	Message	#coin	Rounds [3] (approximate)	Crypto. Tools [4]	Trivial Hash-based [5]
CKPS01-MVBA [1]	$f < n/3$	✓	$\mathcal{O}(\ell n^2 + \lambda n^2 + n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	54	Thld. Sig.	$\mathcal{O}(\ell n^2 + \lambda n^3)$
VABA [2]	$f < n/3$	✓	$\mathcal{O}(\ell n^2 + \lambda n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	19.5	Thld. Sig.	$\mathcal{O}(\ell n^2 + \lambda n^3)$
Dumbo-MVBA [3]	$f < n/3$	✓	$\mathcal{O}(\ell n + \lambda n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	46	Thld. Sig.	$\mathcal{O}(\ell n + \lambda n^3)$
sMVBA [22]	$f < n/3$	✓	$\mathcal{O}(\ell n^2 + \lambda n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	12	Thld. Sig.	$\mathcal{O}(\ell n^2 + \lambda n^3)$
sMVBA*-BLS [22] [6]	$f < n/3$	✓	$\mathcal{O}(\ell n + \lambda n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	28	Thld. Sig.	$\mathcal{O}(\ell n + \lambda n^3)$
sMVBA*-ECDSA [22] [7]	$f < n/3$	✓	$\mathcal{O}(\ell n + \lambda n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	28	ECDSA	$\mathcal{O}(\ell n + \lambda n^3)$
FIN-MVBA [32]	$f < n/3$	✓	$\mathcal{O}(\ell n^2 + \lambda n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(1)$	19.5	hash	-
ELV-HMVBA (sketched in Sec.I-B)	$f < n/3$	✗	$\mathcal{O}(\kappa\ell n + \kappa\lambda n^2 \log n)$	$\mathcal{O}(\kappa n^2)$	$\mathcal{O}(\kappa)$	46	hash	-
Our HMVBA (Sect. V)	$f < n/5$	✓	$\mathcal{O}(\ell n + \lambda n^2 \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	24	hash	-
FLT24-MVBA [33] [8]	$f < n/3$	✓	$\mathcal{O}(\kappa n\ell + \kappa\lambda n^2 \log n)$	$\mathcal{O}(\kappa n^2)$	$\mathcal{O}(\kappa)$	64	hash	-
KNR24-MVBA [34] [8]	$f < n/4$	✓	$\mathcal{O}(\ell n + \lambda n^2 \log n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(1)$	148	hash	-
KNR24-MVBA* [34] [8]	$f < (1/3 - \epsilon)n$	✓	$\mathcal{O}(\gamma_\epsilon(\ell n + \lambda n^2 \log n))$	$\mathcal{O}(n^2)$	$\mathcal{O}(\gamma_\epsilon)$	$(>)\gamma_\epsilon$	hash	-

[1] Following the standard practice in asynchronous consensus literature, we assume a common coin and consistently omit its cost. In this paper, we use f to denote the maximum number of nodes an adversary can corrupt, λ to represent the computational security parameter capturing the signature size and the hash output size, and κ to represent the statistical security parameter that is typically set to a few tens.
[2] The "classical" MVBA schemes [1]–[3], [22], [32] were not paired with detailed security proofs for adaptive security, which do hold if their all components are adaptively secure, particularly, as BLS [27] has been proved to be adaptively secure in a recent work [28].
[3] We measure the exact number of expected rounds, where the network is asynchronous and there are corrupted nodes. For protocols with a statical error [33], we measure the rounds given the error probability of 2^{-40}.
[4] Thld. Sig. stands for Threshold Signature.
[5] The trivial hash-based version means the hash-based MVBA constructions obtained by naively replacing the threshold signature used in the corresponding MVBA scheme by a concatenation of $n - f$ hash-based signatures. The asymptotic communication complexity may only get slightly worse, but the actual cubic term gets a larger coefficient too for much worse concrete complexity.
[6] sMVBA*-BLS is the MVBA scheme obtained by plugging sMVBA into Dumbo-MVBA*'s framework to reduce $\mathcal{O}(\ell n^2)$ term.
[7] sMVBA*-ECDSA replaces the threshold signature in sMVBA*-BLS with the catenation of $n - f$ ECDSA signatures.
[8] [33] and [34] are *subsequent* to this work. In [33], the parameter κ must be set to 69 to ensure a failure probability of at most 2^{-40}. In [34], the parameter γ_ϵ is given by $\gamma_\epsilon = (\lceil \frac{12}{\epsilon^2} \rceil + \lceil \frac{7}{\epsilon} \rceil)^2$ for $\epsilon \in (0, 1)$. If the resilience threshold $1/3 - \epsilon$ is set to be $1/5$, γ_ϵ can grow up to a million, leading to prohibitively high communication complexity, rendering the approach impractical.

literature, the IT setting is sometimes referred to as the signature-free setting [36], [37], which subsumes the error-free setting [38], [39]. This is in contrast with the above "classical" MVBA protocols. However, while enjoying the obvious benefits of using hash functions rather than heavy cryptographic tools like threshold signatures, the state-of-the-art design, FIN-MVBA by Duan et al. [32], suffers from high communication complexity $\mathcal{O}(\ell n^2 + \lambda n^3)$, which is even asymptotically worse than the early construction from Cachin et al.'s [1]. It follows that current hash-based MVBA achieves post-quantum security, but at the cost of larger communication (thus inferior performance), which did not realize its full power. This leads to the natural question:

Can we develop an asynchronous MVBA *that is free of heavy cryptographic operations (using collision resistant hash functions and a common coin), while at the same time, demonstrates performance benefits?*

A. Our Results

Hash-based MVBA **with (Nearly) Optimal Communication and Adaptive Security.** In this paper, we answer the question affirmatively by repeating the successful developments in "classical" MVBA protocols, i.e., match the complexity, and solely making blackbox use of conventional hash functions. Specifically, we present the first hash based MVBA protocol HMVBA with adaptive security, $\mathcal{O}(1)$ rounds, and $\mathcal{O}(\ell n + \lambda n^2 \log n)$ communication. This is achieved via a new "Dispersal-Elect-Agree" paradigm, which makes use of an overlooked primitive of asynchronous multi-valued Byzantine agreement with weak validity (see Sect.I-B for details). We compare our results with existing ones in Table I. A caveat is that our scheme tolerates up to 20% Byzantine nodes, rather than being optimally resilient against 33% corrupted nodes. Nevertheless, it is still practically useful and can be extended to handle more corruptions by leveraging common fallback mechanisms (see Sect.VIII for more discussions).

We also observe a simple hash-based MVBA construction (dubbed ELV-HMVBA in Table I) that enjoys optimal resilience and quadratic communication, under static corruption. While the conventional committee-based approach for reducing communication complexity usually has to sacrifice resilience, this result can be interesting for some applications.

Immediate applications. We can just plug-in our new HMVBA protocol to existing frameworks such as Dumbo-NG [25] and Jumbo [40] to get a better asynchronous BFT protocol, and many more. Notably, our HMVBA also directly implies a hash-based ACS with quadratic communication complexity, specifically $\mathcal{O}(\ell n^2 + \lambda n^2 \log n)$, by instantiating the framework of Cachin et al. [1] with hash-based signatures. In contrast, all existing hash-based ACS schemes [32], [41] incur cubic communication complexity.

TABLE II: Improvements of latency with mini-payloads

Scale (N)	Latency (milisec)			Improvement	
	FIN-MVBA	sMVBA*-BLS	Our HMVBA	FIN-MVBA	sMVBA*-BLS
101	4200	8414	2764	↓34%	↓67%
201	16172	16800	1941	↓88%	↓88%

Implementation and Evaluation. We implemented our HMVBA in Python 3 and deployed it on AWS EC2 `t2.medium` instances evenly distributed across 13 AWS regions[1]. For a fair comparison, we further developed Python implementations of FIN-MVBA [32] and the actual state of the art "classical" MVBA protocol sMVBA*, which is obtained by trivially instantiating the Dumbo-MVBA* framework in [3] with sMVBA in [22]. We tested the three MVBA protocols with various input sizes L and network sizes N. The results demonstrate that (1) The current hash-based MVBA FIN-MVBA is indeed sacrificing performance, as it is consistently worse than sMVBA*. (2) our new HMVBA consistently outperforms the other two MVBAs when the input size is fixed and the scale increases starting from moderate N. As Table II shows, our HMVBA brings significant improvement. Moreover, our MVBA establishes a clearly better throughput-latency trade-off at a reasonable scale. See Sect.VII for details.

B. Challenges and Our Techniques

The goal of MVBA is to decide on a "valid" input, a natural idea is to have all nodes agree on an input from a random node such that, with a constant probability, the value is valid. We found that existing MVBA schemes follow this common approach, which we call "Lock-Elect-Vote" (LEV).

The existing LEV approach. FIN-MVBA [32] employs a reliable broadcast protocol RBC [42] (which ensures agreement among all honest nodes even if the sender is malicious; see Section IV) to "lock" input messages. Then, it invokes a leader election protocol, which is usually realized by a common coin, to decide whose input is going to be the prospective output. Finally, an ABA (asynchronous binary agreement), actually a variant called reproposable ABA [41], is applied to "vote" on the status of the elected leader's RBC instance. When ABA outputs 1, it means at least one honest node inputs 1, which implies that the honest node has received the corresponding value from the elected RBC instance. RBC's property will then ensure all honest nodes will eventually receive that same value. FIN-MVBA gives a hash-based instantiation, as components have efficient hash-based instantiations [37], [42]. However, since using RBC to disseminate an ℓ-bit value to n nodes incurs $\mathcal{O}(\ell n + \lambda n^2)$ bits of communication, invoking n parallel RBC instances leads to the communication complexity of $\mathcal{O}(\ell n^2 + \lambda n^3)$. With threshold signatures, we can employ a "cheaper" broadcast primitive, provable broadcast PB [2], [4] (with communication cost of $\mathcal{O}(\ell n + \lambda n)$) to replace RBC. In doing so, we get rid of the cubic term, since n parallel PB instances only cost us $\mathcal{O}(\ell n^2 + \lambda n^2)$. Unfortunately, threshold signatures typically require algebraic structures that are not available in conventional hash functions or other lightweight tools, let alone the reliance on a trusted setup. [2]

Why hash-based quadratic MVBA is hard under an adaptive adversary? In the LEV, for locking n messages we use n instances of broadcast with a strong consistency guarantee. As mentioned, using n instances of RBC or PB results in different challenges: the former requires $\mathcal{O}(n^3)$ communication cost, while the latter demands efficient and suitable algebraic structures. An alternative method is to use a concatenation of $n - f$ hash-based signatures as the proof, which however blows up the communication cost to $\mathcal{O}(n^3)$ again. MVBA with quadratic communication without using expensive cryptographic tools appears to be beyond the LEV paradigm. We find it easy to construct an MVBA protocol with quadratic communication complexity and optimal resilience, if we only focus on *static adversaries*. At a high-level, we can select a few nodes (κ nodes) in the beginning, such that at least one of them is honest with an overwhelming probability; Then, we let all selected nodes broadcast their inputs (via RBC), and let all nodes agree on the broadcasted values (via ABA) and finally decide on a valid input. We call this construction "Elect-Lock-Vote"(ELV). Due to the failure in the "locking" step, LEV has largely failed to maintain adaptive security. Specifically, as we are using very few ($< f$) parallel broadcast to lock messages, an adaptive adversary can target all the senders to stop the protocol. We are facing a dilemma: on one hand, avoiding cubic communication (while not using threshold signatures) prevents us from locking $\mathcal{O}(n)$ inputs. On the other hand, if there are only $o(n)$ inputs to be locked, an adaptive adversary may simply corrupt all the selected senders and make the protocol stuck.

Our approach towards adaptive security: "Disseminate-Elect-Agree". To break out of this dilemma, as shown in Figure 1, we start with a "dissemination" step (which does not use RBC thus avoiding the high communication). Subsequently, we use a coin to conduct the "election", if the selected value is disseminated by a malicious node, then there is no consistency guarantee. We need a more powerful "Vote" technique to pair with the efficient "dissemination" to compensate for the absence of RBC for locking. To illustrate, let us assume the following "ideal" dissemination to design the remaining techniques, then we discuss how to realize the dissemination part efficiently.

- **Ideal dissemination.** All nodes are engaged in this phase to disseminate their inputs to the entire network. At the end of this phase, every honest node should have the inputs provided by all other honest nodes.

Now after the ideal dissemination and election, if the elected node (as sender in dissemination) is honest, every honest node is holding a same value; otherwise, they may have different values (or some may not have one). To conquer the challenge for agreement on the final output, a binary agreement (ABA) to vote as in previous constructions seems insufficient. Instead, we observe that a classical yet overlooked BFT primitive, Multi-Valued Byzantine Agreement (MBA) with weak validity [45], [46], is closer to our need as the more powerful "Vote" step. *Weak validity* means if all honest nodes have the same

[1]Our code: https://anonymous.4open.science/r/hash-mvba-7C57/.

[2]Remark that from the feasibility point of view, as inspired by recent works [30], [31], one may construct such a threshold signature by making non-blackbox use of a hash-based signature [43] and a hash-based succinct argument system [44], which, however, is much heavier than BLS [28].

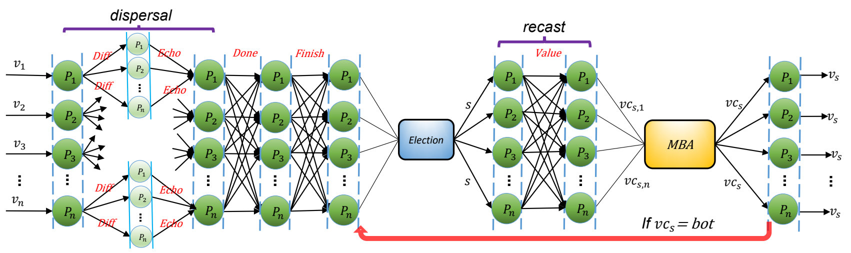


Fig. 1: The execution flow of our HMVBA

input, then they will output that value; no guarantee otherwise. Now, if the selected input is from an honest node, such that every other honest node already has it, then they must decide on this value; if the selected one is malicious, since now MBA has no guarantee, we should at least let the honest nodes be aware of the failure such that they can restart from the coin step. So before invoking an MBA protocol to vote, each honest node will multi-cast its current "notification" (either a received fragment, or nothing, just using $\perp$). More care is needed for the details (see Sec.V). We rephrase this new paradigm as "Disseminate-Elect-Agree".

Realizing dissemination with quadratic communication. We now turn to the construction of dissemination. Note that the ideal dissemination can be trivially realized in a synchronous network; Every node simply multicasts its input such that every other honest node can have it at the end of the round. The communication cost of such dissemination will be merely $\mathcal{O}(\ell n^2)$. However, subtleties arise due to asynchrony: some honest nodes may have not finished the multicast step when the election starts. We will have to relax the requirement of ideal dissemination and only ask for a constant fraction of honest inputs to be disseminated to all honest nodes.

Even for the relaxed dissemination, there are subtleties around. Let us consider a straightforward approach as a baseline, where each node performs the following tasks: (1) multicast its input; (2) when receiving an input value from node $\mathcal{P}_i$ for the first time, respond OK to $\mathcal{P}_i$; (3) whenever receiving OK from $n-f$ distinct nodes, multicast DONE; (4) whenever receiving DONE from $n-f$ distinct nodes, move to the next phase. In doing so, we can guarantee there are at least $n-2f$ honest nodes (we call them good senders hereafter) who managed to deliver their input messages to at least $n-2f$ honest nodes. However, even when a good sender is elected, there can still be f honest nodes (we call them unfortunate receivers) that have not received the corresponding message. What is worse, an adaptive adversary may corrupt the elected good sender, retract the message that has not been delivered, and send different messages to these unfortunate receivers.

We rectify this situation by letting all nodes exchange information about the value received from the elected node, such that honest nodes shall use the value endorsed by the majority of other nodes as input for MBA. When $n \geq 5f+1$, there could be $3f+1$ honest nodes having received the message from an elected good sender. In the step for exchanging information, their voice will form a majority in every honest node's view.

Pushing to optimal communication using erasure codes. All the above discussions assume that every node multicasts its entire input, which results in a communication cost of $\mathcal{O}(\ell n^2)$. However, as in MVBA each node outputs only one value, so it is unnecessary for every node to keep track of all input values from other nodes. We therefore adopt the dispersal-then-recast methodology introduced in [3]. In the MVBA design of [3], instead of having each node directly send its input to everyone, they consider that each node first disperses fragments of its input to all nodes. These fragments have a smaller size compared to the full input value. What's more, all other honest nodes can reconstruct the input of the elected node using a sufficient number of fragments. We bend the methodology into our design of dissemination, using a hash-based Merkle tree to help nodes identify the correct fragments, resulting in $\mathcal{O}(\ell n + \lambda n^2 \log n)$ bit complexity.

Applying non-intrusion secure MBA and further optimizations. Our HMVBA makes novel use of MBA to achieve agreement on messages. However, directly inputting entire messages into MBA can still be communication-intensive, potentially undermining prior efforts. Nevertheless, our dissemination phase already ensures a certain level of data availability: if an honest node uses a message as the input of MBA, all other honest nodes also can obtain this message during recast. Hence, rather than using MBA to agree on the entire message, we leverage it to agree on a short hash **digest**. In conventional MBA, the output of MBA might be a digest provided by the adversary, resulting in the unavailability of the original value for some honest nodes and causing an agreement issue. Fortunately, the *non-intrusion security* property in MBA [47] addresses this concern. This property ensures that if the output of MBA is not $\perp$, it must be the input of an honest node, ensuring data availability for the agreed digest.

We instantiated the MBA (actually IT-secure), featuring $\mathcal{O}(1)$ rounds, $\mathcal{O}(n^2)$ messages, and $\mathcal{O}(\ell n^2)$-bit communication cost. However, for practical efficiency, the MBA in [47] requires 6 additional rounds of multicast beyond its ABA components, which we aim to optimize. By leveraging the fact that our MVBA operates with $n \geq 5f+1$, we design a more efficient IT-secure MBA with only 2 extra rounds of multicast alongside ABA in the same setting, while maintaining the same asymptotic performance as [47].

II. Related Work

(Multi-valued) Asynchronous Byzantine Agreement. The most basic form of asynchronous Byzantine consensus is asynchronous binary agreement (ABA), where each node has a binary value of 0 or 1 as input and will agree on a binary value. The validity of ABA is defined in an *unanimous* manner, *i.e.*, if all honest nodes input the same binary value, they will agree on this value. As there are only two candidate input values, the validity of ABA implies the so-called *strong validity*, *i.e.*, the output is always the input of some honest node, which is a very useful property for applications. Mostefaoui et al. 's seminal work [37] presented an ABA protocol with $\mathcal{O}(1)$ rounds and $\mathcal{O}(n^2)$ communication complexity, not relying on any cryptographic tools beyond the coin. There are follow-up works [48] for improving the concrete performance of [37].

Multi-valued Byzantine agreement (MBA) with weak validity is a natural extension to ABA for handling multi-bit inputs. There is a straightforward reduction from MBA to ABA, by applying multiple ABA instances to agree on each bit. However, for an ℓ-bit input, the expected running time and the message complexity of ℓ parallel ABA instances will be blown up to $\mathcal{O}(\log \ell)$ and $\mathcal{O}(\ell n^2)$, respectively. Mostefaoui and Raynal [47] presented an optimized MBA with $\mathcal{O}(1)$ rounds, $\mathcal{O}(n^2)$ messages, and $\mathcal{O}(\ell n^2)$ communication complexity. For large-size inputs, say $\ell \gg \lambda$, Nayak et al. [49] presented a general framework that reduced the ℓn^2 term to ℓn in communication complexity using pairing-based cryptography tools. Alternatively, Li and Chen [46] presented an MBA with communication complexity of $\mathcal{O}(\ell n + n^2 \log n)$, achieving perfect security without using any cryptographic tools. Note that the MBA discussed above guarantees that when all honest nodes have the same input value, they will agree on that value. But for other cases, there is no guarantee on what value they will agree on; the output could be a default value $\perp$. Some works, including [36], [47] considered a slightly stronger validity called *non-intrusion validity*, which means if the output $v \neq \perp$, then v must be an input of an honest node. The non-intrusion property has been leveraged and explored in consequent works [50], [51]. Compared with [47], our MBA in Section VI improves the concrete communication and rounds cost while assuming $n \geq 5f+1$ which aligns with our MVBA.

Multi-valued Validated Asynchronous Byzantine Agreement. The weak validity of MBA is insufficient for many natural cases. A dream version of validity would be that the nodes always agree on the input of an honest node, which, often called *strong validity*, is known to be out of reach for large input sizes [52]. To address this issue, Cachin et al. [1] introduced *external validity*, named MVBA, which guarantees that the nodes can always agree on a "valid" value satisfying a predefined predicate function, as long as all honest nodes input valid values. Note that MVBA and MBA (with weak validity) are generally incomparable, as MVBA's output may be controlled by the adversary in any input case. However, this issue can be mitigated by carefully designing a predicate that the output should satisfy.

Since the first MVBA was introduced in [1], many subsequent works [2], [3], [22] have been proposed to either improve communication complexity or reduce the number of concrete rounds. However, all these MVBA schemes rely on threshold signatures, whose current constructions are based on algebraic assumptions that are vulnerable to quantum attackers. Additionally, threshold signatures require a trusted setup, which can be problematic in various settings. Due to the drawbacks of using heavy cryptographic tools like threshold signatures, several recent works [29], [32], [53] shifted their focus on studying MVBA in the information-theoretical setting (also known as signature free setting) or the hash-based setting, where the hash function is the only cryptographic tool and used in a blackbox manner. These works actually give a framework that could be instantiated in both settings, while their hash-based instantiations enjoy better performance. Particularly, Das et al. [29], [54] presented MVBA has $\mathcal{O}(\log n)$ rounds, $\mathcal{O}(n^3)$ messages, and the communication complexity of $\mathcal{O}(\ell n^2 + \lambda n^3)$. Duan et al. [32] improved Das et al.'s result to $\mathcal{O}(1)$ rounds. Nonetheless, there exists a significant asymptotic efficiency gap between current hash-based MVBA constructions and classical constructions [1]–[3].

Subsequent Works. There are two follow-up works on hash-based MVBA that aim to improve resilience. In particular, Feng et al. [33] presented a hash-based MVBA protocol with optimal resilience and $\mathcal{O}(\kappa n\ell + \kappa\lambda n^2 \log n)$ communication complexity, where κ is a statistical security parameter. Komatovic et al. [34] introduced a construction that offers sub-optimal yet improved resilience, specifically $1/3 - \epsilon$, with a communication complexity of $\mathcal{O}(\gamma(\epsilon)n\ell + \gamma(\epsilon)\lambda n^2 \log n)$, where $\gamma(\epsilon) = \mathcal{O}\left(\frac{1}{\epsilon^2}\right)$. Both works followed our framework of "Disseminate-Elect-Agree," using the same dissemination phase as ours but designing a stronger agreement phase to achieve better resilience. We stress that [33] and [34] incur significantly higher communication costs and rounds than ours (as shown in Table I), and they did not implement their protocols. In contrast, our protocol has been tested and shown to be significantly more efficient than state-of-the-art MVBA protocols.

III. Model and Goal

A. System model

The system involves a set of n known nodes labeled as $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ which are pairwise connected. Similar to most existing works on Byzantine fault-tolerant consensus such as [32], [37], [47], we assume that communication channels between each pair of nodes are *authenticated*, preventing the adversary from impersonating an uncorrupted node. In practice, authenticated channels can be implemented using cryptographic message authentication codes [55], which require each pair of nodes to share a secret key. Beyond this, our protocol relies solely on a collision-resistant hash function. Notably, we do not use digital signatures in any form and therefore do not assume a PKI setup.

Throughout this paper, we primarily consider an strongly adaptive and computationally bounded adversary ($\mathcal{A}$), capable of corrupting up to $f < n/5$ nodes at any point during the protocol execution[3]. Nodes not corrupted by the adaptive adversary at a certain stage of the protocol are termed "so-far-uncorrupted." However, once the adaptive adversary corrupts a node $\mathcal{P}_i$, that node $\mathcal{P}_i$ becomes fully controlled by the adaptive adversary and can act maliciously. A node is considered honest if it has never been corrupted by the adaptive adversary. Specifically, if a "so-far-uncorrupted" node $\mathcal{P}_i$ sent a message to $\mathcal{P}_j$ and then got corrupted by $\mathcal{A}$ before it was delivered, we allow the adversary to retract this message, preventing its delivery. Additionally, we concentrate on asynchronous networks, where no assumptions are made about the timing of message transmissions. Moreover, the adversary can intentionally delay messages but must eventually deliver all messages sent among honest nodes.

B. Goal: hash-based asynchronous MVBA

Our goal is to design an efficient multi-valued validated Byzantine agreement (MVBA) protocol [1]–[3]. The formal definition of MVBA is as follows.

Definition 1: In an MVBA protocol involving an external global predicate function denoted as $\mathsf{Predicate} : \{0,1\}^\ell \rightarrow \{1,0\}$, each honest node has its input value that conforms to the predefined global function Predicate. The objective is to generate a unanimous output, ensuring that the resulting output also adheres to the predetermined global function Predicate [4]. Formally, the protocol strives to attain the following properties, with all but negligible probability:

- **Termination.** If each honest node $\mathcal{P}_i$ takes an input value v_i such that $\mathsf{Predicate}(v_i) = 1$, then the protocol ensures that every honest node outputs a value v.
- **External-Validity.** If an honest node $\mathcal{P}_i$ outputs a value v, it guarantees that $\mathsf{Predicate}(v) = 1$.
- **Agreement.** If one honest node $\mathcal{P}_i$ outputs v and another honest node $\mathcal{P}_j$ outputs v', it guarantees that v is equal to v', i.e., $v = v'$.

"Quality" was introduced by Abraham et al. [2]. It indicates that the probability of the output value is determined by the adversary. If this probability is less than 1, it can prevent the adversary from entirely determining the output.

- **Quality.** If an honest node outputs an value v, then it is ensured that the probability of v being input by the adversary is bounded by a constant $\mathrm{p} \in (0, 1)$ [56].

IV. Notations and Preliminaries

In this section, we introduce the definitions of several fundamental building blocks that are employed in our paper.

Notations: The notation $\Pi[\mathsf{ID}]$ is employed to denote an instance of the protocol Π with the identifier ID. Additionally, the notation $y \leftarrow \Pi[\mathsf{ID}](x)$ signifies the action of invoking $\Pi[\mathsf{ID}]$ with input x and obtaining y as the output. We sometimes use $[k]$ to represent the integers from 1 to k. Throughout this paper, λ denotes the cryptographic security parameter, representing the length of the hash value.

Collision-resistant Hash Function. A cryptographic collision-resistant hash function ensures that a computationally limited adversary cannot find two distinct inputs that produce the same hash value, except for a negligible probability.

Erasure code scheme. The (k, n)-erasure code scheme [57] consists of two deterministic algorithms, referred to as Enc and Dec. The Enc algorithm takes a vector $\mathbf{v} = (v_1, \cdots, v_k)$ consisting of k data fragments and maps it into a vector $\mathbf{m} = (m_1, \cdots, m_n)$ containing n code fragments. Crucially, the Dec algorithm allows for the reconstruction of $\mathbf{v}$ using any set of k elements from the code vector $\mathbf{m}$. Throughout the paper, we consider a $(f + 1, n)$-erasure code scheme, and we employ the terms fragment and codeword interchangeably when the context is unambiguous.

Vector commitment (VC). A VC scheme is specified as a tuple of algorithms denoted as $(\mathsf{VCom}, \mathsf{Open}, \mathsf{VerifyOpen})$. The VCom algorithm produces a commitment vc for an input vector $\mathbf{m} = (m_1, ..., m_n)$. When provided with both (m_i, i) and vc, the Open algorithm generates a succinct proof π_i. This proof is designed to verify that the element m_i corresponds to the i-th committed element in the vector $\mathbf{m}$. The position proof can be verified by the VerifyOpen algorithm. Specifically, given m_i, i, the commitment vc, and an opening proof π_i, it outputs 0/1 to determine the validity of the proof.

Remark: In this VC scheme, the hiding property is not required, and the VCom algorithm is deterministic. To implement the VC protocol, we employ a Merkle tree based on hash functions, as described in [58]. In this instantiation, the size of commitment vc is $\mathcal{O}(\lambda)$ bits, and the openness proof π is $\mathcal{O}(\lambda \log n)$ bits in size.

Common coin & **Election**. In our protocol design, we follow the approach of prior works [15], [37], [48] by assuming the existence of common coins, a concept introduced by Rabin in [59]. This common coin provides an unpredictable and unbiased random value that is shared among all nodes in the network. We model this common coin as an oracle, denoted as $random()$, which all nodes can query using a string `event`. To prevent the adversary from preemptively knowing the coin values of honest nodes, we impose the condition that $random()$ responds to a query with `event` only when at least $f+1$ nodes have previously queried $random()$ with the same `event`. This requirement ensures that at least one non-faulty node has requested the coin.

The common coin is employed in the random leader election protocol, denoted as $\mathsf{Election}[id]$ [2], which is designed to output a uniformly sampled index $\ell \in [n]$.

Reliable broadcast (RBC). In RBC [42], there exists a sender whose goal is to broadcast a value to all nodes. More formally, an RBC protocol satisfies the following properties: *Totality*: if an honest node outputs v, then all honest nodes output v; *Agreement*: if any two honest nodes output v and v' respectively, then $v = v'$; *Validity*: if the sender is honest and inputs v, then all honest nodes output v.

[3]If there is no ambiguity, we use "adaptive adversary" to represent the strongly adaptive adversary throughout this paper.

[4]For example, in MVBA-based ACS protocols [1], [32], the predicate $\mathsf{Predicate}(v) = 1$ if and only if v represents a set of inputs from at least $n - f$ distinct nodes.

Asynchronous binary agreement (ABA). In an ABA protocol [37], honest nodes provide a single bit as input and output a common bit value $b \in \{0,1\}$ that is the input of at least one honest node. Formally, an ABA protocol aims to fulfill the following properties, with all but negligible probability: *Termination*: if all honest nodes input a bit, either 0 or 1, then every honest node outputs a bit $b \in \{0,1\}$; *Agreement*: if any two honest nodes output b and b' receptively, then $b = b'$; *Validity*: if any honest node outputs a bit $b \in \{0,1\}$, then at least one honest node had b as its input.

Multi-valued Byzantine agreement (MBA). In MBA, honest nodes provide input values $v_i \in \{0,1\}^\ell \cup \{\perp\}$, where the values are not limited to 0/1. Formally, MBA satisfies the following properties except with negligible probability: *Termination*: if all honest nodes input a value v_i, then every honest node output a value v; *Agreement*: if any two honest nodes output v and v' receptively, then $v = v'$; *Weak Validity*: if all honest nodes input the same value v, then all honest nodes output v. Besides the above conventional properties of an MBA, it also satisfies the following *non-intrusion validity* [47], [60]: if one honest node outputs v and $v \neq \perp$, then v is the input of some honest node.

V. Adaptive Asynchronous Multi-valued validated Byzantine agreement

In this section, we introduce our hash-based MVBA protocol, designed to withstand adaptive adversaries and denoted as HMVBA. The HMVBA protocol achieves an optimal time complexity of $\mathcal{O}(1)$ and an optimal message complexity of $\mathcal{O}(n^2)$. Additionally, it achieves optimal communication complexity, specifically $\mathcal{O}(n\ell)$, when the size of the input values ℓ is at least $\lambda n \log n$.

A. Overview of the HMVBA *protocol*

As we discussed in Introduction, our HMVBA follows "Disseminate-Elect-Agree" paradigm. Now we turn to explain how HMVBA realizes each part of the framework. The workflow of our HMVBA is delineated in Figure 1.

To attain the optimal communication complexity of $\mathcal{O}(n\ell)$, we let each node disseminate their input via an erasure-code-based *Dispersal phase*, rather than through simply multicasting. The Merkle tree is utilized to help the network identify the correct codewords. Specifically, as illustrated in Figure 1, when an honest node receives a valid value v, it uses the deterministic Enc algorithm to generate the codewords $\{m_1, m_2, \dots, m_n\}$ and computes the vector commitment vc using the VCom algorithm. The honest node then sends codeword m_i along with some auxiliary information to $\mathcal{P}_i$. This approach ensures a communication complexity of $\mathcal{O}(n\ell)$.

Election can be realized by the underlying common coin. After the sender $\mathcal{P}_s$ is chosen by Election, the network starts to recast the input value of $\mathcal{P}_s$ and enters the "Agree" part, trying to agree on the input of $\mathcal{P}_s$. In our protocol, each invocation of Election ensures that all honest nodes can recast the same valid value with a constant probability. This guarantee is crucial for both the performance and the complexities of the protocol.

B. Details of the HMVBA *protocol*

Our HMVBA protocol is detailed in Algorithm 1. Below is a detailed description of the process of the HMVBA protocol:

1. Dispersal phase (lines 1-16, Algorithm 1). Nodes disperse their input value v through DIFF messages; each DIFF message includes a vector commitment, a codeword, and a position proof. When a node receives a valid DIFF message from a sender for the first time, it responses with an ECHO message to the sender. Once a node receives $n-f$ ECHO messages from distinct nodes, it informs all nodes that its dispersal is completed via DONE messages. When a node receives $n-f$ DONE messages from distinct nodes, it will send a FINISH message to all nodes. If an honest node receives $f+1$ FINISH messages from distinct nodes, it will also send a FINISH message to all nodes if it has not done so already.

2. Election **& Recast phase** (lines 17-33, Algorithm 1). Once a node receives $n-f$ FINISH messages from distinct nodes, it initiates a common coin protocol Election to randomly select a leader node $\mathcal{P}_s$. Nodes exchange the DIFF messages received from the elected node and attempt to reconstruct a value. Specifically, upon receiving the result $\mathcal{P}_s$ from Election, each node $\mathcal{P}_i$ checks if it has previously received a valid DIFF message from the sender $\mathcal{P}_s$. If $\mathcal{P}_i$ has received it, $\mathcal{P}_i$ multicasts $(\text{VALUE}, k, \mathcal{P}_s, \mathsf{vc}, m_i, \pi_i)$. If $\mathcal{P}_i$ has not received a valid message, it multicasts $(\text{VALUE}, k, \mathcal{P}_s, \perp, \perp, \perp)$. Honest nodes wait for $n-f$ VALUE messages from distinct nodes. If there are at least $n-3f \geq 2f+1$ messages carrying the same vc, each node randomly selects $f+1$ messages from these $2f+1$ messages and tries to decode the $f+1$ codewords to generate an output value M_i. If the value M_i satisfies the condition $\mathsf{Predicate}(M_i) = 1$, it will set VCom_i equal to the vector commitment VCom of the value M_i. Otherwise, it will set VCom_i equal to $\perp$.

3. MBA phase (lines 34-43, Algorithm 1). All honest nodes invoke MBA with the vector commitment vc_i as input. Suppose MBA returns a value vc'. If $\mathsf{vc}' \neq \perp$ and vc' is the input of MBA, then output the M_i generated in the recast phase. If $\mathsf{vc}' \neq \perp$ and vc' is not the input of MBA, then wait for $f+1$ valid VALUE messages from distinct nodes such that $|store[\mathsf{vc}']| = f+1$, and then decode it to output M_i. If $\mathsf{vc}' = \perp$, repeat the Election process until an externally valid value is obtained.

C. Security and Complexity analysis

Security analysis. We now establish the security of Algorithm 1 in the following theorem, and provide a proof sketch.

Theorem 1: Assuming the hash function is collision-resistant, and the MBA protocol is adaptively secure, our HMVBA in Algorithm 1, aided by a common coin, is a secure MVBA against any adaptive and computationally bounded adversary corrupting up to f among $n \geq 5f+1$ nodes.

Proof 5.1 (sketched): We first establish the following useful facts about our dissemination phase, and then prove the security properties of our HMVBA based on these facts.

First, we have the following Claim 1: if a "so-far-uncorrupted" node $\mathcal{P}_s$ successfully disperses its input (i.e.,

Algorithm 1 HMVBA protocol with external Predicate, for each node $\mathcal{P}_i$: $n \geq 5f+1$

```
let $S[i] \leftarrow \bot$ for $i \in [n]$, $store \leftarrow \{\ \}$, $flag \leftarrow 0$, $abandons \leftarrow 0$
                                                        ▷ Dispersal
upon receiving an input value $v$ s.t. Predicate$(v) = 1$ do
  $\boldsymbol{m} := \{m_1, \cdots, m_n\} \leftarrow$ Enc$(n, f, v)$, where $v$ is parsed as a $f+1$ vector
  vc $\leftarrow$ VCom$(\boldsymbol{m})$;
  for each $j \in [n]$ do
    $\pi_j \leftarrow$ Open(vc, $m_j, j$)
    send (DIFF, vc, $m_j, \pi_j$) to $\mathcal{P}_j$
upon receiving (DIFF, vc, $m_i, \pi_i$) from $\mathcal{P}_j$ for the first time do
  if $abandons = 0$ and VerifyOpen(vc, $m_i, i, \pi_i$) $= 1$ then
    $S[j] \leftarrow (j$, vc, $m_i, \pi_i)$                    ▷ store fragment
    send (ECHO, 1) to $\mathcal{P}_j$
upon receiving (ECHO, 1) from $n-f$ nodes do
  multicast (DONE, 1)
upon receiving (DONE, 1) from $n-f$ nodes do
  multicast (FINISH, 1)
upon receiving (FINISH, 1) from $f+1$ nodes do
  multicast (FINISH, 1) if it has not yet been sent
upon receiving (FINISH, 1) from $n-f$ nodes do
  $abandons \leftarrow 1$                          ▷ abandon all Dispersal
  for each $k \in \{1, 2, 3, \ldots\}$ do
    $s \leftarrow$ Election$[k]$                          ▷ threshold $f+1$
    if $S[s] := (j$, vc, $m_i, \pi_i)$ then
      multicast (VALUE, $k, s$, vc, $m_i, \pi_i$)
    else
      multicast (VALUE, $k, s, \bot, \bot, \bot$)              ▷ $S[s] = \bot$
    upon receiving (VALUE,$k$,$s$,vc,$m_j$,$\pi_j$) from $\mathcal{P}_j$ for the first time do
      if $m_j \neq \bot$ and VerifyOpen(vc, $m_j, j, \pi_j$) $= 1$ then
        $store$[vc] $\leftarrow store$[vc] $\cup (j, m_j)$
        upon $|store$[vc]$| = n - 3f$ do
                  ▷ pick $f+1$ elements in $store$[vc] when decoding
          $M_i \leftarrow$ Dec($store$[vc])
          if Predicate$(M_i) = 1$ and VCom(Enc$(n, f, M_i)$) = vc then
            vc$_i \leftarrow$ vc, $flag \leftarrow 1$
    upon receiving VALUE from $n-f$ nodes and $flag = 0$ do
      vc$_i \leftarrow \bot$; $flag \leftarrow 1$
    upon $flag = 1$ do
      vc$' \leftarrow$ MBA$[k]$(vc$_i$)                    ▷ see Algorithm 2
      if vc$' \neq \bot$ then
        if vc$' =$ vc$_i$ then
          output $M_i$
        else
          wait until $|store$[vc$']| = f+1$
            output $M_i \leftarrow$ Dec($store$[vc$'$])
      else
        $M_i \leftarrow \bot$; $flag \leftarrow 0$; $store \leftarrow \{\}$
```

multicasts (DONE, 1)), then all honest nodes can recover its original input value M, even if the node later becomes corrupted by an adaptive adversary. The main reason is as follows: when $\mathcal{P}_s$ multicasts DONE, at least $n-f$ nodes have received fragments of message M from $\mathcal{P}_s$ along with the same vector commitment corresponding to M, with at least $n-2f$ of them being honest nodes. Meanwhile, there are up to f honest nodes may not have received any fragments. Subsequently, an adaptive adversary can later corrupt $\mathcal{P}_s$ and send incorrect fragments (not corresponding to the fragments in M) to these f nodes that have not received the correct fragments. In doing so, the adversary can provide up to $2f$ incorrect fragments in the recasting phase. To successfully reconstruct the original message M, it is necessary to receive at least $2f+1$ correct fragments with the same vector commitment among the received $4f+1$ messages, which is why we need to require $n \geq 5f+1$ in the asynchronous setting.

Second, we present the following Claim 2: when an honest node initiates the leader election, at least $n-2f$ "so-far-uncorrupted" nodes have already completed their dispersal, and every honest node can enter the leader election phase. Specifically, an honest node $\mathcal{P}_i$ invokes Election only when it receives $n-f$ FINISH messages, which indicates that at least $f+1$ honest nodes have sent the FINISH messages. According to line 15, every honest node will multicast the FINISH message, so that every honest node can receive $n-f$ FINISH messages to enter the election phase. On the other hand, it is easy to see that at least one honest node has received $n-f$ DONE messages. As a result, at least $n-2f$ honest nodes multicast DONE messages, which also means these $n-2f$ honest nodes have successfully dispersed their input.

Now, we turn to prove the security properties.

Termination: As we argued above, all honest nodes can enter the leader election phase. Then, every honest node exchanges their fragments via VALUE messages. As a result, each honest node can receive at least $n-f$ VALUE messages, allowing all honest nodes to proceed to the next step and invoke MBA with a single input. Moreover, based on Claim 1 and Claim 2, and given that the election function returns an unpredictable and unbiased random value, with a probability of $3/5$, all honest nodes can recast the same valid value that satisfies the Predicate function, leading to the invocation of MBA with the same input, which is not $\bot$. Following the termination of MBA, all honest nodes output the same commitment value vc.

Furthermore, if vc is not $\bot$, then by the *non-intrusion* property of MBA, at least one honest node must have taken vc as input. According to Algorithm 1, that honest node has received at least $n-3f$ valid fragments from VALUE messages corresponding to the vector commitment vc. Therefore, all honest nodes can wait for $f+1$ valid VALUE messages with the same vc to decode an output. According to Algorithm 1, all honest nodes repeatedly invoke the election function until a valid value is output. In summary, all honest nodes can terminate after invoking the election function at most twice.

Agreement: Note that the *agreement* property of MBA ensures that all honest nodes output the same vector commitment vc$'$. According to line 36 of Algorithm 1, if and only if vc$' \neq \bot$, all honest nodes can output a value. Following lines 37 and 40 of Algorithm 1, the output of all honest nodes is determined by two conditions. Assume that one honest node $\mathcal{P}_i$ outputs M, while another honest node $\mathcal{P}_j$ outputs M'. Let us consider the following two cases:

Case 1: If both $\mathcal{P}_i$ and $\mathcal{P}_j$ output by line 37, meaning their input is equal to the output of MBA, then according to lines 30–31 of Algorithm 1, the honest node $\mathcal{P}_i$ has the value M that satisfies VCom(Enc(n, f, M)) = vc$'$, and the honest node $\mathcal{P}_j$ has the value M' that satisfies VCom(Enc(n, f, M')) = vc$'$. Since VCom and Enc are deterministic algorithms, it follows that $M = M'$.

Case 2: If $\mathcal{P}_j$ outputs by line 40, then by the *non-*

intrusion property of MBA, at least one honest node (say $\mathcal{P}_i$) must have taken vc' as input. According to lines 30–31 of Algorithm 1, the output value M of the honest node satisfies $\mathsf{VCom}(\mathsf{Enc}(n, f, M)) = \mathsf{vc}'$. For any pair (j, m_j) that satisfies $\mathsf{VerifyOpen}(\mathsf{vc}', m_j, j, \pi_j) = 1$, the value m_j must be a fragment corresponding to M. Otherwise, $\mathsf{VCom}(\mathsf{Enc}(n, f, M)) \neq \mathsf{vc}'$ would lead to a contradiction. Thus, any set of $f+1$ valid fragments corresponding to vc' can reconstruct the same value M, ensuring that $M = M'$.

External-validity: It is granted as an honest node will only output a value satisfying the external predicate.

Quality: As the network is trying to agree on an input of a randomly selected node. When an honest node who has finished the dispersal phase is selected, then the network can agree on the input of this node. Therefore, there is a constant probability that the output is an honest node's input.

Efficiency analysis. As depicted in Algorithm 1, the cost breakdown of the HMVBA protocol can be summarized into three distinct phases: In the dispersal phase, which incurs $\mathcal{O}(n^2)$ messages and $\mathcal{O}(n\ell + \lambda n^2 \log n)$ bit complexity, where each node sends a total of $\mathcal{O}(n)$ messages, including DIFF, ECHO, DONE, and FINISH messages. In the Election & Recast phase, beyond a single common coin invocation, the recast phase requires one all-to-all multicast of VALUE messages, incurring $\mathcal{O}(n^2)$ messages exchange and $\mathcal{O}(n\ell + \lambda n^2 \log n)$ bits of communication. In the MBA phase, there is only one MBA instance, Assuming the use of the MBA protocol [37], where the time complexity of MBA is $\mathcal{O}(1)$, message complexity is $\mathcal{O}(n^2)$, and communication complexity is $\mathcal{O}(n^2\ell)$. Moreover, the Election & Recast phase and the MBA phase are expected to be repeated two times. To summarize, the HMVBA in Algorithm 1 has constant running time, $\mathcal{O}(n^2)$ message complexity, and $\mathcal{O}(n\ell + \lambda n^2 \log n)$ communication complexity, where ℓ is the input size.

VI. Asynchronous Multi-valued Byzantine Agreement with Weak Vailidity

In this section, we propose a novel MBA that significantly outperforms the one in [47]. Specifically, our MBA reduces the all-to-all broadcast step by at least four compared to the MBA in [47], while maintaining the same asymptotic complexity. Our MBA construction assumes an adaptively secure ABA protocol, which can be instantiated using the IT ABA protocol [37] combined with a common coin, offering $\mathcal{O}(n^2)$ message complexity, $\mathcal{O}(n^2\lambda)$ communication complexity, and $\mathcal{O}(1)$ time complexity.

A. *Overview of the* MBA *protocol*

Our construction is primarily based on the following principle: If all honest nodes input the same value v, then all honest nodes will output v. To achieve this, we first perform a "filter" procedure to retain only the "good cases," where a good case is at least the majority of nodes have the same input. After this filtering process, all honest nodes invoke ABA to determine whether to output a non-$\perp$ value, based on the output of ABA. If ABA outputs 1, it signifies the occurrence of a good case. To achieve agreement, our protocol ensures that all honest nodes output the same value in the good case. To maintain weak validity, we also guarantee that the output value matches the input of the majority of nodes.

It is crucial to ensure that when all honest nodes have the same input, they can collectively identify the occurrence of a good case. This enables all honest nodes to input 1 when invoking ABA. Additionally, we must ensure that even if not all honest nodes input the same values, the protocol will not get stuck. Therefore, in all scenarios, our protocol guarantees that all honest nodes will always have a value as input for ABA, ensuring termination.

Algorithm 2 MBA protocol, for each node $\mathcal{P}_i$: $n \geq 5f+1$

```
    let $flag \leftarrow 0$
 1: upon receiving an input value $v$ do
 2:   multicast $(\text{VALUE}, v)$
 3: upon receiving $(\text{VALUE}, *)$ from $n-f$ nodes do
 4:   if $(\text{VALUE}, v')$ was received from $n-2f$ nodes then
 5:     multicast $(\text{ECHO}, v')$
 6:   else
 7:     multicast $(\text{ECHO}, 0)$
 8: upon receiving $(\text{ECHO}, *)$ from $n-f$ nodes do
 9:   if $(\text{ECHO}, v')$ was received from $n-2f$ nodes and $v' \neq 0$ then
10:     $flag \leftarrow 1$
11:   else
12:     $flag \leftarrow 0$
13: wait $b \leftarrow \mathsf{ABA}(flag)$
14:   if $b = 0$ then
15:     output $\perp$
16:   if $b = 1$ then
17:     wait until receiving $(\text{ECHO}, v')$ from $f$+1 nodes and $v' \neq 0$
18:       output $v'$
```

B. *Details of the* MBA *protocol*

In this section, we present a comprehensive description of the construction of our MBA protocol. The detailed procedure for MBA is outlined in Algorithm 2. The protocol consists of three distinct logical phases, following these sequential steps:

1. Filter phase (lines 1-7, Algorithm 2). When a node $\mathcal{P}_i$ receives an input value v, it multicasts (VALUE, v) to all nodes. All nodes wait for VALUE messages from $n-f$ distinct nodes. Once the node $\mathcal{P}_i$ receives (VALUE, v') from $n-2f$ distinct nodes, where $v' \neq 0$, it multicasts (ECHO, v') message to all nodes. This (ECHO, v') message serves as the signal that at least $n-2f$ honest nodes have the same input. Otherwise, it multicasts $(\text{ECHO}, 0)$ message to all nodes.

2. ABA phase (lines 8-13, Algorithm 2). For any node $\mathcal{P}_i$, upon receiving ECHO messages from $n-f$ distinct nodes, if at least $n-2f$ ECHO messages carry the same non-zero value v', then it will input 1 to ABA; otherwise, it takes 0 as its input.

3. Output phase (lines 14-18, Algorithm 2). In this phase, all nodes output a value based on the output of ABA. If the output is 0, then all nodes output $\perp$. If the output is 1, then all nodes output a non-$\perp$ value. Specifically, if ABA outputs 1, then all honest nodes wait until receiving (ECHO, v') from $f+1$ distinct nodes, where $v' \neq 0$. Then, all nodes output value v'.

C. Security and Complexity analysis

Security analysis. We establish the security of MBA in the following theorem and provide a sketch of the proof.

Theorem 2: Assuming the underlying ABA protocol is adaptively secure, our MBA in Algorithm 2 is a secure MBA against any adaptive and computationally bounded adversary corrupting up to f among $n \geq 5f+1$ nodes.

Proof 6.1 (sketched): Agreement: When an honest node $\mathcal{P}_i$ outputs $v' \neq \perp$, ABA must return 1. Due to the validity of ABA, there must be at least one honest node that provided 1 to ABA, so it has received at least (ECHO, v') from at least $n-2f$ nodes. It follows that at least $n-3f = 2f+1$ honest nodes sent (ECHO, v'), so every honest nodes can receive (ECHO, v') from at least $f+1$ nodes. Then, we show there are no distinct v' and v'', such that both of them are carried by at least $f+1$ ECHO messages. Note an honest node multicasts (ECHO, v') only when it has received message (VALUE, v') from $n-2f$ distinct nodes. Since $n \geq 5f+1$, at least $n-3f \geq 2f+1$ honest nodes multicast (VALUE, v'). As a result, if two different honest nodes $\mathcal{P}_i$ and $\mathcal{P}_j$ multicast (ECHO, v') and (ECHO, v''), respectively, and if $v' \neq 0$ and $v'' \neq 0$, then $v' = v''$. Therefore, if an honest node outputs v', all other honest nodes output v'.

Termination: Due to the termination property of ABA, all honest nodes eventually receive a binary value from ABA. When ABA returns 0, all nodes can terminate with $\perp$. Otherwise, if ABA returns 1, all honest nodes will decide on the same value as we argued above.

Weak validity: When all honest nodes have the same input v, then every honest node can receive enough VALUE and ECHO messages carrying v. Therefore, all honest nodes provide 1 as their inputs to ABA. Due to the validity of ABA, all honest nodes receive 1 from ABA and then decide on the value v.

Non-intrusion: If an honest $\mathcal{P}_i$ returns v', then there exists an honest $\mathcal{P}_j$ having received (ECHO, v') from at least $n-2f$ nodes. It follows that at least $n-3f$ honest nodes received (VALUE, v') from at least $n-2f$ nodes. So v' is the input of at least $n-3f$ honest nodes, while $n-3f > 1$ as $n \geq 5f+1$.

Efficiency of MBA. The cost breakdown of Algorithm 2 consists of three phases. In the filter phase, each node sends n messages, resulting in $\mathcal{O}(n^2)$ messages and $\mathcal{O}(n^2\ell)$ bit complexity due to the message size ℓ. In the ABA phase, beyond the invocation of the common coin, the protocol exchanges $\mathcal{O}(n^2)$ messages with a bit complexity of $\mathcal{O}(n^2)$. Finally, the output phase incurs no communication cost, as it requires no message exchange. Hence, the MBA implemented in Algorithm 2 has constant running time, $\mathcal{O}(n^2)$ message complexity, and $\mathcal{O}(n^2\ell)$ communication complexity, where ℓ is the input size.

VII. Implementation and Evaluations

We implemented and evaluated the performance of HMVBA within a WAN environment. Throughout our study, we systematically compared HMVBA with several common MVBA protocols, including sMVBA* [22] and hash-based FIN-MVBA [32]. In our evaluations, we examined two variants of sMVBA*: sMVBA*-BLS and sMVBA*-ECDSA. In sMVBA*-BLS, we employed sMVBA [22] as the underlying MVBA to instantiate the Dumbo-MVBA* [3], utilizing the BLS threshold signature. Conversely, in sMVBA*-ECDSA, we substituted the BLS threshold signature with the concatenation of $n-f$ ECDSA signatures. Additionally, we emphasize that our experiments did not incorporate any network layer optimizations.

Test environment. The experiments are conducted among AWS EC2 `t2.medium` instances evenly distributed in 13 AWS regions: N. Virginia, Ohio, N. California, Oregon, Canada Central, Mumbai, Tokyo, Seoul, Osaka, Singapore, Sydney, Ireland, and São Paulo. Each `t2.medium` instance has 2 Intel Xeon processors of speed up to 3.4GHz Turbo CPU clock and 4 GB memory. It also provides a baseline bandwidth of 256 Mbps and a peak bandwidth of 1024 Mbps.

The one-shot agreement is evaluated with different input sizes (L) and network sizes (N). Each input L consists of B batches of transactions. In our experiments, a single transaction is represented as a string of 250 bytes, which approximates the size of a typical Bitcoin transaction with one input and two outputs. Hence, we express L as $250 \times B$, where the batch size B ranges from 0 (mini-payload, an empty string) to 7×10^3 in our experiments. Moreover, we conducted tests with six different network sizes, specifically $N = 6$, 16, 31, 61, 101, and 201, while the parameter f is set as the optimal threshold of the protocol being tested [5].

In our test, all N nodes take an input and participate in the instantiation simultaneously. The latency for each node is defined as the time difference between receiving an input and outputting all transactions. Initially, we obtained ten latency measurements by conducting repeated assessments ten times for each specific test configuration under a fixed network size and a fixed input size. Subsequently, the average latency is determined to be the mean of the collected latency values.

Implementation details. All asynchronous protocols are written as multi-process Python 3 programs, and are developed upon the open source code of *Dumbo_NG* [25] [6]. The pair-to-pair communication channels between every two nodes are set up using unauthenticated TCP sockets. The Python program initiates three processes on each node, comprising a protocol process responsible for protocol execution, a client process managing data transmission, and a server process managing data reception. Concurrent tasks within each process are managed by *gevent* [7] Python library.

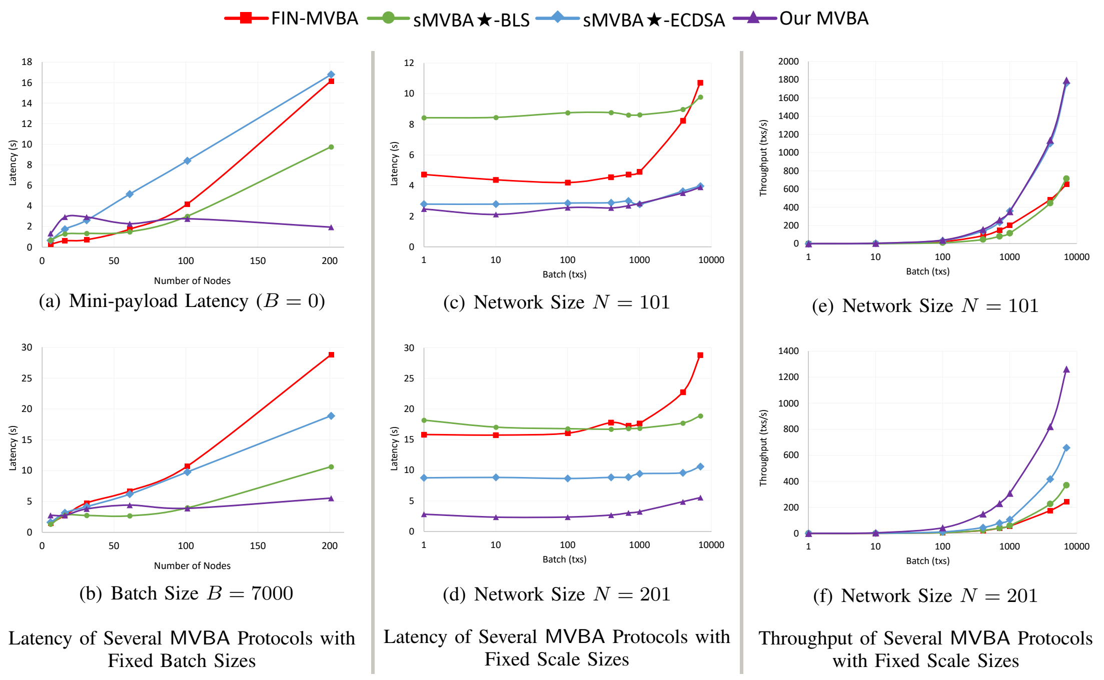
We adapt the source code of sMVBA* [3] in *Dumbo_NG* [25] [8] to fit our testing framework. All components in FIN-MVBA [32] are implemented from scratch, including its

[5]We evaluate the performance of different protocols under the same network size, as most blockchain applications operate with a fixed-size network. A comparison based on the same fault tolerance threshold can be inferred by assessing the performance of HMVBA on a network with $5N/3$ nodes and comparing it to that of other protocols in a network with N nodes.

[6]https://github.com/yylluu/Dumbo_NG

[7]http://www.gevent.org/

[8]https://github.com/yylluu/Dumbo_NG/tree/main/dumbomvbastar



(a) Mini-payload Latency ($B = 0$)

(b) Batch Size $B = 7000$

Latency of Several MVBA Protocols with Fixed Batch Sizes

(c) Network Size $N = 101$

(d) Network Size $N = 201$

Latency of Several MVBA Protocols with Fixed Scale Sizes

(e) Network Size $N = 101$

(f) Network Size $N = 201$

Throughput of Several MVBA Protocols with Fixed Scale Sizes

Fig. 2: Performance of Several MVBA Protocols

WRBC and RABA constructions. For our own HMVBA, we adopted the ABA component from *ADKG* [9] in [29]. Everything else is newly implemented, except for the basic cryptographic components present in *Dumbo_NG*, such as erasure code, Merkle tree, and ECDSA signature; we re-use these components from *Dumbo_NG*. Common coins and leader election protocols are needed by all tested MVBA protocols. They are implemented by hashing the session ID, which is shared across the entire network. Thus, the implementation of these sub-protocols does not introduce any fairness issues.

Latency Improvements with Fixed Input Sizes. Our HMVBA achieves a substantial reduction in basic latency (e.g., batch size $B = 1$) compared to FIN-MVBA by 47% and 82% when $N = 101$ and 201, respectively. Similarly, it also demonstrates latency reductions of 11% and 67% compared to sMVBA*-ECDSA under the same conditions of $N = 101$ and 201. As for sMVBA*-BLS, the reductions in latency are even greater. A mini-payload input, an empty string realized by $B = 0$, is also tested as the input of MVBAs against multiple group sizes. Fig. 2(a) shows that HMVBA outperforms FIN-MVBA and the two variants of sMVBA* for a scale $N \geq 101$. These outcomes align with our asymptotic comparisons, as our MVBA effectively reduces the $\mathcal{O}(\lambda n^3)$ term in the communication cost of FIN-MVBA. On the other hand, on a smaller scale with $N < 101$, the cubic term in FIN-MVBA and sMVBA* might not dominate the overall communication cost. As shown in Table I, they also incur fewer rounds. [10] As a result, the latency of HMVBA is slightly higher than theirs in such cases.

For larger input sizes (e.g., $B = 7000$), our HMVBA continues to outperform FIN-MVBA and achieves latency reductions starting from an even smaller scale (e.g., $N = 31$). This reflects our asymptotic improvement by reducing the $\mathcal{O}(\ell n^2)$ term in the communication cost of FIN-MVBA to $\mathcal{O}(\ell n)$. In contrast, since sMVBA* already benefits from the $\mathcal{O}(\ell n)$ term, the latency of our HMVBA is still higher than that of sMVBA* on a smaller scale. However, HMVBA still can reduce latency under the same conditions that enable it to outperform sMVBA* in basic latency comparisons. This can be indicative of the impact of the gap in computational complexity, as observed when $N = 101$ and 201. Refer to Fig. 2(b).

Latency & Throughput Improvements with Fixed Network Size. To illustrate our advantages in handling various input sizes in a fixed-size network, we demonstrates results of all four MVBA protocols with a fixed two network sizes of $N = 101$ and 201, and various input batch sizes ranging from $B = 1$ to 7000. As depicted in Fig. 2(c) & 2(d), our HMVBA consistently reduces latency across all tested input sizes. Furthermore, with fixed network sizes of $N = 101$ and 201, our HMVBA demonstrates greater throughput across all tested input sizes, cf Fig. 2(e) & 2(f). This aligns with our advantages in less total latency illustrated in Fig. 2(c) & 2(d).

Better Latency-throughput Trade-off. In Fig. 3, the throughput-latency trade-offs in the four MVBA protocols are

[9]https://github.com/sourav1547/adkg/blob/adkg/adkg/broadcast/binaryagreement.py

[10]While the expected number of rounds of sMVBA* is larger than ours, it offers an optimistic path with fewer rounds.

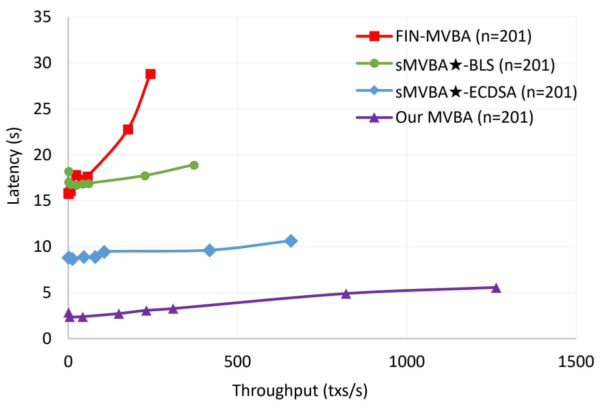


Fig. 3: Latency vs. Throughput of Several MVBA Protocols

demonstrated at $N = 201$. We will also show the advantage of HMVBA at $N = 61, 101$ in the full paper. These curves do not exhibit an L-shape, suggesting that the tested MVBA protocols have not reached the network-bound, and the peak throughput is yet to be observed. In general, MVBA serves as a subroutine within a larger system, such as ACS, rather than functioning as an independent system. In most ACS frameworks, the payload supplied to MVBA is relatively small ($\mathcal{O}(n)$ bits or $\mathcal{O}(n\lambda)$ bits) and it falls within our testing range where the largest batch size tested is 7000, translating to 1.4×10^7 bits. Therefore, we refrain from evaluating our MVBA on larger input sizes or conducting a systematic benchmark analysis of its resource consumption, such as CPU usage and other system-related metrics. Despite not exhausting their bandwidths, our HMVBA has already demonstrated a significant advantage over FIN-MVBA and sMVBA*-BLS and sMVBA*-ECDSA. Specifically, when $N = 201$, our HMVBA achieves a maximum throughput of 1263 transactions per second under the current testing condition, whereas FIN-MVBA has never reached this throughput under the same testing conditions.

VIII. Conclusion and Future Work

Conclusion. We developed an MVBA that relies on collision-resistant hash functions and a common coin, while simultaneously achieving optimal communication and maintaining adaptive security. Additionally, we proposed a new MBA that requires fewer rounds compared to the existing one [47], while maintaining the same asymptotic performance. Extensive experiments demonstrate that HMVBA offers better performance when the system scale is moderately large.

Future work. While the $1/5$ resilience is acceptable and already used in various scenarios, such as those considered in Algorand [61], achieving the optimal resilience of $1/3$ is certainly desirable. Subsequent works [33], [34] have aimed to improve this aspect, with [33] achieving both optimal resilience and quadratic communication complexity. However, significantly higher round costs are incurred in [33], and the protocols are generally much more involved thus hindering practical performance. Therefore, it remains an interesting open problem to achieve optimal resilience while retaining the practical performance of HMVBA.

One promising approach is to build an optimistic asynchronous protocol: in which a *fast path* is usually run with higher performance and lower resilience, and when extreme attacks happen, the participants jointly "switch" to a slower but more resilient protocol. Such an idea got substantial recent attentions, there exist extensive investigations of optimistic consensus protocols [62], [63], [10], [64], [65]. Notably, in all those work, the fast path only ensures security in the very optimistic case that there is no Byzantine node, and the network has to be synchronous.

We believe that a potential approach for enhancing the practical performance of [33] can be to leverage our HMVBA as a *robust* fast path. Namely, when the number of corrupted nodes is below $n/5$, the network can reach a consensus via our HMVBA with fewer rounds; if more than $n/5$ nodes are corrupted, the network can still reach a consensus through the slower but more secure protocol from [33]. One possible way is that we may follow the ideas of [64], [65] to run two protocols (our HMVBA and [33]) in parallel, such that the network normally takes the output of the fast path (HMVBA) but switches to the output of the slow path ([33]) when it is "catching up".

That being said, there are a few challenges to be addressed. First, the safety of HMVBA shall be ensured even when more than $n/5$ (but less than $n/3$) nodes are corrupted, such that only liveness will be influenced; this might be achieved by employing a standard optimal-resilient MBA protocol [47]. Also, the network needs to agree on which path to take, which may require another ABA instance to finalize or employ the detection techniques from [66]. Moreover, the solution involving two parallel chains could be further refined, as they may increase the actual bandwidth requirements compared to sequential chains in practical deployments.

Resolving all these issues to have full-fledged designs to enjoy the best of both (performance benefits of our HMVBA, and resilience benefits of the optimally resilient ones such as [33]) will be interesting future work to tackle.

Acknowledgments

We thank the anonymous reviewers and our shepherd, Alysson Bessani, for their valuable feedback on an earlier version of this paper. This work is supported in part by ARC Discovery Project (DP250101739), and research awards from the Ethereum Foundation, Protocol Labs, and the Stellar Development Foundation.

References

[1] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, "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*, ser. Lecture Notes in Computer Science, vol. 2139. Springer, 2001, pp. 524–541.
[2] I. Abraham, D. Malkhi, and A. Spiegelman, "Asymptotically optimal validated asynchronous Byzantine agreement," in *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. ACM, 2019, pp. 337–346.

[3] Y. Lu, Z. Lu, Q. Tang, and G. Wang, "Dumbo-mvba: Optimal multi-valued validated asynchronous Byzantine agreement, revisited," in *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. ACM, 2020, pp. 129–138.

[4] B. Guo, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo: Faster asynchronous BFT protocols," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 2020, pp. 803–818.

[5] Y. Gao, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Efficient asynchronous Byzantine agreement without private setups," in *42nd IEEE International Conference on Distributed Computing Systems, ICDCS 2022, Bologna, Italy, July 10-13, 2022*. IEEE, 2022, pp. 246–257.

[6] E. Kokoris-Kogias, D. Malkhi, and A. Spiegelman, "Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*. ACM, 2020, pp. 1751–1767.

[7] I. Abraham, P. Jovanovic, M. Maller, S. Meiklejohn, G. Stern, and A. Tomescu, "Reaching consensus for asynchronous distributed key generation," *Distributed Comput.*, vol. 36, no. 3, pp. 219–252, 2023.

[8] T. Yurek, Z. Xiang, Y. Xia, and A. Miller, "Long live the honey badger: Robust asynchronous DPSS and its applications," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. USENIX Association, 2023, pp. 5413–5430.

[9] B. Hu, Z. Zhang, H. Chen, Y. Zhou, H. Jiang, and J. Liu, "Dycaps: Asynchronous proactive secret sharing for dynamic committees," *IACR Cryptol. ePrint Arch.*, p. 1169, 2022.

[10] Y. Lu, Z. Lu, and Q. Tang, "Bolt-dumbo transformer: Asynchronous consensus as fast as the pipelined BFT," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 2159–2173.

[11] R. Gelashvili, L. Kokoris-Kogias, A. Sonnino, A. Spiegelman, and Z. Xiang, "Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback," in *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 13411. Springer, 2022, pp. 296–315.

[12] R. Bacho, D. Collins, C. Liu-Zhang, and J. Loss, "Network-agnostic security comes (almost) for free in DKG and MPC," in *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 14081. Springer, 2023, pp. 71–106.

[13] A. Choudhury and A. Patra, "An efficient framework for unconditionally secure multiparty computation," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 428–468, 2017.

[14] M. Backes, F. Bendun, A. Choudhury, and A. Kate, "Asynchronous MPC with a strict honest majority using non-equivocation," in *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*. ACM, 2014, pp. 10–19.

[15] M. Ben-Or, B. Kelmer, and T. Rabin, "Asynchronous secure computations with optimal resilience (extended abstract)," in *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, Los Angeles, California, USA, August 14-17, 1994*. ACM, 1994, pp. 183–192.

[16] E. Blum, C. L. Zhang, and J. Loss, "Always have a backup plan: Fully secure synchronous MPC with asynchronous fallback," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 12171. Springer, 2020, pp. 707–731.

[17] R. Canetti, "Studies in secure multiparty computation and applications," *Scientific Council of The Weizmann Institute of Science*, 1996.

[18] A. Chopard, M. Hirt, and C. Liu-Zhang, "On communication-efficient asynchronous MPC with adaptive security," in *Theory of Cryptography - 19th International Conference, TCC 2021, Raleigh, NC, USA, November 8-11, 2021, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 13043. Springer, 2021, pp. 35–65.

[19] A. Choudhury and N. Pappu, "Perfectly-secure asynchronous MPC for general adversaries (extended abstract)," in *Progress in Cryptology - INDOCRYPT 2020 - 21st International Conference on Cryptology in India, Bangalore, India, December 13-16, 2020, Proceedings*, ser. Lecture Notes in Computer Science, vol. 12578. Springer, 2020, pp. 786–809.

[20] A. Choudhury and A. Patra, "Optimally resilient asynchronous MPC with linear communication complexity," in *Proceedings of the 2015 International Conference on Distributed Computing and Networking, ICDCN 2015, Goa, India, January 4-7, 2015*. ACM, 2015, pp. 5:1–5:10.

[21] D. Lu, T. Yurek, S. Kulshreshtha, R. Govind, A. Kate, and A. Miller, "Honeybadgermpc and asynchromix: Practical asynchronous MPC and its application to anonymous communication," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 887–903.

[22] B. Guo, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Speeding dumbo: Pushing asynchronous BFT closer to practice," in *29th Annual Network and Distributed System Security Symposium, NDSS 2022, San Diego, California, USA, April 24-28, 2022*. The Internet Society, 2022.

[23] 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, Vienna, Austria, October 24-28, 2016*. ACM, 2016, pp. 31–42.

[24] I. Abraham, T. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi, "Communication complexity of Byzantine agreement, revisited," *Distributed Comput.*, vol. 36, no. 1, pp. 3–28, 2023.

[25] Y. Gao, Y. Lu, Z. Lu, Q. Tang, J. Xu, and Z. Zhang, "Dumbo-ng: Fast asynchronous BFT consensus with throughput-oblivious latency," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 1187–1201.

[26] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2567. Springer, 2003, pp. 31–46.

[27] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *J. Cryptol.*, vol. 17, no. 4, pp. 297–319, 2004.

[28] R. Bacho and J. Loss, "On the adaptive security of the threshold BLS signature scheme," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 193–207.

[29] S. Das, T. Yurek, Z. Xiang, A. Miller, L. Kokoris-Kogias, and L. Ren, "Practical asynchronous distributed key generation," in *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. IEEE, 2022, pp. 2518–2534.

[30] T. Attema, R. Cramer, and M. Rambaud, "Compressed sigma-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures," in *ASIACRYPT (4)*, ser. Lecture Notes in Computer Science, vol. 13093. Springer, 2021, pp. 526–556.

[31] T. Qiu and Q. Tang, "Predicate aggregate signatures and applications," in *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 14439. Springer, 2023, pp. 279–312.

[32] S. Duan, X. Wang, and H. 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 2023, Copenhagen, Denmark, November 26-30, 2023*. ACM, 2023, pp. 815–829.

[33] H. Feng, Z. Lu, and Q. Tang, "$\widetilde{O}$ptimal adaptively secure hash-based asynchronous common subset," *Cryptology ePrint Archive*, 2024.

[34] J. Komatovic, J. Neu, and T. Roughgarden, "Toward optimal-complexity hash-based asynchronous MVBA with optimal resilience," Cryptology ePrint Archive, Paper 2024/1682, 2024.

[35] S. Das, Z. Xiang, L. Kokoris-Kogias, and L. Ren, "Practical asynchronous high-threshold distributed key generation and distributed polynomial sampling," in *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. USENIX Association, 2023, pp. 5359–5376.

[36] M. Correia, N. F. Neves, and P. Veríssimo, "From consensus to atomic broadcast: Time-free Byzantine-resistant protocols without signatures," *Comput. J.*, vol. 49, no. 1, pp. 82–96, 2006.

[37] A. Mostéfaoui, H. Moumen, and M. Raynal, "Signature-free asynchronous binary Byzantine consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time," *J. ACM*, vol. 62, no. 4, pp. 31:1–31:21, 2015.

[38] A. Patra, "Error-free multi-valued broadcast and Byzantine agreement with optimal communication complexity," in *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, ser. Lecture Notes in Computer Science, vol. 7109. Springer, 2011, pp. 34–49.

[39] J. Chen, "Optimal error-free multi-valued Byzantine agreement," in *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*, ser. LIPIcs, vol. 209. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, pp. 17:1–17:19.

[40] H. Cheng, Y. Lu, Z. Lu, Q. Tang, Y. Zhang, and Z. Zhang, "Jumbo: Fully asynchronous bft consensus made truly scalable," *arXiv preprint arXiv:2403.11238*, 2024.

[41] H. Zhang and S. Duan, "PACE: fully parallelizable BFT from reproposable Byzantine agreement," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. ACM, 2022, pp. 3151–3164.

[42] G. Bracha, "Asynchronous Byzantine agreement protocols," *Inf. Comput.*, vol. 75, no. 2, pp. 130–143, 1987.

[43] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: practical stateless hash-based signatures," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 9056. Springer, 2015, pp. 368–397.

[44] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: lightweight sublinear arguments without a trusted setup," *Des. Codes Cryptogr.*, vol. 91, no. 11, pp. 3379–3424, 2023.

[45] A. Patra and C. P. Rangan, "Communication optimal multi-valued asynchronous Byzantine agreement with optimal resilience," in *Information Theoretic Security - 5th International Conference, ICITS 2011, Amsterdam, The Netherlands, May 21-24, 2011. Proceedings*, ser. Lecture Notes in Computer Science, vol. 6673. Springer, 2011, pp. 206–226.

[46] F. Li and J. Chen, "Communication-efficient signature-free asynchronous Byzantine agreement," in *IEEE International Symposium on Information Theory, ISIT 2021, Melbourne, Australia, July 12-20, 2021*. IEEE, 2021, pp. 2864–2869.

[47] A. Mostéfaoui and M. Raynal, "Signature-free asynchronous Byzantine systems: from multivalued to binary consensus with $t < n/3$, $O(n^2)$ messages, and constant time," *Acta Informatica*, vol. 54, no. 5, pp. 501–520, 2017.

[48] T. Crain, "Two more algorithms for randomized signature-free asynchronous binary Byzantine consensus with $t < n/3$ and $O(n^2)$ messages and $O(1)$ round expected termination," *arXiv preprint arXiv:2002.08765*, 2020.

[49] K. Nayak, L. Ren, E. Shi, N. H. Vaidya, and Z. Xiang, "Improved extension protocols for Byzantine broadcast and agreement," in *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, ser. LIPIcs, vol. 179. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, pp. 28:1–28:17.

[50] X. Sui and S. Duan, "Signature-free atomic broadcast with optimal $O(n^2)$ messages and $O(1)$ expected time," *IACR Cryptol. ePrint Arch.*, p. 1549, 2023.

[51] R. Cohen, P. Forghani, J. A. Garay, R. Patel, and V. Zikas, "Concurrent asynchronous Byzantine agreement in expected-constant rounds, revisited," in *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*, ser. Lecture Notes in Computer Science, vol. 14372. Springer, 2023, pp. 422–451.

[52] G. Neiger, "Distributed consensus revisited," *Inf. Process. Lett.*, vol. 49, no. 4, pp. 195–201, 1994.

[53] I. Abraham, G. Asharov, A. Patra, and G. Stern, "Perfectly secure asynchronous agreement on a core set in constant expected time," *IACR Cryptol. ePrint Arch.*, p. 1130, 2023.

[54] S. Das, Z. Xiang, A. Tomescu, A. Spiegelman, B. Pinkas, and L. Ren, "A new paradigm for verifiable secret sharing," *IACR Cryptol. ePrint Arch.*, p. 1196, 2023.

[55] J. M. Turner, "The keyed-hash message authentication code (hmac)," *Federal Information Processing Standards Publication*, vol. 198, no. 1, pp. 1–13, 2008.

[56] G. Goren, Y. Moses, and A. Spiegelman, "Probabilistic indistinguishability and the quality of validity in Byzantine agreement," in *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*. ACM, 2022, pp. 111–125.

[57] R. E. Blahut, *Theory and practice of error control codes*. Addison-Wesley, 1983.

[58] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, ser. Lecture Notes in Computer Science, vol. 293. Springer, 1987, pp. 369–378.

[59] M. O. Rabin, "Randomized Byzantine generals," in *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 1983, pp. 403–409.

[60] M. Ben-Or and R. El-Yaniv, "Resilient-optimal interactive consistency in constant time," *Distributed Comput.*, vol. 16, no. 4, pp. 249–262, 2003.

[61] 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, Shanghai, China, October 28-31, 2017*. ACM, 2017, pp. 51–68.

[62] R. Pass and E. Shi, "Thunderella: Blockchains with optimistic instant confirmation," in *Advances in Cryptology–EUROCRYPT 2018: 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29-May 3, 2018 Proceedings, Part II 37*. Springer, 2018, pp. 3–33.

[63] P.-L. Aublin, R. Guerraoui, N. Knežević, V. Quéma, and M. Vukolić, "The next 700 bft protocols," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 4, pp. 1–45, 2015.

[64] X. Dai, B. Zhang, H. Jin, and L. Ren, "Parbft: Faster asynchronous BFT consensus with a parallel optimistic path," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*. ACM, 2023, pp. 504–518.

[65] E. Blum, J. Katz, J. Loss, K. Nayak, and S. Ochsenreither, "Abraxas: Throughput-efficient hybrid asynchronous consensus," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*. ACM, 2023, pp. 519–533.

[66] C. Berger, L. Rodrigues, H. P. Reiser, V. Cogo, and A. Bessani, "Chasing lightspeed consensus: Fast wide-area Byzantine replication with mercury," in *Proceedings of the 25th International Middleware Conference*, 2024, pp. 158–171.